การศึกษา PLC Siemens S7-1200 โดยใช้ Modbus TCP/IP สำหรับกระบวนการควบคุม

Study of PLC Siemens S7-1200 by using Modbus TCP/IP for process control

สรวิชญ์ บุญจั่น

ศุภกรณ์ ศรีราชา

วราวิทย์ ศิริยงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรของปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

Study of PLC Siemens S7-1200 by using Modbus TCP/IP for process control

SORRAWIT BOONJAN
SUPAKORN SRIRACHA
WARAWIT SIRIYONG

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

หัวข้อปริญญานิพนธ์ การศึกษา PLC Siemens S7-1200 โดยใช้ Modbus TCP/IP สำหรับกระบวนการควบคุม
Study of PLC Siemens S7-1200 by using Modbus TCP/IP for process control

นักศึกษาผู้จัดทำ นายสรวิชญ์ บุญจั่น รหัสนักศึกษา 59011362
นายศุภกรณ์ ศรีราชา รหัสนักศึกษา 59011317
นายวราวิทย์ ศิริยงค์ รหัสนักศึกษา 59011193

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.นรินทร์ ธรรมารักษ์วัฒนะ

ปีการศึกษา 2562

บทคัดย่อ

ปริญญานิพนธ์นี้จัดทำขึ้นโดยมีวัตถุประสงค์ เพื่อศึกษาและสร้างกระบวนการเรียนรู้การสื่อสารทางอุตสาหกรรม ชนิด Modbus TCP/IP เนื่องจากกระบวนการทางอุตสาหกรรมนั้นจำเป็นต้องใช้โปรโตคอลจึงได้เริ่มต้นศึกษารายละเอียดของโปรโตคอลชนิดนี้โดยการสร้างแบบจำลองกระบวนการและศึกษาการใช้งานของโปรโตคอลนี้ว่ามีความแตกต่างอย่างไรในการใช้งาน ซึ่งกระบวนการดังกล่าวจำเป็นต้องศึกษาและเลือกใช้ PLC ที่เหมาะสมกับโปรโตคอลชนิดนี้และการนำมาปรับใช้ โดยจะมีการจำลองแพลนต์ปฏิบัติการ เพื่อเขียนกระบวนการทดลอง ให้คนที่ต้องการศึกษาเรื่องนี้สามารถมาศึกษาต่อได้โดยง่าย และมีความเข้าใจการทำงานของโปรโตคอลชนิดนี้และการเลือกใช้ที่สอดคล้องกับความต้องการของงานทางอุตสาหกรรมหากต้องไปทำการติดตั้งหรือดัดแปลงโปรโตคอลตามอุตสาหกรรมต่างๆ เป็นการสร้างเสริมความเข้าใจทำให้สามารถนำความรู้ไปใช้งานจริงได้

Thesis Title	Study of PLC Siemens S7-1200 by using Modbus TCP/IP for process control
Authors	Mr. Sorrawit Boonjan
	Mr. Supakorn Sriracha
	Mr. Warawit Siriyong
Thesis Advisor	Asst. Prof. Narin Tammarugwattana
Year	2019

ABSTRACT

The objectives of the thesis were to study and create industrial communication (Modbus TCP/IP) learning process. Because industrial process require protocol. Therefore need to study the details of the protocol as mentioned by creating process model and studying how these protocol different in work. So this process require to study and choose the right PLC suite to protocol and adapting it, With simulations of plants for write the process to people who want to study about this, bring this thesis to study easier and understand the operation of these protocol. So the thesis have to creating understanding and applying knowledge to work.

กิตติกรรมประกาศ

ทางคณะผู้จัดทำโครงงานขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.นรินทร์ ธรรมารักษ์วัฒนะ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานนี้ อีกทั้งท่านได้สละเวลาอันมีค่าที่ได้ให้คำปรึกษาคำแนะนำต่างๆจนทำให้โครงงานนี้ได้บรรลุวัตถุประสงค์ไปด้วยดี

ขอขอบพระคุณอาจารย์ทุกท่านที่ได้สิทธิประสาทวิชาความรู้ในด้านวิศวกรรม ทำให้สามารถนำความรู้มาประยุกต์ในในโครงงานได้เป็นอย่างดี

ขอขอบพระคุณ บิดามารดา ที่ให้การเลี้ยงดู ดูแลเอาใจใส่ และคอยเป็นกำลังใจให้ตลอดเวลาเสมอมา ตลอดจนพี่ๆ เพื่อนๆ ทุกคนที่ให้ความช่วยเหลือในด้านต่างๆ ตลอดเวลาทำโครงงานฉบับนี้

