การประยุกต์การควบคุมด้วย PLC/Fuzzy Logic ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว แสดงผลจัดเก็บข้อมูลและรายงานผลด้วย SCADA

The Applying PLC/Fuzzy Logic Control of Counterbalance Sphere Object on Single Plane With Genesis SCADA

คุณากร พฤกษากร

วรพงศ์ สุภรัตน์

พระงาม ไตรรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2563

การประยุกต์การควบคุมด้วย PLC/Fuzzy Logic ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว แสดงผลจัดเก็บข้อมูลและรายงานผลด้วย SCADA

The Applying PLC/Fuzzy Logic Control of Counterbalance Sphere Object on Single Plane With Genesis SCADA

คุณากร พฤกษากร

วรพงศ์ สุภรัตน์

พระงาม ไตรรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2563

The Applying PLC/Fuzzy Logic Control of Counterbalance Sphere Object on Single Plane With Genesis SCADA

Kunakorn Pruksakorn

Worapong Suparat

Prangam Trairat

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 2020

ปริญญานิพนธ์ปีการศึกษา 2563

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ การประยุกต์การควบคุมด้วย PLC/Fuzzy Logic ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว แสดงผล จัดเก็บข้อมูล และ รายงานผลด้วย SCADA

The Applying PLC/Fuzzy Logic Control of Counterbalance Sphere Object On Single Plane With Genesis SCADA

นักศึกษาผู้จัดทำ นายคุณากร พฤกษากร **รหัสนักศึกษา** 60010119

นายวรพงศ์ สุภรัตน์ **รหัสนักศึกษา** 60010893

นายพระงาม ไตรรัตน์ **รหัสนักศึกษา** 60010668

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2563

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.ดร. สุพรรณ กุลพาณิชย์	
รศ. วิริยะ กองรัตน์	

หัวข้อปริญญานิพนธ์ การประยุกต์การควบคุมด้วย PLC/Fuzzy Logic ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว แสดงผล จัดเก็บข้อมูลและรายงานผลด้วย SCADA

The Applying PLC/Fuzzy Logic Control of Counterbalance Sphere Object On Single Plane With Genesis SCADA

นักศึกษาผู้จัดทำ นายคุณากร พฤกษากร รหัสนักศึกษา 60010119

นายวรพงศ์ สุภรัตน์ รหัสนักศึกษา 60010893

นายพระงาม ไตรรัตน์ รหัสนักศึกษา 60010668

อาจารย์ที่ปรึกษา รศ.ดร. สุพรรณ กุลพาณิชย์

รศ. วิริยะ กองรัตน์

ปีการศึกษา 2563

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอโครงงาน Machine Vision โดยอาศัยแบบจำลองกระบวนการควบคุมการเคลื่อนที่วัตถุทรงกลมให้มีความสมดุลของแรงบนคานระนาบเดียวโดยมีจุดหมุนอยู่ที่กึ่งกลางคาน และในขณะเดียวกันที่ปลายของคานมี Servo Motor เป็นอุปกรณ์ขับ (Actuator) โดยอาศัยเครื่องควบคุม PLC เป็นเครื่องควบคุมหลัก และเขียนโปรแกรมควบคุม Servo Motor ด้วย CX-Programmer นอกจากนี้ยังมีกล้อง Pixy Camera ทำงานร่วมกับบอร์ด Arduino เพื่อตรวจจับการเคลื่อนที่และส่งค่าตำแหน่งของวัตถุบนคานแบบ On-Line ผ่านพอร์ทสื่อสารข้อมูล RS-232C ด้วย Host-Link Protocol ใช้เป็นสัญญาณป้อนกลับแบบวงรอบปิด (Closed-loop Control) ให้กับเครื่องควบคุม PLC ในการประมวลผลจะอาศัยหลักการของ Fuzzy Logic ที่สามารถสร้างและออกแบบเงื่อนไขเพื่อควบคุมให้วัตถุไปหยุดอยู่ที่ตำแหน่งใด ๆ บนคานได้ตามต้องการ สามารถควบคุมได้ตั้งแต่ระยะ 5 ถึง 35 เซนติเมตร โดยอ้างอิงจากด้านที่มีตะกร้าให้เป็นจุดเริ่มต้น สังเกตผลตอบสนองต่อเวลาการควบคุมของตัวแปรกระบวนการ Set Value(SV) , Present Value(PV) , Manipulate Value(MV) และสร้าง Database ใน MS SQL Server โดยนำค่า SV , PV , กราฟฟิก และปุ่มคำสั่งต่าง ๆ มาแสดงบน Genesis SCADA ผ่านโปรแกรม GraphWorX32 และทำรายงานในรูปแบบของกราฟแนวโน้ม (Trend Graph) ผ่าน TrendWorX32 Report

Thesis Title	The Applying PLC/Fuzzy Logic Control of Counterbalance Sphere Object On Single Plane With Genesis SCADA		
Producer	Kunakorn	Pruksakorn	60010119
	Worapong	Suparat	60010893
	Prangam	Trairat	60010668
Thesis Advisor	Assoc. Prof. Dr. Suphan Gulpanich		
	Assoc. Prof. Viriya Kongratana		
Year	2020		

Abstract

This thesis presents a Machine Vision using a model of a process for controlling the movement of a spherical object to balance the force on a single plane beam with the pivot point in the center of the beam. Causing the beam to tilt up-down by programming the servo motor control through the program CX-Programmer Using a PLC controller as the main control and a Pixy camera works with the Arduino board to detect movement and send on-line object position on the beam via RS-232C communication port with Host-Link Protocol. Closed-loop control to the PLC controller in processing is based on the principle of Fuzzy Logic that can create and design conditions to control the object to stop at any position on the beam as needed. The ball can be controlled from 5 to 35 centimeters referring to the basket side as the starting point. Observe the response to control times of process variables, Set Value (SV), Present Value (PV), Manipulate Value (MV) and create a database in MS SQL Server using SV, PV, Graphics, and command buttons. They can be displayed on Genesis SCADA via GraphWorX32 and make a report in the form of a Trend Graph via TrendWorX32 Report.

กิตติกรรมประกาศ

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

ขอขอบคุณผู้แต่งหนังสือเอกสารอ้างอิงและเว็บไซต์ต่างๆ คณะผู้จัดทำได้นำมาใช้อ้างอิง ประกอบการศึกษา และจัดทำปริญญานิพนธ์ฉบับนี้จนปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

โครงงานวิจัยฉบับนี้เป็นการต่อยอดจากงานวิจัยที่ผ่านมา (The Applying PLC/PID Control of Counterbalance Sphere Object On Single Plane) ซึ่งแต่เดิมเป็นการควบคุมการเคลื่อนที่ของลูกปิงปองบนคานด้วยฟังค์ชั่น PID และสั่งงานแสดงผลผ่าน HMI (Human Machine Interface) ดังนั้นในงานวิจัยนี้ขอนำเสนอการควบคุมด้วย Fuzzy Logic Control เพื่อเป็นการเปรียบเทียบข้อดีและข้อด้อยของการทำงานทั้ง 2 แบบ กับแบบจำลอง Counterbalance และเพิ่มเติมด้วยการแสดงผลพร้อมจัดเก็บข้อมูลด้วย Genesis SCADA

1.2 วัตถุประสงค์ของโครงการ

1. เพื่อพิสูจน์ทฤษฎีการควบคุมด้วย Fuzzy Logic
2. เปรียบเทียบการควบคุมด้วย PID กับ การควบคุมด้วย Fuzzy Logic

1.3 ขอบเขตโครงงาน

1. ศึกษาการทำงานของแบบจำลองคานสมดุล
   - โครงสร้างแต่ละส่วนประกอบของแบบจำลอง
   - ศึกษาชุดขับเคลื่อนทางกล และ ชุดควบคุมทางไฟฟ้าที่เป็น DC Motor, Servo Motor และ Servo Drive
2. ศึกษาการควบคุมด้วย PLC ของตะกร้าใส่ปิงปอง และคานกระดก
   - ทดสอบการควบคุมลูกบอลบนคานเป็นแบบ Manual ด้วย Joystick
   - ทดสอบการควบคุมลูกบอลบนคานด้วยฟังค์ชั่น PID

3. ศึกษาการควบคุมด้วย Fuzzy Logic Control
   - กำหนดตัวแปรและตัวควบคุม Fuzzy เพื่อกำหนดเงื่อนไขหรือกฎการควบคุม
4. เปรียบเทียบการควบคุมระหว่าง Fuzzy กับ PID
   - สรุปข้อดีและข้อด้อยของแต่ละแบบที่ใช้ควบคุมแบบจำลองตัวอย่าง
5. ศึกษาการทำงานร่วมกันระหว่าง PLC กับ Genesis SCADA
   - การสื่อสารข้อมูลระหว่างกันในรูปแบบ Modbus Protocol
   - สร้างงาน Graphic และ ฐานข้อมูล เพื่อแสดงและจัดเก็บข้อมูลขณะควบคุมทำงานแบบจำลอง
   - แสดง Report สถานะของแบบจำลอง

1.4 วิธีที่ใช้ในโครงงาน

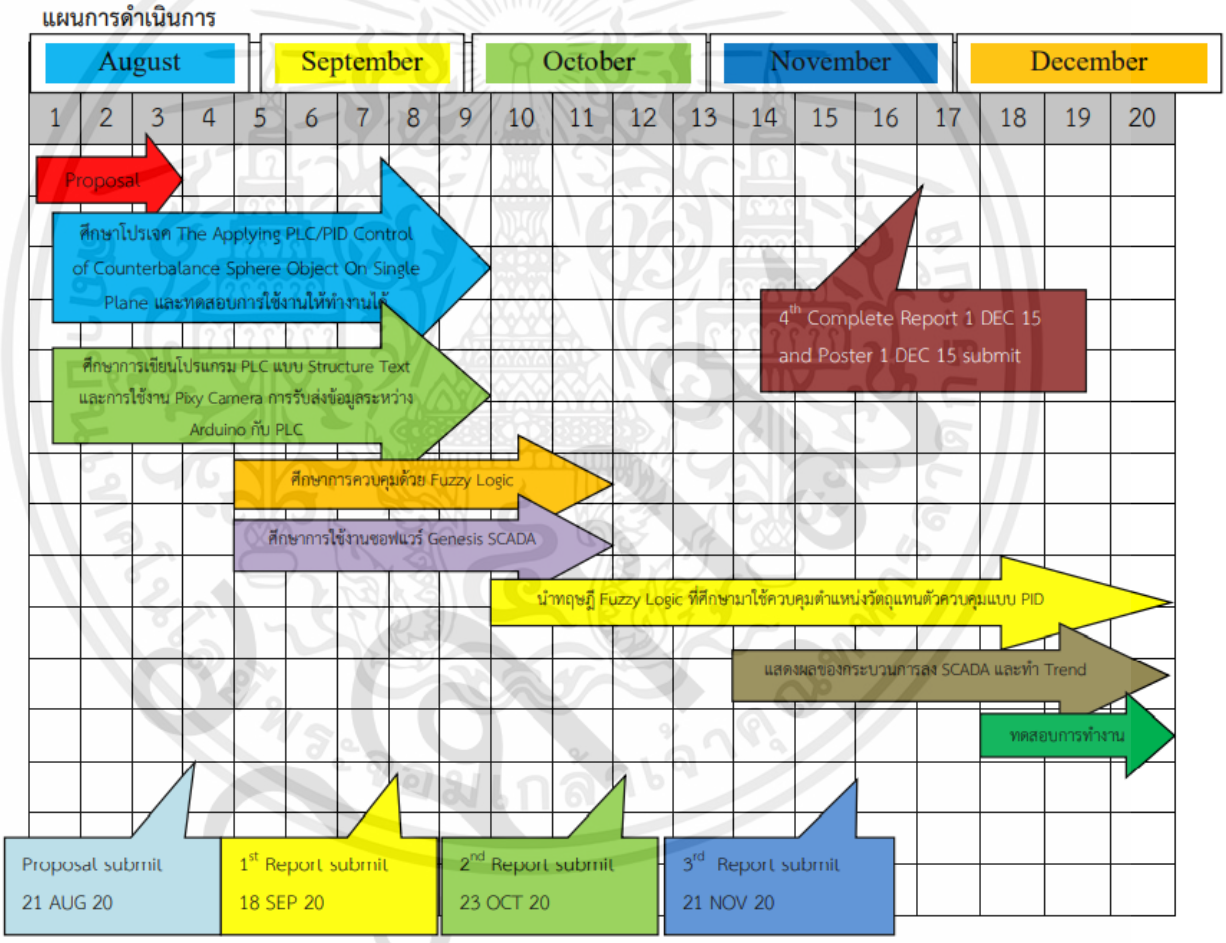
1. ศึกษาการทำงานของ PLC (Programmable Logic Controller) ซึ่งเป็นอุปกรณ์ที่ใช้ในการควบคุม
2. ศึกษาการทำงานของกล้อง PIXY MON ซึ่งใช้เป็นอุปกรณ์ใช้จับตำแหน่งของวัตถุศึกษาการทำงานของ Arduino UNO ซึ่งใช้เป็นอุปกรณ์ที่ใช้ในการแปลงสัญญาณ ตำแหน่งจากกล้อง PIXY MON ให้แก่ PLC
3. ศึกษาการทำงานของการส่งสัญญาณแบบ Serial Port ระหว่าง Arduino กับ PLC ผ่าน Host Link Protocol และหน้าจอ HMI กับ PLC โดยผ่านสายส่ง RS-232
4. ศึกษาการเขียนโปรแกรม CX-Programmer ซึ่งเป็นโปรแกรมที่ใช้สำหรับ PLC ทั้งภาษา Ladder Diagram และ Structured Text
5. ศึกษาการโปรแกรม Arduino IDE ซึ่งเป็นโปรแกรมที่ใช้สำหรับ Arduino
6. ศึกษาการควบคุมวัตถุให้สมดุลบนคานด้วยมือ(Joy Stick) เพื่อให้เข้าใจการเคลื่อนที่ของวัตถุ
7. ศึกษาทฤษฎี Fuzzy Logic ที่นำมาใช้ควบคุมแทนทฤษฎี PID
8. ศึกษาการตั้งกฎ Fuzzy Logic เพื่อใช้ในการควบคุมวัตถุให้อยู่ตามตำแหน่งที่เราต้องการ
9. ศึกษาการติดต่อระหว่าง PLC กับ OPC Server ด้วยซอฟแวร์ KEPServerEX
10. ศึกษาการทำงานร่วมกันระหว่าง PLC กับ Genesis SCADA ผ่าน Modbus Protocol
11. ศึกษาการทำงานของ Genesis SCADA เพื่อรับค่า Set Value และส่งให้กับ PLC

12. ศึกษาการสร้างงาน Graphic และ Database เพื่อแสดงและจัดเก็บข้อมูลขณะควบคุมแบบจำลอง
13. ศึกษาการทำ Report ด้วยซอฟต์แวร์ Genesis ผ่าน TrendWorX32 report

1.5 แผนการดำเนินโครงงาน

1. แผนการดำเนินโครงงานภาคเรียนที่ 1

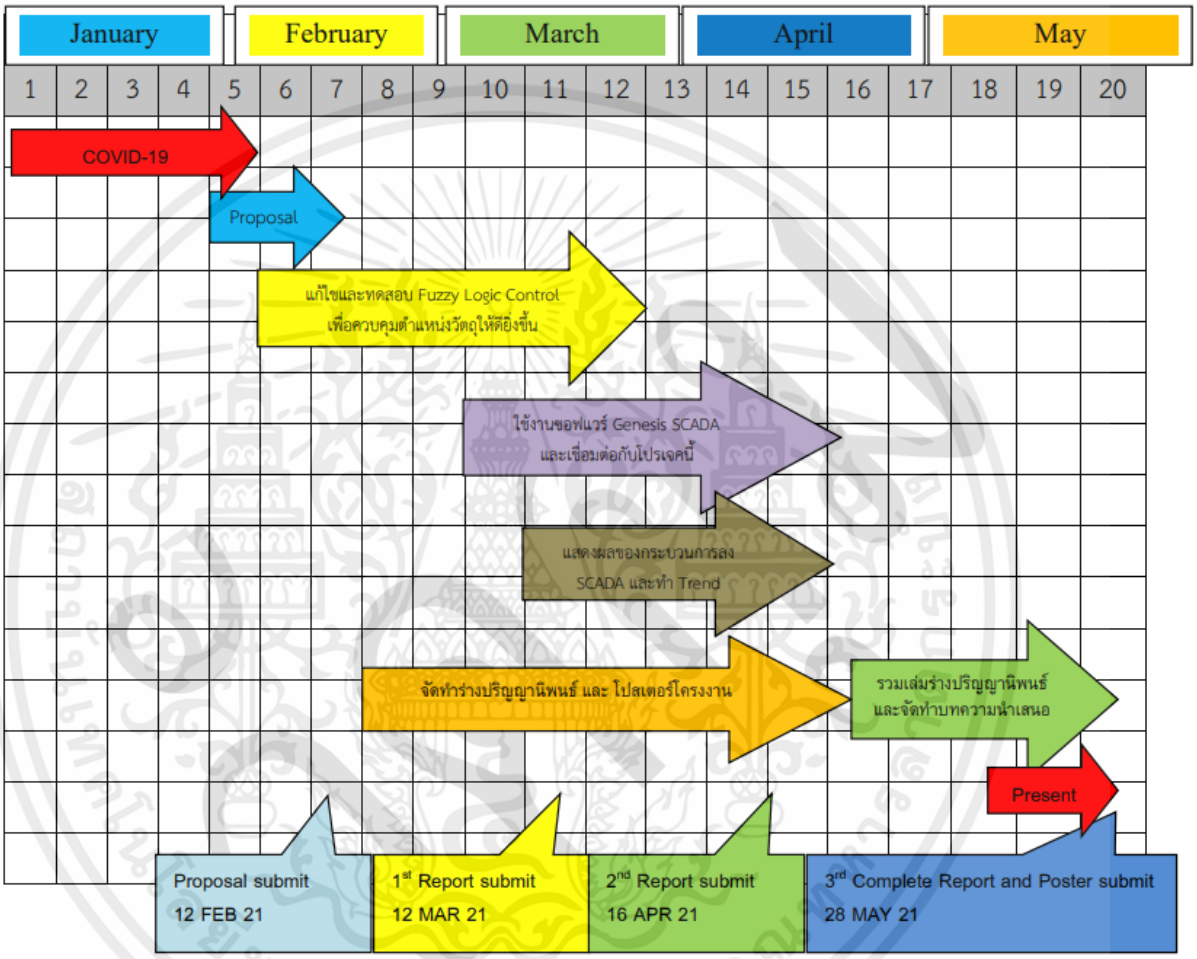
ตารางที่ 1.1 แผนการดำเนินโครงงานภาคเรียนที่ 1



2. แผนการดำเนินโครงงานภาคเรียนที่ 2

ตารางที่ 1.2 แผนการดำเนินโครงงานภาคเรียนที่ 2

แผนการดำเนินการ



1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. การใช้งาน Hardware PLC Omron CP1H และ การเขียน Software ให้กับ PLC ผ่านโปรแกรมCX-Programmer
2. การใช้งาน Servo Drive + Servo Motor และการ interface เข้ากับเครื่องควบคุม PLC
3. การใช้งาน Hardware และ Software ของ Pixy Camera และการส่งตำแหน่ง Object ไปยัง PLC ผ่าน Host Link Protocol
4. การประยุกต์ร่วมกันระหว่าง PLC & Servo Motor และ PLC & Pixy Camera
5. การติดต่อระหว่าง Hardware กับ OPC Server ด้วยซอฟต์แวร์ KEPServerEX
6. การใช้งานตัวควบคุมแบบ PID และ Fuzzy Logic
7. การแสดงผล, การทำ Database และการทำ Report ด้วยซอฟต์แวร์ Genesis

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 Programmable Logic Control (PLC)

2.1.1 Programmable Logic Control (PLC) คืออะไร

Programmable Logic Control (PLC) คืออุปกรณ์ที่ใช้ควบคุมกระบวนการทำงานต่างๆ ของเครื่องจักรในระบบอุตสาหกรรม โดยภายในมี Microprocessor ทำหน้าที่สั่งการ ซึ่ง PLC จะมีส่วนที่เป็นอินพุตและเอาต์พุตที่สามารถต่อออกไปใช้งานได้ทันที ตัวตรวจวัดหรือสวิตช์ต่างๆ จะต่อเข้ากับอินพุต ส่วนเอาต์พุตจะใช้ต่อออกไปควบคุมการทำงานของอุปกรณ์หรือเครื่องจักรเป้าหมาย ผู้ใช้งานสามารถออกแบบวงจรของการควบคุมได้โดยการป้อนเป็นโปรแกรมคำสั่งเข้าไปใน PLC นอกจากนั้นยังสามารถใช้งานร่วมกับอุปกรณ์อื่นๆเช่น เครื่องพิมพ์ (Printer), เครื่องอ่านบาร์โค๊ด (Barcode Reader) เป็นต้น ปัจจุบันนอกจากเครื่อง PLC จะสามารถใช้งานแบบเดี่ยว (Stand Alone) แล้วยังสามารถใช้งานแบบต่อร่วม PLC หลายๆ ตัวเข้าด้วยกัน (Network) เพื่อควบคุมการทำงานของระบบให้มีประสิทธิภาพมากยิ่งขึ้น ดังนั้นจะเห็นได้ว่าการใช้งาน PLC มีความยืดหยุ่นมาก จึงเป็นที่นิยมใช้งานในโรงงานอุตสาหกรรม

2.2.2 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

ตารางที่ 2.1 Specification PLC CP1H

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.)		

2.1.3 ส่วนประกอบของ PLC CP1H-XA

PLC CP1H-XA ประกอบไปด้วยส่วนต่างๆ ดังรูปที่ 2.2

2.1.3.1 Peripheral USB port เป็นจุดเชื่อมต่อสำหรับส่งข้อมูลผ่านสาย 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

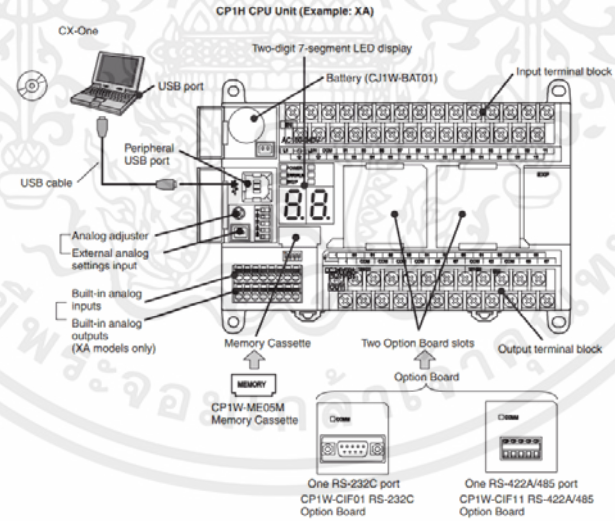
2.1.3.7 Battery (CJ1W-BAT01) เป็นตัวให้พลังงานนำไปสำรองข้อมูลเมื่อขาดไฟเลี้ยง

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

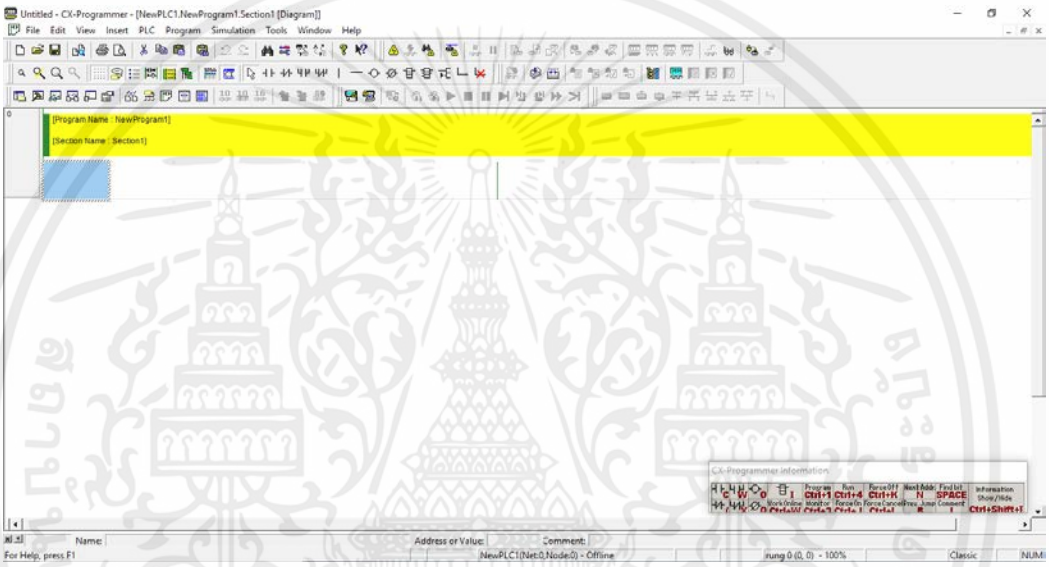
2.1.3.11 External analog settings input



รูปที่ 2.2 ส่วนประกอบ PLC CP1H-XA

2.2 CX-Programmer

CX-Programmer เป็นโปรแกรมที่ใช้สำหรับการติดต่อระหว่าง PLC กับคอมพิวเตอร์ ในการควบคุมคำสั่ง การกำหนดสัญญาณ Input และOutput การเรียงลำดับความสำคัญ การกำหนดลักษณะการตั้งค่าและการกำหนดพื้นที่หน่วยความจำ



รูปที่ 2.3 แสดงโปรแกรม CX-Programmer

2.3 กล้อง Pixy

2.3.1 กล้อง Pixy คืออะไร

รูปที่ 2.4 กล้อง Pixy 2

Pixy คือกล้องที่มีเซนเซอร์ตรวจจับภาพที่รวดเร็ว โดยส่วนใหญ่จะใช้งานที่เกี่ยวข้องกับการประดิษฐ์หุ่นยนต์หรือโครงการอื่นๆ ที่ใกล้เคียงกัน ซึ่งเราสามารถใช้วิธีสอนให้ Pixy จดจำลักษณะของวัตถุด้วยการกดปุ่มที่อยู่บนตัว Pixy นอกจากนี้ Pixy ยังสามารถใช้ในการใช้งานประเภทอื่นๆ ได้อีกมากมายไม่เฉพาะที่เกี่ยวกับหุ่นยนต์ Pixy มีศักยภาพมากพอที่จะตรวจจับและติดตามวัตถุได้ในระดับร้อยชิ้นในเวลาเดียวกันและสามารถส่งข้อมูลที่ตรวจจับมาได้ในทันที

2.3.2 กล้อง Pixy 2

Pixy2 คือกล้องที่ถูกออกแบบมาให้มีขนาดเล็กเพื่อใช้ในการตรวจจับวัตถุ, ตรวจจับเส้นทางการเคลื่อนที่ หรือใช้ในการอ่านบาร์โค้ดอย่างง่าย ซึ่งกระบวนการทำงานของ Pixy2 มันจะตรวจจับความแตกต่างของลักษณะรูปร่างและโทนสีของวัตถุที่เราต้องการ โดยแต่ละวัตถุจะมีเอกลักษณ์ของตัวเองซึ่งเราจะเรียกว่า “Signature“ โดยที่กล้อง Pixy2 จะสามารถตรวจจับความแตกต่างของวัตถุได้มากถึง 7 ลักษณะ นอกจากนี้ตัวกล้องเองยังมีอัลกอริทึมที่ใช้สำหรับติดตามทางเดิน ซึ่งความแตกต่างของกล้อง Pixy2 กับกล้องตัวอื่นคือสามารถใช้ตรวจจับสิ่งที่อยู่ข้างหน้าเพื่อกำหนดเส้นทางการเดินหรือเรียกว่า “เวกเตอร์” คล้ายกับการเดินหรือขับรถในชีวิตประจำวันของมนุษย์ที่จะต้องอาศัยการมองไปข้างหน้าเพื่อตัดสินใจว่าจะเดินหน้า,เลี้ยวซ้าย,เลี้ยวขวาหรือหยุด ในกรณีของ Pixy2 นั้นก็เช่นกัน

รูปที่ 2.5 กล่องแพคเกจกล้อง Pixy 2

2.4 PixyMon

2.4.1 PixyMon

PixyMon คือซอฟแวร์ที่ออกแบบมาเพื่อใช้งานร่วมกับกล้อง Pixy เป็นฟรีซอฟแวร์สามารถดาวน์โหลดมาใช้งานได้โดยไม่เสียค่าใช้จ่าย โดยที่เราสามารถใช้งานเพื่อเขียนชุดคำสั่งหรือแก้ไขโปรแกรมคำสั่งของกล้อง Pixy ให้ควบคุมโหมดการทำงาน ตั้งค่าให้สามารถตรวจจับวัตถุชนิดที่ต้องการ หรือตั้งค่าพื้นฐานของกล้อง ซึ่งโปรแกรม PixyMon นี้สามารถใช้งานทั้งใน Windows ,Mac OS X. และ Linux

รูปที่ 2.6 แสดงโปรแกรม PixyMon

2.4.2 การใช้งานโปรแกรม PixyMon เบื้องต้น

วิธีการใช้งานกล้อง Pixy เบื้องต้นนั้นผู้ใช้งานจำเป็นจะต้องตั้งค่าการตรวจจับวัตถุ ขนาด และสี ภายในโปรแกรม PixyMon

2.5 Arduino Uno

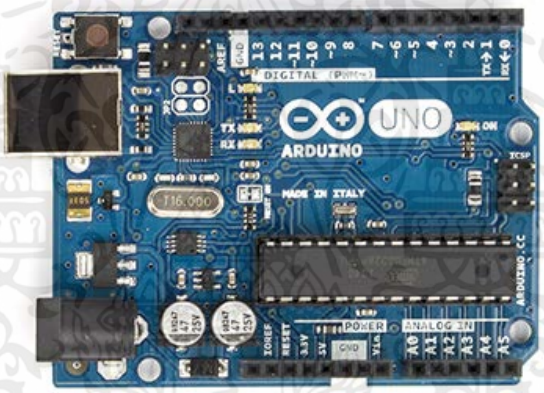
2.5.1 Arduino คืออะไร

Arduino คือ บอร์ดตระกูล AVR ที่ประกอบด้วยชิปไอซีของไมโครคอนโทรลเลอร์ตระกูลต่างๆ โดยที่ซอฟต์แวร์ของบอร์ด Arduino นี้สามารถใช้งานร่วมกับภาษา C ที่เป็นลักษณะเฉพาะ ซึ่งเป็นชุดโปรแกรมคำสั่งในไลบารี่ของ Arduino ที่สร้างขึ้นมาเพื่อใช้ในการสั่งงานการควบคุมไมโครคอนโทรลเลอร์ที่แตกต่างกันให้สามารถใช้งานโค้ดตัวเดียวกันได้ ในปัจจุบันบริษัทที่ทำการผลิตได้มี

การออกแบบตัวบอร์ดทดลองและผลิตออกมาขายอีกหลายรูปแบบ เพื่อใช้งานร่วมกับซอฟต์แวร์ IDE ของตนเอง เป็นสาเหตุหลักที่ทำให้ Arduino กลายเป็นนิยมมาก เพราะซอฟแวร์ที่ใช้งานร่วมกันสามารถดาวน์โหลดมาติดตั้งได้ฟรี และบริษัทยังแจกแปลนของตัวบอร์ดเพื่อให้สามารถถูกนำไปพัฒนาต่อได้

2.5.2 Arduino Uno

Arduino Uno เป็นบอร์ด Arduino รุ่นแรกที่ผลิตออกมา มีขนาดประมาณ 68.6x53.4 mm เป็นบอร์ดมาตรฐานที่นิยมใช้งานมากที่สุด เนื่องจากมีความเหมาะสมสำหรับการเริ่มต้นเรียนรู้ศึกษา Arduino และมี Shields ให้เลือกใช้งานได้มากกว่าบอร์ด Arduino รุ่นอื่นๆ ที่ออกแบบมาเฉพาะมากกว่า โดยบอร์ด Arduino Uno ได้มีการพัฒนาเรื่อยมา ตั้งแต่ R2 R3 และรุ่นย่อยที่เปลี่ยนชิปไอซีเป็นแบบ SMD



รูปที่ 2.7 Arduino Uno

2.6 Arduino ide

2.6.1 Arduino ide

Arduino ide คือซอฟต์แวร์ที่ถูกสร้างขึ้นมาเพื่อใช้ในการเขียนโปรแกรมชุดคำสั่งที่ใช้งานร่วมกับบอร์ด Arduino ทุกๆรุ่น โดยภายในจะมีเครื่องมือสำหรับเชื่อมต่อกับ Arduino เช่น การค้นหา Arduino ที่ติดต่อกับเครื่องคอมพิวเตอร์ การเลือกรุ่น Arduino ที่ต่ออยู่เพื่อตรวจสอบว่าขนาดของโปรแกรมที่เขียน หรือไลบรารี่ต่างๆซับพอร์ตกับ Arduino รุ่นนั้นๆไหม อีกทั้งยังมีโปรแกรมติดต่อผ่านซีเรียลโดยตรงสำหรับคอมพิวเตอร์

รูปที่ 2.8 แสดงโปรแกรม Arduino ide

2.7 Servo Motor

2.7.1 Servo Motor

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

รูปที่ 2.9 Servo Motor

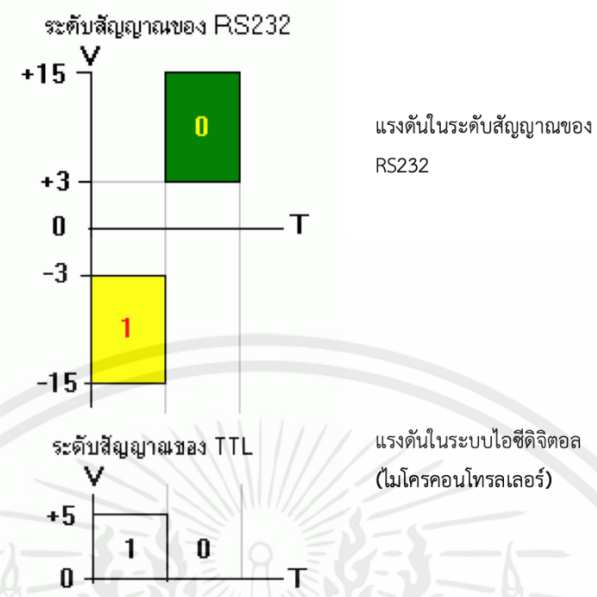
2.7.2 Servo Drive

Servo Drive คือ อุปกรณ์ที่ใช้ในอุตสาหกรรมเพื่อสั่งการ Servo Motor โดยจะทำการรับสัญญาณคำสั่งมาจาก controller หรือ ระบบควบคุมสัญญาณ และส่งกระแสไฟฟ้าเพื่อเชื่อมต่อไปยัง Servo Drive เป็นตัวผ่านการสั่งการไปยัง Servo Motor ด้วยการใช้โปรแกรมของตัวงานนั้นๆ ซึ่งจะทำการควบคุมการเคลื่อนที่ ความเร็วของการหมุน ระยะทางที่มอเตอร์หมุน และกำลังที่จะใช้หมุน (กรณีที่เครื่องจักรอุตสาหกรรมต้องใช้กำลังการทำงานสูง)

รูปที่ 2.10 Servo Drive

2.8 การสื่อสารข้อมูลผ่านพอร์ต อนุกรม RS232

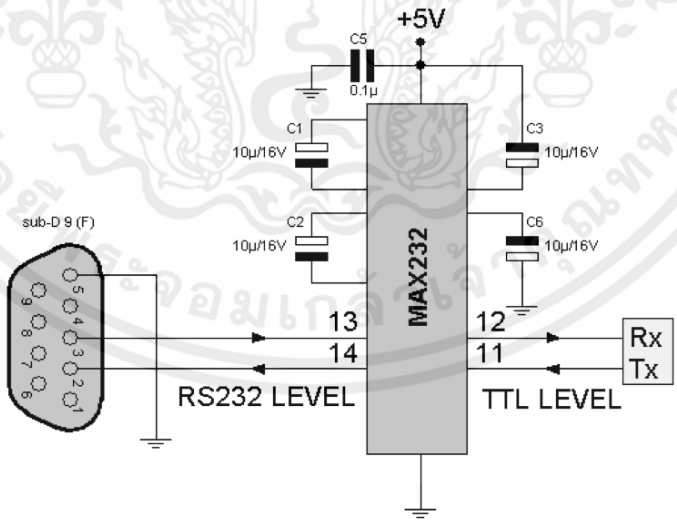
มาตรฐานการเชื่อมต่อแบบอนุกรม EIA RS-232 (x) ถูกกำหนดให้เป็นมาตรฐานอุตสาหกรรมโดยคณะกรรมการสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association) ออกแบบมาเพื่อใช้ในงานการเชื่อมต่อที่สอดคล้องกันระหว่างอุปกรณ์คอมพิวเตอร์ต่างๆ เป็นการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง มีระดับการรับส่งสัญญาณตั้งแต่ 3 โวลต์ จนถึง 15 โวลต์ สำหรับลอจิก “0” และระดับแรงดัน -3 โวลต์ จนถึง -15 โวลต์ สำหรับลอจิก “1” ซึ่งเป็นระดับแรงดันที่แตกต่างจากระบบไอซีดิจิตอลทั่วไป ดังนั้นการต่อใช้งานร่วมกับวงจรไอซีดิจิตอลจึงต้องมีอุปกรณ์ปรับเปลี่ยนระดับแรงดันจาก 3 – 15 โวลต์ ให้มีระดับแรงดัน 0 -5 โวลต์ ในภาคการส่งข้อมูล และปรับเปลี่ยนระดับแรงดันจาก 0 – 5 โวลต์ จากไมโครคอนโทรลเลอร์ให้เป็นระดับแรงดันที่สูงกว่า +3 หรือต่ำกว่า -3 ในภาคการรับข้อมูล โดยอาจจะต่อวงจรจากทรายซิสเตอร์ หรือใช้ไอซีสำเร็จรูปก็ได้