สุดท้ายนี้ทางผู้จัดทำโครงงานฉบับนี้หวังเป็นอย่างยิ่งว่า โครงงานฉบับนี้จะเป็นแนวทางและเป็นประโยชน์ต่อคนรุ่นหลังต่อไป

สรวิชญ์ บุญจั่น
ศุภกรณ์ ศรีราชา
วราวิทย์ ศิริยงค์

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

กระบวนการสื่อสารทางอุตสาหกรรมเป็นกระบวนการที่สำคัญ และใช้งานอยู่ทั่วไปในโรงงานอุตสาหกรรมต่างๆ เช่น อุตสาหกรรมอาหาร อุตสาหกรรมน้ำมัน อุตสาหกรรมการผลิต อุตสาหกรรมเคมีภัณฑ์และอื่นๆ อีกมากมาย ซึ่งเป็นการนำประโยชน์ของเทคโนโลยีการสื่อสารมาใช้งานในกระบวนการทางอุตสาหกรรมต่างๆ ในระบบของการควบคุมด้วย PLC เช่น การใช้ระบบสื่อสารระหว่าง PLC กับเครื่องมือวัดเพื่อควบคุมระบบ รับ-เก็บ-จ่าย ของโรงงานแก๊ส การใช้ระบบสื่อสารระหว่าง PLC กับคอนโทรลวาล์วในการระบายน้ำ เป็นต้น จึงเล็งเห็นความสำคัญของเทคโนโลยีการสื่อสารนี้ จึงได้ทำการศึกษาโปรโตคอล คือ Modbus TCP/IP เพื่อทำความเข้าใจ และสามารถนำไปใช้เป็นต้นแบบและเขียนกระบวนการทดลองเพื่อใช้ในการศึกษา

1.2 วัตถุประสงค์ของปริญญานิพนธ์

เพื่อศึกษากระบวนการสื่อสารทางอุตสาหกรรม โดยการจำลองกระบวนการทางอุตสาหกรรมผ่านการควบคุมด้วย PLC และเครื่องมือวัดชนิดต่างๆ เพื่อเป็นต้นแบบในการศึกษาและกระบวนการทดลองการใช้งานโปรโตคอลชนิด Modbus TCP/IP และสามารถใช้ในกระบวนการทางอุตสาหกรรมต่างๆได้อย่างเหมาะสม

1.3 เป้าหมายและขอบเขตของโครงงาน

1. ศึกษาการทำงานของโปรโตคอล Modbus TCP/IP
2. ศึกษาการใช้งานโปรแกรม TIA Portal V14
3. ศึกษาการทำงานของ PLC
4. แสดงผลค่าและตัวแปรต่างๆ ในโปรแกรม TIA Portal V14
5. สร้างตัวอย่างแบบจำลองเพื่อแสดงผลของกระบวนการควบคุม

1.4 ขั้นตอนการศึกษา

1. ศึกษาการทำงานของโปรโตคอลชนิดต่างๆ
2. ศึกษาการใช้งานโปรแกรม TIA Portal V14
3. ศึกษาการทำงานของตัวควบคุม PLC

4. เก็บข้อมูลต่างๆที่ได้จากกระบวนการ
5. เขียนกระบวนการและทดลองใช้โปรโตคอลชนิดนี้จำลองการควบคุมกระบวนการขึ้นมา
6. สร้างแบบจำลองเพื่อใช้ประกอบการทดลองให้เห็นภาพโดยง่าย
7. เขียนรายงานเล่มโครงงานและสอบ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้เกี่ยวกับการสื่อสารทางอุตสาหกรรม
2. ได้รับความรู้ในการเขียนโปรแกรมควบคุม PLC
3. ได้รับความรู้เกี่ยวกับการวิเคราะห์ใช้งานโปรโตคอลชนิด Modbus TCP/IP ให้เหมาะกับงานทางอุตสาหกรรมต่างๆ
4. ได้ศึกษาการใช้งานอุปกรณ์ทางด้านวัดคุม เช่น PLC เป็นต้น
5. ผู้ที่ต้องการศึกษาเรื่องนี้สามารถนำชุดปฏิบัติการนี้ไปทดลองและศึกษาต่อได้