รูปที่ 2.11 แรงดันในระดับสัญญาณของ RS-232

ไอซี MAX232

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



รูปที่ 2.12 ภาพแสดงตำแหน่งขาของ MAX232 และการเชื่อมต่อ

2.8.1 RS-232

ในโครงงานนี้เราใช้สาย RS-232 ในการรับส่งข้อมูลระหว่าง PLC กับ Arduino และ PLC กับหน้าจอแสดงผล HMI แล้วใช้สาย USB2.0 ต่อจาก PLC เข้าคอมพิวเตอร์ เพื่อตั้งค่า Arduino 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 2.13 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

การสื่อสารแบบ RS232 ถูกคิดค้นขึ้นมาตั้งแต่ปี 1960 ซึ่งในปัจจุบันได้มีการสื่อสารรูปแบบใหม่ที่พัฒนาให้ดีขึ้นกว่า แต่ RS232 นั้นก็ยังมีข้อดีอยู่ คือ มีอุปกรณ์รองรับเยอะ เพราะมีอยู่ในเมนบอร์ดคอมพิวเตอร์แทบทุกรุ่น หรือที่เรียกว่า Serial port ซึ่งทำให้การสื่อสารแบบ RS232 ไม่จำเป็นต้องใช้ Converter (ตัวแปลง สัญญาณ) ในการเชื่อมต่อกับคอมพิวเตอร์ แตกต่างจากมาตรฐานใหม่อย่าง RS422, RS485 ที่แม้จะมีข้อดีมากกว่า แต่จำเป็นต้องใช้ Converter แปลงสัญญาณ ในปัจจุบันเมนบอร์ดรุ่นใหม่ๆ ได้นำ Serial port ออกจากเมนบอร์ดและเพิ่ม Port การสื่อสารใหม่ที่เป็นที่นิยมคือการสื่อสารแบบ USB ซึ่งจะมาแทนที่การสื่อสารแบบเดิมอย่าง RS232 ให้เสื่อมความนิยมไปในที่สุด

2.8.2 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 คือหมายเลขเครื่องในการเชื่อมต่อที่เป็นโครงข่ายแบบหลายจุดนั้นเครื่องควบคุมที่เชื่อมต่ออยู่ในระบบจะมีมากกว่า 1 เครื่อง การกำหนดว่าต้องการส่งข้อมูลให้กับเครื่องควบคุมตัวใด
- เพื่อใช้เลือกเครื่องควบคุมที่ XX ในระบบเครือข่าย Host Link
- Header เป็นส่วนของคำสั่งหลักที่จะกำหนดว่าต้องการกระทำกับข้อมูลส่วนใด เช่น ต้องการอ่านข้อมูลของอินพุท ต้องการเขียนข้อมูลให้กับตัวนั้น เป็นต้น
- Text เป็นส่วนของข้อมูล เช่น ค่าที่อ่านได้จากอินพุท หรือ ค่าใช้จะต้องเขียนในพื้นที่ต่างๆ
- FCS (Frame Check Sequence) เป็นส่วนควบคุมความผิดพลาดของข้อมูลซึ่งได้จากการคำนวณ
- * (Terminate) เป็นส่วนที่ใช้ในการปิดท้ายคำสั่งให้ทราบว่าจบบล็อก
- CR (Caries Return) เพื่อให้ข้อความที่ตามมาขึ้นบรรทัดใหม่

ตารางที่ 2.2 ตารางการคำนวณหาค่า 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 2.3 ตัวอย่างตาราง Command Block

CHAR	ASCII BINARY								HEX
	7	6	5	4	3	2	1	0	
@	0	1	0	0	0	0	0	0	40
0	0	0	1	1	0	0	0	0	30
	0	1	1	1	0	0	0	0	
0	0	0	1	1	0	0	0	0	30
	0	1	0	0	0	0	0	0	
R	0	1	0	1	0	0	1	0	52
	0	0	0	1	0	0	1	0	
R	0	1	0	1	0	0	1	0	52
	0	1	0	0	0	0	0	0	
0	0	0	1	1	0	0	0	0	30
	0	1	1	1	0	0	0	0	
0	0	0	1	1	0	0	0	0	30
	0	1	0	0	0	0	0	0	
3	0	0	1	1	0	0	1	1	33
	0	1	1	1	0	0	1	1	
0	0	0	1	1	0	0	0	0	30
	0	1	0	0	0	0	1	1	
0	0	0	1	1	0	0	0	0	30
	0	1	1	1	0	0	1	1	
0	0	0	1	1	0	0	0	0	30
	0	1	0	0	0	0	1	1	
0	0	0	1	1	0	0	0	0	30
	0	1	1	1	0	0	1	1	
2	0	0	1	1	0	0	1	0	32
FCS	0	1	0	0	0	0	0	1	41

จากตัวอย่างนี้เป็นก่ารทดลองป้อนคำสั่งเพื่ออ่านข้อมูลจาก CH 30 และ CH 31 ผลลัพท์ของ FCS คือ 41 บล็อกคำสั่งที่สมบูรณ์คือ @00RR0030000241* และผลตอบสนองที่ได้คือ @00RR00E96E7B1C48* ในแต่ละส่วนอธิบายได้ดังนี้

@00RR หมายถึง การอ่านข้อมูลในพื้นที่ IR จากเครื่องควบคุมหมายเลข 00

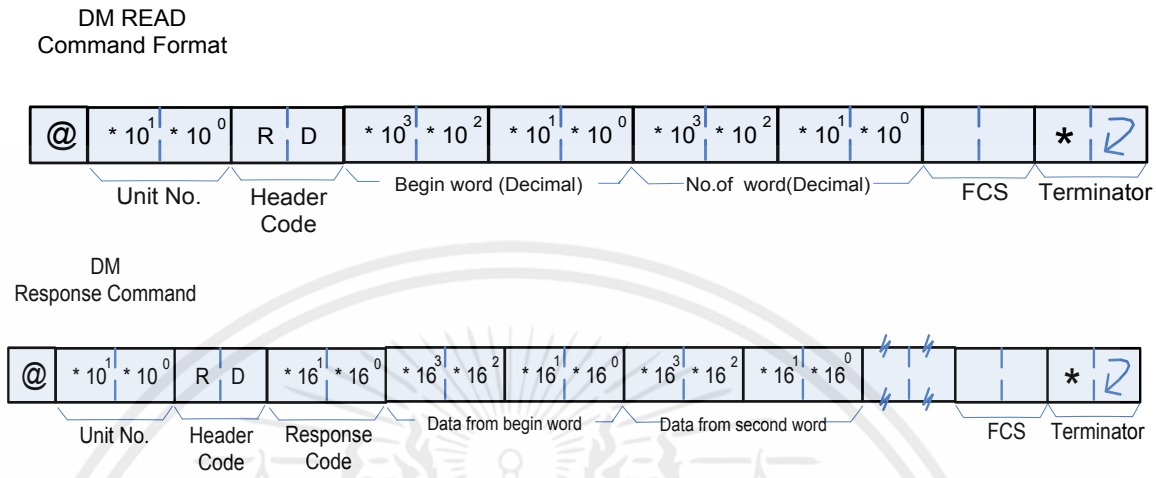
00 หมายถึง Response Code แสดงรายละเอียดของความผิดพลาดในบล็อกตอบสนอง

E964 หมายถึง ข้อมูลจาก CH 30 เป็นเลขฐาน 16

7B1C หมายถึง ข้อมูลจาก CH 31 เป็นเลขฐาน 16

48 หมายถึง FCS ที่เครื่องควบคุมส่งมาในบล็อกตอบสนอง

DM Read Command & Response Command



2.9 Fuzzy Logic

ฟัซซี่ลอจิกหรือการใช้เหตุผลแบบประมาณ ซึ่งแตกต่างจากตรรกะบูลีนแบบดั้งเดิม (traditional Boolean logic) ที่เป็นการใช้เหตุผลในลักษณะ ถูก/ผิด, ใช่/ไม่ใช่ แบบเด็ดขาด

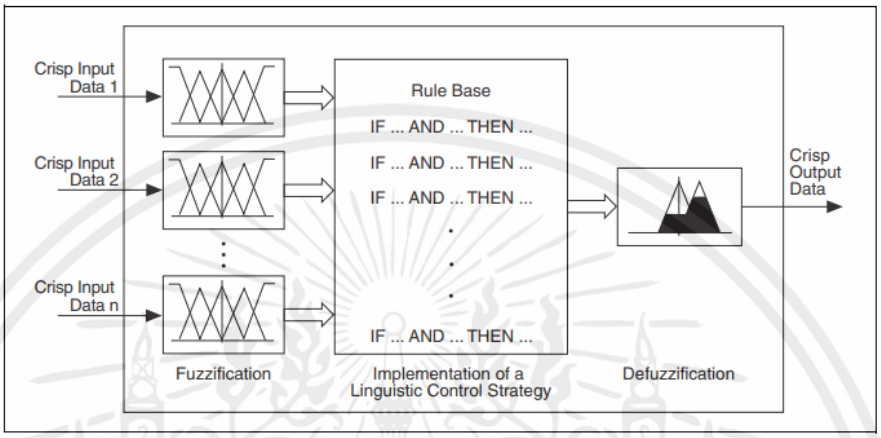
ค่าระดับความจริงของ Fuzzy Logic ใช้สำหรับระบุค่าความเป็นสมาชิกของเซต (Degree of Membership) ตัวอย่างเช่น สมมติว่า นาย A กำลังเดินเข้าห้องนอน สถานะของนาย A ตามตรรกะบูลีนแบบดั้งเดิม (traditional Boolean logic) คือ "อยู่ในห้องนอน" หรือ "อยู่นอกห้องนอน" แต่ถ้านาย A ยืนอยู่ระหว่างประตูห้องนอน จะพิจารณาได้ว่าเขา "อยู่ในห้องนอนบางส่วน" ระดับของสถานะนี้จะระบุด้วยค่าความเป็นสมาชิกของเซต สมมุติเขาเพิ่งจะก้าวปลายนิ้วเท้าผ่านข้ามไประตูเข้าห้องนอน เราอาจกล่าวว่า นาย ก นั้น 0.99 "อยู่นอกบ้าน"

Fuzzy Logic สามารถระบุค่าความเป็นสมาชิกของเซต (Degree of Membership) ด้วยค่าระหว่าง 0 และ 1 ซึ่งมีประโยชน์ในการจำลองระดับซึ่งสามารถระบุด้วยคำพูด "เล็กน้อย" "ค่อนข้าง" "มาก" โดยใช้ค่าความเป็นสมาชิกของเซต

2.9.1 Fuzzy Logic Control System

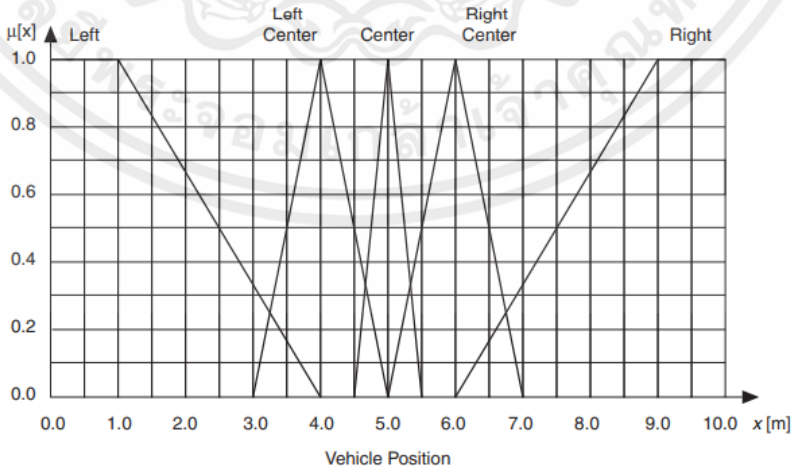
ระบบควบคุมแบบฟัซซี่ลอจิก (Fuzzy Logic Control System) คือ ระบบกฎพื้นฐานอันเป็นกฎฟัซซี่ (Fuzzy rule) ที่สามารถแทนส่วนที่ใช้แทนการตัดสินใจทางกล (Human mechanical) เพื่อให้ได้คำตอบที่แน่นอนของระบบ จุดประสงค์หลักของระบบควบคุมแบบฟัซซี่ลอจิกคือ ใช้แทนส่วน

ระบบปฏิบัติการของมนุษย์ (Human Operator) ด้วยกฎพื้นฐานของฟัซซี่ (Fuzzy Rule Based System) โดยทั่วไประบบควบคุมฟัซซี่ลอจิกจะมี 3 ส่วน



รูปที่ 2.14 กระบวนการของ Fuzzy Logic Control System

ส่วนที่ 1 เรียกว่า “ฟัซซิฟิเคชั่น” (Fuzzification) เป็นกระบวนการเชื่อมโยงค่าอินพุตหรือเอาต์พุตกับเงื่อนไขทางภาษา (Left, Left Center, ..., Right) หรือก็คือการกำหนด Membership Function



รูปที่ 2.15 ตัวอย่างการกำหนด Membership Functions

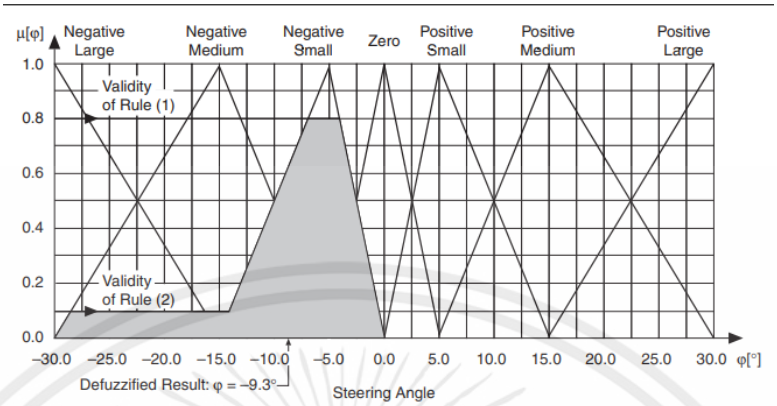
ส่วนที่ 2 เป็นส่วนของการประมวลผลตามกฎพื้นฐาน (Rules Based) ที่ได้มีการกำหนดไว้ ตัวอย่างเช่น อุณหภูมิปัจจุบันของห้องคือ 50 องศา ซึ่งสอดคล้องกับเงื่อนไขทางภาษา Cold ที่ค่าความเป็นสมาชิกของเซต 0.4 สมมติว่าอุณหภูมิที่ต้องการคือ 70 องศา ซึ่งสอดคล้องกับเงื่อนไขทางภาษา moderate ที่ค่าความเป็นสมาชิกของเซต 0.8 ตัวควบคุมฟัซซี่ก็จะเรียกใช้กฎพื้นฐาน ถ้าหากอุณหภูมิปัจจุบันคือ Cold และอุณหภูมิที่ต้องการคือ moderate ดังนั้นการตั้งค่าฮีตเตอร์คือ Low โดยค่าความเป็นสมาชิกของเซตจะมีวิธีการคำนวณดังนี้

- AND (Minimum) : $\mu A \bullet B = \min(\mu A, \mu B)$
- AND (Product) : $\mu A \bullet B = (\mu A, \mu B)$
- OR (Maximum) : $\mu A + B = \max(\mu A, \mu B)$
- OR (Probabilistic) : ((A+B)-(AB))

สมมติว่ากฎที่เรียกใช้ในตัวอย่างนี้ใช้ AND (Minimum) ดังนั้นการตั้งค่าฮีตเตอร์คือ Low โดยค่าความเป็นสมาชิกของเซตจะเท่ากับ 0.4

ส่วนที่ 3 เรียกว่า “ดีฟัซซิฟิเคชั่น” (Defuzzification) เป็นกระบวนการในการแปลงค่าความเป็นสมาชิกของตัวแปรภาษาศาสตร์เอาต์พุตภายในคำศัพท์ทางภาษาให้เป็นค่าตัวเลขที่ชัดเจน ตัวควบคุมแบบฟัซซีสามารถใช้วิธีการทางคณิตศาสตร์วิธีใดวิธีหนึ่งในการคำนวณ

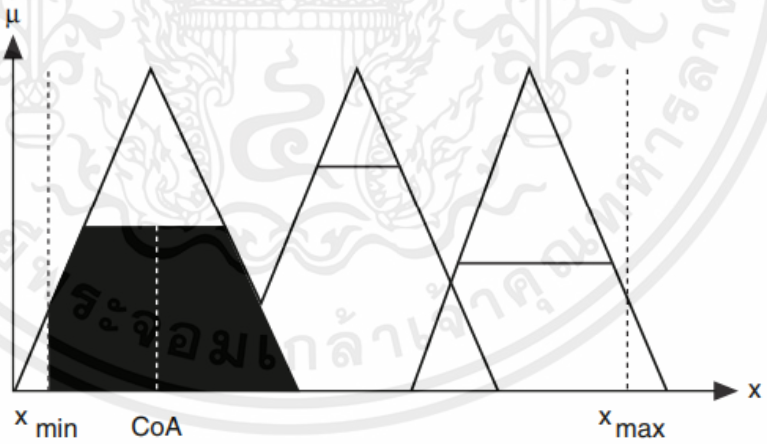
- Center of Area



$$CoA = \frac{\int_{x_{min}}^{x_{max}} f(x) \cdot x \, dx}{\int_{x_{min}}^{x_{max}} f(x) dx}$$

รูปที่ 2.16 Center of Area

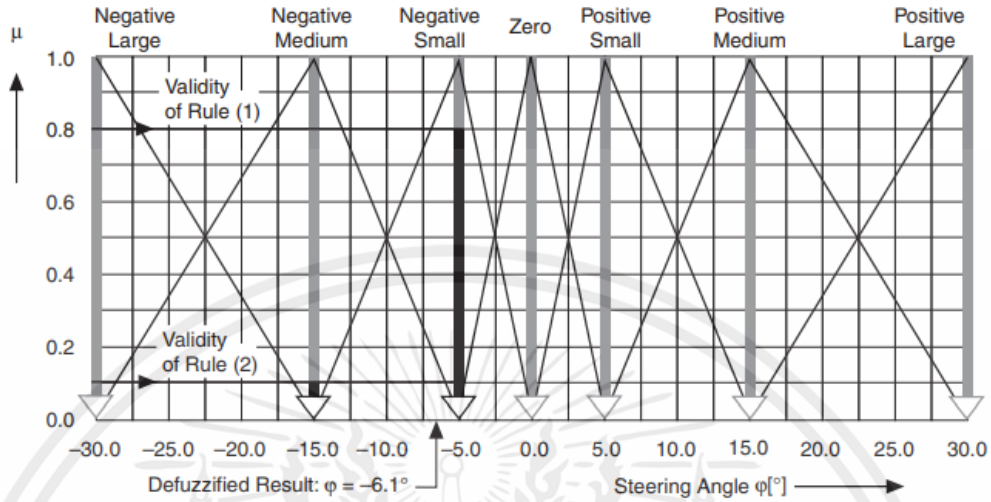
- Center of Sums



$$x_{final} = \frac{(CoA_1 area_1 + CoA_2 area_2 + \ldots + CoA_n area_n)}{(area_1 + area_2 + \ldots + area_n)}$$

รูปที่ 2.17 Center of Sums

- Center of Maximum



$$x_{final} = \frac{(x_1\mu_1 + x_2\mu_2 + \ldots + x_n\mu_n)}{(\mu_1 + \mu_2 + \ldots + \mu_n)}$$

รูปที่ 2.18 Center of Maximum

2.10 SCADA

SCADA นั้นย่อมาจากคำว่า Supervisory Control And Data Acquisition เป็นระบบตรวจสอบและวิเคราะห์ข้อมูลแบบ Real-time โดยส่งข้อมูลสู่ส่วนกลางของระบบ SCADA นอกจากนั้น SCADA ยังทำหน้าที่คำนวณและประมวลผลข้อมูลจากอุปกรณ์ฮาร์ดแวร์ เช่น DCS, PLC, Controller, RTU พร้อมแสดงข้อมูลไปที่หน้าจอ หรือส่งสัญญาณเพื่อควบคุมฮาร์ดแวร์นั้นๆ เช่น ถ้าหากความดันของอุปกรณ์สูงเกินค่าที่ต้องการ ให้ปิดการทำงานของอุปกรณ์นั้น โดยควบคุมผ่าน PLC หรือController ที่เชื่อมต่ออยู่เป็นต้น ข้อดีของ SCADA คือสามารถเก็บรวบรวมข้อมูลที่ได้จากระบบควบคุมทั้งหมดไว้ในฐานข้อมูลเพื่อนำไปใช้งานต่อได้ ทั้งนี้ SCADA ยังสามารถเข้าไปมีส่วนในงานควบคุมต่างๆที่ต้องการแสดงผล หรือแลกเปลี่ยนข้อมูลในระบบควบคุมต่าง ๆ จากส่วนกลาง เพื่อมองเห็นการภาพรวมการทำงานทั้งหมดของระบบต่างๆที่สัมพันธ์กัน

SCADA สามารถลดความขัดข้องในกระบวนการอุตสาหกรรม/วิศวกรรมได้เนื่องจากผู้ใช้รับทราบเหตุการณ์และแก้ไขได้ทันท่วงที ทำให้ช่วยลดDown Timeช่วยให้การทำงานหรือการผลิตมีความต่อเนื่อง ซึ่งส่งผลต่อศักยภาพการผลิต นอกจากนี้ยังสรุปปัญหาที่เกิดขึ้นพร้อมสภาพแวดล้อม/พารามิเตอร์ต่างๆที่สนใจที่ช่วยวิเคราะห์สาเหตุของปัญหาได้ และยังเพิ่มศักยภาพการบริหารธุรกิจและอุตสาหกรรมเนื่องจากผู้บริหารสามารถตัดสินใจบนพื้นฐานข้อมูลที่แม่นยำและรวดเร็ว โดยข้อมูลมาจากรายงานที่รวบรวมและสรุปผลด้วยฟีเจอร์ของSCADAซึ่งมีความเที่ยงตรงและรวดเร็วกว่ามนุษย์ ดังนั้นธุรกิจและอุตสาหกรรมที่ใช้ระบบSCADAจึงมีความได้เปรียบทางธุรกิจมากกว่า

SCADA เริ่มใช้งานในคอมพิวเตอร์ตั้งแต่ระบบปฏิบัติการ DOS, VMS และ UNIX จนมาถึงระบบปฏิบัติการยุคปัจจุบันซึ่งขยายไคลเอ็นท์ไปถึงคอมพิวเตอร์พกพาและSmart Phone/Tablet

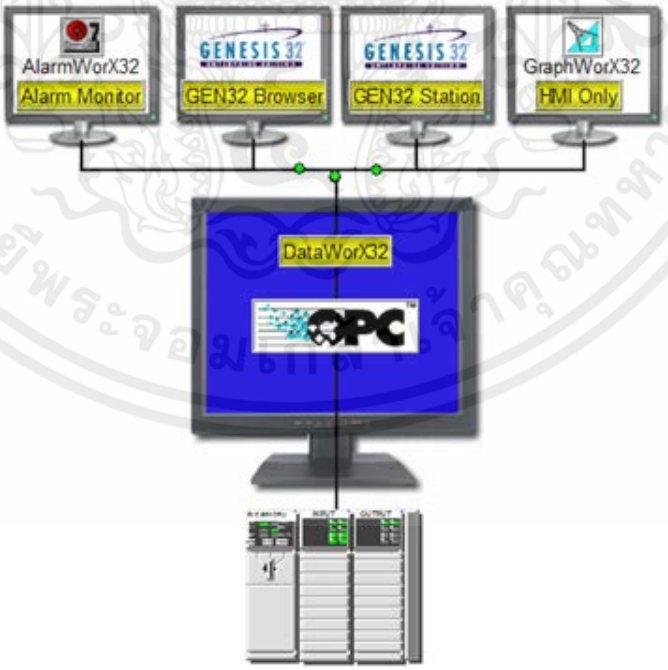
2.10.1 GENESIS32

GENESIS32 คือชุดซอร์ฟแวร์ประเภท Web-enabled Industrial Automation ผลิตโดยบริษัท ICONICS ที่ทำงานบนระบบปฏิบัติการวินโดวส์โดยใช้เทคโนโลยีการสื่อสารข้อมูลของ OPC แบบ Client – Server เพื่อสร้างแอพลิเคชั่นสำหรับระบบ SCADA และ Human Machine Interface (HMI) รองรับการทำงานแบบ Multi-processor และ Hyper Threading system

GENESIS32 ประกอบด้วยโมดูลย่อยต่าง ๆ ที่ทำงานเป็นอิสระต่อกันเพื่อสร้างแอพลิเคชั่นแบบต่าง ๆ ตามลักษณะงาน โดยประกอบด้วยโมดูลหลัก3โมดูลคือ โมดูลสำหรับแอพลิเคชั่นแบบกราฟิก แอพลิเคชั่นสำหรับรวบรวมข้อมูล / พล็อตกราฟข้อมูลแบบต่อเนื่อง (Trending) และโมดูลสำหรับสร้างระบบแจ้งเตือน (Alarm) ตามลำดับดังนี้

- GraphWorX32 โมดูลหลักสำหรับสร้างระบบกราฟิก, การแสดงผลและสั่งการ
- TrendWorX32 โมดูลแสดงกราฟTrend แบบ Real Time และ History
- TrendWorX Automatic Report (Excel, Text) ออกแบบ Template เองได้ ใช้สูตร Excel ได้ ใส่ Chart ได้, สร้างรายงาน On Event ได้ และส่งออกอัตโนมัติทางอีเมล
- AlarmWorX32 โมดูลตรวจจับและแสดงผล Alarm ตามเงื่อนไขที่สร้างไว้
- Web-based Workbench โมดูลจัดการคอนฟิก GENESIS32 โมดูลต่างๆผ่านเว็บ
- Modbus Ethernet OPC Server
- Modbus Serial OPC Server

- SNMP Connectivity ติดต่ออุปกรณ์ที่สนับสนุน SNMP เช่น PC/Printer/Router/UPS/เป็นต้น
- Unified Data Manager โมดูลสร้าง Expession, Register, Trigger, Conversion
- Unified Data Browser โมดูลBrowserเพื่อเลือกสัญญาณต่างๆมาใช้งานใน GraphWorX, TrendWorX, AlarmWorX เป็นต้น
- MonitorWorX32 โมดูลตรวจสอบสถานะการทำงาน ไลเซนส์ จำนวนtagที่ใช้งานอยู่ที่เหลืออยู่
- ScriptWorX/ ScriptWorX-2010 โมดูลเขียนScriptแบบVB
- ProjectWorX32 โมดูลเพื่อการบีบอัดและแจกจ่ายไฟล์ต่างๆโดยรวมเป็นโปรเจค
- GEN32-Symbols กราฟิกไลบรารี่รวบรวมสัญลักษณ์กราฟิกในอุตสาหกรรมต่างๆเพื่อใช้ในการออกแบบหน้าแสดงผลกราฟิก
- GenBroker Networking โมดูลจัดการติดต่อClient/Serverผ่านระบบเครือข่าย
- Database Connectors ส่วนติดต่อฐานข้อมูล เช่นMicrosoft SQL Server, Microsoft Access, SAP, ODBC(กรณีOracle, SAP ท่านต้องมีไลเซนส์ในการติดต่อเพิ่มเติม)



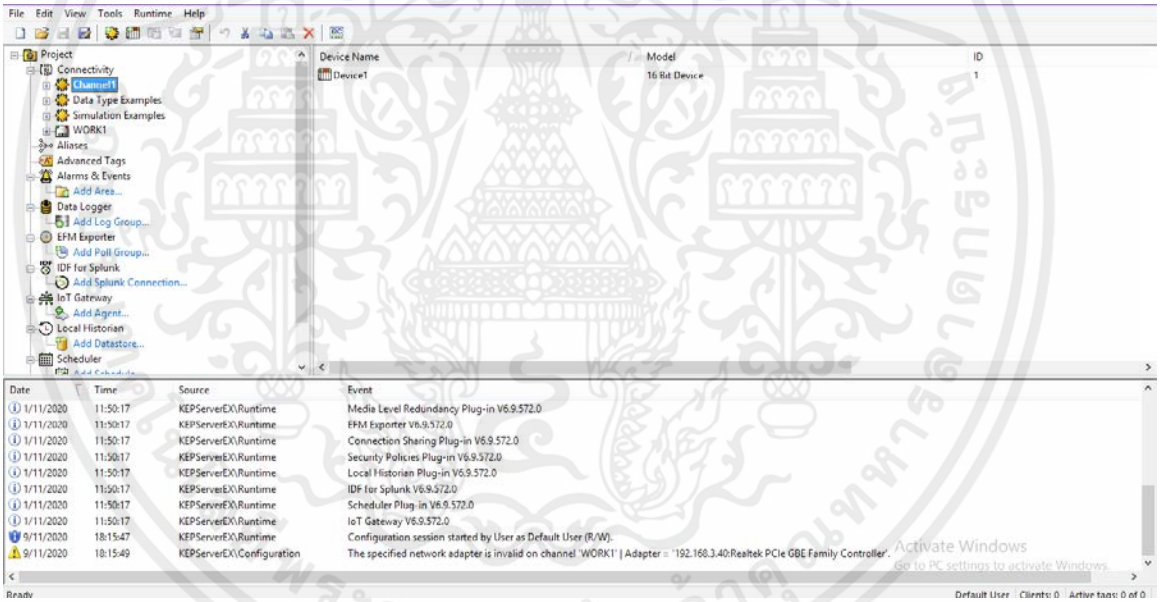
รูปที่ 2.19 โครงสร้างการทำงานของGENESIS32 SCADA

2.11 OPC Server

SCADA โดยทั่วไปประกอบด้วยส่วนรับข้อมูลจาก Device ต่าง ๆ ซึ่งมีทั้งส่วนรับข้อมูลที่เรียกว่า OPC Server และ Driver ในปัจจุบันนิยมใช้ OPC server มากกว่าเนื่องจากมีมาตรฐานที่ไม่ขึ้นกับค่ายหรือยี่ห้อ ซ้ำยังมีให้เลือกซื้อจากผู้ผลิตจำนวนมาก และมีองค์กรกลาง คือ OPC Foundation เป็นผู้ดูแลตรวจสอบมาตรฐานและพัฒนามาตรฐานของ OPCซึ่งย่อมา จาก OLE for Process Control เมื่อ OPC Server ซึ่งเป็นซอร์ฟแวร์รับส่งข้อมูลระหว่าง Device ส่งให้ซอร์ฟแวร์ SCADA เพื่อนำไปแสดงผลในแบบกราฟิก เก็บข้อมูล พล็อตสัญญาณ และแจ้งเตือน

2.11.1 KEPServerEX

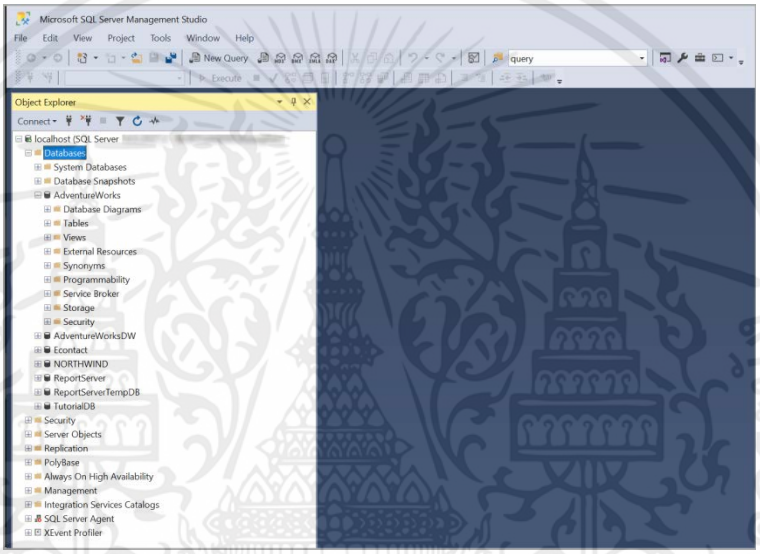
ใช้เป็นที่แลกเปลี่ยนข้อมูลระหว่าง SCADA ซึ่งเป็นของฟรี และสนับสนุนรูปแบบข้อมูลทุกประเภท (Word, DWord, String, bit, Float, Double, Long, Short, BCD)



รูปที่ 2.20 โปรแกรม KEPServerEX

2.12 SQL Server Management Studio (SSMS)

ใช้ในการสร้าง Database ซึ่ง SQL Server Management Studio เป็นแอปพลิเคชั่นซอฟต์แวร์ที่ใช้สำหรับการเข้าถึงข้อมูล(accessing), การกำหนดค่า (configuring), การจัดการ (managing), การบริหาร (administering) และการพัฒนา (developing) ทุกองค์ประกอบของ SQL Server โดย SSMS ได้รวบรวมเครื่องมือต่างๆ ในรูปแบบ ของรูปภาพกราฟิก



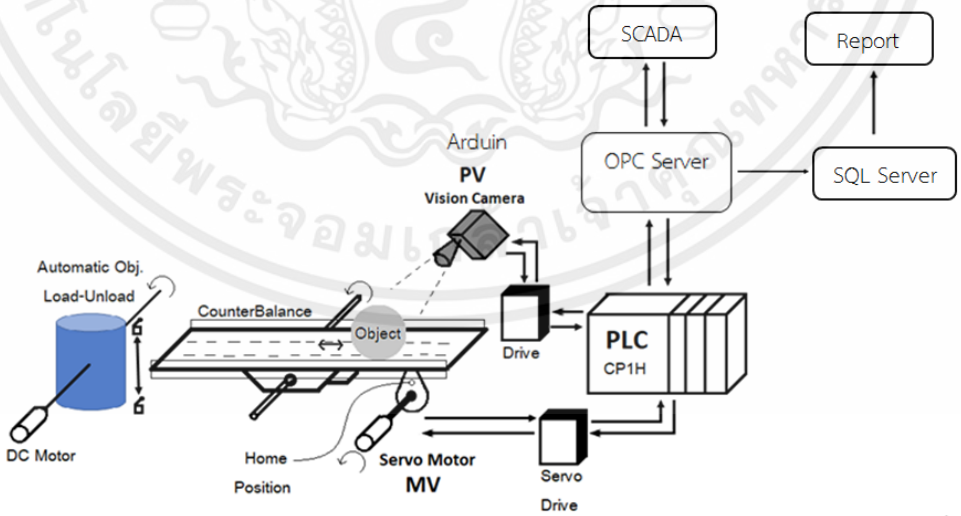
รูปที่ 2.21 โปรแกรม SQL Server Management Studio

บทที่ 3

วิธีการดำเนินงาน

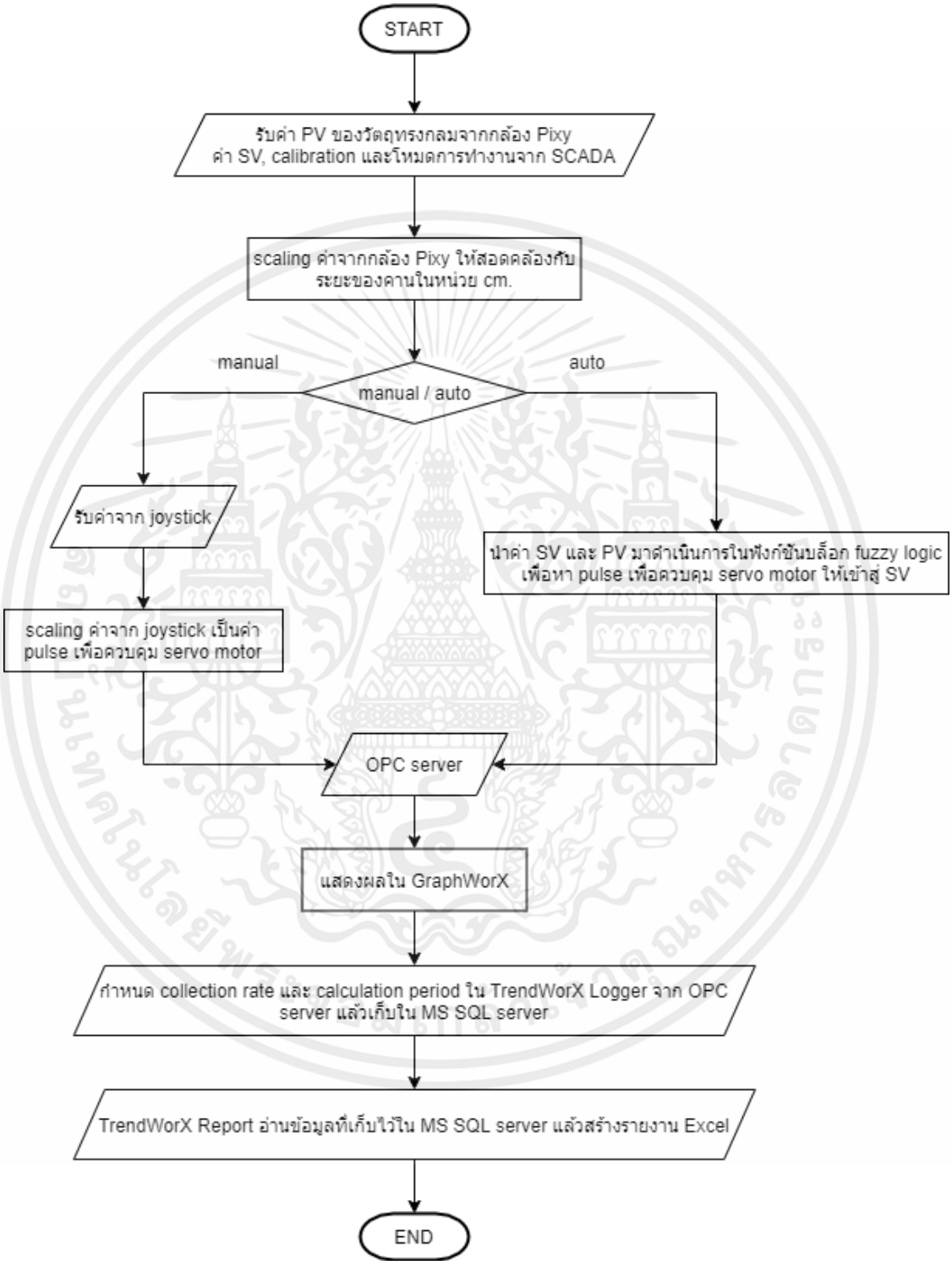
3.1 หลักการสร้างและออกแบบ

ในการศึกษาการควบคุมตำแหน่งของลูกปิงปองในระนาบเดียวจะใช้หลักการสมดุลของโมเมนต์ (การหมุนของคาน) โดยการจะให้เคลื่อนที่ไปยังตำแหน่ง SV (Set Value) ได้นั้น จะต้องมีการรับค่าตำแหน่งของลูกปิงปอง PV (Process Variable) มาจาก Pixy Camera ซึ่งเป็นโมดูลที่ใช้งานร่วมกับ Arduino และจะนำค่า PV (Process Variable) ป้อนกลับ (Feedback) ไปยัง PLC ผ่าน RS-232 ในรูปแบบของ Host Link Protocol เพื่อนำข้อมูลที่ได้มาเปรียบเทียบกับ SV (Set Value) จากนั้น Controller ทราบค่า Error แล้วจึงนำผลมาวิเคราะห์ด้วย Fuzzy Logic เพื่อควบคุม Servo motor และในขณะเดียวกันที่ปลายของ Servo motor จะติดอยู่กับลูกเบี้ยวซึ่งยึดติดอยู่กับคาน เมื่อไรที่ Servo motor หมุน ก็จะส่งผลให้คานเกิดการกระดกขึ้นลง ดังนั้นวัตถุทรงกลมสามารถเคลื่อนที่ไปยังตำแหน่งที่ต้องการได้ โดยเขียนโปรแกรมควบคุม Servo motor ผ่านโปรแกรม CX-Programmer และสร้าง Database ใน MS SQL Server และนำค่า PV, กราฟฟิก และปุ่มคำสั่งมาแสดงบน Genesis SCADA ผ่านโปรแกรม GraphWorX32 และทำ Report ผ่าน TrendWorX32 report