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

ในบทนี้จะกล่าวถึงทฤษฎีของตัวเรื่องที่สนใจศึกษานั่นก็คือ Modbus TCP/IP รวมถึงส่วนของอุปกรณ์สำคัญต่างๆ นั่นก็คือ PLC โปรแกรม Tia portal V.14 อุปกรณ์ Arduino แหล่งจ่ายไฟและตัวต้านทาน

2.2 PLC

PLC ย่อมาจากคำว่า "Programmable Logic Controller" เป็นอุปกรณ์ควบคุมอิเล็กทรอนิกส์ที่มีหน่วยความจำในการเก็บโปรแกรมสำหรับควบคุมการทำงานของอุปกรณ์ต่างๆ ที่ต่อกับขั้วเข้าและขั้วออกของมัน PLC นี้ยังมีชื่อเรียกอย่างอื่น เช่น PC ซึ่งยอมาจาก "Programmable Controller" และ SC ซึ่งย่อมาจาก "Sequence Controller" PLC ขนาดเล็กอาจเรียกว่า ซีเควนเซอร์ PLC ถือเป็นอุปกรณ์ควบคุมสำคัญมากตัวหนึ่ง ในการควบคุมเครื่องจักรต่างๆ ในโรงงานอุตสาหกรรมให้ทำงานแบบอัตโนมัติในระบบ FA (Factory Automation) PLC จะถูกใช้ในการเพิ่มประสิทธิภาพการทำงานของเครื่องจักร ทำให้เครื่องจักรสามารถทำงานได้เองโดยอัตโนมัติเป็นการลดภาระหน้าที่ของคนงาน PLC นั้นมีทั้งที่มีขนาดใหญ่หรืออาจเป็นระบบควบคุมสายพานลำเลียงในโรงงานจนกระทั่งถึง PLC ขนาดเล็กซึ่งใช้ในการควบคุมเครื่องจักรแต่ละเครื่อง

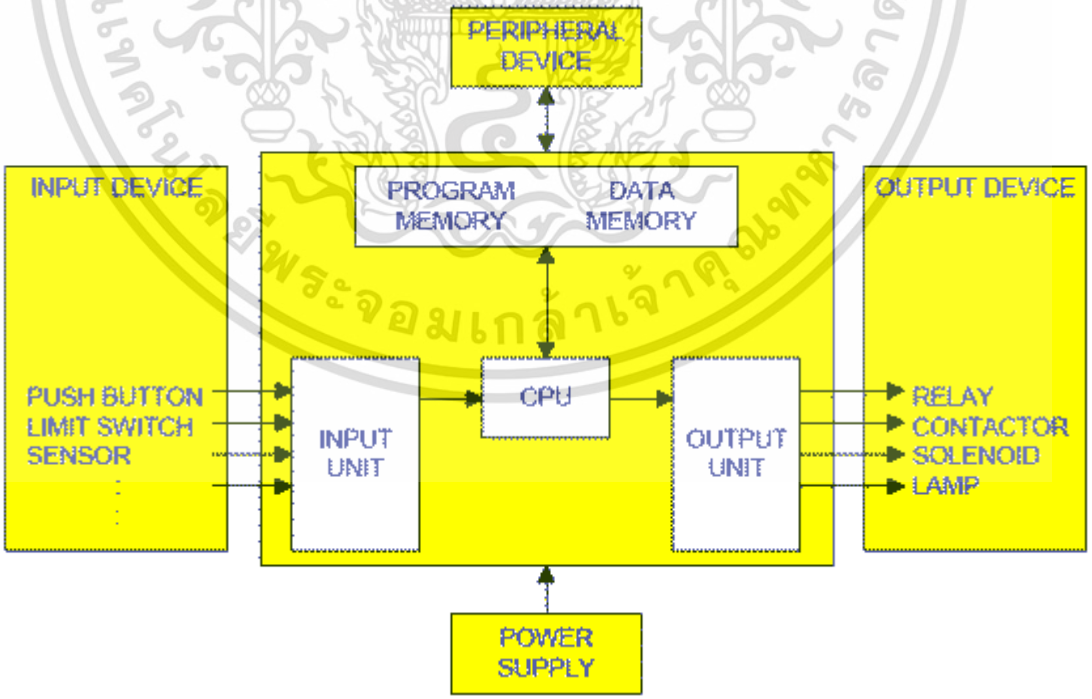
2.3 โครงสร้างของ PLC

โครงสร้างภายในของ PLC แต่ละส่วนนั้นจะประกอบกันทำงานเป็นระบบควบคุมหรือที่เรียกว่า PLC ซึ่งประกอบไปด้วยส่วนสำคัญคือ Unit ทั้ง 5 ส่วน เมื่อประกอบเข้าด้วยกันแล้วก็จะกลายเป็น PLC ชุดหนึ่งที่สามารถทำงานได้แต่ละ Unit จะมีหน้าที่และคุณสมบัติดังนี้

1. CPU (Central Processing Unit)
2. หน่วยความจำ (Memory Unit)
3. ภาคอินพุต (Input Unit)
4. ภาคเอาต์พุต (Output Unit)
5. ภาคแหล่งจ่ายพลังงาน (Power Supply Unit)

รูปที่ 2.1 PLC

2.4 ลักษณะโครงสร้างภายในของ PLC



รูปที่ 2.2 โครงสร้างภายใน PLC

2.4.1 CPU

CPU หรือหน่วยประมวลผลกลาง ทำหน้าที่ทำงานตามคำสั่งของส่วนต่างๆ ตามที่ได้รับมา ผลจากการประมวลผลก็จะถูกส่งออกไปยังส่วนต่างๆ ตามที่ได้ระบุไว้ด้วยคำสั่งนั่นเอง CPU จะใช้เวลาในการประมวลผลช้าหรือเร็วขึ้นอยู่กับการเลือกขนาดของ CPU และความยาวของโปรแกรมที่เขียนด้วย ปกติแล้ว CPU จะใช้ไมโครโปรเซสเซอร์ขนาดตั้งแต่ 4 บิต 8 บิต 16 บิต 32 บิต 64 บิต 128 บิต มาทำงาน โดย CPU ในแต่ละขนาดก็จะมีความสามารถไม่เท่ากันจึงทำให้ PLC ในแต่ละรุ่น แต่ละยี่ห้อนั้นจะมีความสามารถแตกต่างกันนั่นเอง หรือแม้กระทั่งว่าภายใน PLC บางรุ่นจะใช้ไมโครโปรเซสเซอร์มากถึง 2 ตัวมาช่วยกันทำงาน จึงทำให้เวลาประมวลผลก็จะเร็วกว่า PLC ที่ใช้ไมโครโปรเซสเซอร์เพียงแค่ตัวเดียว

โดยปกติแล้วในการเลือกใช้งาน PLC นั้นจะเลือกจากการประยุกต์ใช้งานจึงทำให้ผู้ใช้งาน ไม่รู้ว่าผู้ผลิตใช้ไมโครโปรเซสเซอร์รุ่นหรือเบอร์อะไรในการสร้างเครื่อง PLC เวลาพิจารณาเลือกใช้ PLC ซึ่งไม่มีการระบุเบอร์หรือรุ่นของไมโครโปรเซสเซอร์นั้น ผู้ที่ใช้งานสามารถเลือกจากคุณสมบัติอื่นๆ เช่น จำนวนอินพุต/เอาต์พุต ความเร็วในการประมวลผลของคำสั่ง ขนาดความจุของโปรแกรม และข้อมูล เป็นต้น

2.4.2 หน่วยความจำ (Memory Unit)

หน่วยความจำเป็นอุปกรณ์ที่ใช้เก็บโปรแกรมและข้อมูลต่างๆ ของ PLC กรณีที่ต้องการสั่งให้ PLC ทำงาน โดย PLC จะนำเอาโปรแกรมและข้อมูลในหน่วยความจำมาประมวลผลการทำงาน สำหรับหน่วยความจำที่ใช้งานมีด้วยกัน 2 ชนิด คือ หน่วยความจำชั่วคราว (RAM: Random Access Memory) และหน่วยความจำถาวร (ROM: Read Only Memory)

หน่วยความจำชั่วคราว (RAM: Random Access Memory) โปรแกรมและข้อมูลที่ถูกสร้างขึ้นโดยผู้ใช้ก็จะถูกจัดเก็บในส่วนนี้ คุณสมบัติของหน่วยความจำชั่วคราวนั้นเมื่อไม่มีไฟเลี้ยงจะทำให้โปรแกรมและข้อมูลหายไปทันที ดังนั้นภายใน PLC จะพบว่ามีแบตเตอรี่สำรองข้อมูลเอาไว้สำรองข้อมูลกรณีที่แหล่งจ่ายไฟหลัก ไม่จ่ายไฟให้กับ PLC ข้อควรระวังคือ ไม่ควรที่จะถอดแบตเตอรี่สำรองในกรณีที่ไม่มีไฟจ่ายให้ PLC