รูปที่ 3.1 โครงสร้างของ FUZZY CONTROL OF COUNTERBALANCE SPHERE OBJECT ONSINGLE PLANE

3.2 แผนผังการดำเนินงาน



รูปที่ 3.2 Flowchart แสดงหลักการทำงาน

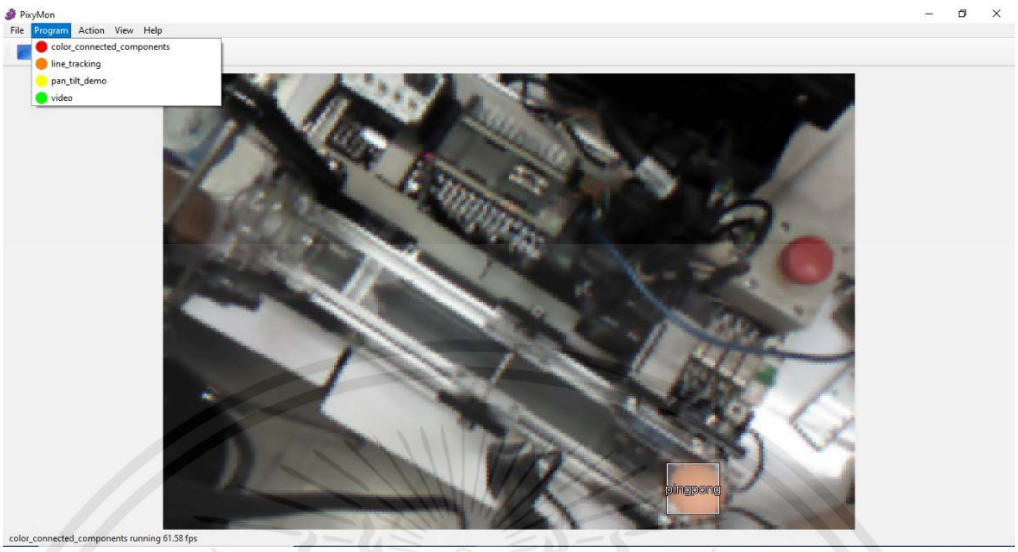
3.3 การทำงานของ Vision camera และ Arduino code

3.3.1 การใช้งานโปรแกรมสแกนวัตถุ

การใช้งานวัตถุต้องเชื่อมสายเชื่อมต่อระหว่าง Arduino และ Pixy2 เพื่อจ่ายไฟเลี้ยงให้ตัวกล้องหรืออีกวิธีเราสามารถใช้สาย USB เชื่อมต่อตัวกล้องเข้ากับคอมพิวเตอร์ได้โดยตรงเราสามารถตั้งค่าการทำงานหรือสอนให้กล้องเรียนรู้การจำแนกแยกสีโดยโปรแกรม Pixymon

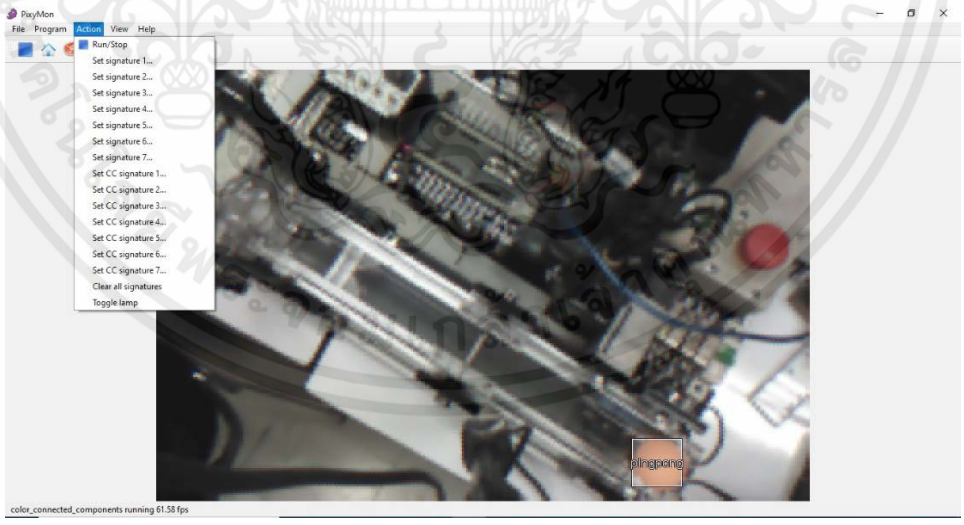
รูปที่ 3.3 หน้าตาโปรแกรม pixymon ส่วนที่ 1

ในส่วนแรก คือส่วนของ File สามารถตั้งค่าต่างๆของกล้องได้จากส่วนนี้ เช่น ค่าพารามิเตอร์ของกล้อง ค่าความกว้างของวัตถุที่ต้องการตรวจจับ หรือแม้กระทั่งตั้งชื่อของวัตถุที่เราตรวจจับได้เช่นในตัวอย่างนี้เราได้ตั้งค่าลูกปิงปองที่จับได้ว่า “pingpong”



รูปที่ 3.4 หน้าตาโปรแกรม pixymon ส่วนที่ 2

ในส่วนที่สองของโปรแกรมเราสามารถเลือกได้ว่าต้องการใช้ option ไหนของตัวกล้องซึ่งกล้อง pixy2 มีทั้งหมด 4 options ให้เลือก โดยในโปรเจคของเราเราได้เลือกใช้ color_connected_componernts (ccc) หรือจะเรียกกว่า ความสามารถในการตรวจจับและจำแนกแยกสีของวัตถุ



รูปที่ 3.5 หน้าตาโปรแกรม pixymon ส่วนที่ 3

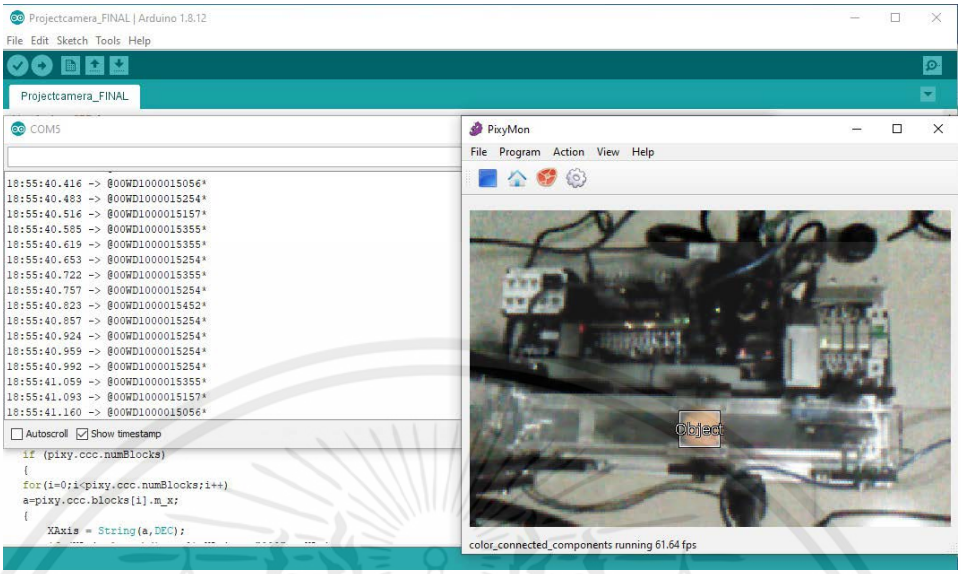
ส่วนที่ 3 ของโปรแกรมจะเกี่ยวข้องกับการกำหนดตั้งค่าสีที่ต้องการให้ Pixy ตรวจจับได้ (ตัวอย่าง ตรวจจับสีส้ม) โดย Pixy สามารถตรวจจับได้ 7 สี (แดง ส้ม เหลือง เขียว ฟ้า น้ำเงิน ม่วง) ถ้าต้องการตรวจจับเพียงสีเดียว ให้ทำการเคลียร์ค่าทั้งหมดก่อน ไปที่ action > clear all signatures หลังจากนั้น ไปที่ เมนู Action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

3.3.2 การใช้ Libraries Pixy บน Arduino

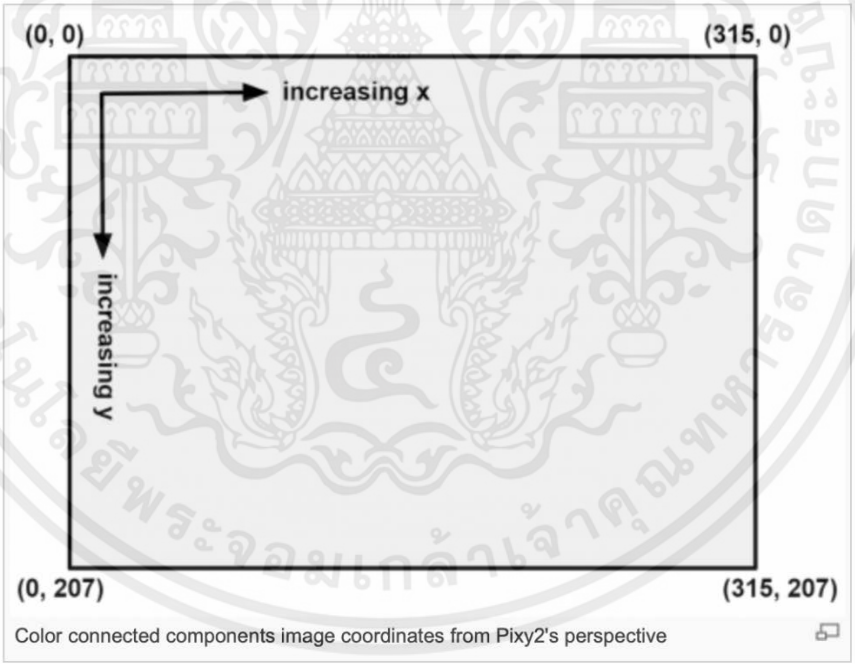
กล้อง pixy2 จะมี libraries ที่สามารถใช้งานร่วมกันระหว่างกล้องและตัวบอร์ด Arduino โดยเป็น free software ซึ่งเราสามารถเข้าไป download ได้ที่ https://pixycam.com/downloads-pixy2/ แตกไฟล์แล้วไปไว้ใน Folder ที่ติดตั้ง Arduino > Libraries. หลังจากนั้นเปิดโปรแกรม Arduino เลือก example > Pixy > hello_world แล้ว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

รูปที่ 3.6 โปรแกรม IDE

ในโปรเจคของเราได้นำตัวอย่างมาปรับแก้เพื่อตรวจจับค่าของตำแหน่งที่วัตถุทรงกลมของเราเคลื่อนที่ไปโดยค่าที่แสดงออกมาคือค่าที่กล้องสามารถตรวจสอบได้ใน 1 block (ความกว้างสูงสุดที่กล้องสามารถตรวจจับได้) 0 ถึง 315



รูปที่ 3.7 แสดงค่าที่ส่งให้ PLC ทาง Serial monitor คู่กับตรวจจับว่าวัตถุปัจจุบันนั้นอยู่ตำแหน่งไหน



รูปที่ 3.8 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;
```

กำหนดตัวแปรชื่อต่างๆ

โดยมี **String , byte** เป็นประเภทของตัวแปร

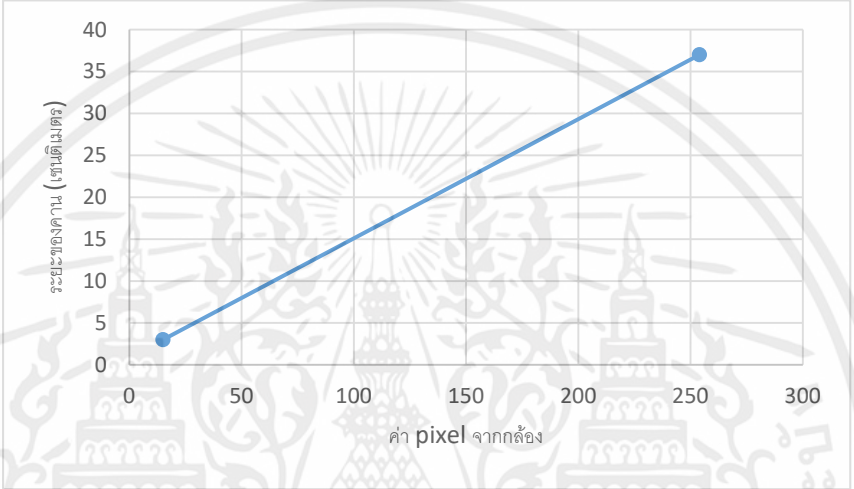
```
pixy.ccc.getBlocks(); //เรียกใช้ฟังชั่นให้กล้อง pixy ตรวจจับสีได้ทั้งหมดใน blocks
if (pixy.ccc.numBlocks) //ถ้าตรวจเจอให้แสดงค่าออกมา
{
  for (i=0;i<pixy.ccc.numBlocks;i++) //ใช้คำสั่งเพื่อให้ค่าที่ตรวจจับเพิ่มขึ้นหรือลดลงตาม
แนวแกน
  a=pixy.ccc.blocks[i].m_x; //กำหนดตัวแปร a เท่ากับค่าที่กล้องตรวจจับได้ในแกน 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;
    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

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

3.4 การสเกลค่าจาก vision camera ให้สอดคล้องกับระยะของคานโดยใช้ structure text ใน cx-programmer

เนื่องจากเรารู้ระยะที่แน่นอนของคานในหน่วยเซนติเมตร (3 – 37 เซนติเมตร) และค่าจากกล้องที่ส่งมาจาก Arduino ผ่านพอร์ตอนุกรมไปยัง PLC เป็นข้อมูลฐานสิบหกในหน่วยของ pixel (0015 - 00FE) ดังนั้นต้องทำการสเกลค่าจากกล้องให้เป็นเซนติเมตรโดยใช้สมการเชิงเส้น



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	REAL	No		0.0	
Outputs	Output_int	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;
Output := (Slope*(Real_Input-Real_Input_min))+Real_Output_min;
Output_int:=REAL_TO_DINT(Output);
```

รูปที่ 3.9 การสเกลค่าข้อมูลจากกล้องให้สอดคล้องกับระยะของคานโดยเขียน Structure Text ใน CX-Programmer

3.5 การควบคุมสมดุลของวัตถุด้วย PLC โดยการควบคุมแบบแมนนวล (Manual Mode) ใน CX-Programmer

เนื่องจากในโหมดการทำงานอัตโนมัตินั้นใช้ทฤษฎี Fuzzy Logic ในการควบคุม ซึ่งจำเป็นต้องเข้าใจผลตอบสนองของกระบวนการในแต่ละตำแหน่งเป็นอย่างมากในการกำหนด membership function และการกำหนด rules based จึงต้องศึกษาการเคลื่อนที่ของวัตถุทรงกลมโดยใช้ joystick ผ่านการควบคุมในโหมดแมนนวลเพื่อหา pule output ที่เหมาะสม ณ ตำแหน่งนั้นเพื่อนำไปควบคุม server 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 3.10 การใช้ joystick ในการกำหนด pulse โดยเขียน Structure Text ใน CX-Programmer

รูปที่ 3.11 นำค่าจาก joystick มาควบคุม servo motor โดยเขียน ladder ใน CX-Programmer

3.6 การควบคุมสมดุลของวัตถุด้วย PLC โดยการควบคุมแบบอัตโนมัติ (Auto Mode) ใน CX-Programmer

หลักการทำงานการควบคุมสมดุลของวัตถุด้วย PLC โดยการควบคุมแบบอัตโนมัติ (Auto Mode) สำหรับการควบคุมแบบอัตโนมัติ จะใช้กล้องเป็นตัวจับตำแหน่งของวัตถุเพื่อใช้ในการควบคุมแบบ FUZZY LOGIC โดยตำแหน่งวัตถุที่ได้จากกล้อง (PV) มาเปรียบเทียบกับค่า Setpoint (SV) ที่เรากำหนดจาก SCADA เป็นค่าเป้าหมายของการควบคุม เพื่อนำ Output มาขับเคลื่อน Servo Motor ให้ปรับระดับระนาบเพื่อให้วัตถุเข้าสู่สมดุล

3.6.1 Fuzzification

เขียนโปรแกรม Structure text เพื่อกำหนด membership function ให้กับ input

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	Y1	REAL	No		0.0	
Outputs	Y2	REAL	No		0.0	
Outputs	NXL	BOOL	No		FALSE	
Outputs	NL	BOOL	No		FALSE	
Outputs	NM	BOOL	No		FALSE	
Outputs	NS	BOOL	No		FALSE	
Outputs	Z	BOOL	No		FALSE	
Outputs	PS	BOOL	No		FALSE	
Outputs	PM	BOOL	No		FALSE	
Outputs	PL	BOOL	No		FALSE	
Outputs	PXL	BOOL	No		FALSE	
Internals	SVcm	DINT	No	D24		
Internals	SV_real	REAL	No		0.0	
Internals	PVcm	REAL	No	D5000		
Internals	error	DINT	No		0	
Internals	error_real	REAL	No		0.0	
Internals	a	INT[2]	No	D204		
Internals	y	REAL[2]	No	D200		

[Function block Name : Error]

```
SV_real:=DINT_TO_REAL(SVcm);
error_real:=SV_real-PVcm;
IF (error_real<=-21.0) THEN
        y[0]:=1.0; Y1:=y[0]; (* NXL *)
        a[0]:=0;
        NXL:=TRUE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (error_real>-21.0) AND (error_real<=-15.0) THEN
        y[0]:=(error_real/-6.0)-(15.0/6.0); Y1:=y[0]; (* NXL *)
        y[1]:=(error_real/6.0)+(21.0/6.0); Y2:=y[1]; (* NL *)
        a[0]:=0;
        a[1]:=1;
        NXL:=TRUE;
        NL:=TRUE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;

ELSIF (error_real>-15.0) AND (error_real<=-9.0) THEN
        y[0]:=(error_real/-6.0)-(9.0/6.0); Y1:=y[0]; (* NL *)
        y[1]:=(error_real/6.0)+(15.0/6.0); Y2:=y[1]; (* NM *)
        a[0]:=1;
        a[1]:=2;
        NXL:=FALSE;
        NL:=TRUE;
        NM:=TRUE;
        NS:=FALSE;
```

```
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (error_real>-9.0) AND (error_real<=-3.0) THEN
        y[0]:=(error_real/-6.0)-(3.0/6.0); Y1:=y[0]; (* NM *)
        y[1]:=(error_real/6.0)+(9.0/6.0); Y2:=y[1]; (* NS *)
        a[0]:=2;
        a[1]:=3;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=TRUE;
        NS:=TRUE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (error_real>-3.0) AND (error_real<=0.0) THEN
        y[0]:=(error_real/-3.0);Y1:=y[0]; (* NS *)
        y[1]:=(error_real/3.0)+1.0 ; Y2:=y[1]; (* Z *)
        a[0]:=3;
        a[1]:=4;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=TRUE;
        Z:=TRUE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
```

```
ELSIF (error_real>0.0) AND (error_real<=3.0) THEN
        y[0]:=(error_real/-3.0)+1.0;Y1:=y[0]; (* Z *)
        y[1]:=(error_real/3.0) ; Y2:=y[1]; (* PS *)
        a[0]:=4;
        a[1]:=5;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=TRUE;
        PS:=TRUE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (error_real>3.0) AND (error_real<=9.0) THEN
        y[0]:=(error_real/-6.0)+(9.0/6.0);Y1:=y[0]; (* PS *)
        y[1]:=(error_real/6.0)-(3.0/6.0) ; Y2:=y[1]; (* PM *)
        a[0]:=5;
        a[1]:=6;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=TRUE;
        PM:=TRUE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (error_real>9.0) AND (error_real<=15.0) THEN
        y[0]:=(error_real/-6.0)+(15.0/6.0);Y1:=y[0]; (* PM *)
        y[1]:=(error_real/6.0)-(9.0/6.0) ; Y2:=y[1]; (* PL *)
        a[0]:=6;
        a[1]:=7;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=TRUE;
        PL:=TRUE;
        PXL:=FALSE;
```

```
ELSIF (error_real>15.0) AND (error_real<=21.0) THEN
        y[0]:=(error_real/-6.0)+(21.0/6.0);Y1:=y[0]; (* PL *)
        y[1]:=(error_real/6.0)-(15.0/6.0); Y2:=y[1]; (* PXL *)
        a[0]:=7;
        a[1]:=8;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=TRUE;
        PXL:=TRUE;
ELSIF (error_real>21.0) THEN
        y[0]:=1.0;Y1:=y[0]; (* PXL *)
        a[0]:=8;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=TRUE;
END_IF;
(*0=NXL, 1=NL, 2=NM, 3=NS, 4=Z, 5=PS, 6=PM, 7=PL, 8=PXL *)
```

รูปที่ 3.12 การเขียน Structure text กำหนด membership function ของ input error

Variable Type	Name	Data Type	Retained	AT	Initial Value	Comment
Inputs	EN	BOOL	No		FALSE	Controls execution of the Function Block.
Inputs	SVcm	DINT	No		0	
Inputs	PVcm	REAL	No		0.0	
Outputs	ENO	BOOL	No		FALSE	Indicates successful execution of the Function Block.
Outputs	DError	REAL	No		0.0	
Internals	ERRORcm	DINT	No		0	
Internals	ERRORcm_Real	REAL	No		0.0	
Internals	Error1	REAL	No		0.0	
Internals	Error2	REAL	No		0.0	
Internals	Timer1	BOOL	No	1010.00		
Internals	SV_real	REAL	No		0.0	

[Function block Name : DE]

```
SV_real:=DINT_TO_REAL(SVcm);
ERRORcm_Real:=SV_real-PVcm;
IF (Timer1=FALSE) THEN
        Error1:=ERRORcm_Real;
ELSIF (Timer1=TRUE) THEN
        Error2:=ERRORcm_Real;
        DError:=(Error2-Error1)/0.1;
END_IF;
```

รูปที่ 3.13 Structure text หาอัตราการเปลี่ยนแปลงของ error ($\Delta \mathbf{E}$)

Variable Type	Name	Data Type	Retained	AT	Initial Value	Comment
Inputs	EN	BOOL	No		FALSE	Controls execution of the Function Block.
Inputs	Derror_Real	REAL	No		0.0	
Outputs	ENO	BOOL	No		FALSE	Indicates successful execution of the Function Block.
Outputs	Y1	REAL	No		0.0	
Outputs	Y2	REAL	No		0.0	
Outputs	NXL	BOOL	No		FALSE	
Outputs	NL	BOOL	No		FALSE	
Outputs	NM	BOOL	No		FALSE	
Outputs	NS	BOOL	No		FALSE	
Outputs	Z	BOOL	No		FALSE	
Outputs	PS	BOOL	No		FALSE	
Outputs	PM	BOOL	No		FALSE	
Outputs	PL	BOOL	No		FALSE	
Outputs	PXL	BOOL	No		FALSE	
Internals	y	REAL[2]	No	D212		
Internals	a	INT[2]	No	D216		

[Function block Name : DE2]

```
IF (Derror_Real<=-16.0) THEN
        y[0]:=1.0; Y1:=y[0]; (* NXL *)
        a[0]:=0;
        NXL:=TRUE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (Derror_Real>-16.0) AND (Derror_Real<=-12.0) THEN
        y[0]:=(Derror_Real/-4.0)-(3.0); Y1:=y[0]; (* NXL *)
        y[1]:=(Derror_Real/4.0)+(4.0); Y2:=y[1]; (* NL *)
        a[0]:=0;
        a[1]:=1;
        NXL:=TRUE;
        NL:=TRUE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
```

```
ELSIF (Derror_Real>-12.0) AND (Derror_Real<=-8.0) THEN
        y[0]:=(Derror_Real/-4.0)-(2.0); Y1:=y[0]; (* NL *)
        y[1]:=(Derror_Real/4.0)+(3.0); Y2:=y[1]; (* NM *)
        a[0]:=1;
        a[1]:=2;
        NXL:=FALSE;
        NL:=TRUE;
        NM:=TRUE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (Derror_Real>-8.0) AND (Derror_Real<=-4.0) THEN
        y[0]:=(Derror_Real/-4.0)-(1.0); Y1:=y[0]; (* NM *)
        y[1]:=(Derror_Real/4.0)+(2.0); Y2:=y[1]; (* NS *)
        a[0]:=2;
        a[1]:=3;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=TRUE;
        NS:=TRUE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (Derror_Real>-4.0) AND (Derror_Real<=-0.0) THEN
        y[0]:=(Derror_Real/-4.0);Y1:=y[0]; (* NS *)
        y[1]:=(Derror_Real/4.0)+1.0; Y2:=y[1]; (* Z *)
        a[0]:=3;
        a[1]:=4;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=TRUE;
        Z:=TRUE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
```

```
ELSIF (Derror_Real>0.0) AND (Derror_Real<=4.0) THEN
        y[0]:=(Derror_Real/-4.0)+1.0;Y1:=y[0]; (* Z *)
        y[1]:=(Derror_Real/4.0); Y2:=y[1]; (* PS *)
        a[0]:=4;
        a[1]:=5;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=TRUE;
        PS:=TRUE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (Derror_Real>4.0) AND (Derror_Real<=8.0) THEN
        y[0]:=(Derror_Real/-4.0)+2.0;Y1:=y[0]; (* PS *)
        y[1]:=(Derror_Real/4.0)-1.0; Y2:=y[1]; (* PM *)
        a[0]:=5;
        a[1]:=6;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=TRUE;
        PM:=TRUE;
        PL:=FALSE;
        PXL:=FALSE;
ELSIF (Derror_Real>8.0) AND (Derror_Real<=12.0) THEN
        y[0]:=(Derror_Real/-4.0)+(3.0);Y1:=y[0]; (* PM *)
        y[1]:=(Derror_Real/4.0)-(2.0); Y2:=y[1]; (* PL *)
        a[0]:=6;
        a[1]:=7;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=TRUE;
        PL:=TRUE;
        PXL:=FALSE;
```

```
ELSIF (Derror_Real>12.0) AND (Derror_Real<=16.0) THEN
        y[0]:=(Derror_Real/-4.0)+(4.0);Y1:=y[0]; (* PL *)
        y[1]:=(Derror_Real/4.0)-(3.0); Y2:=y[1]; (* PXL *)
        a[0]:=7;
        a[1]:=8;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=TRUE;
        PXL:=TRUE;
ELSIF (Derror_Real>16.0) THEN
        y[0]:=1.0;Y1:=y[0]; (* PXL *)
        a[0]:=8;
        NXL:=FALSE;
        NL:=FALSE;
        NM:=FALSE;
        NS:=FALSE;
        Z:=FALSE;
        PS:=FALSE;
        PM:=FALSE;
        PL:=FALSE;
        PXL:=TRUE;
END_IF;
(*0=NXL, 1=NL, 2=NM, 3=NS, 4=Z, 5=PS, 6=PM, 7=PL, 8=PXL *)
```

รูปที่ 3.14 การเขียน Structure text กำหนด membership function ของ input $\frac{\mathbf{dE}}{\mathbf{dt}}$

3.6.2 Rules Based

เขียนโปรแกรม Structure text เพื่อนำ error และ อัตราการเปลี่ยนแปลงของ error มาประมวลผลตามกฎพื้นฐาน (Rules Based)

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	LXL	BOOL	No		FALSE	
Outputs	LL	BOOL	No		FALSE	
Outputs	LM	BOOL	No		FALSE	
Outputs	LS	BOOL	No		FALSE	
Outputs	C	BOOL	No		FALSE	
Outputs	RS	BOOL	No		FALSE	
Outputs	RM	BOOL	No		FALSE	
Outputs	RL	BOOL	No		FALSE	
Outputs	RXL	BOOL	No		FALSE	
Internals	X	REAL[6]	No	D238		
Internals	Y	INT[2]	No	D204		
Internals	Z	INT[2]	No	D216		
Internals	A	INT	No	D1200		
Internals	B	INT	No	D1201		
Internals	i	INT	No		0	
Internals	j	INT	No		1	

[Function block Name : rules]

```
LXL:=FALSE; LL:=FALSE; LM:=FALSE; LS:=FALSE; C:=FALSE; RS:=FALSE; RM:=FALSE; RL:=FALSE; RXL:=FALSE;
FOR i:=0 TO 3 DO
    IF i=0 THEN A:=Y[0]; B:=Z[0];
    ELSIF i=1 THEN A:=Y[0]; B:=Z[1];
    ELSIF i=2 THEN A:=Y[1]; B:=Z[0];
    ELSIF i=3 THEN A:=Y[1]; B:=Z[1];
    END_IF;
        IF ((A=4) AND (B=8)) OR ((A=5) AND (B=8)) OR ((A=6) AND (B=7)) OR ( (A=6) AND (B=8)) OR ((A=7) AND (B=6)) OR ((A=7) AND (B=7)) OR ((A=7) AND (B=8)) OR ((A=8) AND
(B=4)) OR ((A=8) AND (B=5)) OR ((A=8) AND (B=6)) OR ((A=8) AND (B=7)) OR ((A=8) AND (B=8))
        THEN X[i]:=169.0; LXL:=TRUE;
        ELSIF ((A=3) AND (B=8)) OR ((A=4) AND (B=7)) OR ((A=5) AND (B=6)) OR ((A=5) AND (B=7)) OR ((A=6) AND (B=5)) OR ((A=6) AND (B=6)) OR ((A=7) AND (B=4)) OR ((A=7)
AND (B=5))
        THEN X[i]:=560.0; LL:=TRUE;
        ELSIF ((A=2) AND (B=8)) OR ((A=3) AND (B=7)) OR ((A=4) AND (B=6)) OR ((A=5) AND (B=5)) OR ((A=6) AND (B=4)) OR ((A=7) AND (B=3)) OR ((A=8) AND (B=2))
        THEN X[i]:=800.0; LM:=TRUE;
        ELSIF ((A=1) AND (B=8)) OR ((A=2) AND (B=7)) OR ((A=3) AND (B=6)) OR ((A=4) AND (B=5)) OR ((A=5) AND (B=4)) OR ((A=6) AND (B=3)) OR ((A=7) AND (B=2)) OR ((A=8)
```

```
            THEN X[i]:=560.0; LL:=TRUE;
            ELSIF ((A=2) AND (B=8)) OR ((A=3) AND (B=7)) OR ((A=4) AND (B=6)) OR ((A=5) AND (B=5)) OR ((A=6) AND (B=4)) OR ((A=7) AND (B=3)) OR ((A=8) AND (B=2))
            THEN X[i]:=800.0; LM:=TRUE;
            ELSIF ((A=1) AND (B=8)) OR ((A=2) AND (B=7)) OR ((A=3) AND (B=6)) OR ((A=4) AND (B=5)) OR ((A=5) AND (B=4)) OR ((A=6) AND (B=3)) OR ((A=7) AND (B=2)) OR ((A=8)
AND (B=1))
            THEN X[i]:=980.0; LS:=TRUE;
            ELSIF ((A=0) AND (B=8)) OR ((A=1) AND (B=7)) OR ((A=2) AND (B=6)) OR ((A=3) AND (B=5)) OR ((A=4) AND (B=4)) OR ((A=5) AND (B=3)) OR ((A=6) AND (B=2)) OR ((A=7)
AND (B=1)) OR ((A=8) AND (B=0))
            THEN X[i]:=1040.0; C:=TRUE;
            ELSIF ((A=0) AND (B=7)) OR ((A=1) AND (B=6)) OR ((A=2) AND (B=5)) OR ((A=3) AND (B=4)) OR ((A=4) AND (B=3)) OR ((A=5) AND (B=2)) OR ((A=6) AND (B=1)) OR ((A=7)
AND (B=0))
            THEN X[i]:=1300.0; RS:=TRUE;
            ELSIF ((A=0) AND (B=6)) OR ((A=1) AND (B=5)) OR ((A=2) AND (B=4)) OR ((A=3) AND (B=3)) OR ((A=4) AND (B=2)) OR ((A=5) AND (B=1)) OR ((A=6) AND (B=0))
            THEN X[i]:=1420.0; RM:=TRUE;
            ELSIF ((A=0) AND (B=5)) OR ((A=1) AND (B=3)) OR ((A=1) AND (B=4)) OR ((A=2) AND (B=2)) OR ((A=2) AND (B=3)) OR ((A=3) AND (B=1)) OR ((A=3) AND (B=2)) OR ((A=4)
AND (B=1)) OR ((A=5) AND (B=0))
            THEN X[i]:=1620.0; RL:=TRUE;
            ELSIF ((A=0) AND (B=0)) OR ((A=0) AND (B=1)) OR ((A=0) AND (B=2)) OR ((A=0) AND (B=3)) OR ((A=0) AND (B=4)) OR ((A=1) AND (B=0)) OR ((A=1) AND (B=1)) OR ((A=1)
AND (B=2)) OR ((A=2) AND (B=0)) OR ((A=2) AND (B=1)) OR ((A=3) AND (B=0)) OR ((A=4) AND (B=0))
            THEN X[i]:=1926.0; RXL:=TRUE;
            END_IF;
END_FOR;
```

รูปที่ 3.15 การเขียน Structure text กำหนด membership function ของ pulse output และ rule based

3.6.3 Defuzzification

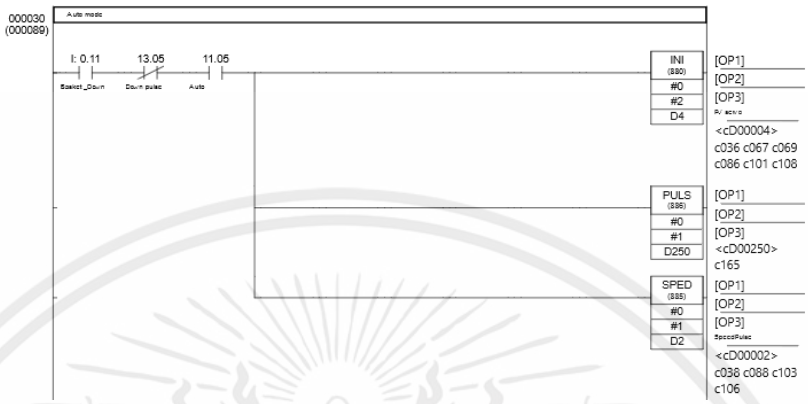
Variable Type	Name	Data Type	Retained	AT	Initial Value	Comment
Inputs	EN	BOOL	No		FALSE	Controls execution of the Function Block.
Outputs	ENO	BOOL	No		FALSE	Indicates successful execution of the Function Block.
Outputs	defuzzy	DWORD	No		1040	
Outputs	defuzzy_real	REAL	No		0.0	
Outputs	defuzzy_dint	DINT	No		0	
Internals	X	REAL[4]	No	D238		
Internals	Degree	REAL[4]	No	D222		