หน่วยความจำถาวร (ROM: Read Only Memory) เป็นหน่วยความจำอีกชนิดหนึ่งโดยที่ข้อมูลในหน่วยความจำถาวรนั้นไม่จำเป็นต้องมีแบตเตอรี่สำรองข้อมูล แต่ก็มีปัญหาเรื่องเวลาในการเข้าถึงข้อมูลที่ช้ากว่า หน่วยความจำชั่วคราว จึงปรากฏให้ผู้ใช้เห็นว่า PLC จะมีหน่วยความจำใช้งานทั้งหน่วยความจำชั่วคราวและหน่วยความจำถาวร ร่วมกันอยู่

หน่วยความจำถาวรแบ่งออกเป็น 3 ชนิดดังนี้

-PROM (Programmable ROM) จัดเป็นหน่วยความจำถาวรรุ่นแรกๆ ที่สามารถเขียนข้อมูลลงชิพได้เพียงครั้งเดียว ถ้าเขียนแล้วข้อมูลไม่สมบูรณ์ชิพก็จะเสียทันทีโดยไม่สามารถนำกลับมาเขียนใหม่ได้อีก

-EPROM (Erasable Programmable Read Only Memory) หน่วยความจำชนิดนี้จะต้องใช้เครื่องมือพิเศษในการเขียนโปรแกรม การลบโปรแกรมทำได้โดยใช้แสงอัลตราไวโอเลต หรือตากแดดร้อนๆนานๆ มีข้อดีตรงที่โปรแกรมจะไม่สูญหาย แม้ไฟดับจึงเหมาะกับการใช้งานที่ไม่ต้องการเปลี่ยนโปรแกรม

-EEPROM (Electrical Erasable Programmable Read Only Memory) หน่วยความจำชนิดนี้ไม่ต้องใช้เครื่องมือพิเศษในการเขียน หรือลบโปรแกรม โดยจะใช้วิธีการทางไฟฟ้าเหมือนกับ หน่วยความจำชั่วคราว นอกจากนั้นก็ไม่จำเป็นต้องมีแบตเตอรี่สำรองไฟเมื่อไฟดับ ซึ่ง EEPROM จะรวมคุณสมบัติที่ดีของทั้ง หน่วยความจำชั่วคราว และ EPROM เอาไว้ด้วยกัน

2.4.3 การใช้งานหน่วยความจำใน PLC

-หน่วยความจำชั่วคราวจะใช้เก็บโปรแกรมและข้อมูลที่ทำงานจากการสั่ง RUN/STOP

-หน่วยความจำถาวรจะใช้เก็บซอฟต์แวร์ระบบ และใช้เป็นชุดสำรองโปรแกรมหรือข้อมูลเพื่อป้องกันในกรณีที่โปรแกรมหรือข้อมูลในหน่วยความจำชั่วคราวหายไป ผู้ใช้สามารถที่จะถ่ายโปรแกรมและข้อมูลเข้าไปที่ หน่วยความจำชั่วคราว ใหม่ได้

2.4.4 ภาคอินพุต (Input Unit)

ภาคอินพุตของ PLC จะทำหน้าที่รับสัญญาณอินพุตเข้ามาแล้วแปลงสัญญาณเพื่อที่จะส่งเข้าไปภายใน PLC อุปกรณ์อินพุต (Input Device) ต่างๆ ที่จะนำมาต่อกับภาคอินพุตได้นั้น เช่น Relay, Limit Switch, Inverter, Encoder, Temperature Controller เพื่อส่งไปยัง CPU เพื่อประมวลผลตามโปรแกรมคำสั่งของผู้ใช้ โดยปกติแล้วหน้าที่ของหน่วยอินพุตคือ

-แปลงระดับสัญญาณเข้าไปให้เป็นระดับสัญญาณที่เหมาะสมให้กับระบบการทำงานของ CPU

-แบ่งสัญญาณภายนอกและภายในออกจากกัน เพื่อที่จะต้องการป้องกันไม่ให้หน่วยประมวลผลได้รับความเสียหาย

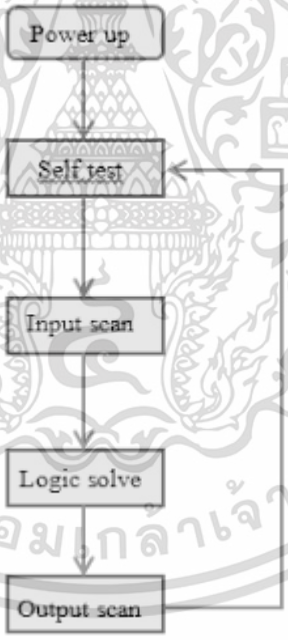
-แก้ปัญหาการสั่นสะเทือนของหน้าสัมผัส

2.4.5 ภาคเอาต์พุต (Output Unit)