```
[Function block Name : Defuzzy]


defuzzy_real:=((Degree[0]*X[0])+(Degree[1]*X[1])+(Degree[2]*X[2])+(Degree[3]*X[3]))/(Degree[0]+Degree[1]+Degree[2]+Degree[3]);
defuzzy_dint:=REAL_TO_DINT(defuzzy_real);
defuzzy:=DINT_TO_DWORD(defuzzy_dint);
```

รูปที่ 3.16 การเขียน Structure text ดำเนินการในขั้นตอน Defuzzification

3.6.4 นำ pulse output จาก defuzzification มาเขียน Ladder ควบคุม Servo motor

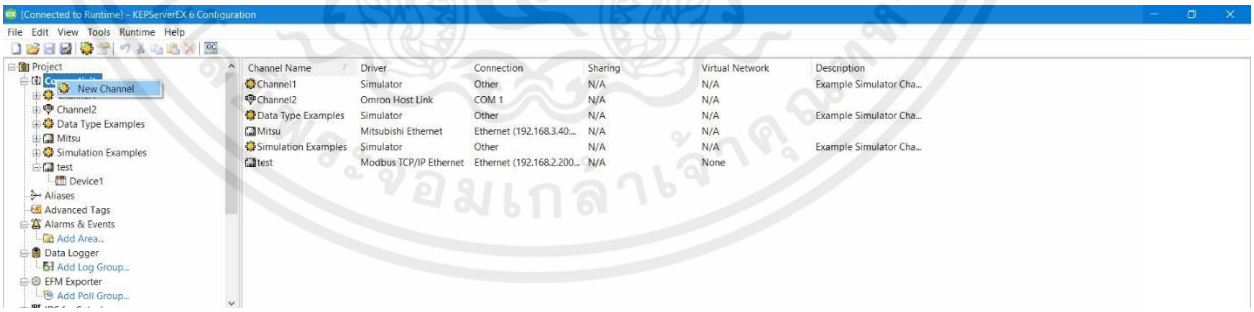


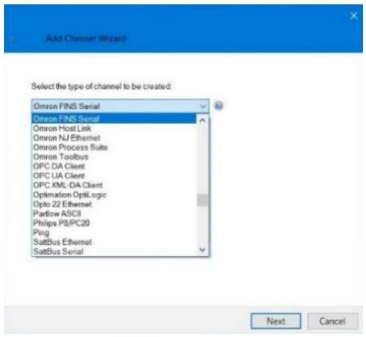
รูปที่ 3.17 Ladder Diagram ในการควบคุม Servo Motor ด้วย Pulse output จาก Defuzzification

3.7 การเชื่อมต่อข้อมูลระหว่าง PLC และ OPC server ด้วยซอฟต์แวร์ KEPServerEX 6 Configuration

3.7.1 การสร้าง channel เพื่อกำหนดประเภทการสื่อสาร

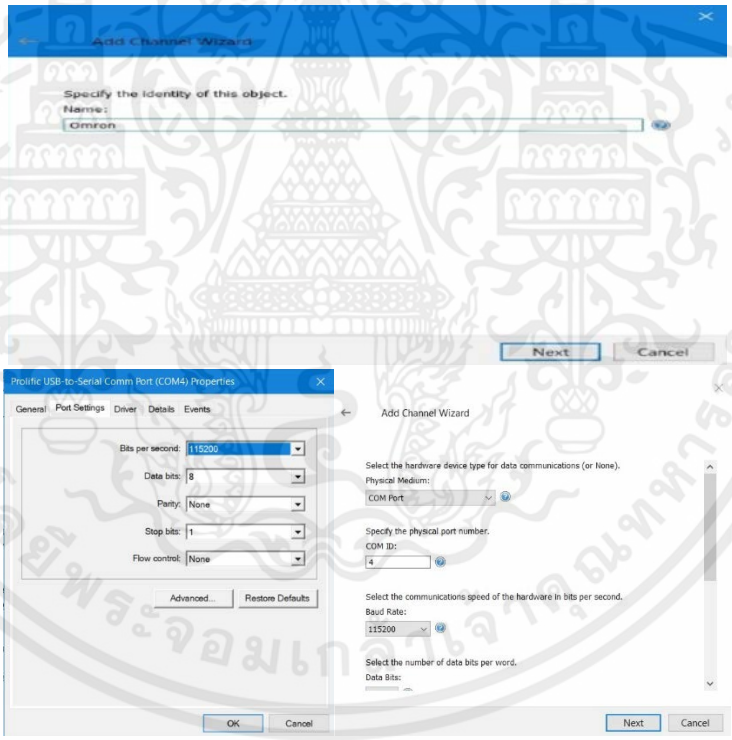
โดยคลิก New channel ที่ Connectivity (เลือก OMRON FINS Serial)





รูปที่ 3.18 เลือกการสื่อการให้สอดคล้องกับ PLC ที่ใช้

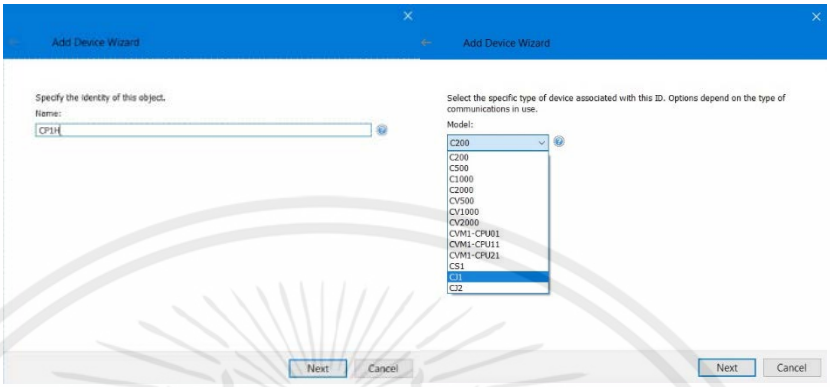
3.7.2 ตั้งชื่อ channel และกำหนด COM Port, Baud Rate, Data bit, Stop bit, Parity ให้ตรงกับ Port settings ที่เชื่อมต่อกับคอมพิวเตอร์



รูปที่ 3.19 ตั้งค่า Port ให้ตรงกับ Port settings ใน Device manager

3.7.3 ตั้งที่ Device และเลือก Device ให้ตรงกับรุ่น PLC ที่ใช้

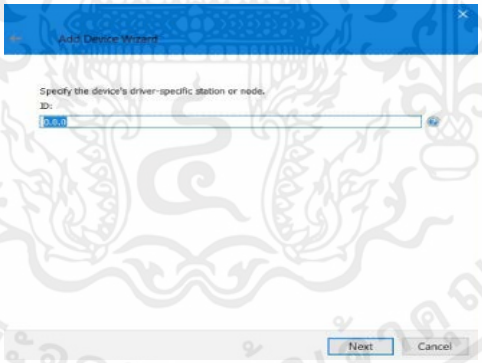
(ในที่นี้ใช้ PLC รุ่น CP1H ซึ่งสามารถใช้ CJ1 ได้)



รูปที่ 3.20 เลือกรุ่น PLC ให้ตรงกับที่ใช้

3.7.4 กำหนด Network ID หรือ Network Address

ซึ่งในที่นี้ใช้เป็น 0.0.0 หมายถึงต่อกับ PLC โดยตรง และหลังจากนี้ให้ตั้งค่าตามค่าเริ่มต้นของโปรแกรม



รูปที่ 3.21 กำหนด Network ID

3.7.5 เพิ่ม Tag ใน Device ที่สร้างมาจากขั้นตอนก่อนหน้า

โดยเลือก New tag และตั้งชื่อพร้อมระบุตำแหน่งของ Data memory, Data type, Scan rate จาก PLC

Tag Name	Address	Data Type	Scan Rate
auto	CIO0015.00	Boolean	100
board	CIO0011.03	Boolean	100
down pulse	CIO0013.05	Boolean	100
LOAD	CIO0001.06	Boolean	100
lower LS	CIO0000.11	Boolean	100
max	CIO0012.01	Boolean	100
min	CIO0012.00	Boolean	100
PV	D00044	DWord	100
sequence	CIO0013.06	Boolean	100
SERVO PV	D00030	DWord	100
SET SERVO	CIO0013.00	Boolean	100
stop	CIO0014.03	Boolean	100
SV	D00024	DWord	100

รูปที่ 3.22 Tag ที่ดึงมาจาก PLC ทั้งหมด

3.8 ออกแบบกราฟิกเพื่อการแสดงผลใน GraphWorX32

3.8.1 ออกแบบหน้าเริ่มต้นเพื่อใช้เลือกโหมดการควบคุมระหว่าง Manual และ Auto

โดยนำ OPC tag ชื่อ “auto” ที่ตำแหน่ง CIO15.00 มาเชื่อมกับปุ่มคำสั่ง

รูปที่ 3.23 การแสดงผลในหน้าเริ่มต้น

3.8.2 ออกแบบหน้าโหมดการควบคุมแบบแมนนวล

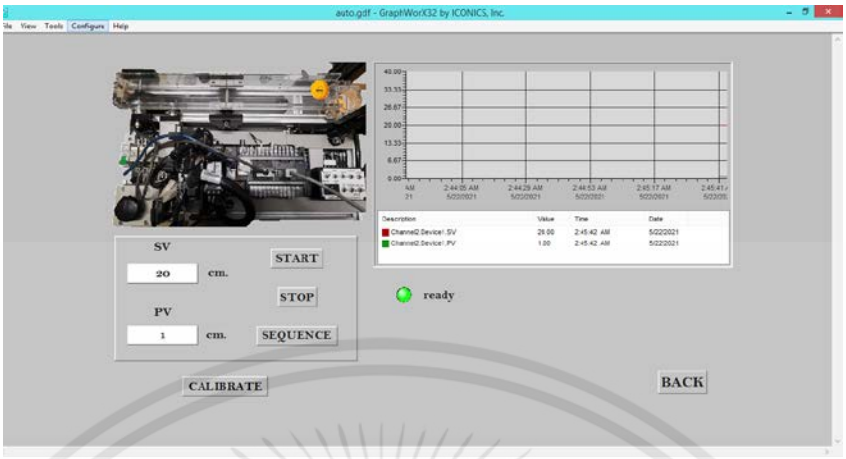
โดยนำ OPC tag ชื่อ “LOAD” ที่ตำแหน่ง CIO1.06 มาใช้สั่งอัพโหลดวัตถุทรงกลมเข้าสู่คาน ส่วน OPC tag ชื่อ “SET SERVO” ที่ตำแหน่ง CIO13.00 มาใช้ในการปรับระนาบของคานให้เอียงทำมุม 0 องศาเพื่อใช้เป็นตำแหน่งเริ่มต้น ส่วน OPC tag ชื่อ “SERVO PV” ที่ตำแหน่ง D30 มาใช้แสดง pulse

ของ Servo motor ในขณะที่ปรับ Joystick และแสดงการกระดกขึ้นลงของคานผ่านตัวโมเดล ส่วน OPC tag ชื่อ “board และ lower LS” ที่ตำแหน่ง CIO11.03 และ CIO0.11 ตามลำดับ คือ limit switch ใช้ในการแสดงสถานะพร้อมทำงาน

รูปที่ 3.24 การแสดงผลในหน้า Manual

3.8.3 ออกแบบหน้าโหมดการควบคุมแบบอัตโนมัติ

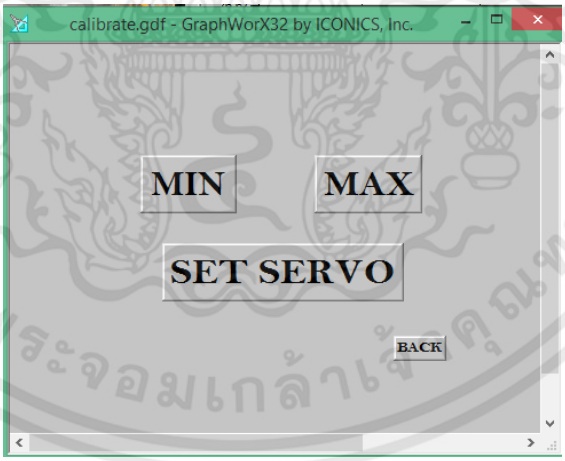
โดยนำ OPC tag ชื่อ “LOAD” ที่ตำแหน่ง CIO1.06 มาใช้สั่งอัพโหลดวัตถุทรงกลมเข้าสู่คาน ส่วน OPC tag ชื่อ “stop” ที่ตำแหน่ง CIO14.03 มาใช้สั่งวัตถุทรงกลมลงจากคาน ส่วน OPC tag ชื่อ “sequence” ที่ตำแหน่ง CIO13.06 มาใช้สั่งให้วัตถุทรงกลมเคลื่อนที่ไปที่ตำแหน่ง 10, 20, 30 ซ.ม. ตามลำดับ ส่วน OPC tag ชื่อ “SV” ที่ตำแหน่ง D24 ใช้ในการป้อนค่าตำแหน่งที่ต้องการให้วัตถุทรงกลมเคลื่อนที่ไปอยู่ ณ ตำแหน่งนั้น ส่วน OPC tag ชื่อ “PV” ที่ตำแหน่ง D44 ใช้แสดงตำแหน่งของวัตถุทรงกลมในรูปแบบของตัวเลขและกราฟ



รูปที่ 3.25 การแสดงผลในหน้า Auto

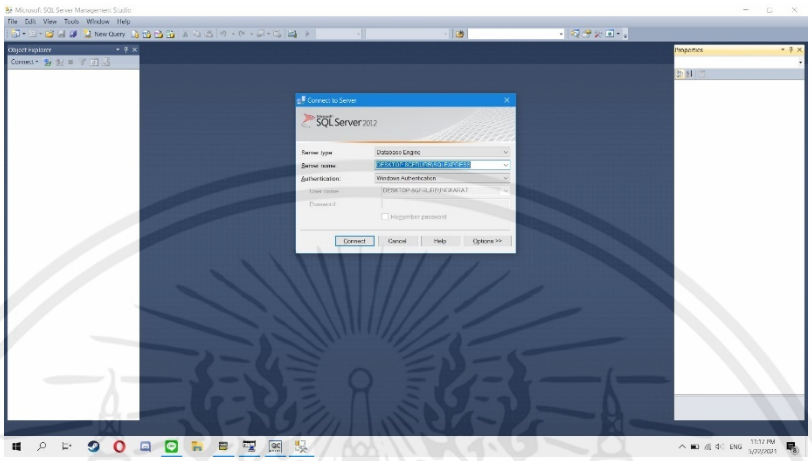
3.8.4 ออกแบบหน้า Calibration

โดยนำ OPC tag ชื่อ “max” ที่ตำแหน่ง CIO12.01 มาใช้เก็บตำแหน่งของวัตถุทรงกลมที่จุดปลายของคาน ส่วน OPC tag ชื่อ “min” ที่ตำแหน่ง CIO12.00 มาใช้เก็บตำแหน่งของวัตถุทรงกลมที่จุดเริ่มต้นของคาน เพื่อนำค่า max และ min ไปคำนวณในรูปที่ 3.7

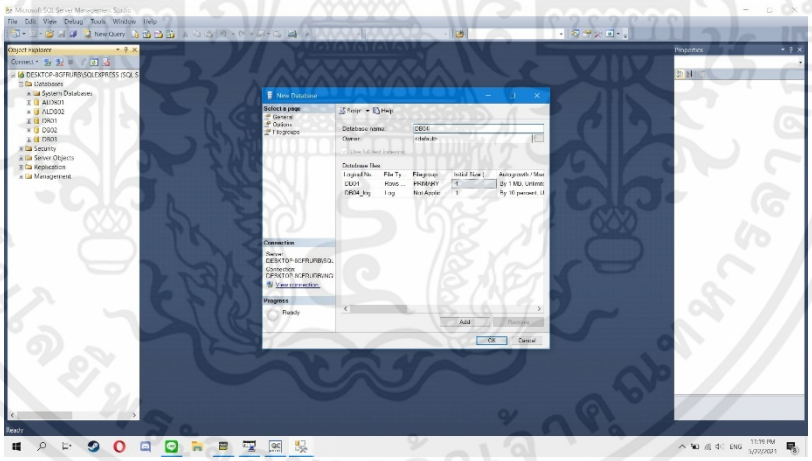


รูปที่ 3.26 การแสดงผลในหน้า calibration

3.9 การสร้าง Database ใน MS SQL Server



รูปที่ 3.27 เปิดโปรแกรม SQL Server Management Studio และเลือก Sever แล้วกด Connect

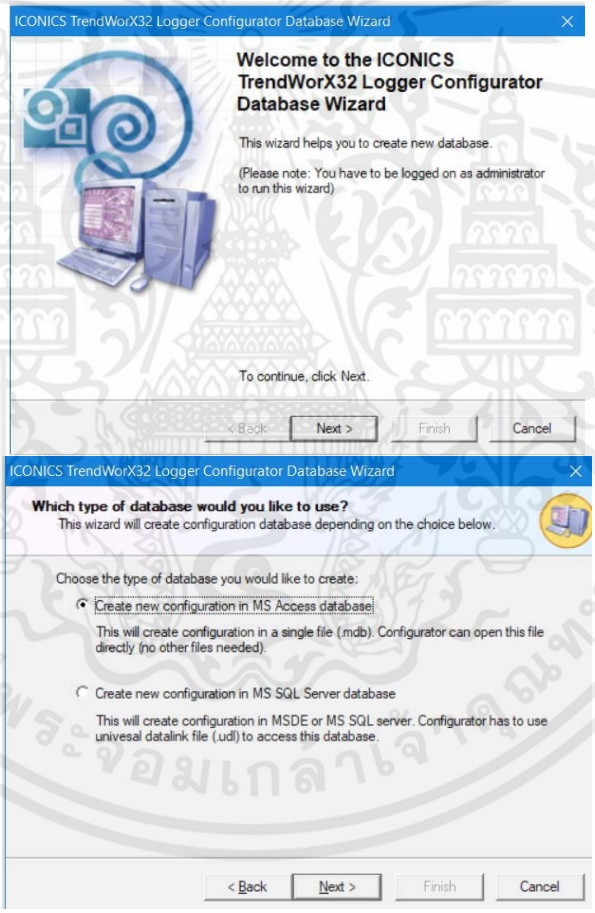


รูปที่ 3.28 สร้าง New database แล้วตั้งชื่อ Database name

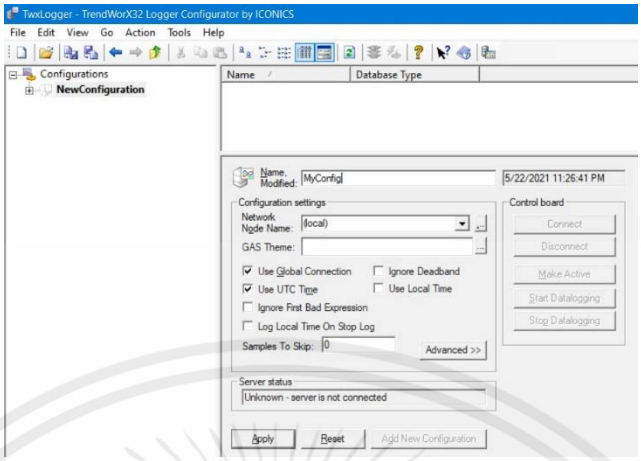
3.10 การเก็บข้อมูลแบบ History ด้วยซอฟต์แวร์ TrendWorX32 Configurator

สร้างไฟล์ Configuration ที่กำหนดว่าจะเก็บข้อมูลจาก OPC tag อะไร เก็บด้วย Collection rate และ Calculation period เท่าไร โดยใช้โปรแกรม TrendWorX32 Configurator และเก็บค่าเหล่านี้ไว้ที่ฐานข้อมูลใน MS SQL server

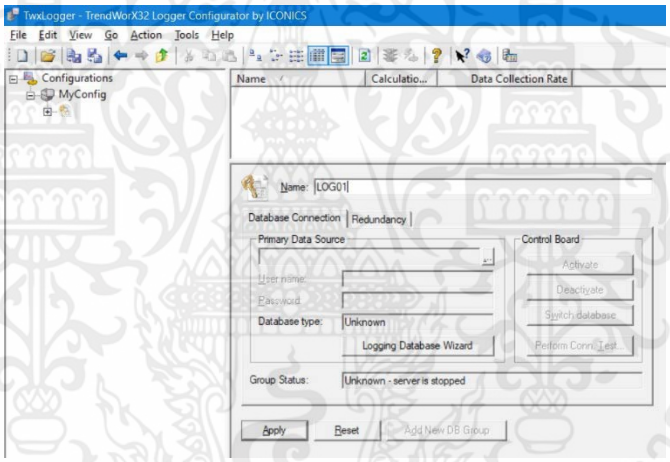
3.10.1 การสร้างคอนฟิกเก็บข้อมูลใน TrendWorX32 Configurator



รูปที่ 3.29 กด New File เพื่อสร้างไฟล์คอนฟิกแบบ MS Access

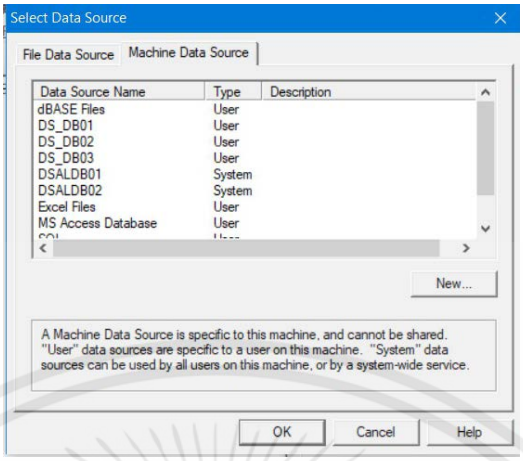


รูปที่ 3.30 คลิก New Configuration แล้วตั้งชื่อในช่อง Name, Modified และคลิก Apply

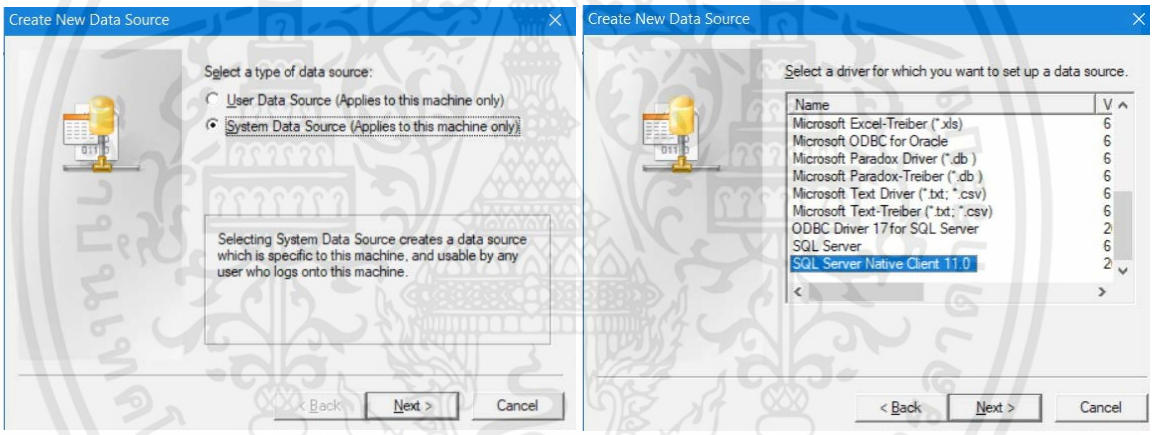


รูปที่ 3.31 สร้าง New Data Group แล้วคลิกช่อง Primary Data Source เพื่อเลือก Data Source Name

ในกรณีนี้เราจะสร้าง Data Source Name (DSN) ขึ้นมาใหม่เพื่อชี้ไปยังฐานข้อมูลที่จะเก็บข้อมูล ดังนั้นเลือกแท็ป Machine Data Source แล้วคลิก New

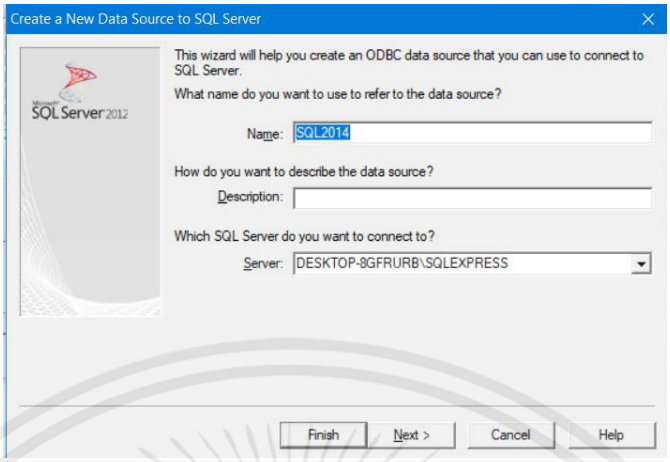


รูปที่ 3.32 หน้าต่าง Select Data Source



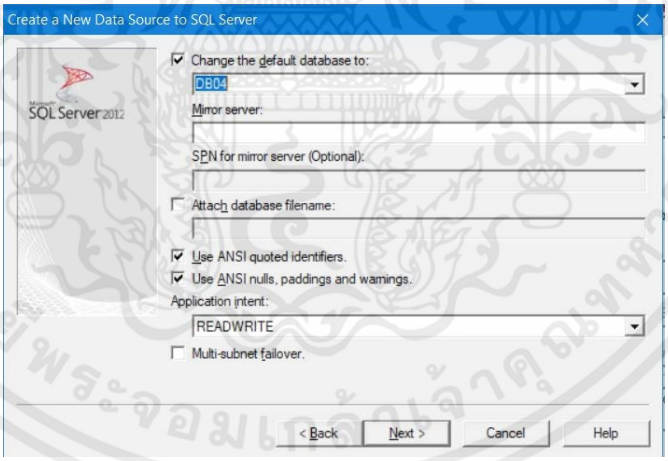
รูปที่ 3.33 เลือก System Data Source แล้วเลือก SQL Server Native Client เนื่องจากจะเก็บข้อมูลไว้ใน MS SQL Server

ในส่วนต่อไปจะเจอกับหน้าต่าง Create a New Data Source to SQL Server ให้ตั้งชื่อในช่อง Name และกำหนดว่าจะติดต่อกับ SQL Server ชื่ออะไร โดยสามารถเปิดดูใน Microsoft SQL Management Studio จะพบชื่อ Server ดังรูป 3.24 ในที่นี้ชื่อว่า DESKTOP-8GFRURB\SQLEXPRESS

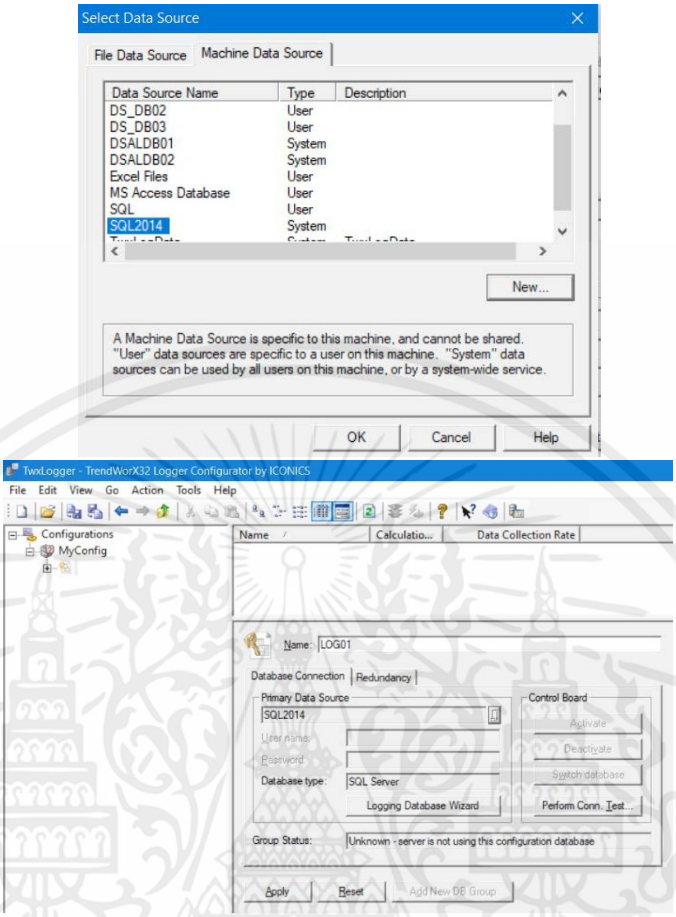


รูปที่ 3.34 การกำหนดว่าจะติดต่อกับ SQL Server ชื่ออะไร

ต่อไปจะเป็นการระบุว่าจะเก็บข้อมูลไว้ใน Database ชื่ออะไร (ต้องสร้าง Database ไว้ใน MS SQL Server ก่อน ในรูปที่ 3.25)

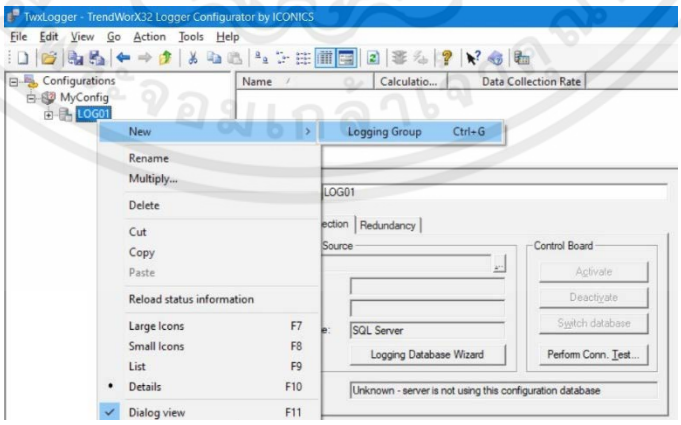


รูปที่ 3.35 เลือกชื่อ Database ที่จะเก็บข้อมูล



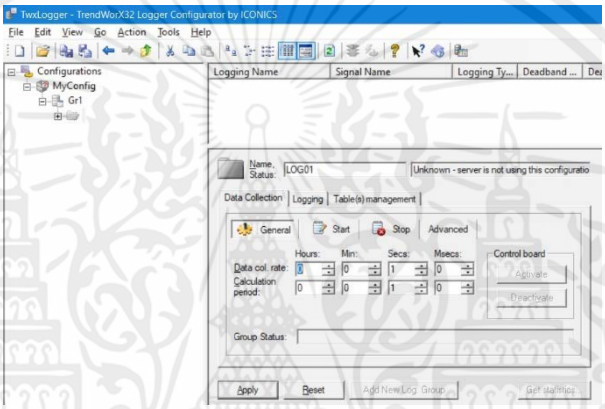
รูปที่ 3.36 โปรแกรมจะแสดง Data Source ที่สร้างใหม่ จากนั้นให้เลือกและคลิก OK แล้วคลิก Apply

3.10.2 การระบุสัญญาณที่จะเก็บ



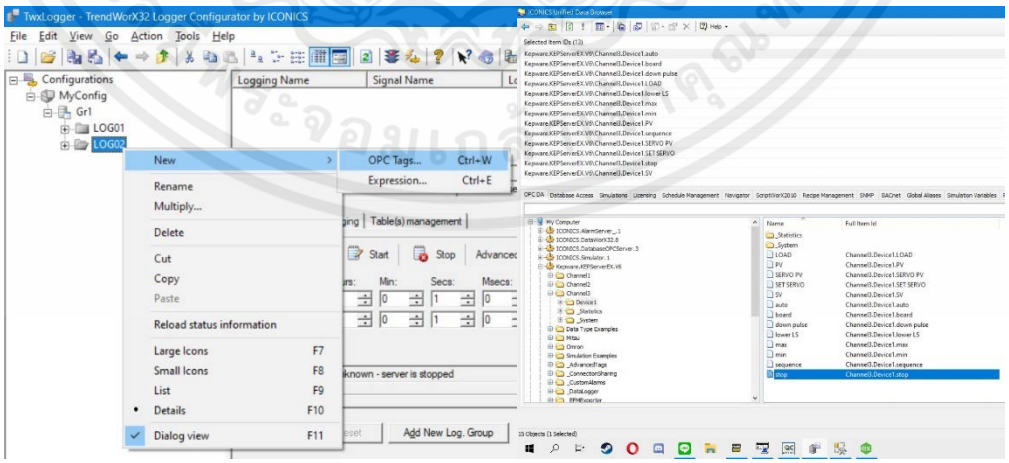
รูปที่ 3.37 สร้างกลุ่มของสัญญาณ (Logging Group)

ต่อไปจะเป็นการกำหนด Collection Rate และ Calculation Period โดย Collection Rate คือความถี่ในการเก็บตัวอย่างสัญญาณ, Calculation Period คือ ช่วงเวลาในการคำนวณ เช่น คำนวณค่า Average, Max, Min, Std, ฯลฯ โดย Calculation Period จะน้อยกว่า Collection Rate ไม่ได้ ยกตัวอย่างเช่น ถ้าต้องการเก็บข้อมูลทุกๆ 1 วินาที แล้วคำนวณค่า Average ของสัญญาณทุกๆ 1 นาที ก็ต้องใช้ Collection Rate เท่ากับ 1 วินาที และ Calculation Period เท่ากับ 1 นาที ค่าสัญญาณที่เก็บมาทุก 1 วินาทีเมื่อครบ 1 นาทีก็จะได้สัญญาณ 60 ตัวอย่างมาคำนวณค่า Average แล้วเก็บลงใน Database