ภาคเอาต์พุตของ PLC ทำหน้าที่ส่งสัญญาณออกไปขับโหลดชนิดต่างๆ ตามเงื่อนไขที่ได้เขียนโปรแกรมเอาไว้ ซึ่งหน่วยเอาต์พุตทำหน้าที่รับข้อมูลจากตัวประมวลผลแล้วส่งต่อข้อมูลไปควบคุมอุปกรณ์ภายนอกเช่น ควบคุมหลอดไฟ มอเตอร์ และวาล์ว เป็นต้น

2.4.6 การทำงานของ PLC

PLC ส่วนใหญ่จะมีลำดับการทำงานพื้นฐาน อยู่ 4 ขั้นตอนและจะทำงานซ้ำๆกันหลายครั้งภายในเวลาหนึ่งวินาทีและเมื่อเริ่มต้นจ่ายไฟให้กับ PLC มันจะเริ่มตรวจสอบการทำงานของฮาร์ดแวร์และซอฟต์แวร์เพื่อที่จะหาข้อบกพร่อง แต่ถ้าไม่มีปัญหาใดๆมันจะนำเอาข้อมูลในอินพุต (สัญญาณอินพุตต่างๆ) เข้ามาเก็บไว้ในหน่วยความจำซึ่งจะเรียกว่า สแกนอินพุต หลังจากนั้น PLC ก็จะประมวลผล ตามโปรแกรมแลดเดอร์ โดยจะใช้ข้อมูลจากหน่วยความจำการประมวลผลดังกล่าวนี้เรียกว่า สแกนลอจิก ในขณะที่ PLC ประมวลผลตามโปรแกรมแลดเดอร์นั้นค่าเอาต์พุตของโปรแกรมแลดเดอร์ ก็จะเปลี่ยนแปลงไปตามเงื่อนไขต่างๆของโปรแกรม แต่การเปลี่ยนแปลงนี้จะอยู่ในหน่วยความจำชั่วคราวเท่านั้น เมื่อการสแกนแลดเดอร์ทำงานเสร็จแล้ว ข้อมูลด้านเอาต์พุตในหน่วยความจำชั่วคราวนี้จะถูกส่งไปที่ Unit เอาต์พุตทำให้อุปกรณ์ที่ต่ออยู่ภายนอกทำงานหรือไม่ทำงานตามผลลัพธ์ที่ได้จากการประมวลผล ซึ่งเรียกว่า สแกนเอาต์พุต เมื่อสิ้นสุดการสแกนเอาต์พุต PLC จะกลับไปเริ่มต้นการทำงานใหม่ ซึ่งกระบวนการดังกล่าว นี้จะใช้เวลา 5 –10 mS หรือเร็วกว่าขึ้นอยู่กับความเร็วในการทำงานของ CPU ตามรูปที่ 2.3



รูปที่ 2.3 วงรอบการสแกนของ PLC

2.4.7 การสแกนอินพุตและเอาต์พุต

เมื่ออินพุตต่างๆ ที่ต่อเข้ากับ PLC นั้นถูกสแกน มันจะเก็บค่าหรือสถานะต่างๆ ไว้ในหน่วยความจำและเมื่อเอาต์พุตที่ต่อกับ PLC ถูกสแกนก็จะทำการคัดลอกข้อมูลจากหน่วยความจำส่งออกไปให้เอาต์พุต และเมื่อทำการสแกนแลดเดอร์หรือประมวลผลแลดเดอร์ PLC จะใช้ค่าหรือ

ข้อมูลในหน่วยความจำเท่านั้น โดยไม่สนใจค่าหรือข้อมูลจริงของอินพุตและเอาต์พุตในขณะนั้นในทำนองเดียวกันถ้าเอาต์พุตต้องเปลี่ยนแปลงอยู่ตลอดเวลา เมื่อประมวลผลในแต่ละคำสั่งของโปรแกรมแลดเดอร์แทนที่จะประมวลผลให้จบทั้งโปรแกรม จะทำให้ PLC ทำงานได้ช้ามาก เพราะต้องคัดลอกข้อมูลไปที่ Unit เอาต์พุตทุกครั้งที่ค่าเปลี่ยนแปลง จากการประมวลผลสัญญาณอินพุตที่ดีจะต้องมีคุณสมบัติและหน้าที่ดังนี้

-ทำให้สัญญาณที่เข้าได้ระดับที่เหมาะสมกับ PLC

-การส่งสัญญาณระหว่างอินพุตกับ CPU จะติดต่อกันด้วยลำแสง โดยอาศัยอุปกรณ์ประเภท โฟโตทรานซิสเตอร์เพื่อต้องการจะแยกสัญญาณทางไฟฟ้าให้ออกจากกันเป็นการป้องกันไม่ให้ CPU เสียหายเมื่ออินพุตเกิดลัดวงจร

-หน้าสัมผัสจะต้องไม่สั่นสะเทือน ในส่วนของเอาต์พุตจะทำหน้าที่รับค่าสภาวะที่ได้จากการประมวลผลของ CPU แล้วนำค่าเหล่านี้ไปควบคุมอุปกรณ์ต่างๆ เช่น รีเลย์ หรือหลอดไฟ เป็นต้น นอกจากนั้นแล้ว ยังทำหน้าที่แยกสัญญาณของ CPU ออกจากอุปกรณ์ด้านเอาต์พุต ซึ่งปกติแล้วเอาต์พุตนี้จะสามารถขับโหลดได้ด้วยกระแสไฟฟ้าประมาณ 1–2 แอมแปร์แต่ถ้าโหลดนั้นต้องการกระแสไฟฟ้ามากกว่านี้ จะต้องต่อเข้ากับอุปกรณ์ขับอื่นเพื่อขยายให้รับกระแสไฟฟ้ามากขึ้น เช่น รีเลย์แม็กเนติกคอนแทกเตอร์ เป็นต้น

2.4.8 การแสดงสถานะของ PLC

เนื่องจาก PLC มีข้อจำกัดเรื่องอุปกรณ์อินพุตและเอาต์พุตและที่ด้านหน้าของ PLC จะมีไฟแสดงสถานะที่จำกัดเพียงไม่กี่ดวงเท่านั้น เช่น

- Power : ไฟนี้จะติดตลอดเวลาเมื่อจ่ายไฟให้กับ PLC
- Run : ไฟนี้จะใช้แสดงว่าโปรแกรมกำลังทำงานอยู่หรือไม่
- Error : ไฟนี้จะติดเมื่อ PLC พบวาฮาร์ดแวร์ที่สำคัญหรือโปรแกรมมีข้อบกพร่อง

2.5 หลักการทำงานของ PLC S7-1200

ในแต่ละรอบการสแกนของ PLC นั้นประกอบด้วยการเขียนเอาต์พุต อ่านอินพุต ทำงานตามโปรแกรมที่ได้เขียนไว้ และทำในส่วนของการบำรุงรักษาระบบหรือการประมวลผลพื้นหลัง

ภายใต้เงื่อนไขปกติทั้ง ดิจิตอลและอนาล็อก I/O จะถูกอัพเดทข้อมูลพร้อมๆกันในแต่ละรอบการสแกน โดยอาศัยหน่วยความจำภายในที่เรียกว่า ภาพกระบวนการ ดังนั้น ภาพกระบวนการ จะประกอบด้วยการจับสัญญาณอย่างรวดเร็วของอินพุตและเอาต์พุตของ CPU กับสัญญาณต่างๆ

- CPU จะทำการอ่านอินพุตก่อนที่จะทำการทำโปรแกรมผู้ใช้และเก็บค่าอินพุต เหล่านี้ไว้ในภาพกระบวนการ พื้นที่อินพุต เพื่อให้มั่นใจได้ว่าค่าต่างๆ เหล่านี้จะยังคงค่าเดิมตลอดการทำงานของคำสั่ง

- CPU ทำงานตามลอจิกที่เขียนไว้ในคำสั่งและทำการอัพเดทค่าเอาต์พุตในภาพกระบวนการพื้นที่เอาต์พุต แทนที่จะเขียนค่าลงไปที่ เอาต์พุต จริงๆโดยตรง
- หลังจากที่ทำคำสั่งต่างๆในโปรแกรมผู้ใช้แล้ว CPU จะทำการเขียนค่าเอาต์พุตจากภาพกระบวนการพื้นที่เอาต์พุต ไปยัง เอาต์พุต จริงๆ