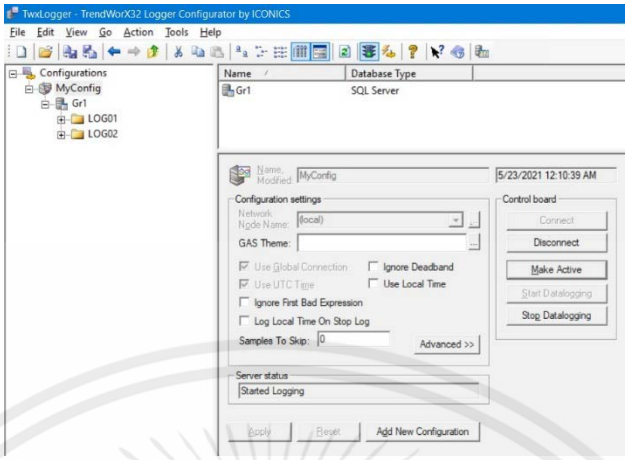


รูปที่ 3.38 กำหนด Data Collection Rate และ Calculation Period

ต่อไปเป็นการเพิ่ม OPC Tag ที่สร้างไว้ในรูปที่ 3.20 ลงใน Logging Group ที่สร้างไว้ในรูปที่ 3.33 โดยคลิกที่ New OPC Tags



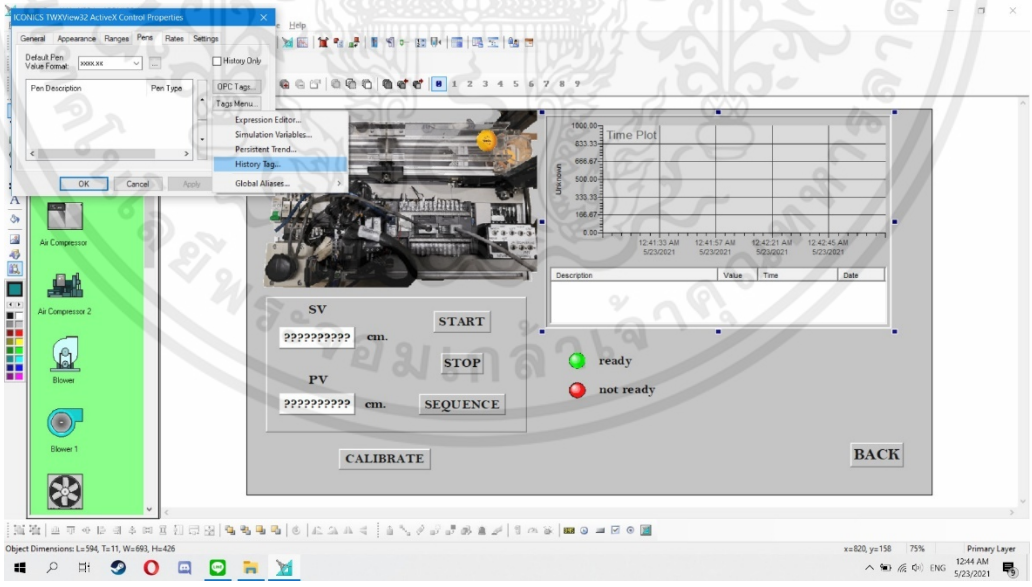
รูปที่ 3.39 เพิ่ม OPC Tag ที่สร้างไว้ใน KEPServerEX ลงใน Logging Group



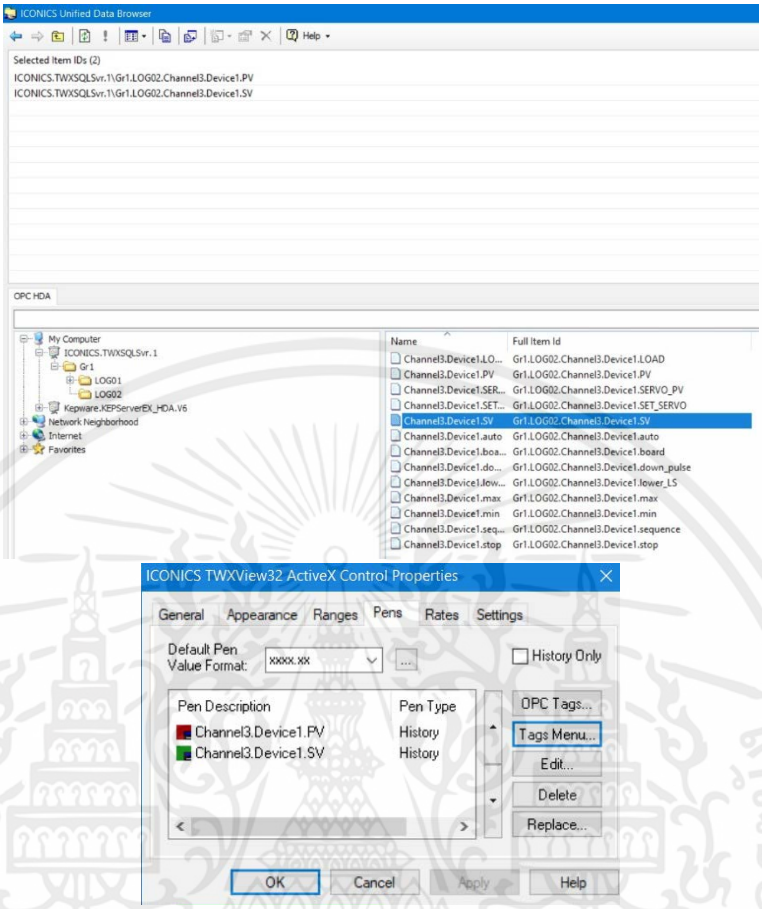
รูปที่ 3.40 คลิก Make Active เพื่ออัพเดทการเปลี่ยนแปลง และเริ่มการทำงาน

3.11 การแสดงผลข้อมูลที่เก็บไว้ด้วย TrendWorX32 Viewer

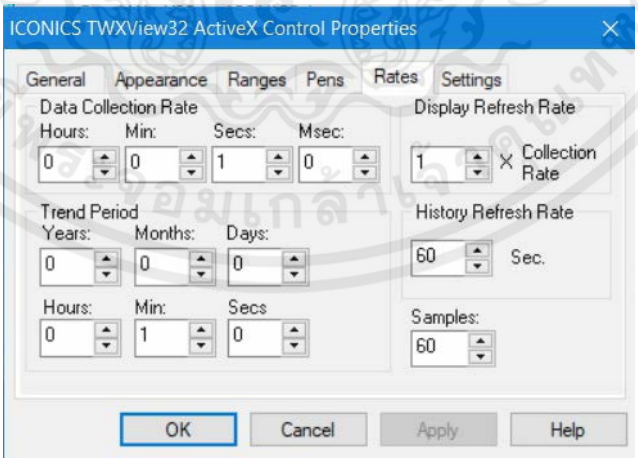
เมื่อเก็บข้อมูลไว้ใน MS SQL server แล้ว การที่จะนำข้อมูลออกมาแสดงแบบ History จะใช้เครื่องมือที่มีชื่อว่า TrendWorx32 Viewer ซึ่งอยู่ใน GraphWorX32 เพื่อนำค่ามาแสดงเป็นกราฟ



รูปที่ 3.41 เลือกข้อมูลที่แสดงแบบ History Tag



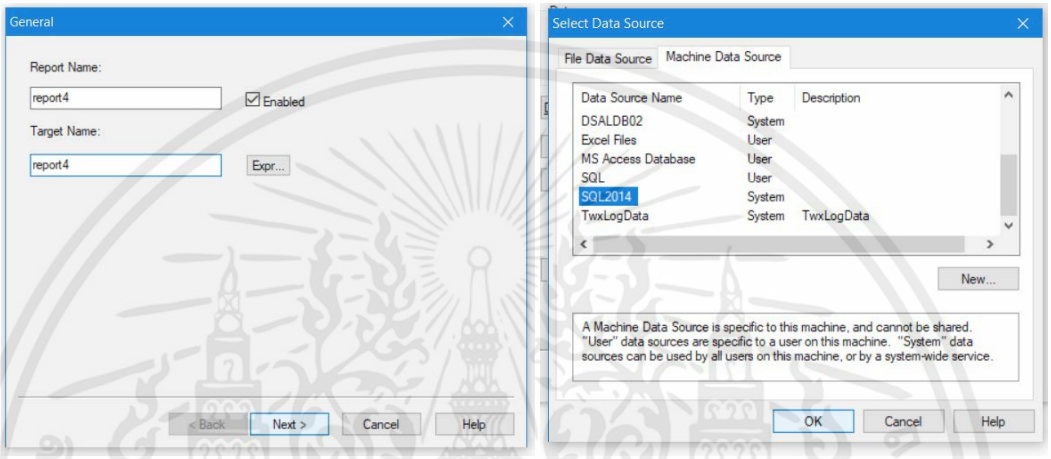
รูปที่ 3.42 เลือก Tag ที่อยู่ใน Logging Group ที่ต้องการ



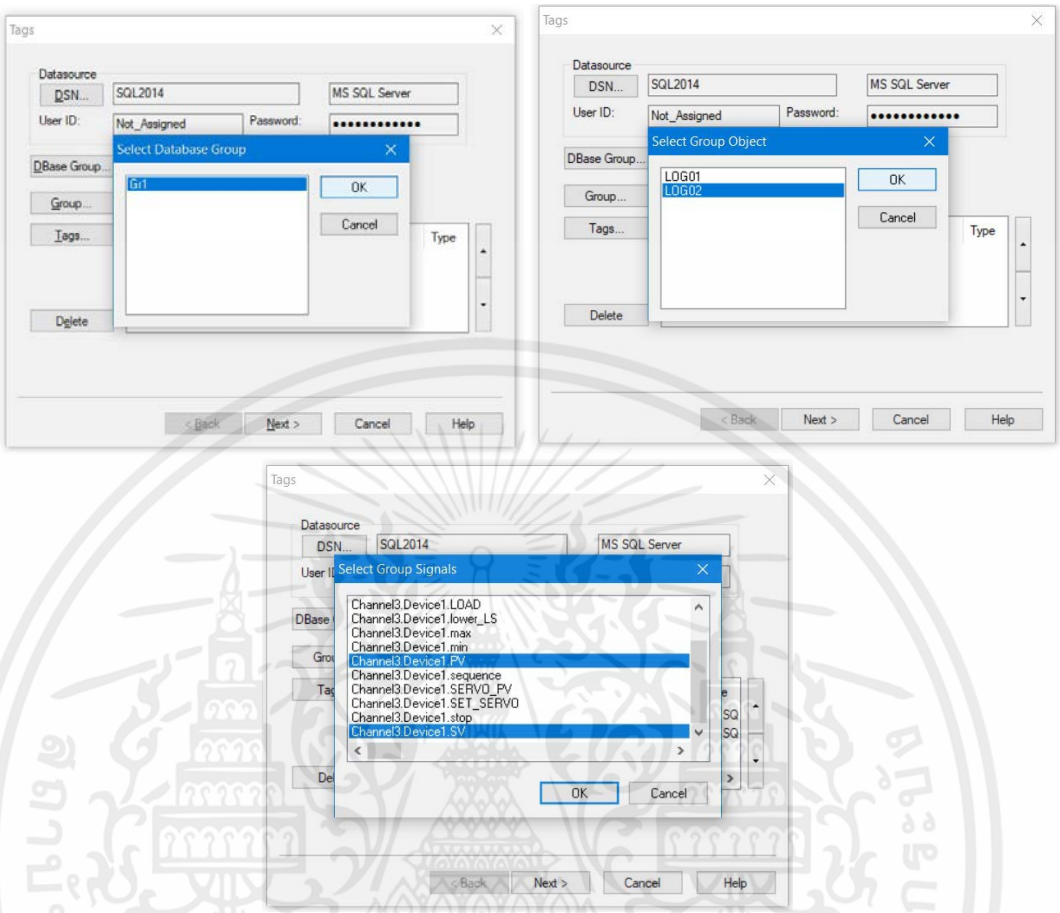
รูปที่ 3.43 เลือก Collection Rate, Period, History Refresh Rate ให้เหมาะกับคอนฟิกใน TrendWorX32 Configurator

3.12 การสร้างรายงานจากข้อมูลที่เก็บไว้ใน Database ด้วย TrendWorX32 Reporting

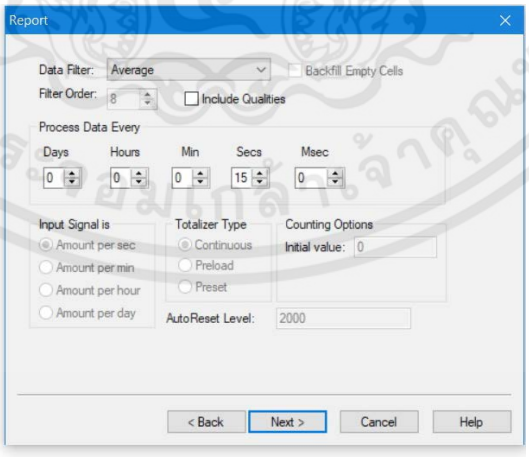
สร้างรายงานผ่านโปรแกรม TrendWorX32 Reporting โดยใช้ Data source name ตัวเดียวกับที่ TrendWorX32 Configurator ใช้เก็บข้อมูลในรูปที่ 3.32



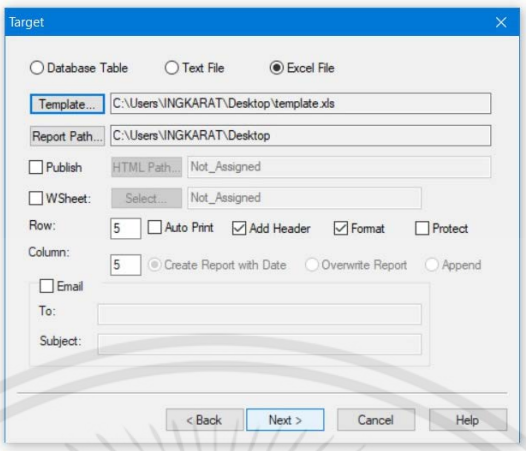
รูปที่ 3.44 ตั้งชื่อรายงาน และเลือก DSN เดียวกับที่ TrendWorX32 Configuratorในรูปที่ 3.32



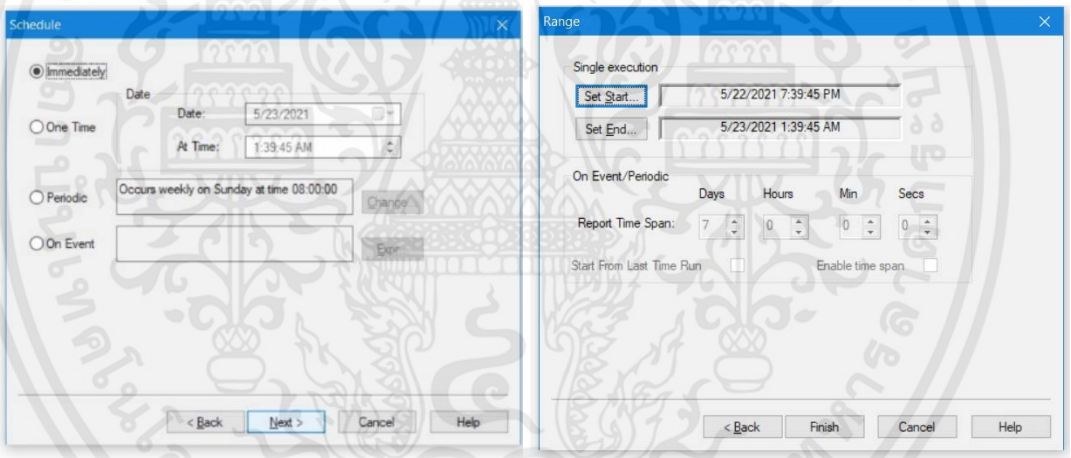
รูปที่ 3.45 เลือก Database Group, Logging Group, Tags ที่สร้างไว้ในขั้นตอนที่ 3.10



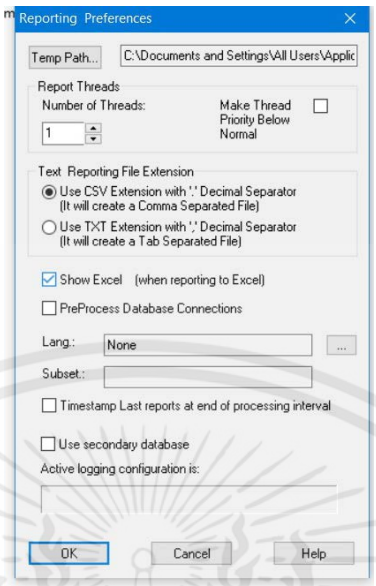
รูปที่ 3.46 เลือก Data Filter เป็น Average ในแต่ละ 15วินาที



รูปที่ 3.47 เลือกรูปแบบรายงานเป็น MS Excel, เลือกไฟล์ Template และโฟลเดอร์เก็บไฟล์รายงาน



รูปที่ 3.48 เลือกช่วงเวลาการสร้างรายงาน



รูปที่ 3.49 เลือกให้แสดงรายงานอัตโนมัติเมื่อสร้างรายงานเสร็จ

บทที่ 4

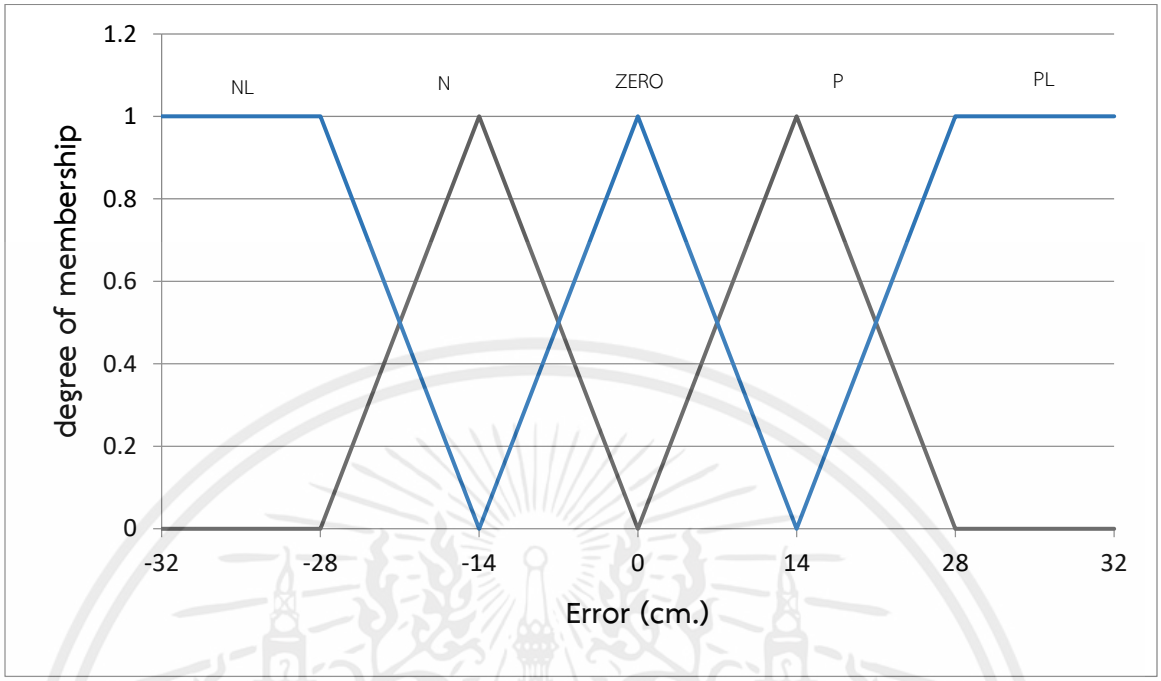
ผลการทดลอง

4.1 กล่าวนำ

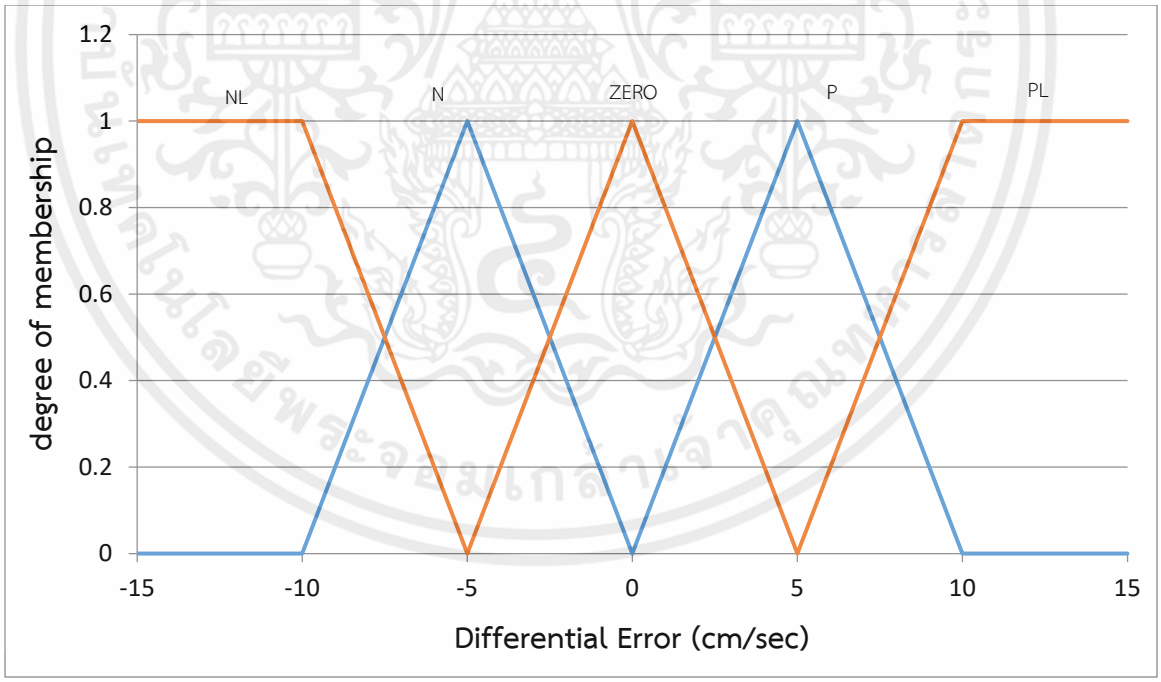
จากการศึกษาทฤษฎีการควบคุมฟัซซีลอจิกร่วมกับการควบคุมลูกบอลด้วย joystick ใน Manual Mode ทำให้ผู้จัดทำสามารถออกแบบ membership function ของตัวแปรที่ใช้ในการควบคุม และ rule based เพื่อที่จะหาค่า Defuzzification ซึ่งเป็นค่าที่ป้อนให้กับ servo motor เพื่อควบคุมลูกบอลในตำแหน่งต่าง ๆ ตัวแปรที่ใช้ในการควบคุมหรือตัวแปรอินพุตคือ Error และ Differential Error ตัวแปรควบคุมหรือตัวแปรเอาท์พุตคือ Pulse Output โดยการออกแบบฟัซซีลอจิกมีดังต่อไปนี้

4.2 การกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบต่างๆ

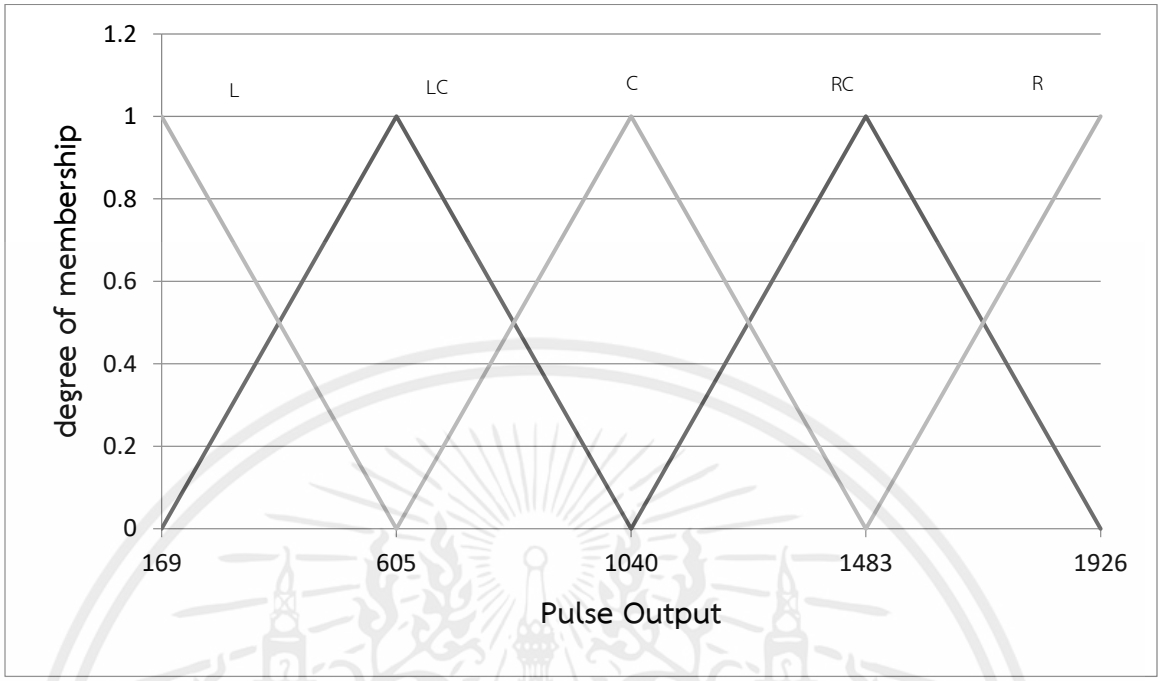
4.2.1 การกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 1



รูปที่ 4.1 membership function ของ Error แบบที่ 1



รูปที่ 4.2 membership function ของ Differential Error แบบที่ 1

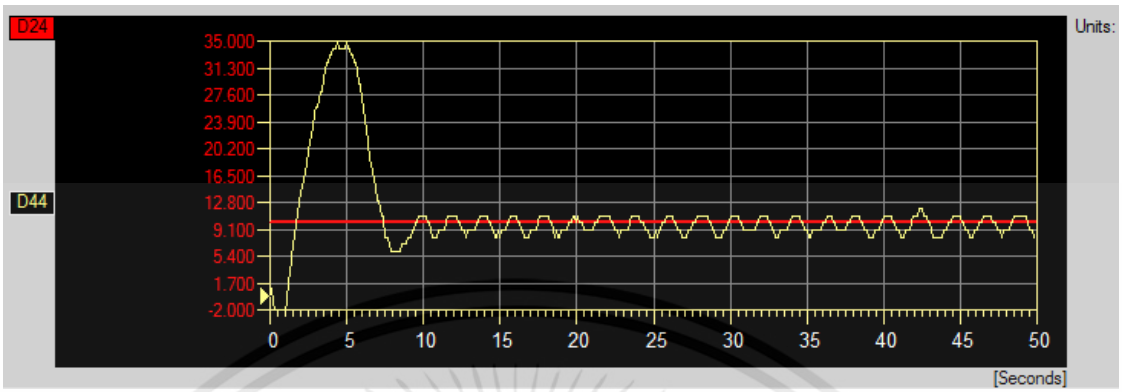


รูปที่ 4.3 membership function ของ Pulse Output แบบที่ 1

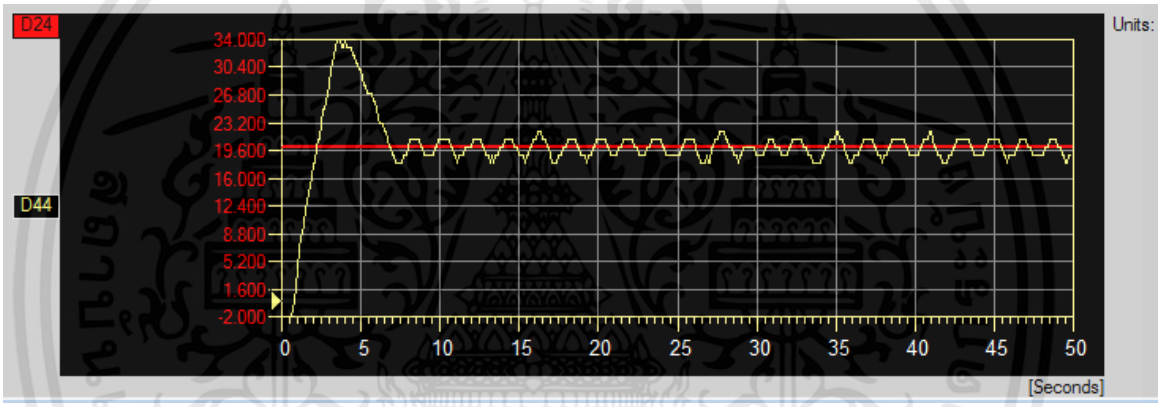
ตารางที่ 4.1 Rule Based แบบที่ 1

Differential Error / Error	NL	N	Z	P	PL
NL	R	RC	RC	RC	C
N	R	RC	RC	RC	C
Z	R	RC	C	LC	L
P	C	LC	LC	LC	L
PL	C	LC	LC	LC	L

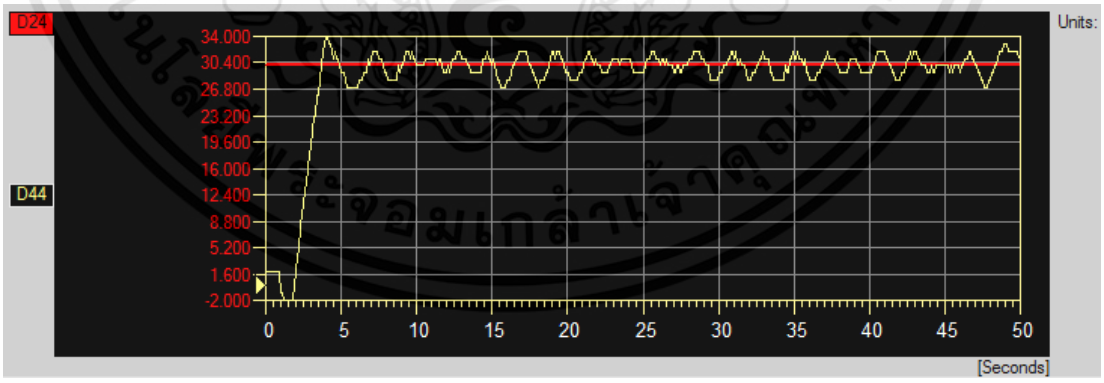
ผลการตอบสนอง



รูปที่ 4.4 ผลตอบสนองที่ sv=10 ตามเงื่อนไขที่ 1



รูปที่ 4.5 ผลตอบสนองที่ sv=20 ตามเงื่อนไขที่ 1

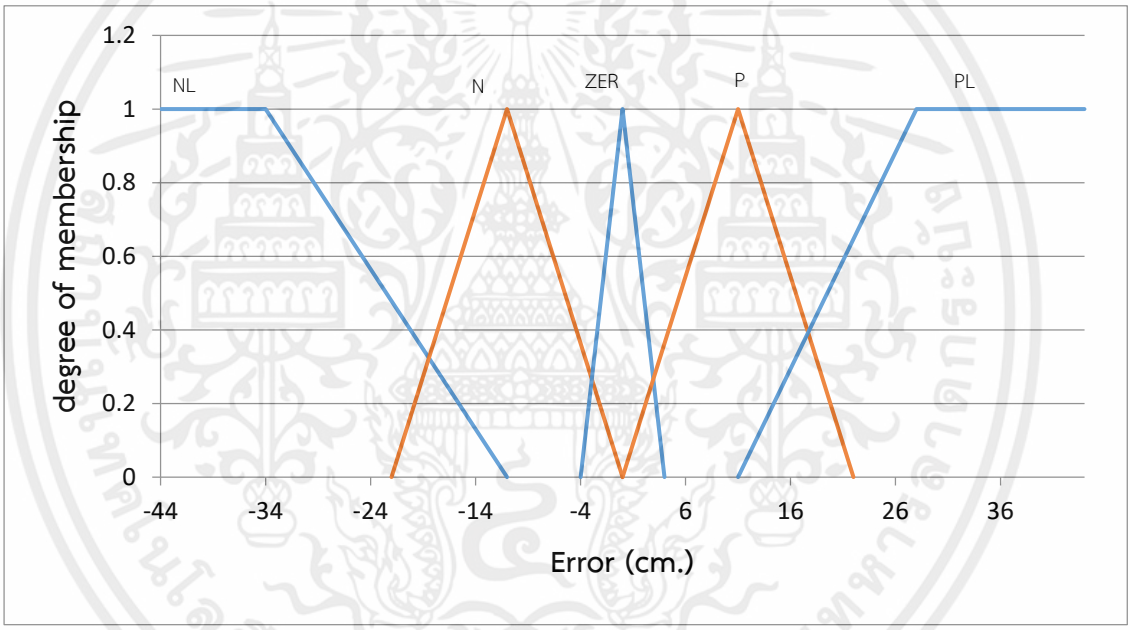


รูปที่ 4.6 ผลตอบสนองที่ sv=30 ตามเงื่อนไขที่ 1

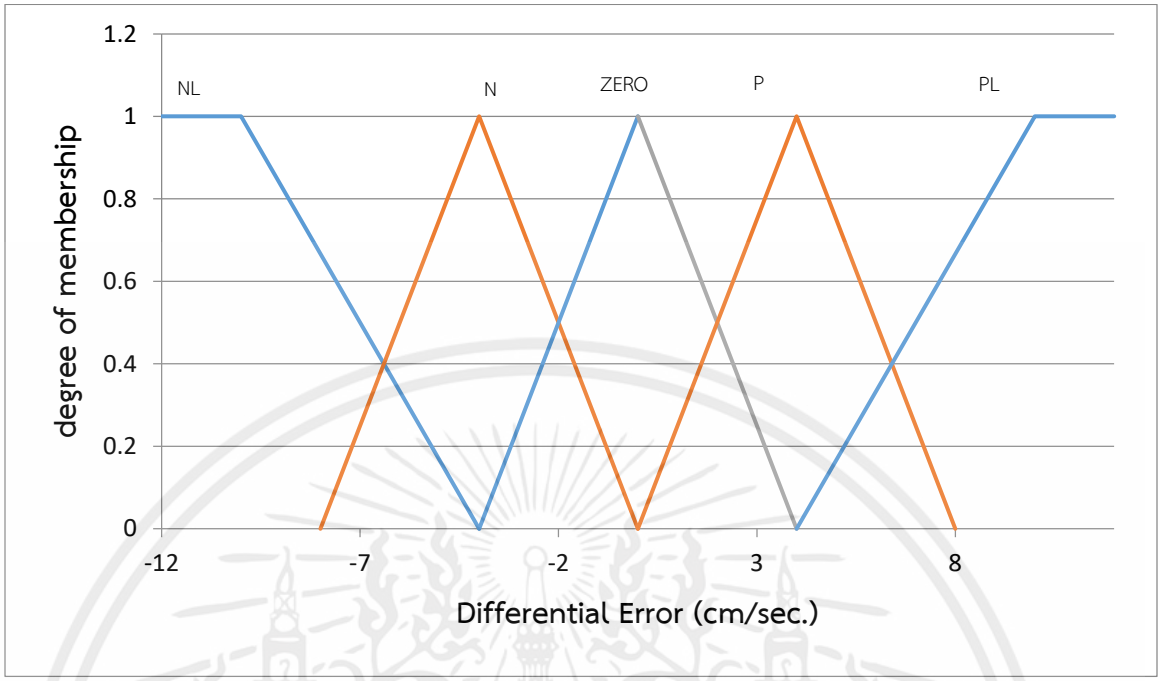
<u>สรุปผลการทดลอง</u>

จากการทดลองกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 1 จะเห็นว่าผลตอบสนองไม่สามารถเข้าสู่ Setpoint ทั้งที่ตำแหน่ง 10, 20 และ 30 เซนติเมตร และ พบว่าผลตอบสนอง Oscillate จาก Setpoint ช่วง ± 3 cm

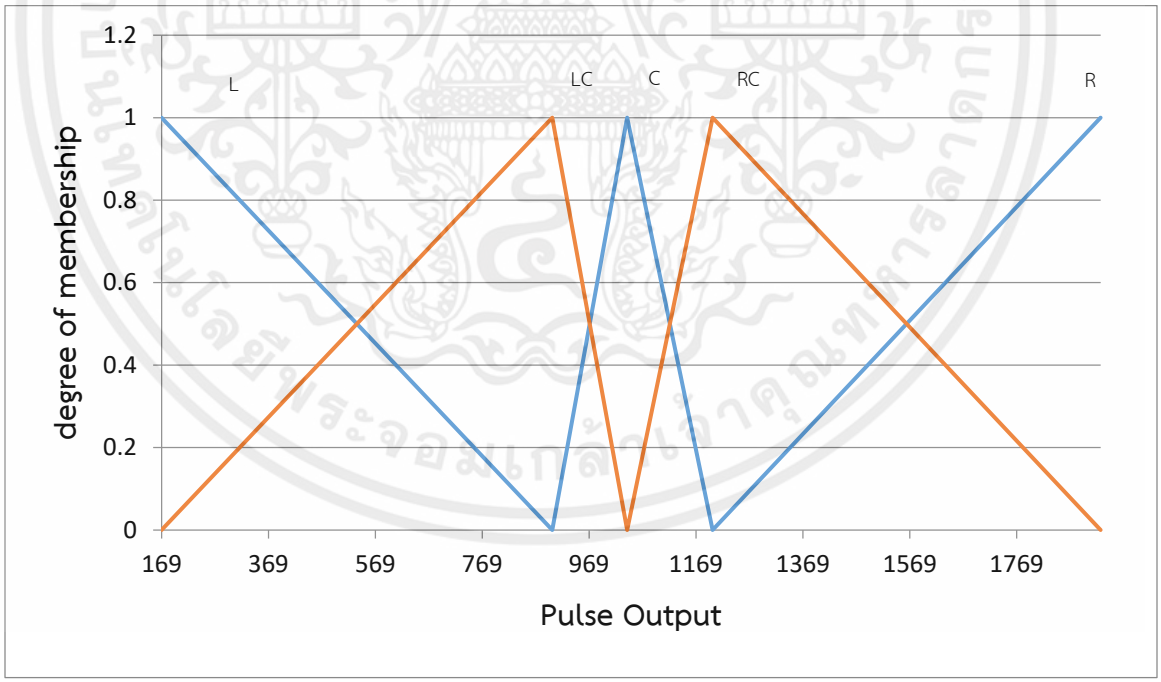
4.2.2 การกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 2



รูปที่ 4.7 membership function ของ Error แบบที่ 2



รูปที่ 4.8 membership function ของ Differential Error แบบที่ 2

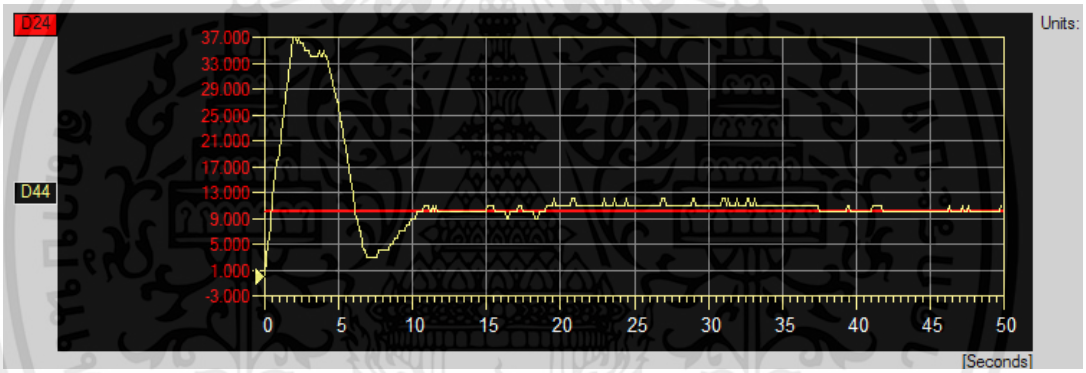


รูปที่ 4.9 membership function ของ Pulse Output แบบที่ 2

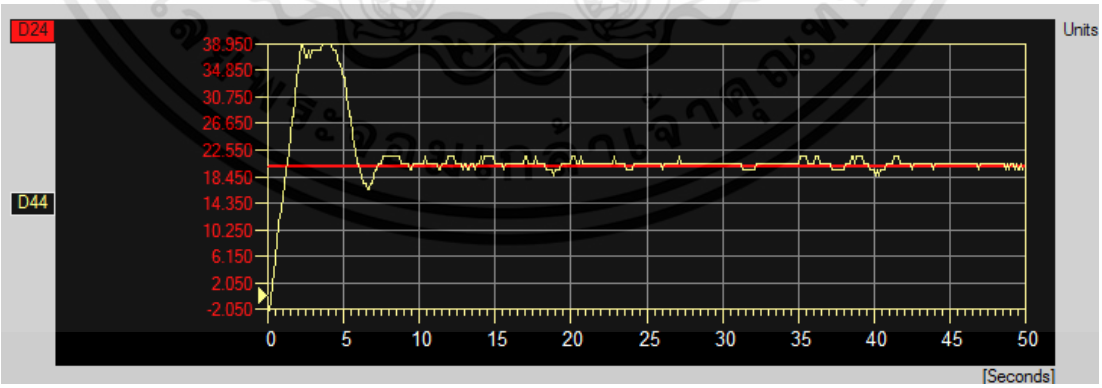
ตารางที่ 4.2 Rule Based แบบที่ 2

Differential Error / Error	NL	N	Z	P	PL
NL	R	R	R	RC	RC
N	R	RC	RC	C	C
Z	R	RC	C	LC	L
P	C	C	LC	LC	L
PL	LC	LC	L	L	L

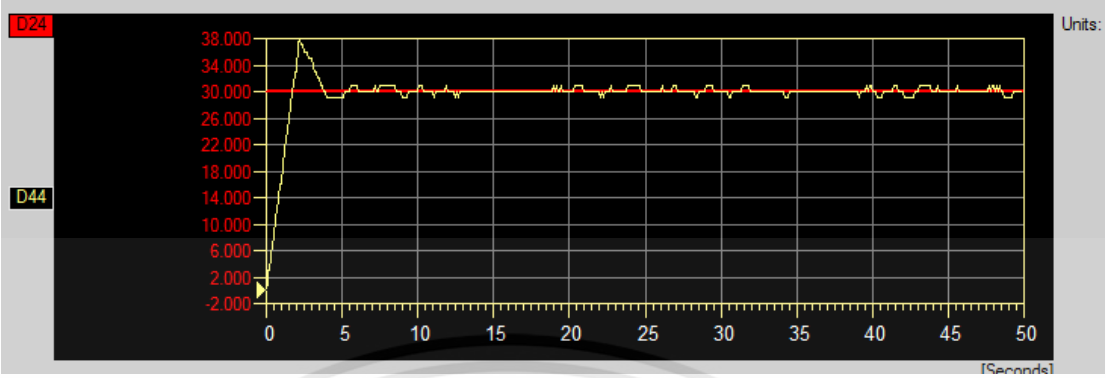
ผลการตอบสนอง



รูปที่ 4.10 ผลตอบสนองที่ sv=10 ตามเงื่อนไขที่ 2



รูปที่ 4.11 ผลตอบสนองที่ sv=20 ตามเงื่อนไขที่ 2



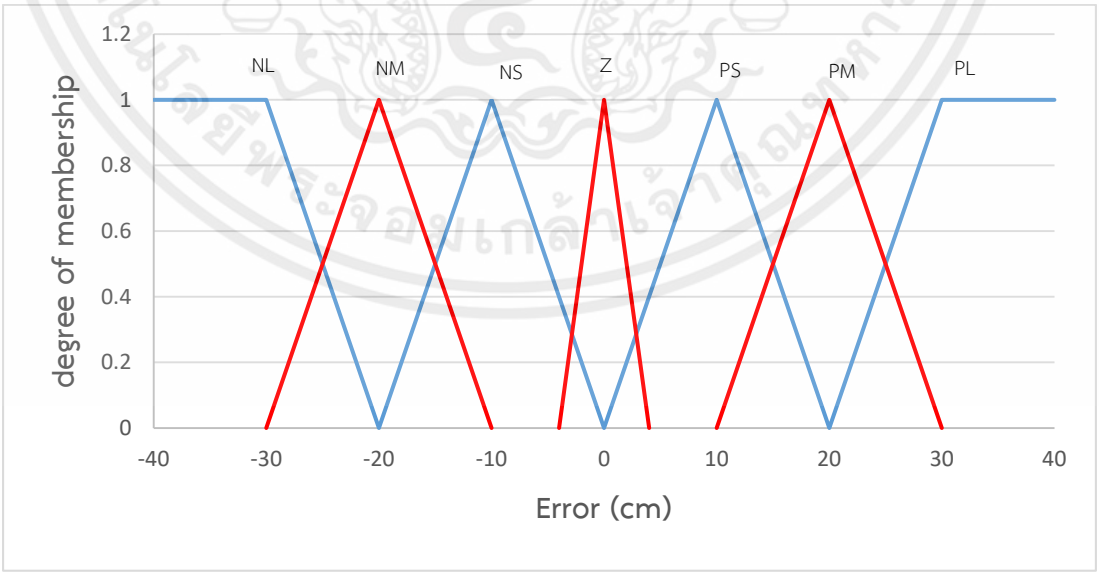
รูปที่ 4.12 ผลตอบสนองที่ sv=30 ตามเงื่อนไขที่ 2

<u>สรุปผลการทดลอง</u>

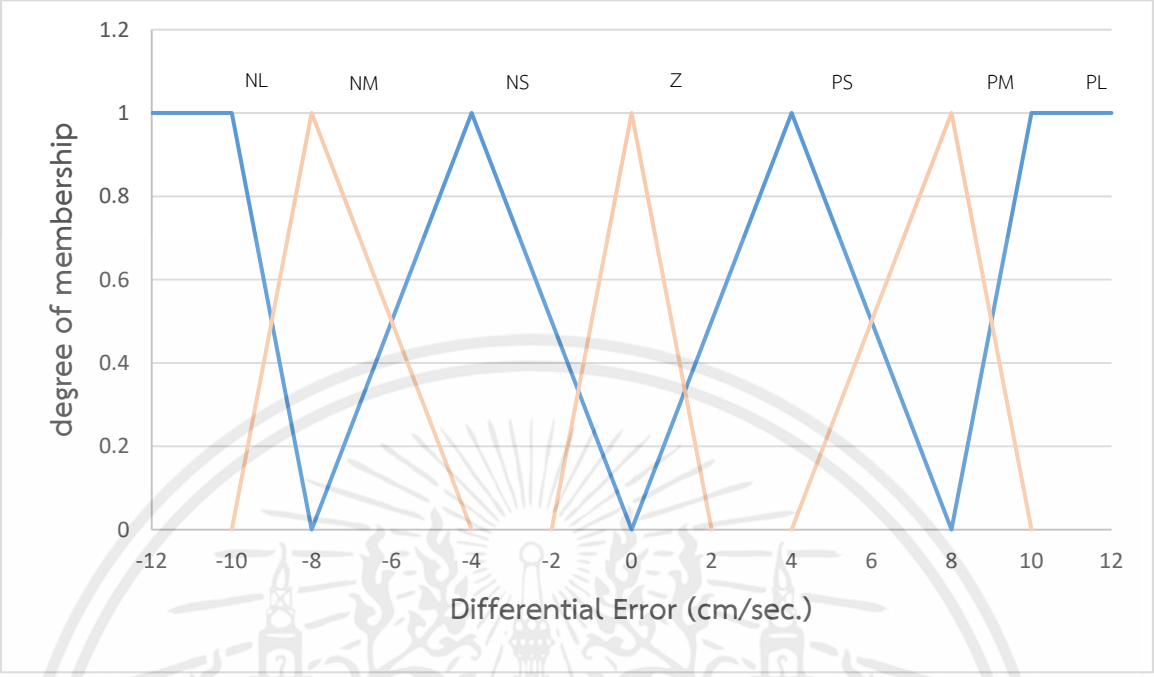
จากการทดลองกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 2 จะเห็นว่าผลตอบสนองยังไม่สามารถเข้าสู่ Setpoint ได้ ทั้งที่ตำแหน่ง 10, 20 และ 30 เซนติเมตร ที่ตำแหน่ง 10 และ 20 เซนติเมตร พบว่ามี undershoot ประมาณ 4 ถึง 6 เซนติเมตร และ พบว่าผลตอบสนอง Oscillate จาก Setpoint ช่วง $\pm$ 2 cm

4.2.3 การกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 3

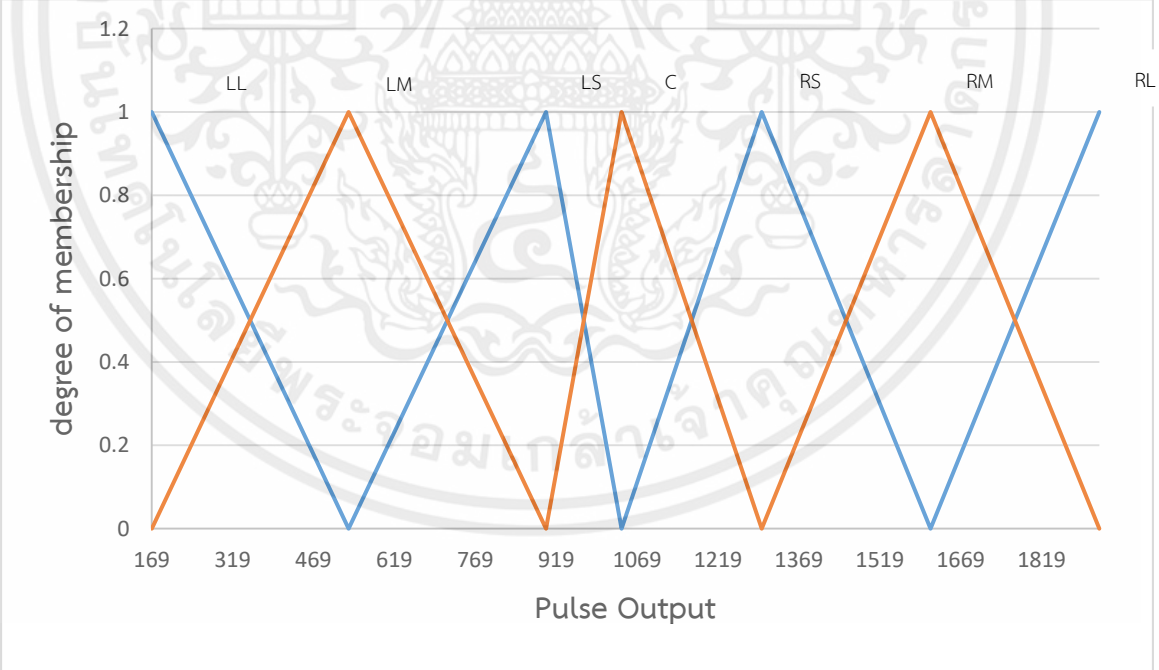
(เพิ่มจาก 5 input/output เป็น 7 input/output)



รูปที่ 4.13 membership function ของ Error แบบที่ 3



รูปที่ 4.14 membership function ของ Differential Error แบบที่ 3

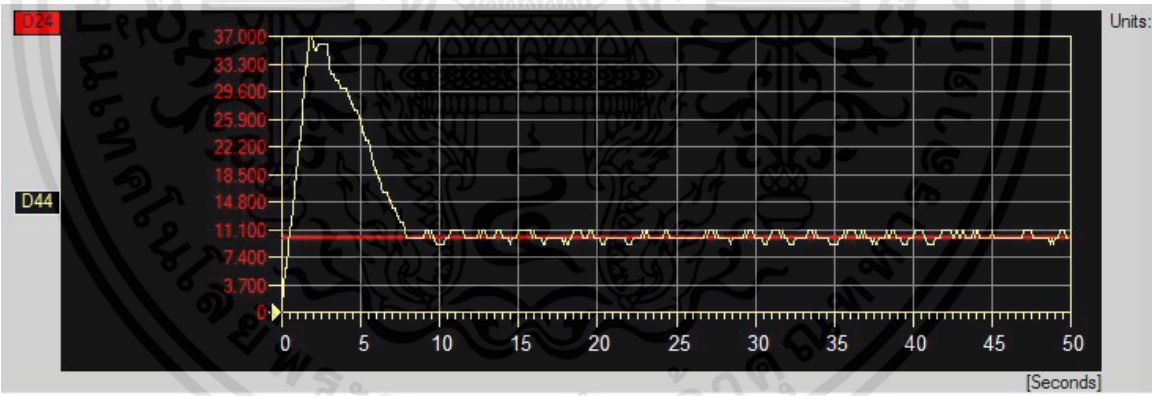


รูปที่ 4.15 membership function ของ Pulse Output แบบที่ 3

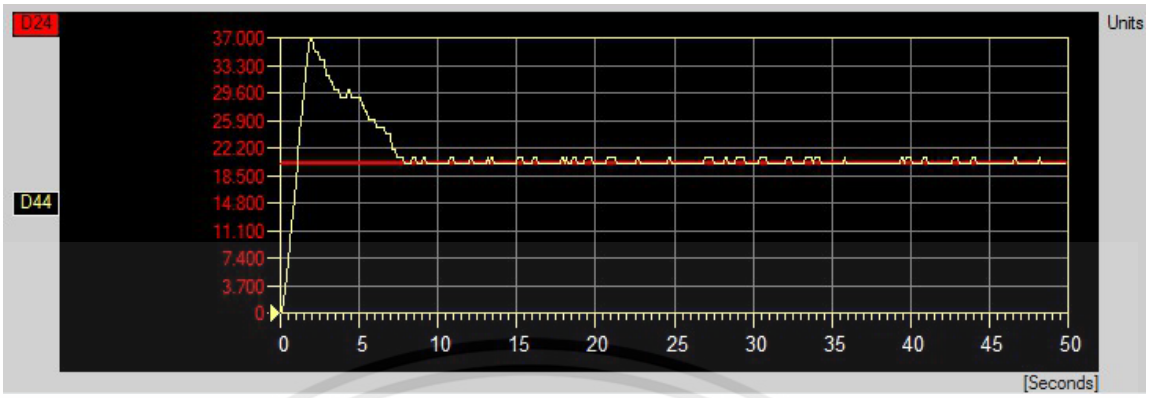
ตารางที่ 4.3 Rule Based แบบที่ 3

Error \ Differential Error	NL	NM	NS	Z	PS	PM	PL
NL	RL	RL	RL	RL	RM	RS	C
NM	RL	RL	RM	RM	RS	C	LS
NS	RL	RM	RM	RS	C	LS	LM
Z	RL	RM	RS	C	LS	LM	LL
PS	RM	RS	C	LS	LM	LM	LL
PM	RS	C	LS	LM	LM	LL	LL
PL	C	LS	LM	LL	LL	LL	LL

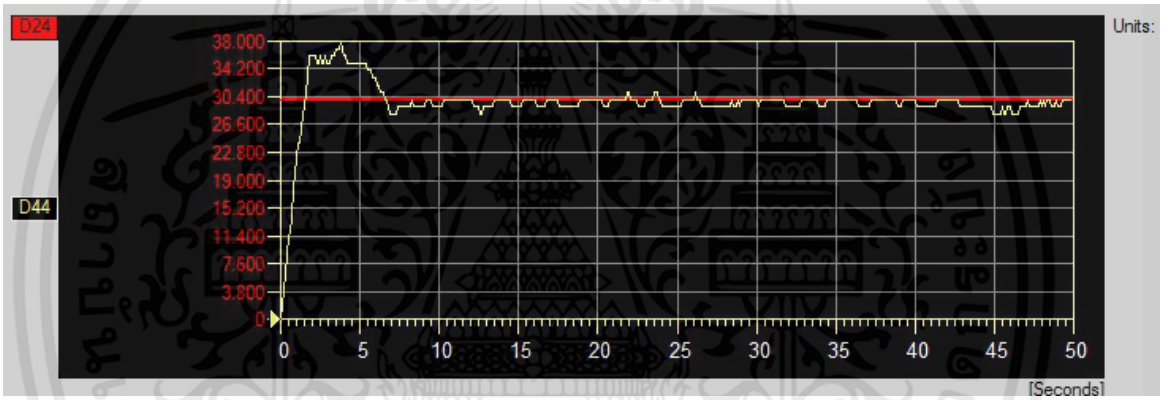
ผลการตอบสนอง



รูปที่ 4.16 ผลตอบสนองที่ sv=10 ตามเงื่อนไขที่ 3



รูปที่ 4.17 ผลตอบสนองที่ sv=20 ตามเงื่อนไขที่ 3



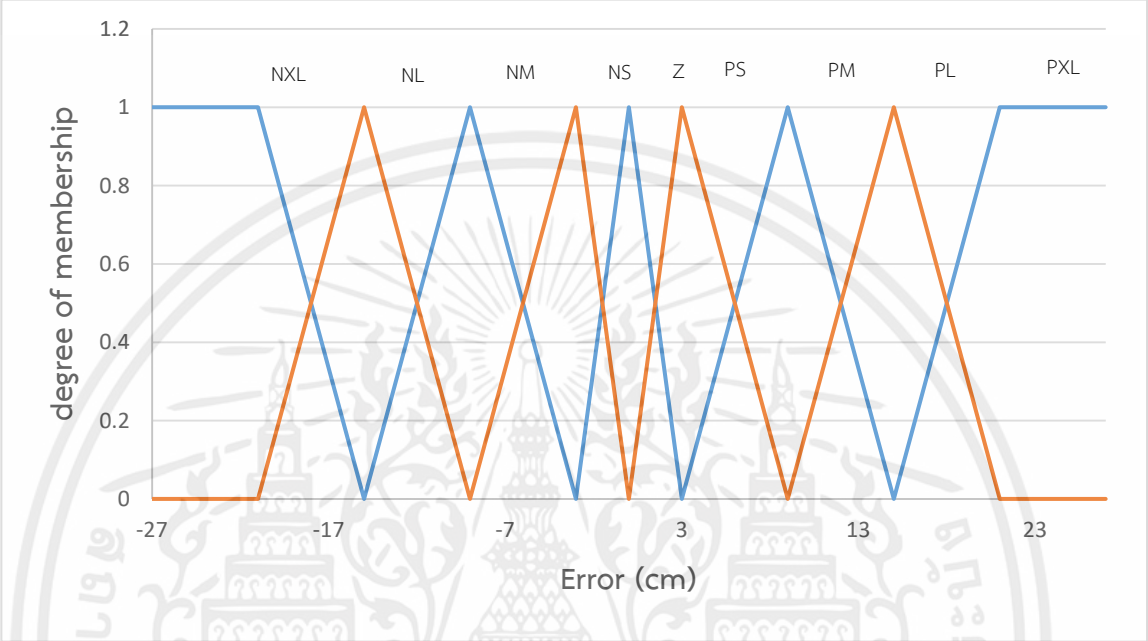
รูปที่ 4.18 ผลตอบสนองที่ sv=30 ตามเงื่อนไขที่ 3

<u>สรุปผลการทดลอง</u>

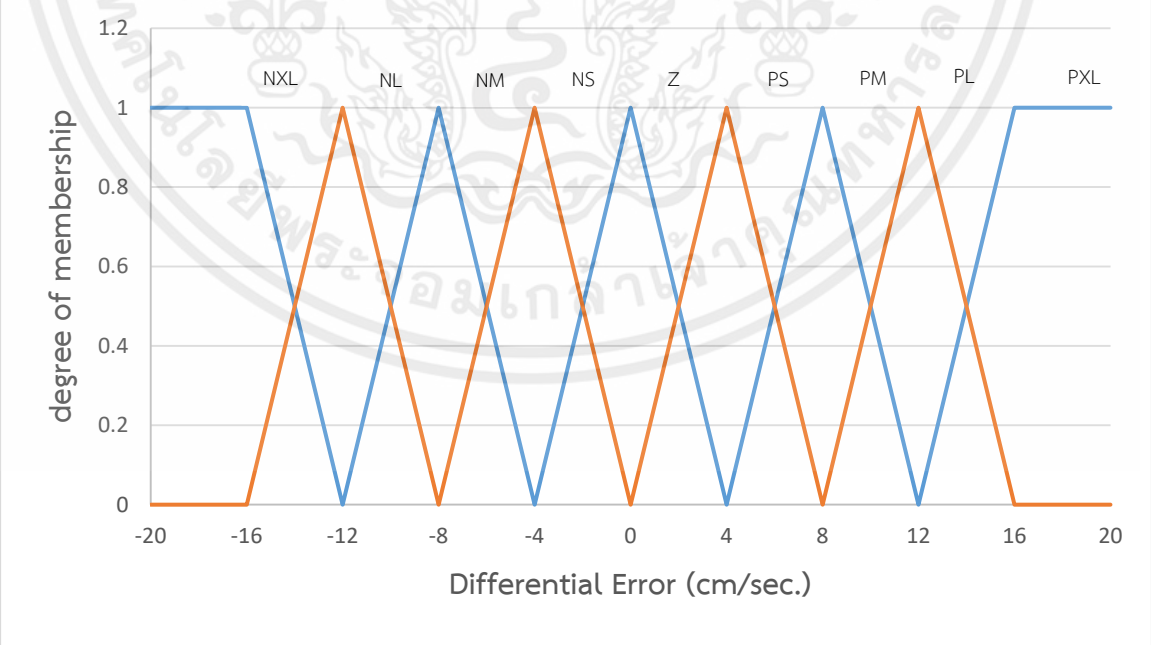
จากการทดลองกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 3 จะเห็นว่าผลตอบสนองเข้าสู่ Setpoint เร็วขึ้น ทั้งที่ตำแหน่ง 10, 20 และ 30 เซนติเมตร พบว่าเกิด undershoot น้อยลง และผลตอบสนอง Oscillate จาก Setpoint ช่วง $\pm$ 2 cm

4.2.4 การกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 4

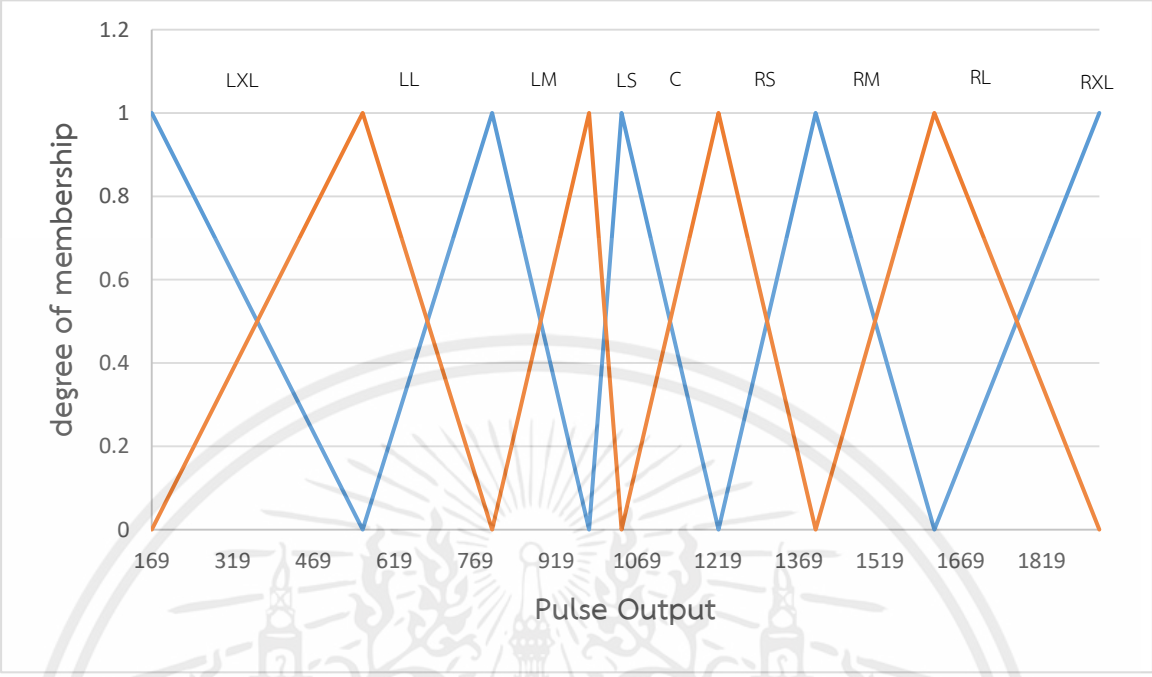
(เพิ่มจาก 7 input/output เป็น 9 input/output)



รูปที่ 4.19 membership function ของ Error แบบที่ 4



รูปที่ 4.20 membership function ของ Differential Error แบบที่ 4

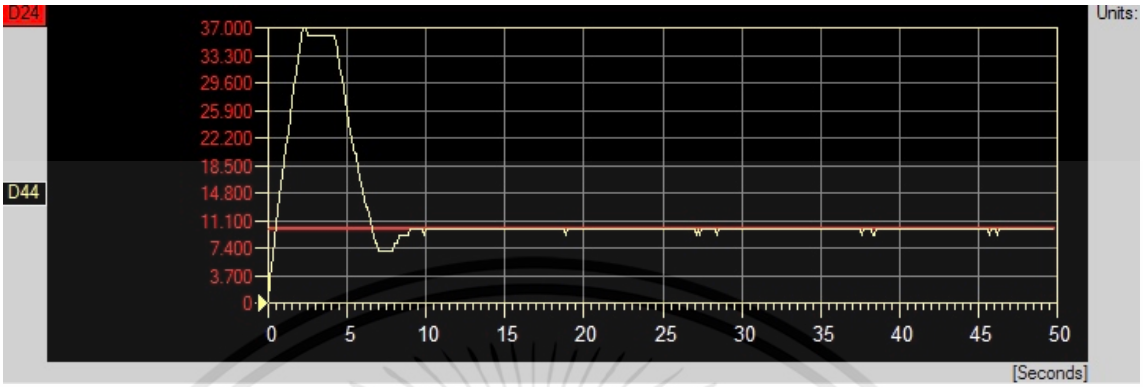


รูปที่ 4.21 membership function ของ Pulse Output แบบที่ 4

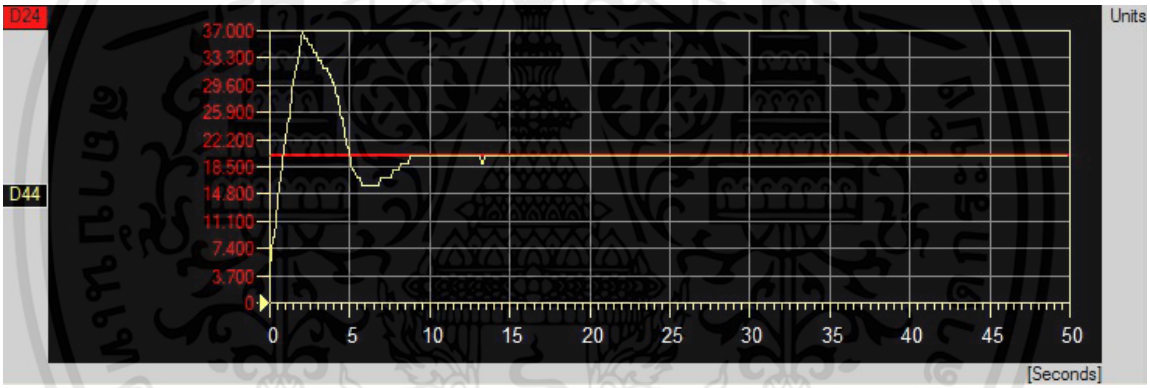
ตารางที่ 4.4 Rule Based แบบที่ 4

Differential Error / Error	NXL	NL	NM	NS	Z	PS	PM	PL	PXL
NXL	RXL	RXL	RXL	RXL	RXL	RL	RM	RS	C
NL	RXL	RXL	RXL	RL	RL	RM	RS	C	LS
NM	RXL	RXL	RL	RL	RM	RS	C	LS	LM
NS	RXL	RL	RL	RM	RS	C	LS	LM	LL
Z	RXL	RL	RM	RS	C	LS	LM	LL	LXL
PS	RL	RM	RS	C	LS	LM	LL	LL	LXL
PM	RM	RS	C	LS	LM	LL	LL	LXL	LXL
PL	RS	C	LS	LM	LL	LL	LXL	LXL	LXL
PXL	C	LS	LM	LL	LXL	LXL	LXL	LXL	LXL

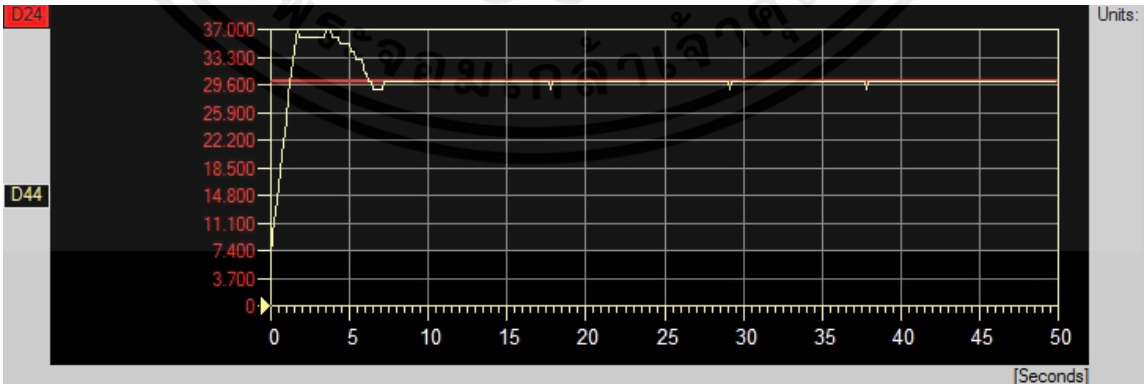
ผลการตอบสนอง



รูปที่ 4.22 ผลตอบสนองที่ sv=10 ตามเงื่อนไขที่ 4



รูปที่ 4.23 ผลตอบสนองที่ sv=20 ตามเงื่อนไขที่ 4



รูปที่ 4.24 ผลตอบสนองที่ sv=30 ตามเงื่อนไขที่ 4

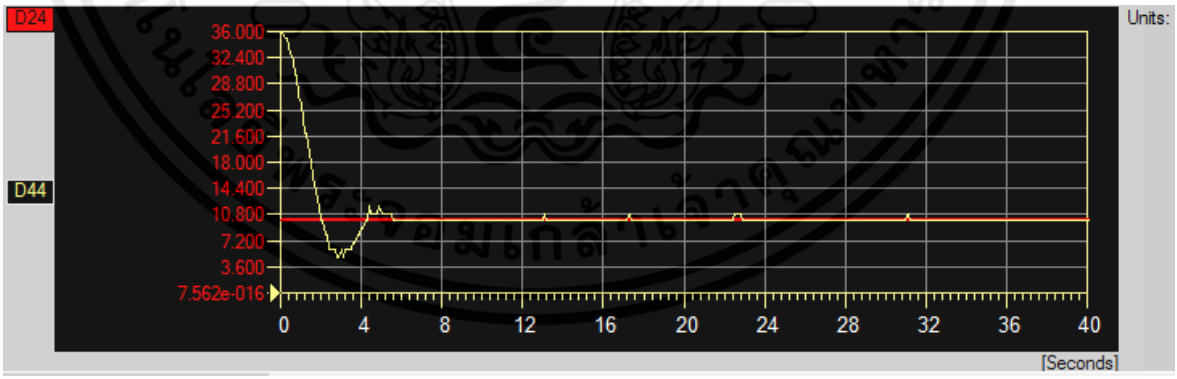
<u>สรุปผลการทดลอง</u>

จากการทดลองกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 4 จะเห็นว่าผลตอบสนองเข้าสู่ Setpoint เร็วขึ้น ทั้งที่ตำแหน่ง 10, 20 และ 30 เซนติเมตร และผลตอบสนอง Oscillate จาก Setpoint ช่วง ± 1 cm

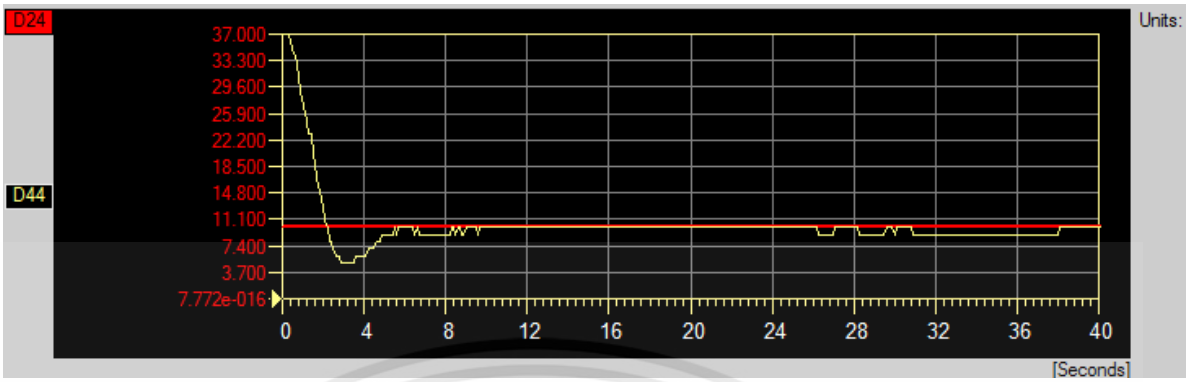
4.3 การทดลองผลตอบสนองของ membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 4 ในแต่ละ Setpoint 10, 20 และ 30 เซนติเมตร

จากการทดลองในหัวข้อที่ 4.1 เราได้ผลตอบสนองที่ดีที่สุดจากการกำหนด membership function ของ Error, Differential Error, pulse output และ rule based แบบที่ 4 (9 input/output) จึงทำการเก็บกราฟผลตอบสนองที่ตำแหน่งต่าง ๆ ตำแหน่งละ 6 ครั้ง และ นำมาหาค่าเวลาที่เข้าสู่ Setpoint ค่า Overshoot, ค่า Maximum, ค่าMinimum และ ค่า Average ของแต่ละ Setpoint เพื่อเป็นการเปรียบเทียบผลตอบสนองของแต่ละช่วง

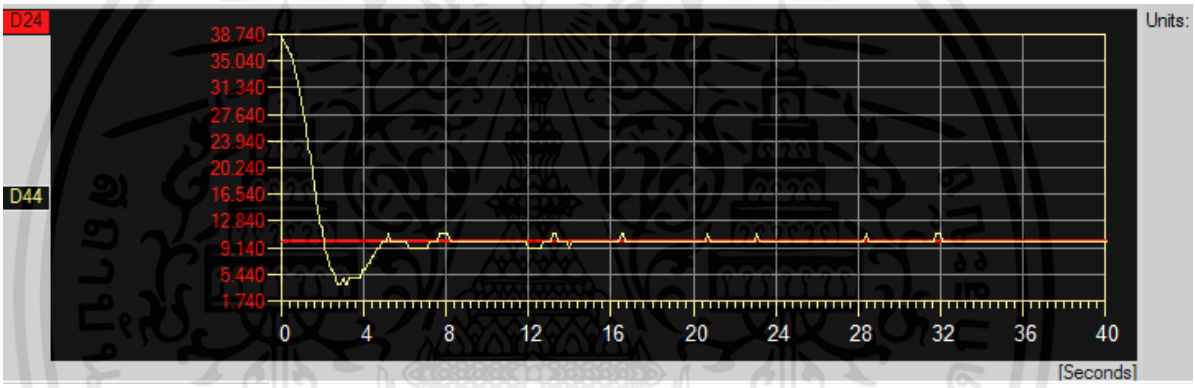
4.3.1 การทดลองผลตอบสนองที่ตำแหน่ง 10 เซนติเมตร



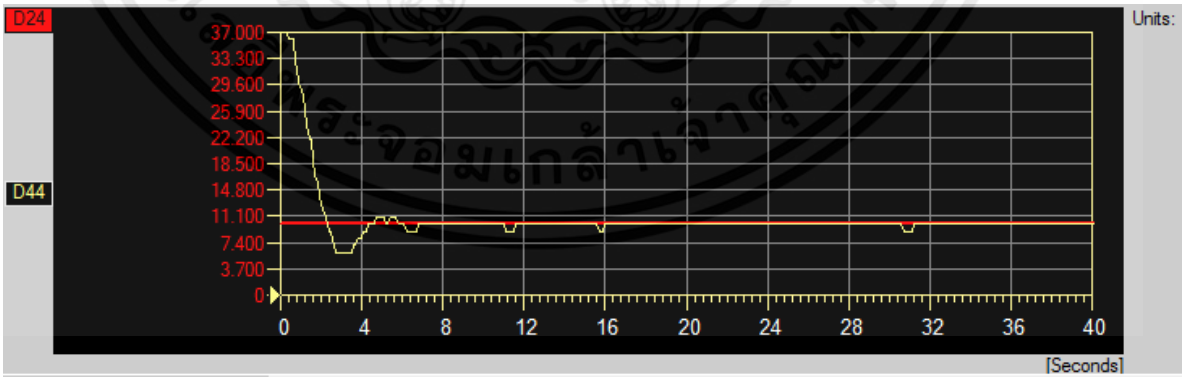
รูปที่ 4.25 ผลตอบสนองที่ sv=10 ครั้งที่ 1



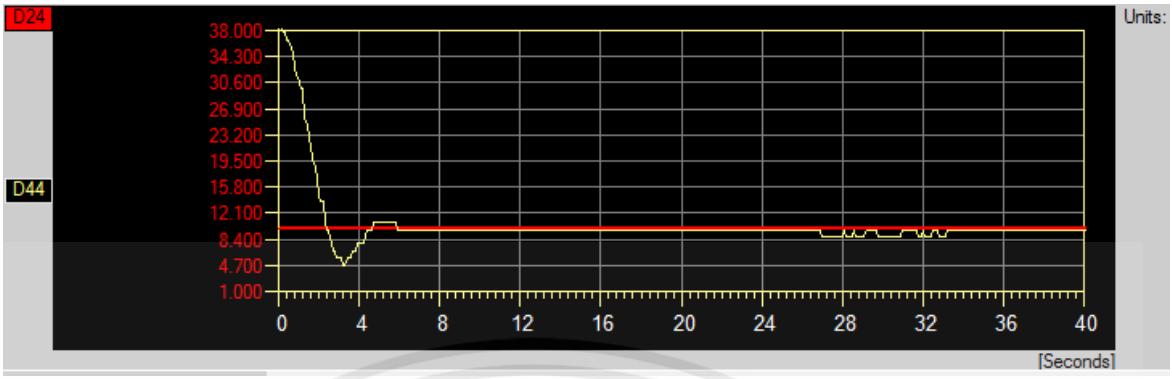
รูปที่ 4.26 ผลตอบสนองที่ sv=10 ครั้งที่ 2



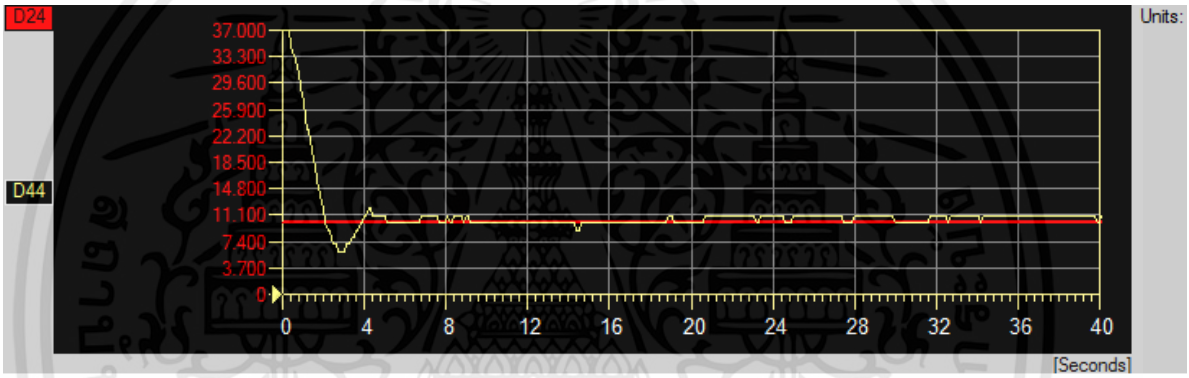
รูปที่ 4.27 ผลตอบสนองที่ sv=10 ครั้งที่ 3



รูปที่ 4.28 ผลตอบสนองที่ sv=10 ครั้งที่ 4



รูปที่ 4.29 ผลตอบสนองที่ sv=10 ครั้งที่ 5



รูปที่ 4.30 ผลตอบสนองที่ sv=10 ครั้งที่ 6

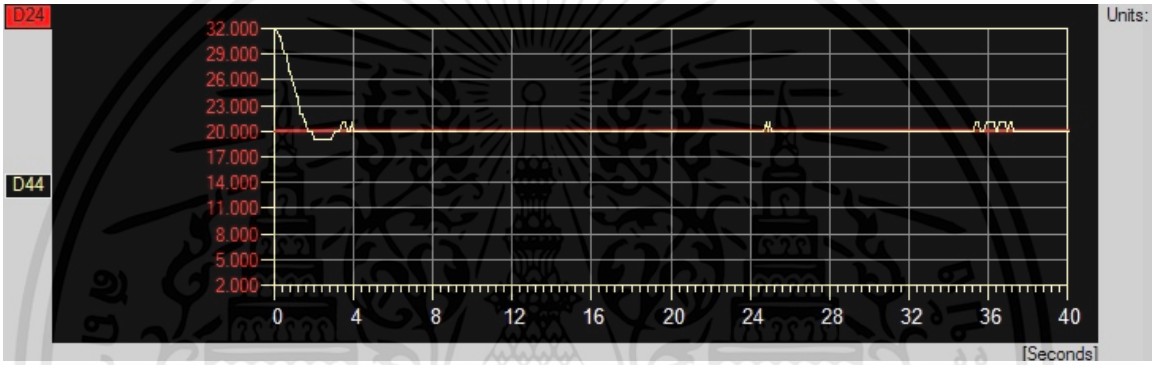
ตารางที่ 4.5 ตารางผลการทดลองที่ตำแหน่ง 10 cm

ครั้งที่	เวลาเข้าสู่ Setpoint		Overshoot	
1	5.6 s		50 %	
2	5.5 s		50 %	
3	5.3 s		60 %	
4	5.2 s		40 %	
5	5.9 s		60 %	
6	5.1 s		40 %	
	Avg.	5.43 s	Avg.	50 %
	Max	5.9 s	Max	60 %
	Min	5.1 s	Min	40 %

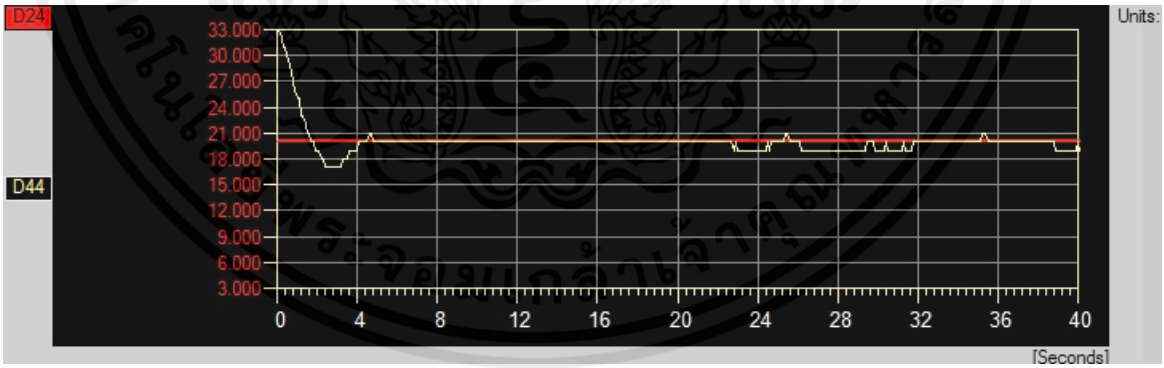
สรุปผลการทดลอง

จากการทดลองที่ตำแหน่ง Setpoint 10 เซนติเมตร ค่าเฉลี่ยของเวลาเข้าสู่ Setpoint เท่ากับ 5.43 วินาทีค่ามากที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 5.9 วินาที ค่าน้อยที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 5.1 วินาทีและ ค่าเฉลี่ย Overshoot เท่ากับ 50% ค่ามากที่สุดของ Overshoot เท่ากับ 60% ค่าน้อยที่สุดของ Overshoot เท่ากับ 40% และ ผลตอบสนอง Oscillate จาก Setpoint ช่วง ± 1 cm

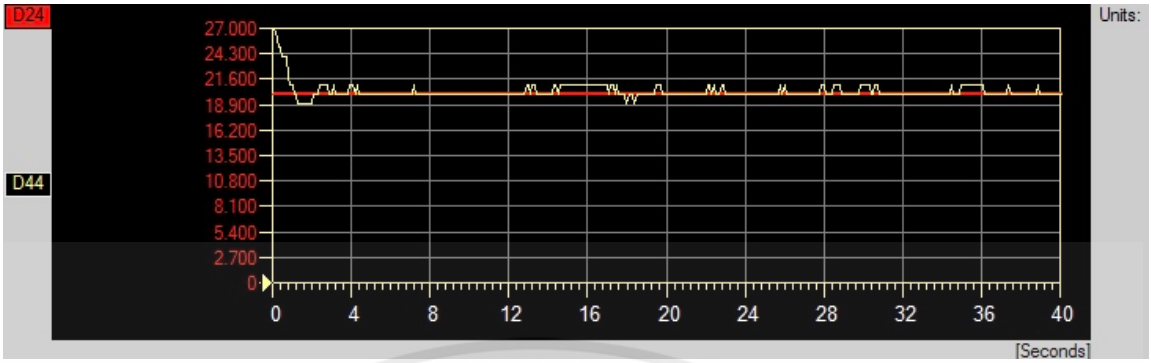
4.3.2 การทดลองผลตอบสนองที่ตำแหน่ง 20 เซนติเมตร



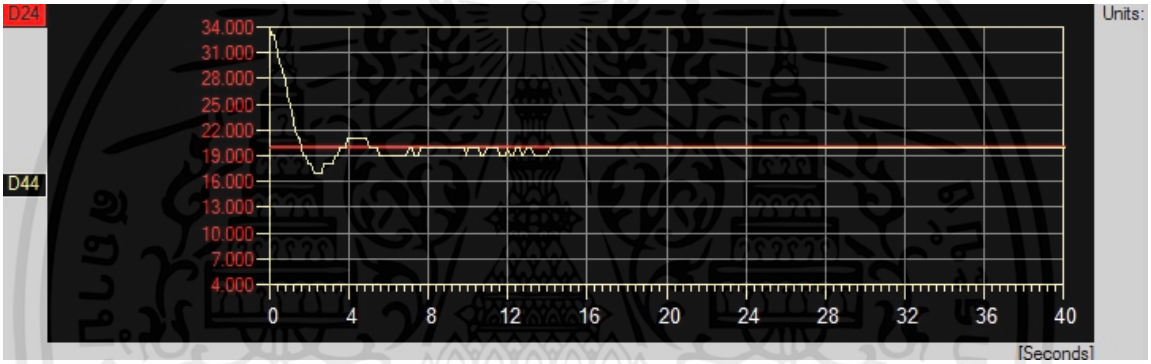
รูปที่ 4.31 ผลตอบสนองที่ sv=20 ครั้งที่ 1



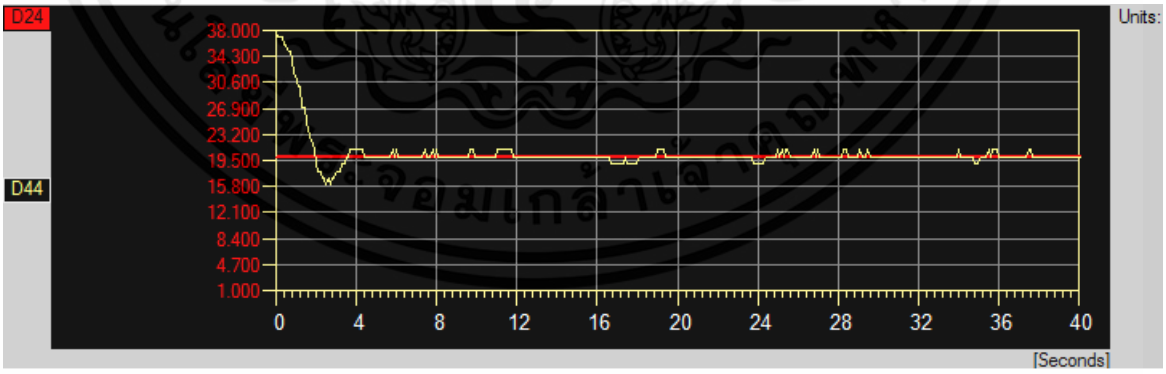
รูปที่ 4.32 ผลตอบสนองที่ sv=20 ครั้งที่ 2



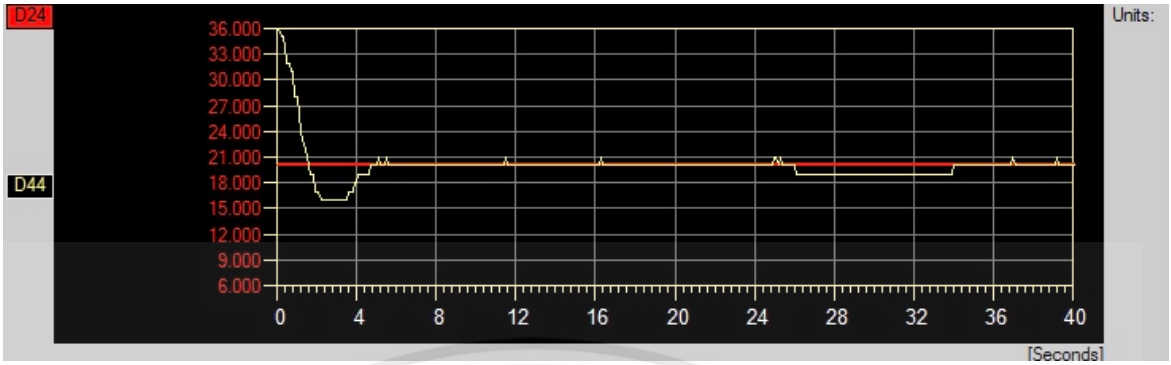
รูปที่ 4.33 ผลตอบสนองที่ sv=20 ครั้งที่ 3



รูปที่ 4.34 ผลตอบสนองที่ sv=20 ครั้งที่ 4



รูปที่ 4.35 ผลตอบสนองที่ sv=20 ครั้งที่ 5



รูปที่ 4.36 ผลตอบสนองที่ sv=20 ครั้งที่ 6

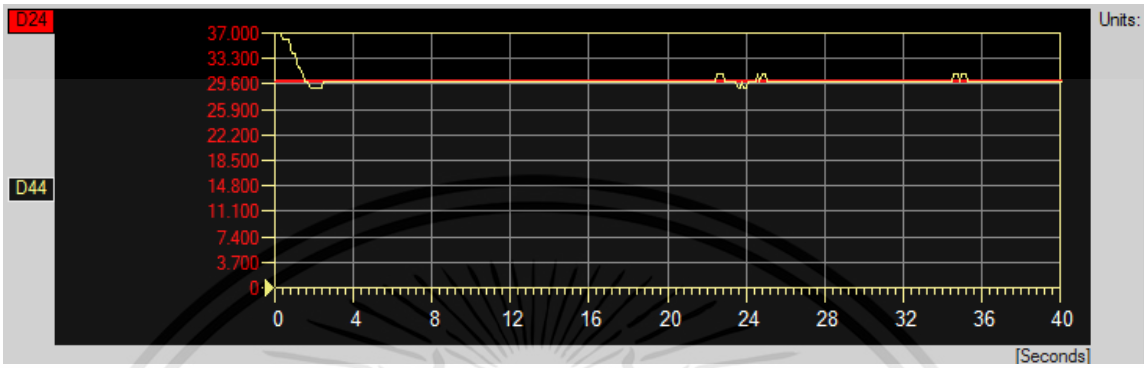
ตารางที่ 4.6 ตารางผลการทดลองที่ตำแหน่ง 20 cm

ครั้งที่	เวลาเข้าสู่ Setpoint		Overshoot	
1	4 s		5 %	
2	5 s		15 %	
3	4.6 s		5.5 %	
4	5.4 s		15 %	
5	4.4 s		20 %	
6	4.8 s		20 %	
	Avg.	**4.7 s**	Avg.	**13.417 %**
	Max	**5.4 s**	Max	**20 %**
	Min	**4 s**	Min	**5 %**

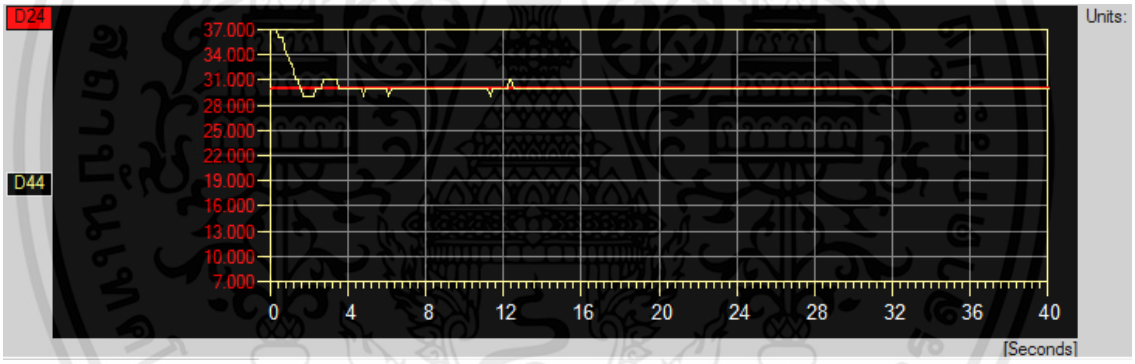
<u>สรุปผลการทดลอง</u>

จากการทดลองที่ตำแหน่ง Setpoint 20 เซนติเมตร ค่าเฉลี่ยของเวลาเข้าสู่ Setpoint เท่ากับ 4.7 วินาทีค่ามากที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 5.4 วินาที ค่าน้อยที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 4 วินาทีและ ค่าเฉลี่ย Overshoot เท่ากับ 13.417% ค่ามากที่สุดของ Overshoot เท่ากับ 20% ค่าน้อยที่สุดของ Overshoot เท่ากับ 5% และ ผลตอบสนอง Oscillate จาก Setpoint ช่วง $\pm$ 1 cm

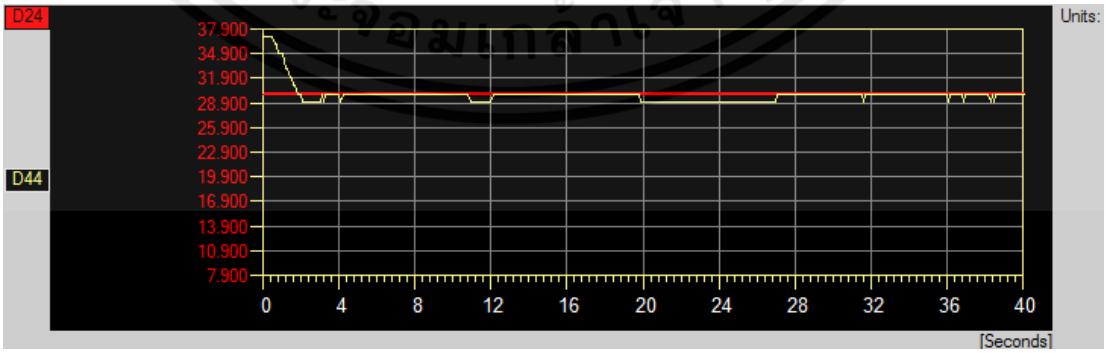
4.3.3 การทดลองผลตอบสนองที่ตำแหน่ง 30 เซนติเมตร



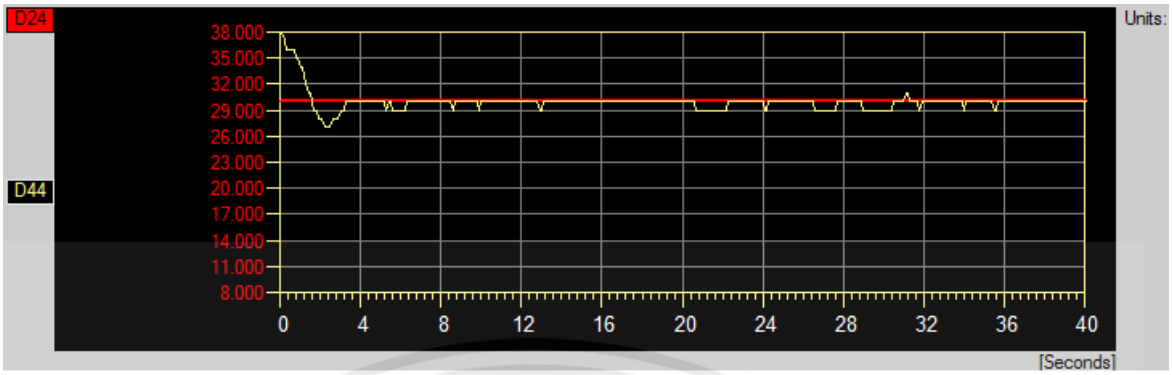
รูปที่ 4.37 ผลตอบสนองที่ sv=30 ครั้งที่ 1



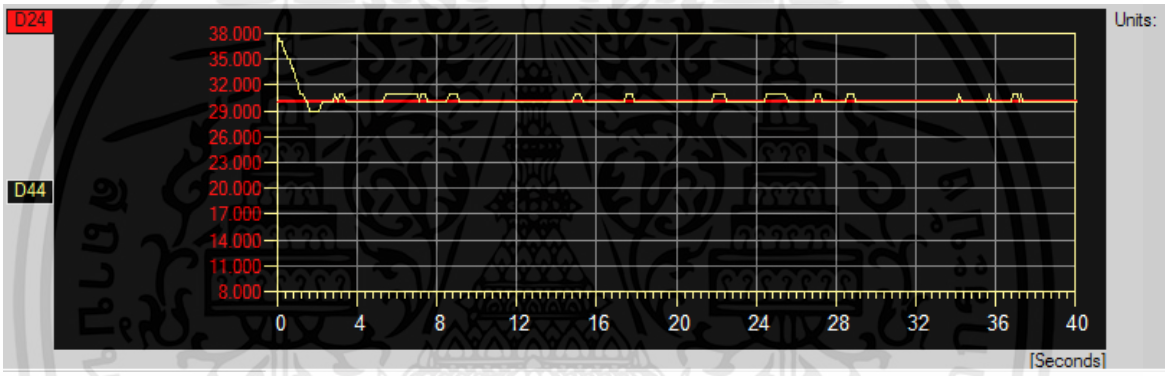
รูปที่ 4.38 ผลตอบสนองที่ sv=30 ครั้งที่ 2



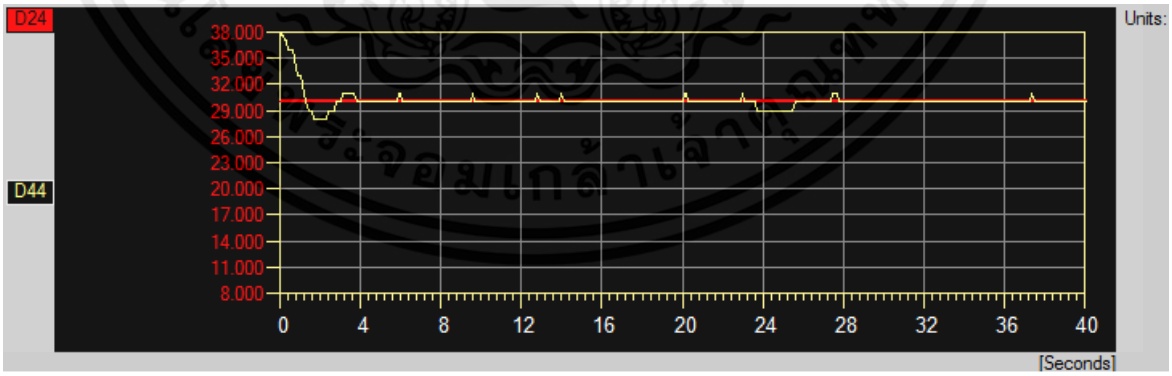
รูปที่ 4.39 ผลตอบสนองที่ sv=30 ครั้งที่ 3



รูปที่ 4.40 ผลตอบสนองที่ sv=30 ครั้งที่ 4



รูปที่ 4.41 ผลตอบสนองที่ sv=30 ครั้งที่ 5



รูปที่ 4.42 ผลตอบสนองที่ sv=30 ครั้งที่ 6

ตารางที่ 4.7 ตารางผลการทดลองที่ตำแหน่ง 30 cm

ครั้งที่	เวลาเข้าสู่ Setpoint		Overshoot	
1	2.5 s		3.3 %	
2	3.5 s		3.3 %	
3	3.1 s		3.3 %	
4	3.7 s		3.3 %	
5	2.2 s		3.3 %	
6	3.8 s		6.7 %	
	Avg.	3.12 s	Avg.	3.9 %
	Max	3.8 s	Max	6.7 %
	Min	2.2 s	Min	3.3 %

สรุปผลการทดลอง

จากการทดลองที่ตำแหน่ง Setpoint 30 เซนติเมตร ค่าเฉลี่ยของเวลาเข้าสู่ Setpoint เท่ากับ 3.12 วินาทีค่ามากที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 3.8 วินาที ค่าน้อยที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 2.2 วินาทีและ ค่าเฉลี่ย Overshoot เท่ากับ 3.9% ค่ามากที่สุดของ Overshoot เท่ากับ 6.7% ค่าน้อยที่สุดของ Overshoot เท่ากับ 3.3% และ ผลตอบสนอง Oscillate จาก Setpoint ช่วง ± 1 cm

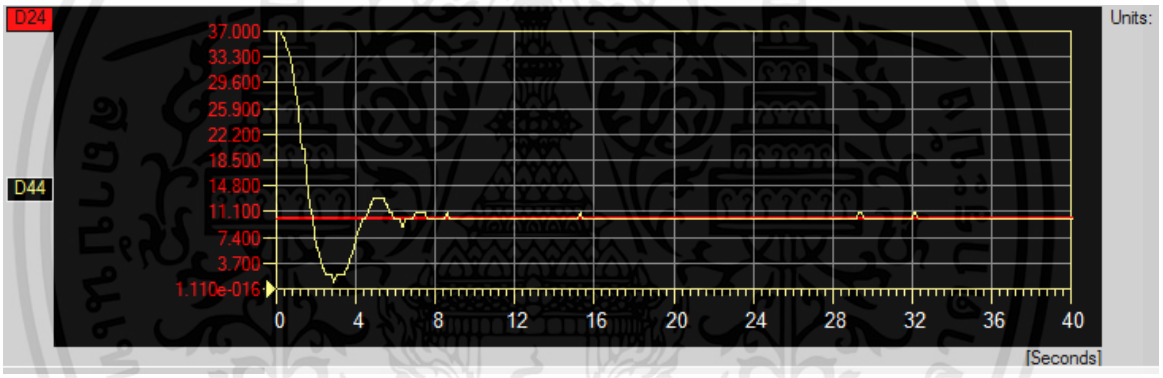
สรุปผลการทดลองทั้งหมด

ที่ตำแหน่ง 10 เซนติเมตร ได้ค่าเฉลี่ยเวลาเข้าสู่ Setpoint = 5.43 วินาที และ ได้ค่าเฉลี่ย Overshoot = 50.0% ที่ตำแหน่ง 20 เซนติเมตร ได้ค่าเฉลี่ยเวลาเข้าสู่ Setpoint = 4.7 วินาที และ ได้ค่าเฉลี่ย Overshoot = 13.417 % ที่ตำแหน่ง 30 เซนติเมตร ได้ค่าเฉลี่ยเวลาเข้าสู่ Setpoint = 3.12 วินาที และ ได้ค่าเฉลี่ย Overshoot = 3.9% จะเห็นได้ว่าตำแหน่ง Setpoint ยิ่งห่างจากตำแหน่งเริ่มต้น (ที่ตำแหน่ง 37 เซนติเมตร) มากเท่าใด ระยะเวลาที่ใช้ในการเข้าสู่ Setpoint จะมากขึ้นและจะเกิด Overshoot มากขึ้นด้วย

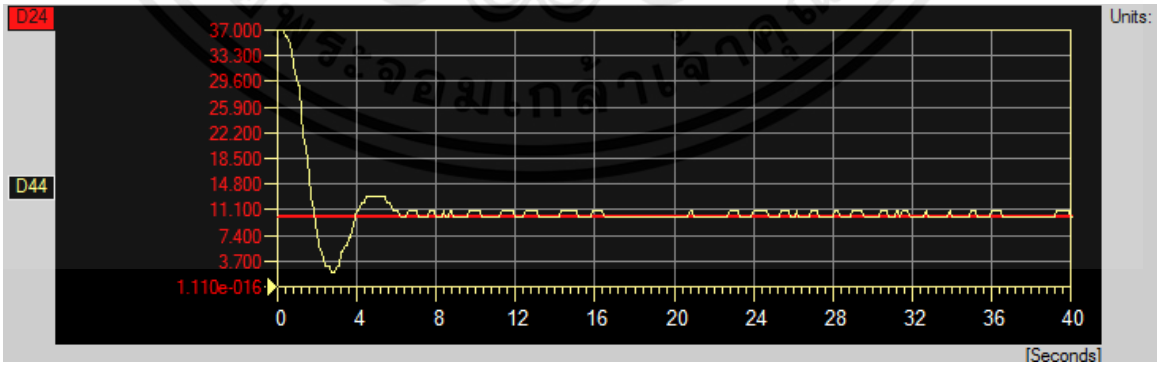
4.4 การทดลองผลตอบสนองในแต่ละ Setpoint 10, 20 และ 30 เซนติเมตร โดยใช้ทฤษฎีควบคุมพีไอดี

การทดลองผลตอบสนองของพีไอดี นำมาทำลองเพื่อเป็นการเปรียบเทียบกับทฤษฎีควบคุมฟัซซีลอจิกโดยเปรียบเทียบเวลาที่ใช้ในการเข้าสู่ Setpoint และ Overshoot ของทั้งสองทฤษฎีการควบคุม เพื่อให้ทราบถึงข้อดีข้อเสียของทั้งสองทฤษฎีการควบคุม โดยค่า PID ที่นำมาใช้ในการควบคุมมาจากโปรเจคการประยุกต์การควบคุมด้วย PLC/PID ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว (The Applying PLC/PID Control of Counterbalance Sphere Object On Single Plane) ของปีการศึกษา 2562 โดยปรับค่า P (Tp = 1000) ค่า I (Ti = 9999) และ D (Td=45)

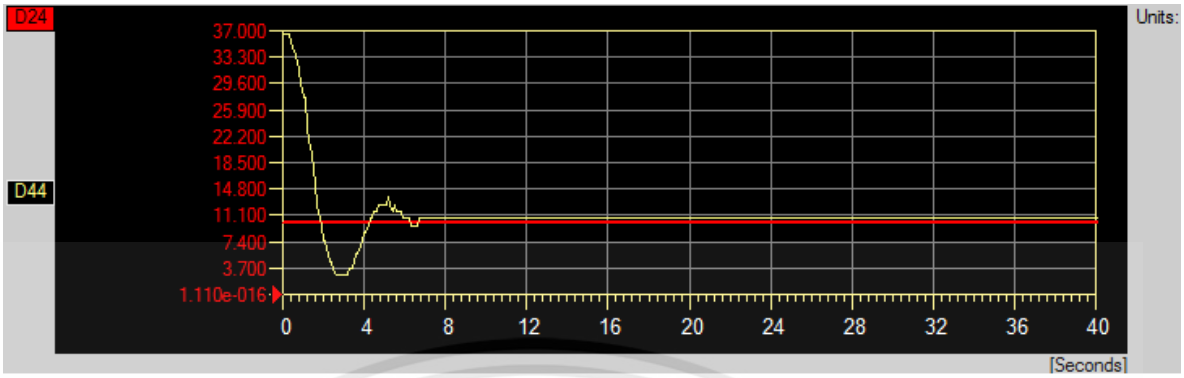
4.4.1 การทดลองผลตอบสนองที่ตำแหน่ง 10 เซนติเมตร



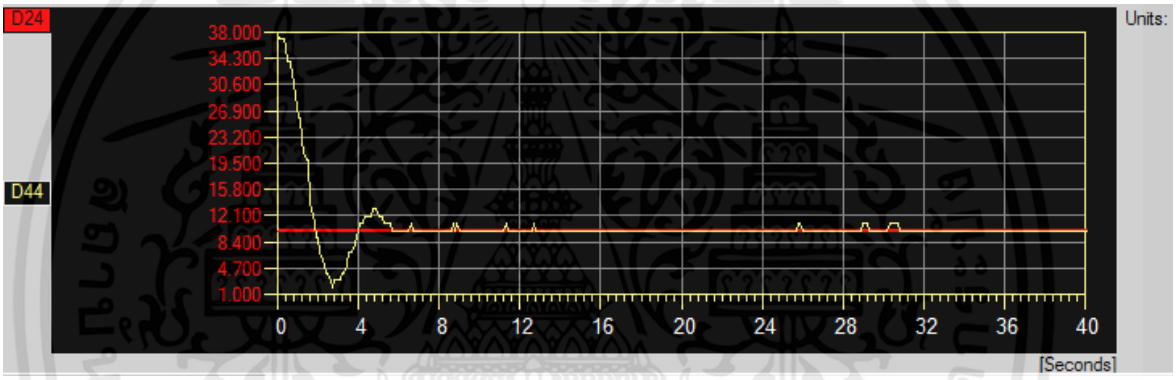
รูปที่ 4.43 ผลตอบสนองของ PID ที่ sv=10 ครั้งที่ 1



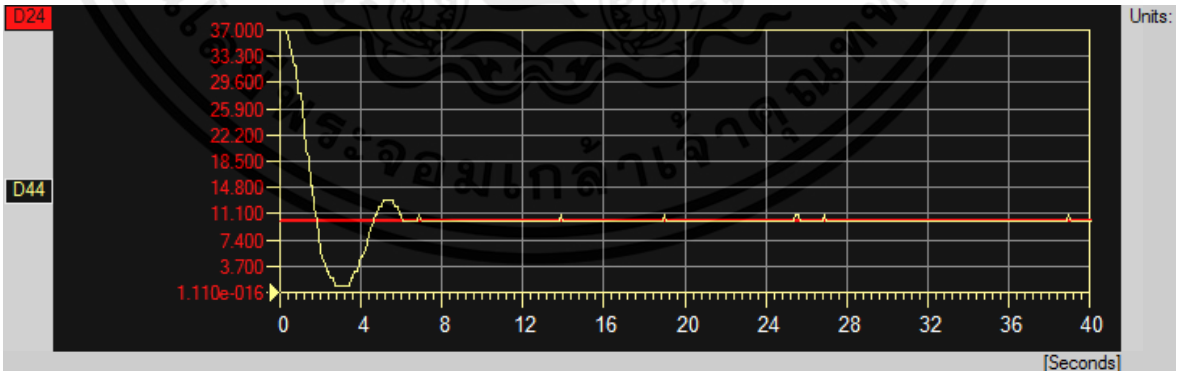
รูปที่ 4.44 ผลตอบสนองของ PID ที่ sv=10 ครั้งที่ 2



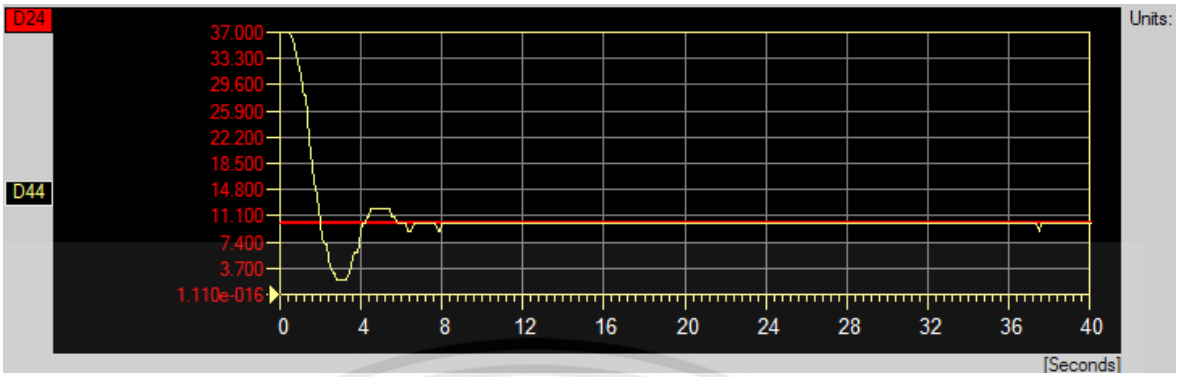
รูปที่ 4.45 ผลตอบสนองของ PID ที่ sv=10 ครั้งที่ 3



รูปที่ 4.46 ผลตอบสนองของ PID ที่ sv=10 ครั้งที่ 4



รูปที่ 4.47 ผลตอบสนองของ PID ที่ sv=10 ครั้งที่ 5



รูปที่ 4.48 ผลตอบสนองของ PID ที่ sv=10 ครั้งที่ 6

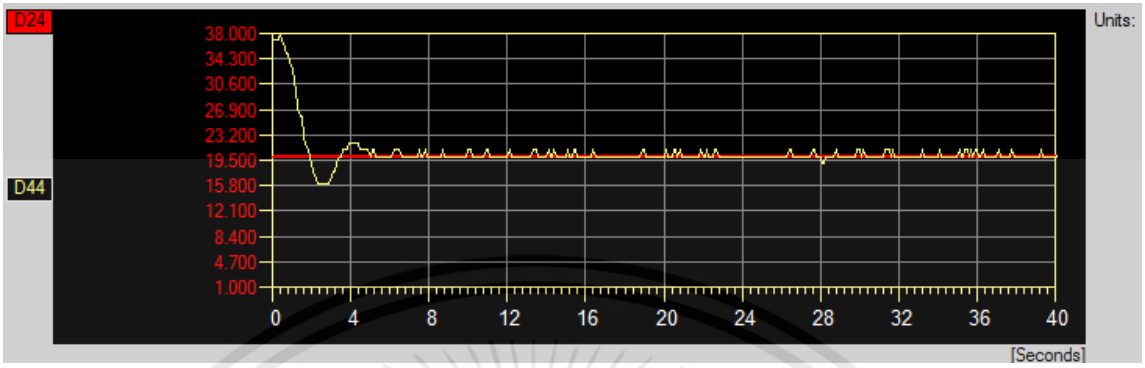
ตารางที่ 4.8 ตารางผลการทดลองของ PID ที่ตำแหน่ง 10 cm

ครั้งที่	เวลาเข้าสู่ Setpoint		Overshoot	
1	5.9 s		90 %	
2	6.3 s		80 %	
3	6.3 s		70 %	
4	5.7 s		80 %	
5	6.2 s		90 %	
6	5.9 s		80 %	
	Avg.	6.05 s	Avg.	81.7 %
	Max	6.3 s	Max	90 %
	Min	5.7 s	Min	70 %

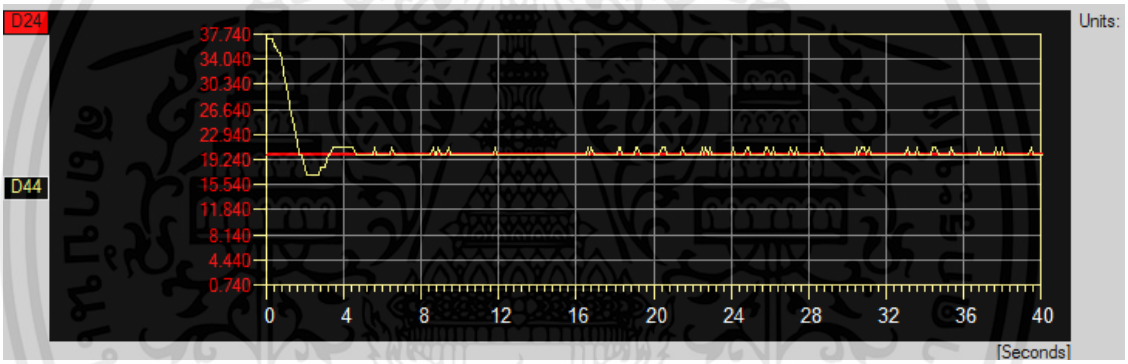
<u>สรุปผลการทดลอง</u>

จากการทดลองที่ตำแหน่ง Setpoint 10 เซนติเมตร ค่าเฉลี่ยของเวลาเข้าสู่ Setpoint เท่ากับ 6.05 วินาทีค่ามากที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 6.3 วินาที ค่าน้อยที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 5.7 วินาทีและ ค่าเฉลี่ย Overshoot เท่ากับ 81.7% ค่ามากที่สุดของ Overshoot เท่ากับ 90% ค่าน้อยที่สุดของ Overshoot เท่ากับ 70% และ ผลตอบสนอง Oscillate จาก Setpoint ช่วง $\pm$ 1 cm

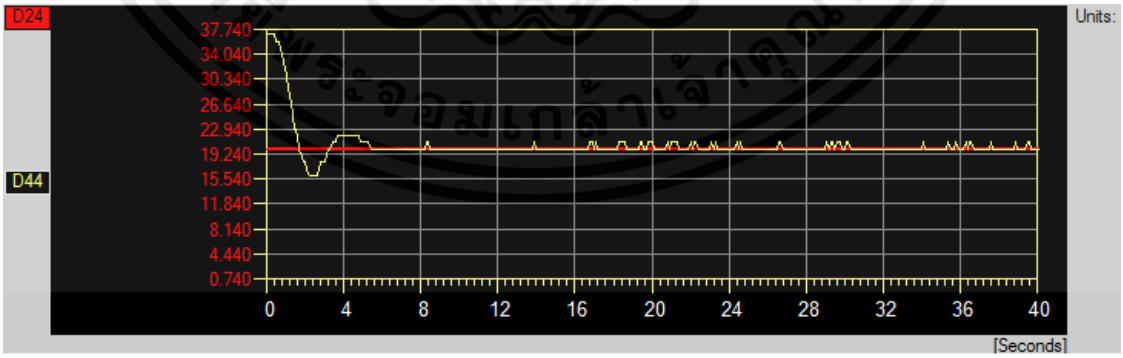
4.4.2 การทดลองผลตอบสนองที่ตำแหน่ง 20 เซนติเมตร



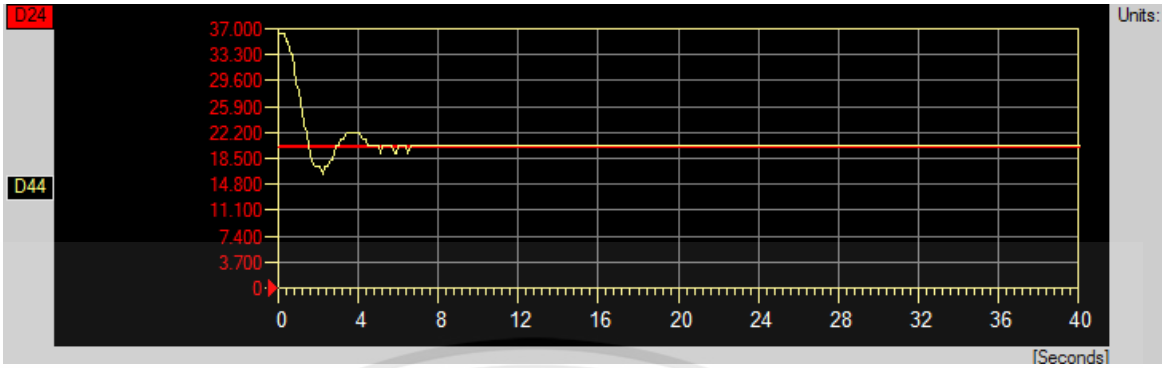
รูปที่ 4.49 ผลตอบสนองของ PID ที่ sv=20 ครั้งที่ 1



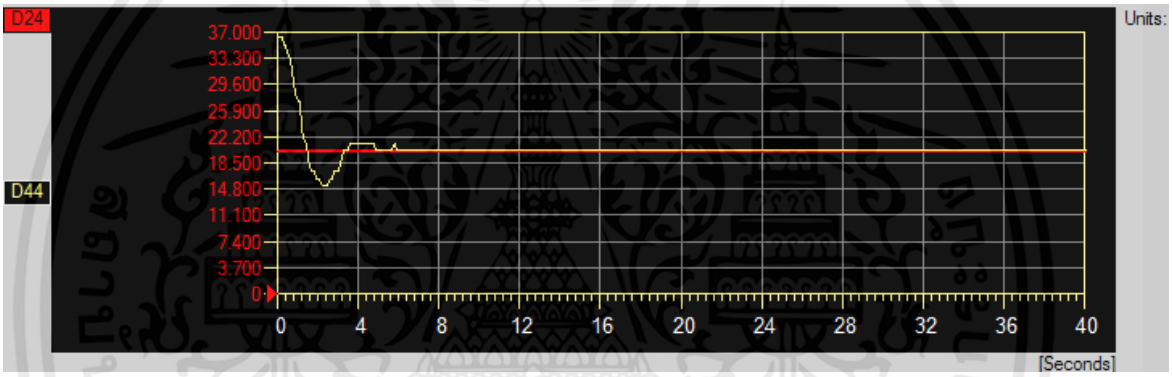
รูปที่ 4.50 ผลตอบสนองของ PID ที่ sv=20 ครั้งที่ 2



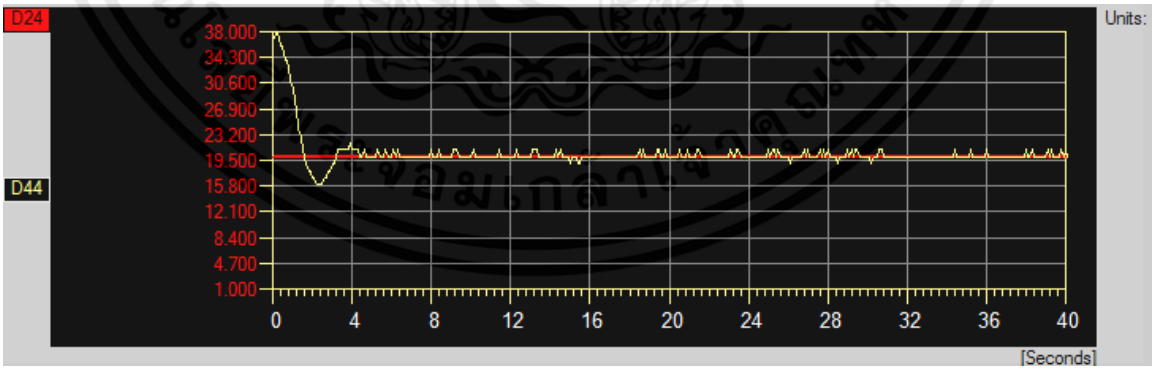
รูปที่ 4.51 ผลตอบสนองของ PID ที่ sv=20 ครั้งที่ 3



รูปที่ 4.52 ผลตอบสนองของ PID ที่ sv=20 ครั้งที่ 4



รูปที่ 4.53 ผลตอบสนองของ PID ที่ sv=20 ครั้งที่ 5



รูปที่ 4.54 ผลตอบสนองของ PID ที่ sv=20 ครั้งที่ 6

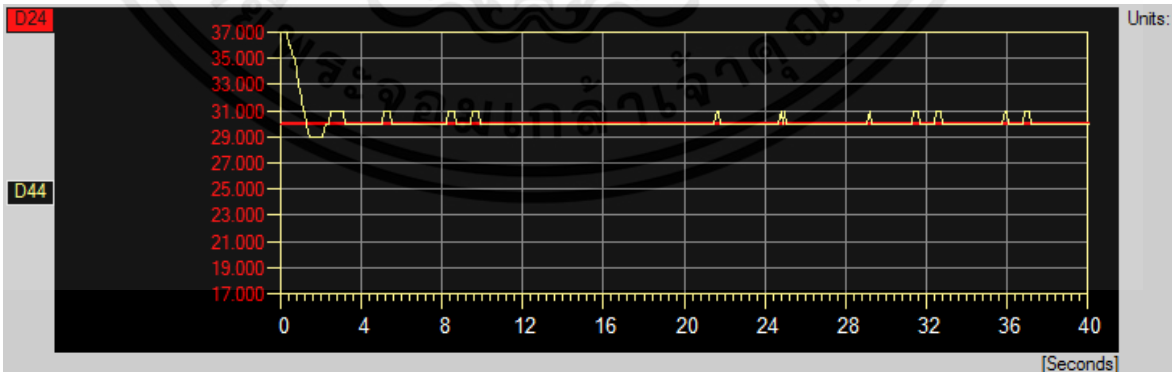
ตารางที่ 4.9 ตารางผลการทดลองของ PID ที่ตำแหน่ง 20 cm

ครั้งที่	เวลาเข้าสู่ Setpoint		Overshoot	
1	5.3 s		15 %	
2	5.4 s		15 %	
3	4.6 s		15 %	
4	4.9 s		20 %	
5	4.6 s		15 %	
6	4.7 s		20 %	
	Avg.	4.92 s	Avg.	16.7 %
	Max	5.4 s	Max	20 %
	Min	4.6 s	Min	15 %

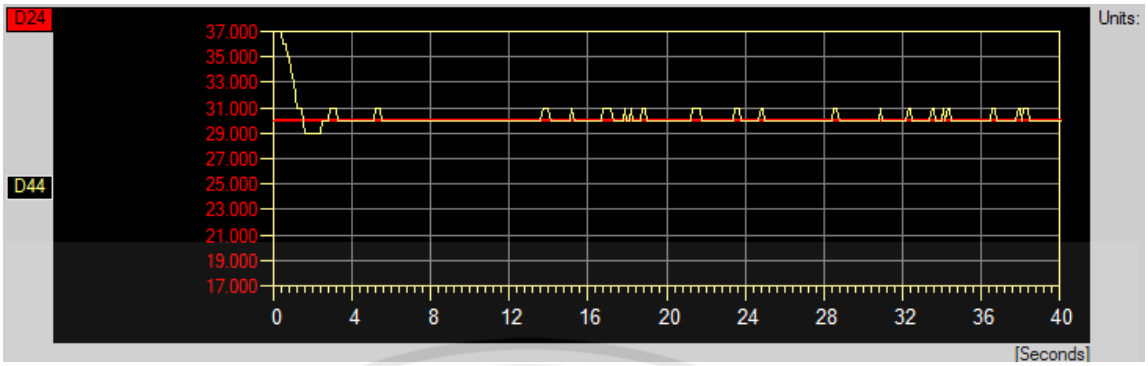
สรุปผลการทดลอง

จากการทดลองที่ตำแหน่ง Setpoint 20 เซนติเมตร ค่าเฉลี่ยของเวลาเข้าสู่ Setpoint เท่ากับ 4.92 วินาทีค่ามากที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 5.4 วินาที ค่าน้อยที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 4.6 วินาทีและ ค่าเฉลี่ย Overshoot เท่ากับ 16.7% ค่ามากที่สุดของ Overshoot เท่ากับ 20% ค่าน้อยที่สุดของ Overshoot เท่ากับ 15% และ ผลตอบสนอง Oscillate จาก Setpoint ช่วง $\pm$ 1 cm

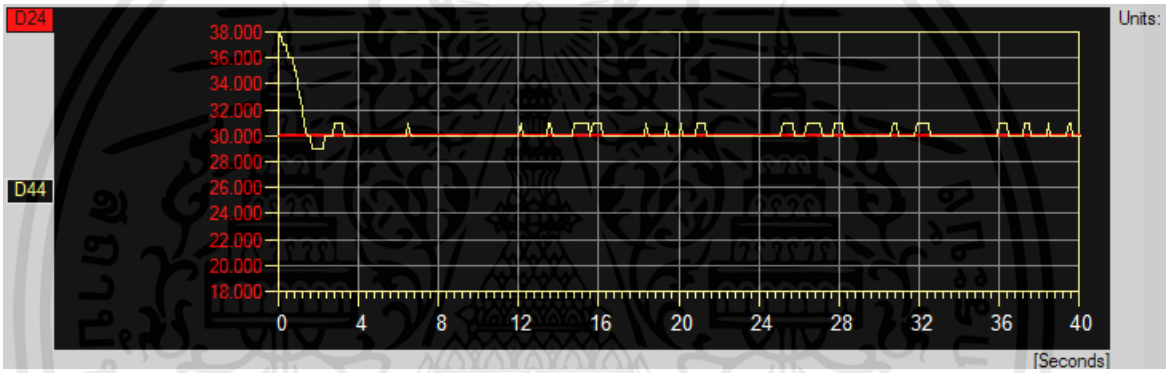
4.4.3 การทดลองผลตอบสนองที่ตำแหน่ง 30 เซนติเมตร



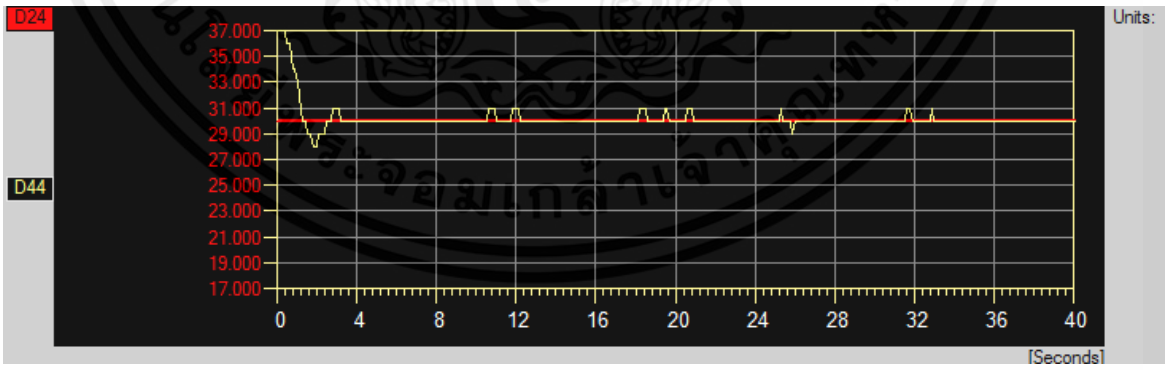
รูปที่ 4.55 ผลตอบสนองของ PID ที่ sv=30 ครั้งที่ 1



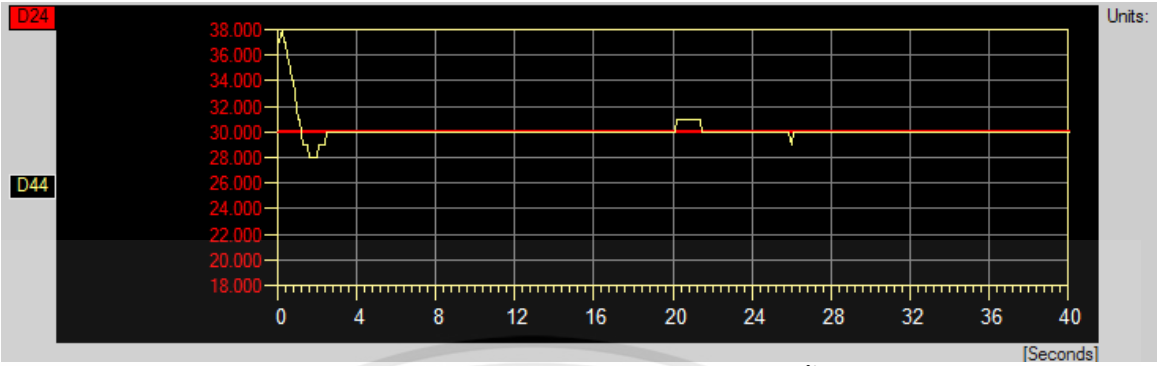
รูปที่ 4.56 ผลตอบสนองของ PID ที่ sv=30 ครั้งที่ 2



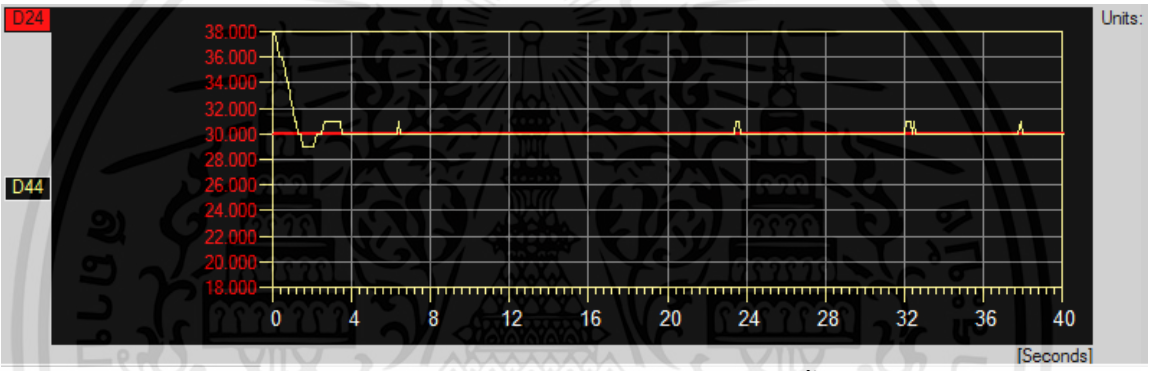
รูปที่ 4.57 ผลตอบสนองของ PID ที่ sv=30 ครั้งที่ 3



รูปที่ 4.58 ผลตอบสนองของ PID ที่ sv=30 ครั้งที่ 4



รูปที่ 4.59 ผลตอบสนองของ PID ที่ sv=30 ครั้งที่ 5



รูปที่ 4.60 ผลตอบสนองของ PID ที่ sv=30 ครั้งที่ 6

ตารางที่ 4.10 ตารางผลการทดลองที่ตำแหน่ง 30 cm

ครั้งที่	เวลาเข้าสู่ Setpoint		Overshoot	
1	3.2 s		3.3 %	
2	3.3 s		3.3 %	
3	3.3 s		3.3 %	
4	3.2 s		6.7 %	
5	2.7 s		6.7 %	
6	3.5 s		3.3 %	
	Avg.	3.27 s	Avg.	5 %
	Max	3.5 s	Max	6.7 %
	Min	2.7 s	Min	3.3 %

สรุปผลการทดลอง

จากการทดลองที่ตำแหน่ง Setpoint 30 เซนติเมตร ค่าเฉลี่ยของเวลาเข้าสู่ Setpoint เท่ากับ 3.27 วินาทีค่ามากที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 3.5 วินาที ค่าน้อยที่สุดของเวลาเข้าสู่ Setpoint เท่ากับ 2.7 วินาทีและ ค่าเฉลี่ย Overshoot เท่ากับ 5% ค่ามากที่สุดของ Overshoot เท่ากับ 6.7% ค่าน้อยที่สุดของ Overshoot เท่ากับ 3.3% และ ผลตอบสนอง Oscillate จาก Setpoint ช่วง ± 1 cm

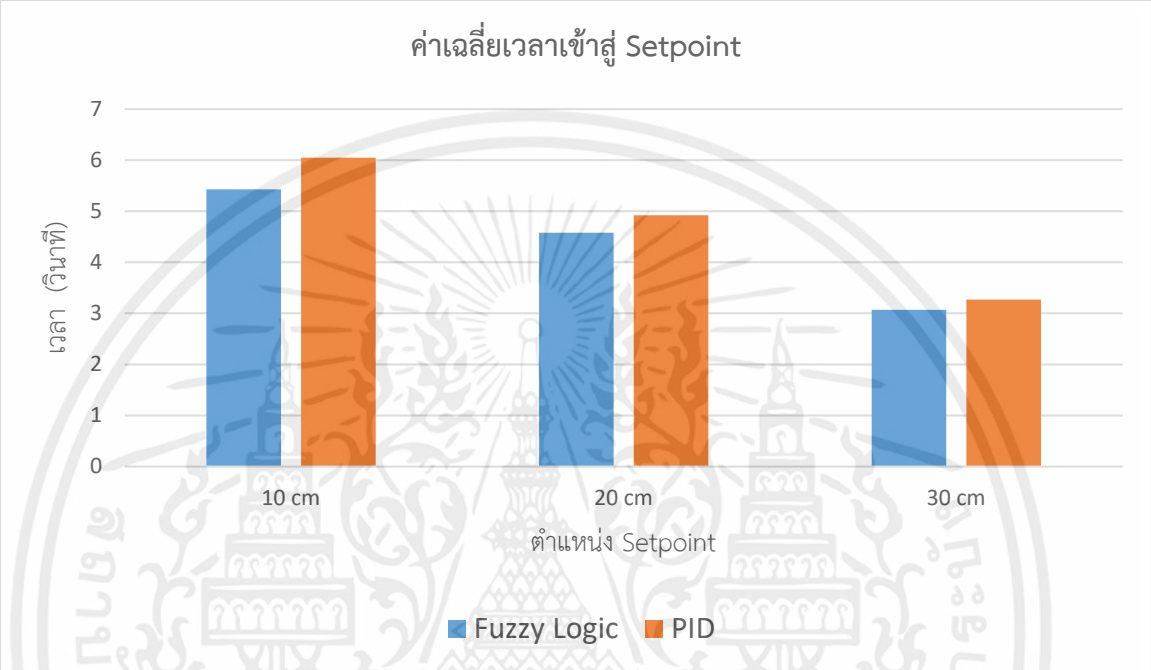
สรุปผลการทดลองทั้งหมด

ที่ตำแหน่ง 10 เซนติเมตร ได้ค่าเฉลี่ยเวลาเข้าสู่ Setpoint = 6.05 วินาที และ ได้ค่าเฉลี่ย Overshoot = 81.7% ที่ตำแหน่ง 20 เซนติเมตร ได้ค่าเฉลี่ยเวลาเข้าสู่ Setpoint = 4.92 วินาที และ ได้ค่าเฉลี่ย Overshoot = 16.7 % ที่ตำแหน่ง 30 เซนติเมตร ได้ค่าเฉลี่ยเวลาเข้าสู่ Setpoint = 3.27 วินาที และ ได้ค่าเฉลี่ย Overshoot = 4.4 % จะเห็นได้ว่าตำแหน่ง Setpoint ยิ่งห่างจากตำแหน่งเริ่มต้น(ที่ตำแหน่ง 37 เซนติเมตร) มากเท่าใด ระยะเวลาที่ใช้ในการเข้าสู่ Setpoint จะมากขึ้นและจะเกิด Overshoot มากขึ้นด้วย

4.5 การเปรียบเทียบการทดลองของทฤษฎีควบคุมฟัซซีลอจิกและพีไอดี ในแต่ละ Setpoint 10, 20 และ 30 เซนติเมตร

จากการทดลองในหัวข้อที่ 4.2 และ 4.3 เป็นการทดลองผลตอบสนองของฟัซซีลอจิก โดยใช้ membership function ของ Error, Differential Error, pulse output, rule based แบบที่ 4 และการทดลองผลตอบสนองโดยใช้ทฤษฎีควบคุมพีไอดี ในหัวข้อนี้เราจะนำทั้งสองทฤษฎีการควบคุมมาเปรียบเทียบกันในแต่ละตำแหน่ง Setpoint 10, 20 และ 30 เซนติเมตร

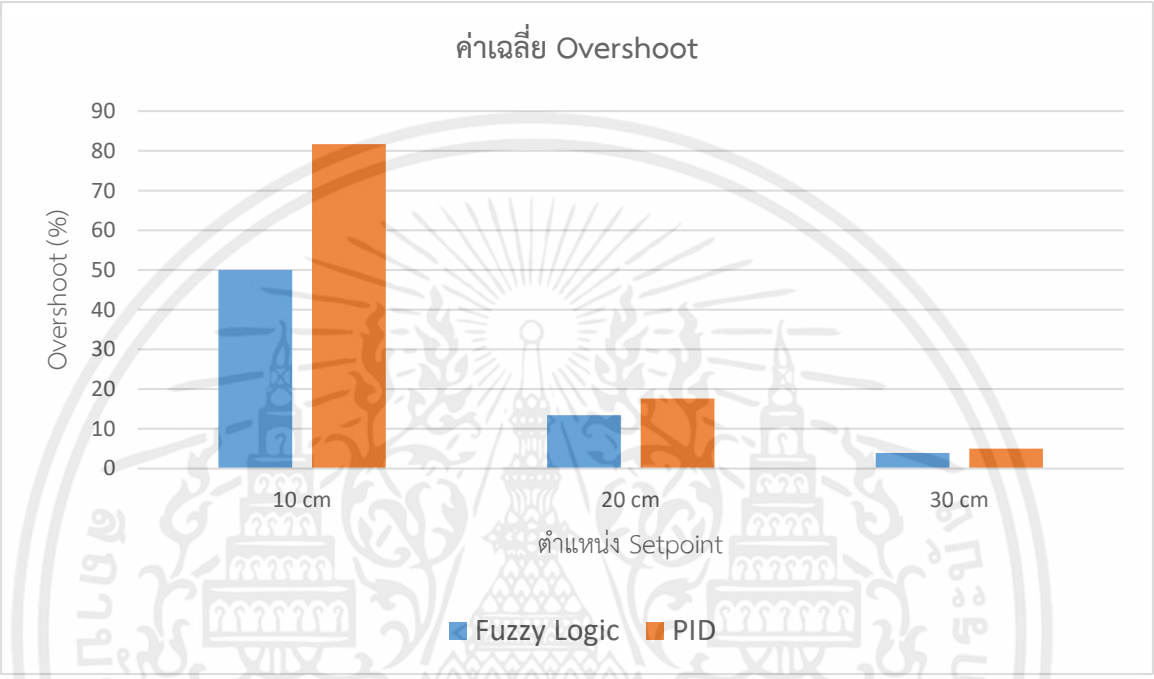
4.5.1 กราฟแสดงความสัมพันธ์ของค่าเฉลี่ยเวลาเข้าสู่ Setpoint กับ ตำแหน่ง 10, 20 และ 30 เซนติเมตร ระหว่างทฤษฎีควบคุมฟัซซีลอจิกและพีไอดี



รูปที่ 4.61 รูปภาพแสดงกราฟค่าเฉลี่ยเวลาเข้าสู่ Setpoint

ทฤษฎีควบคุมแบบฟัซซีลอจิกมีระยะเวลาเฉลี่ยในการเข้าสู่ Setpoint ที่เร็วกว่าทฤษฎีควบคุมแบบพีไอดี ในทุกตำแหน่ง Setpoint 10, 20 และ 30 เซนติเมตร

4.5.2 กราฟแสดงความสัมพันธ์ของค่าเฉลี่ย Overshoot กับ ตำแหน่ง 10, 20 และ 30 เซนติเมตร ระหว่างทฤษฎีควบคุมฟัซซีลอจิกและพีไอดี



รูปที่ 4.62 รูปภาพแสดงกราฟค่าเฉลี่ย Overshoot

ทฤษฎีควบคุมแบบฟัซซีลอจิกมี Overshoot น้อยกว่าทฤษฎีควบคุมแบบพีไอดี ในทุกตำแหน่ง Setpoint 10, 20 และ 30 เซนติเมตร

4.5.3 ตารางสรุปข้อดี-ข้อเสียของทฤษฎีควบคุมแบบฟัซซีลอจิกและพีไอดี

ตารางที่ 4.11 ตารางสรุปข้อดี-ข้อเสียของ Fuzzy Logic และ PID

	Fuzzy Logic Control	PID Control
ระยะเวลาที่ใช้ในการเข้าสู่ Setpoint	เร็วกว่า	ช้ากว่า
Overshoot	น้อยกว่า	มากกว่า
ความนิ่งของระบบ	นิ่งน้อยกว่า	นิ่งมากกว่า

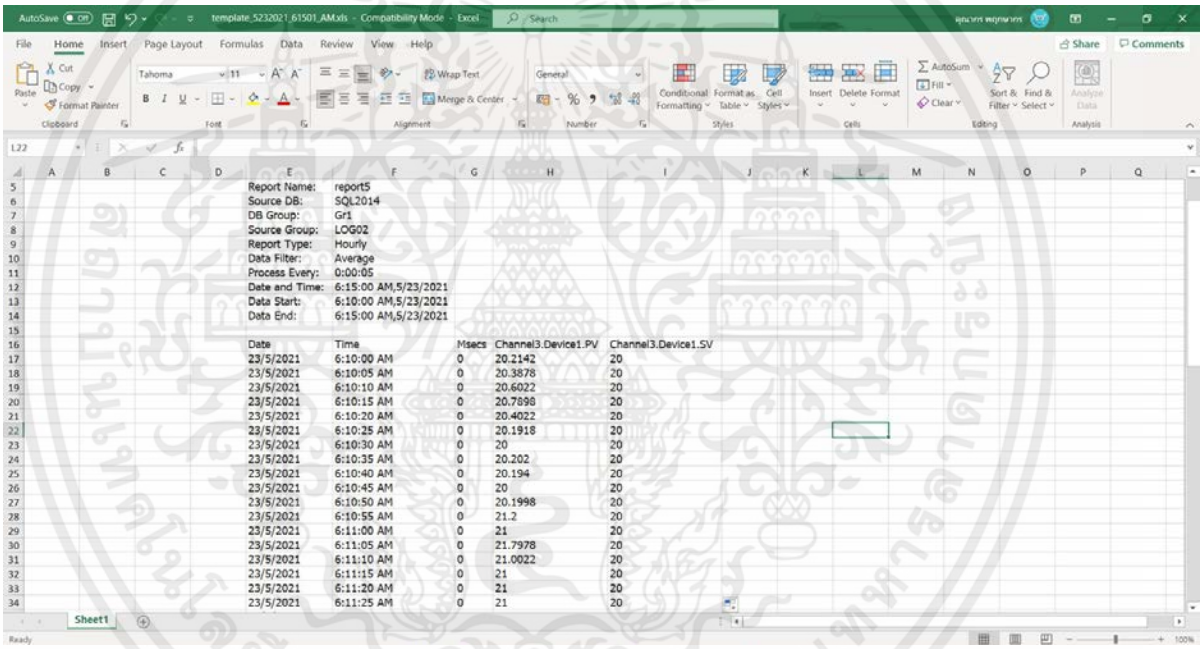
<u>สรุปผลการทดลอง</u>

ทฤษฎีควบคุมแบบฟัซซีลอจิกมีระยะเวลาที่ใช้ในการเข้าสู่ Setpoint เร็วกว่าทฤษฎีพีไอดี และ Overshoot น้อยกว่า แต่ทฤษฎีพีไอดีเมื่อเข้าสู่ Setpoint แล้วจะค่อนข้างนิ่งกว่าทฤษฎีของฟัซซีลอจิก ดังนั้นในตอนแรกที่เป็นการปรับแบบหยาบ ๆ ที่ต้องการความรวดเร็วในการเข้าสู่ Setpoint ควรใช้ทฤษฎีควบคุมฟัซซีลอจิก เพราะเข้าสู่ Setpoint ได้เร็วกว่า แต่เมื่อระบบเข้าสู่ Setpoint แล้ว ควรใช้ทฤษฎีควบคุมแบบพีไอดีในการปรับแบบละเอียด เพราะให้ผลลัพธ์นิ่งกว่า ดังนั้นทฤษฎีทั้งสองมีข้อดี-ข้อเสียในแต่ละด้าน การใช้ข้อดีของทั้งสองทฤษฎีร่วมกันจะให้ผลลัพธ์ที่ดีที่สุด

4.6 ผลลัพธ์การสร้างรายงานจาก Database ด้วย TrendWorX32 Reporting

หลังจากสร้างคอนฟิกกำหนด Collection rate และ Calculation period จาก OPC tags ใน Logging group ที่สร้างไว้ในโปรแกรม TrendWorX32 Configurator เพื่อเก็บข้อมูลไว้ใน SQL Server แล้ว การแสดงรายงานในรูปของไฟล์ .xls สามารถทำได้โดยสร้างคอนฟิกกำหนดว่าจะดึงข้อมูลจากฐานข้อมูลชื่ออะไร, Tag อะไรบ้างใน SQL Server ทุกๆเวลาเท่าไร และให้แสดงรายงานตอนไหน

รายงานการสรุปผลนี้ถูกสร้างขึ้นเมื่อวันที่ 5 พฤษภาคม 2564 โดยการนำ tag PV และ SV ใน SQL Server มาหาค่าเฉลี่ยทุกๆ 5 วินาที แล้วบันทึกลงรายงาน โดยรายงานนี้จะสรุปผลในทุกๆ 5นาที



Report Name:	report5				
Source DB:	SQL2014				
DB Group:	Gr1				
Source Group:	LOG02				
Report Type:	Hourly				
Data Filter:	Average				
Process Every:	0:00:05				
Date and Time:	6:15:00 AM,5/23/2021				
Data Start:	6:10:00 AM,5/23/2021				
Data End:	6:15:00 AM,5/23/2021				
Date	Time	Msecs	Channel3.Device1.PV	Channel3.Device1.SV	
23/5/2021	6:10:00 AM	0	20.2142	20	
23/5/2021	6:10:05 AM	0	20.3878	20	
23/5/2021	6:10:10 AM	0	20.6022	20	
23/5/2021	6:10:15 AM	0	20.7898	20	
23/5/2021	6:10:20 AM	0	20.4022	20	
23/5/2021	6:10:25 AM	0	20.1918	20	
23/5/2021	6:10:30 AM	0	20	20	
23/5/2021	6:10:35 AM	0	20.202	20	
23/5/2021	6:10:40 AM	0	20.194	20	
23/5/2021	6:10:45 AM	0	20	20	
23/5/2021	6:10:50 AM	0	20.1998	20	
23/5/2021	6:10:55 AM	0	21.2	20	
23/5/2021	6:11:00 AM	0	21	20	
23/5/2021	6:11:05 AM	0	21.7978	20	
23/5/2021	6:11:10 AM	0	21.0022	20	
23/5/2021	6:11:15 AM	0	21	20	
23/5/2021	6:11:20 AM	0	21	20	
23/5/2021	6:11:25 AM	0	21	20	

รูปที่ 4.63 รายงานสรุปผลในรูปของไฟล์ xls

บทที่ 5

สรุปผลและวิจารณ์ปริญญานิพนธ์

5.1 สรุปผลการทดลอง

เนื่องจากในโหมดการทำงานแบบอัตโนมัตินั้นใช้ทฤษฎี Fuzzy Logic ในการควบคุม ซึ่งต้องเข้าใจผลตอบสนองของกระบวนการในแต่ละตำแหน่งเป็นอย่างมากในการกำหนด Membership function, Rules based จึงต้องศึกษาการเคลื่อนที่ของวัตถุทรงกลมโดยใช้ joystick ผ่านการควบคุมในโหมดแมนนวลเพื่อหา pule output ที่เหมาะสม ณ ตำแหน่งนั้นๆ โดยในส่วนของการทำงานแบบอัตโนมัติเราจะเริ่มออกแบบตัวควบคุมโดยใช้ทฤษฎี Fuzzy Logic ในการควบคุมตำแหน่งที่ Setpoint 20 cm. ก่อน เนื่องจากเป็นตำแหน่งที่ตรงกับกล้อง Pixy และเป็นจุดศูนย์กลางของคาน ค่าที่อ่านได้จะมีความแม่นยำมากกว่าตำแหน่ง Setpoint 10 และ 30 cm. เนื่องจากเวลาคานกระดกขึ้นลงจะเป็นการเคลื่อนที่ในแกน Z ซึ่งกล้อง Pixy สามารถอ่านตำแหน่งในแกน X และ Y ดังนั้นที่ตำแหน่ง 10 และ 30 cm. จึงมีโอกาส oscillate มาก

ในส่วนของผลตอบสนองในโหมดการทำงานแบบอัตโนมัติที่ตำแหน่ง Setpoint 30 cm. จะมีเวลาเข้าสู่ Steady state และ Overshoot น้อยที่สุด เนื่องจากอยู่ใกล้จุดเริ่มต้น ที่ตำแหน่ง Setpoint 10 cm. จะมีเวลาเข้าสู่ Steady state และ Overshoot มากที่สุด

เมื่อนำการควบคุมที่ใช้ทฤษฎี Fuzzy logic มาเปรียบเทียบกับการควบคุมที่ใช้ทฤษฎี PID (Tp = 1000, Ti = 9999 และ Td = 45) ซึ่งเป็นโปรเจคเก่าที่รุ่นพี่ออกแบบไว้ ผลปรากฏว่าระยะเวลาที่ใช้ในการเข้าสู่ Setpoint และ Overshoot น้อยกว่าทฤษฎี PID แต่ทฤษฎี PID เมื่อเข้าสู่ Setpoint แล้วจะค่อนข้างนิ่งกว่าทฤษฎี Fuzzy logic

5.2 ปัญหาที่พบในระหว่างดำเนินงาน

5.2.1 ใช้เวลาทำความเข้าใจทฤษฎี Fuzzy Logic นานเกินไป

การเขียนโปรแกรมควบคุมตำแหน่งของวัตถุทรงกลมบนคานระนาบเดียวด้วยทฤษฎี Fuzzy Logic ลงใน PLC ซึ่งไม่มีฟังก์ชันบล็อกสำเร็จรูป จึงต้องเขียนด้วยการประยุกต์รวมเอา Ladder และ Structure text มาใช้ ซึ่งต้องใช้เวลาในการศึกษาทฤษฎีให้เข้าใจอย่างถ่องแท้ก่อน จึงจะสามารถสรุปเขียนเป็นโปรแกรมได้ ซึ่งใช้เวลานานกว่าเป้าหมาย

5.2.2 การคิดกฎต่าง ๆ เพื่อสร้าง Rule based

เนื่องจากทฤษฎี Fuzzy Logic เป็นทฤษฎีในการควบคุมที่ยืดหยุ่น ขึ้นอยู่กับความคิดเห็นและความเข้าใจในคุณลักษณะเฉพาะของกระบวนการ ซึ่งแต่ละคนมีความคิดที่แตกต่างกันออกไป ทำให้ใช้เวลาในการถกเถียงเพื่อสรุปกฎที่ดีที่สุดเป็นเวลานาน

5.2.3 การบันทึกผลตอบสนองของระบบ

การบันทึกกราฟผลตอบสนองของทฤษฎี Fuzzy Logic เริ่มบันทึกที่ 3 cm. เป็นจุดเริ่มต้น เนื่องจากต้องนำผลตอบสนองของทฤษฎี Fuzzy Logic เปรียบเทียบกับผลตอบสนองของทฤษฎี PID ซึ่งใช้ตำแหน่ง 37 cm. เป็นจุดเริ่มต้น ทำให้ต้องทำการบันทึกกราฟใหม่ให้จุดเริ่มต้นตรงกัน

5.2.4 ปัญหาอันเนื่องมาจากสถานการณ์การแพร่ระบาดของโรคโควิด-19

การแพร่ระบาดทำให้ต้องย้ายโปรเจคกลับมาทำที่หอพัก ซึ่งมีแสงไม่เพียงพอที่จะทำให้กล้อง Pixy ตรวจจับวัตถุได้ จึงจำเป็นต้องใช้โคมไฟเข้ามาช่วย ทำให้กล้องสามารถตรวจจับวัตถุได้

5.2.5 การติดตั้งโปรแกรม GENNESIS32

เนื่องจาก GENNESIS32 จะให้สิทธิ์การใช้งานกับ USER ที่เป็น Administrator ทำให้ต้องติดตั้งโปรแกรมใหม่

5.3 แนวทางในการพัฒนา

การศึกษาการควบคุมด้วย PLC/Fuzzy Logic ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว แสดงผลจัดเก็บข้อมูลและรายงานผลด้วย SCADA ทำให้เห็นความแตกต่างระหว่างทฤษฎีการควบคุมแบบ Fuzzy Logic และ PID เพื่อเป็นประโยชน์ต่อนักศึกษารุ่นต่อไปในการศึกษาและต่อยอด

บรรณานุกรม

[1] Omron Co., Ltd, “SYSMAC CP Series CP1H CPU Unit OPERATION MANUAL”, ตุลาคม 2557, [ระบบออนไลน์] แหล่งที่มา : https://assets.omron.eu/downloads/manual/en/v2/w450_cp1h_cpu_unit_operation_manual_en.pdf (27 พฤษภาคม 2564).

[2] Omron Co., Ltd, “Function Block/ Structured Text Introduction Guide”, 2557, [ระบบออนไลน์] แหล่งที่มา : http://omrondoc.ru/C/R144-E1-03.pdf (27 พฤษภาคม 2564).

[3] “Hooking up Pixy to a Microcontroller (like an Arduino)”, 18 กันยายน 2563, [ระบบออนไลน์] แหล่งที่มา : https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:hooking_up_pixy_to_a_microcontroller_-28like_an_arduino-29 (27 พฤษภาคม 2564).

[4] Omron Co., Ltd, “Fuzzy Logic Unit C200H-FZ001 - Support Omron”, กันยายน 2542, [ระบบออนไลน์] แหล่งที่มา : https://www.support-omron.fr/telechargements/documentations/2017-07-24%20-%2009-23-13%20-%20653606089/W208-E1-02+C200H-FZ001+Operation_Manual.pdf (27 พฤษภาคม 2564).

[5] National Instruments Corporate Headquarters, “LabVIEW, PID and Fuzzy Logic Toolkit User Manual”, มิถุนายน 2552, [ระบบออนไลน์] แหล่งที่มา : https://www.ni.com/pdf/manuals/372192d.pdf (27 พฤษภาคม 2564).

[6] Professor Hua Li and Professor Madan M. Gupta, “Fuzzy Logic and Intelligent Systems”, 1995, หน้า 97-174

[7] M. Amjad, Kashif M.I., S.S Abdullah, and Z. Shareef, “Fuzzy Logic control of ball and beam system”, July 2010

[8] EDA International Ltd. “Create SCADA system with GENESIS software”, 2017, หน้า 19-164

[9] EDA International Ltd. “Connect KepServerEX with OMRON CP1L, CP1H, CP1E” สิงหาคม 2555, [ระบบออนไลน์] แหล่งที่มา : http://edacloud.dyndns.org/index.php/component/content/article/1005--kepserverex--omron-cp1l-cp1h-cp1e (27 พฤษภาคม 2564).

[10] Aruna Lakmal, “SQL ODBC Connection Failure : SQLState: ‘08001’ SQL Server Error: 2” ธันวาคม 2561, [ระบบออนไลน์] แหล่งที่มา : https://www.techcrumble.net/2017/07/sql-odbc-connection-failure-sqlstate-08001-sql-server-error-2/amp/ (27 พฤษภาคม 2564).