OB : (Organization Blocks) เอาไว้ใช้สำหรับเขียนโปรแกรม OB บางตัวเอาไว้ใช้ในหน้าที่พิเศษ เช่น interrupt เป็นต้น แต่ก็สามารถสร้าง OB สำหรับคำสั่งทั่วไปได้ CPU จะเรียกใช้ OB ตามลำดับความสำคัญ โดยลำดับความสำคัญ OB ที่สูงกว่าจะเรียกใช้ก่อนตัวที่ ลำดับความสำคัญต่ำกว่า โดยลำดับความสำคัญที่ต่ำที่สุดคือ 1 และสูงสุดคือ 26 เนื่องจากกระบวนการของ CPU จะทำงานตามเหตการณ์ โดยในแต่ละเหตการณ์ จะไปทำการ trig เพื่อให้ interrupt OB แต่ละตัวทำงาน สามารถระบุ interrupt OB จากเหตการณ์ โดยการสร้างบล็อกหรือตอนทำการตั้งค่าอุปกรณ์หรือด้วยคำสั่ง ATTACH/DETACH

FC & FB (Functions & Function Blocks) : ประกอบด้วยโค้ดโปรแกรมอยู่ภายใน การใช้งาน FB จะไปเชื่อมต่อกับดาต้าบล็อกด้วยเพื่อที่จะใช้ดาต้าบล็อกในการรักษาค่าสถานะต่างๆเอาไว้ระหว่างการทำงาน

ดาต้าบล็อก : เป็นตัวเก็บข้อมูลและสามารถถูกใช้โดยโปรแกรมบล็อกได้

หน่วยความจำโหลด : เป็นการจัดเก็บแบบไม่ลบที่ใช้สำหรับ ผู้ใช้โปรแกรม ข้อมูล และการตั้งค่า เมื่อมีการดาวน์โหลดโปรเจคลง CPU (FC,FB, OB, DB,การตั้งค่าฮาร์ดแวร์,Technology objects) มันจะทำการโหลดลง หน่วยความจำโหลด เป็นอันดับแรก พื้นที่หน่วยความจำโหลดนี้สามารถเป็นได้ทั้ง การ์ดหน่วยความจำ(ถ้ามี) หรือพื้นที่ใน CPU เองก็ได้ (แต่การ์ดหน่วยความจำจะรองรับพื้นที่มากกว่า) เนื่องจากเป็นการจัดเก็บแบบไม่ลบพื้นที่จึงทำให้ไม่สูญหายแม้ไม่มีการจ่ายไฟให้มัน

หน่วยความจำในการทำงาน : เป็นพื้นที่แบบเปลี่ยนแปลงได้สำหรับบางส่วนของตัว ผู้ใช้โปรเจค ขณะที่มีการทำงานของโปรแกรมผู้ใช้ CPU จะทำการคัดลอกค่าเหล่านี้จากหน่วยความจำโหลดไปยัง หน่วยความจำในการทำงาน ในขณะที่กำลังทำงานโปรแกรมผู้ใช้และเนื่องจากเป็นพื้นที่แบบเปลี่ยนแปลงได้ จึงทำให้ไม่สามารถจดจำค่าได้เมื่อไฟดับ

หน่วยความจำแบบเก็บ : เป็น การจัดเก็บแบบไม่ลบ ที่อยู่ในส่วนหนึ่งของหน่วยความจำในการทำงาน โดยจะใช้เพื่อเก็บค่าของผู้ใช้เอาไว้ไม่ให้หายเมื่อไฟดับ โดย CPU จะทำการคืนค่าให้อีกครั้งเมื่อมีการจ่ายไฟมาใหม่

การใช้ SIMATIC การ์ดหน่วยความจำก็เป็นอีกทางเลือกหนึ่งเพื่อเก็บ โปรแกรมผู้ใช้หรือใช้เพื่อถ่ายโอนโปรแกรม เมื่อไรก็ตามที่ใช้การ์ดหน่วยความจำ ตัว CPU จะรันโปรแกรมผ่านทางการ์ดหน่วยความจำแทนหน่วยความจำของ CPU โดยการใช้ SIMATIC การ์ดหน่วยความจำสามารถใช้เพื่อเป็น การ์ดโปรแกรม สำหรับถ่ายโอนการ์ด เพื่อเป็นที่เก็บข้อมูลหรือเพื่ออัพเดทเฟิร์มแวร์ ดังนี้

1. ใช้ถ่ายโอนโปรแกรมเพื่อคัดลอกโปรเจคไปยัง CPU หลายๆ ตัวโดยไม่ต้องใช้ซอฟต์แวร์ STEP7 โดยถ่ายโอนการ์ดจะคัดลอกโปรเจคจากการ์ดลงหน่วยความจำของ CPU และต้องเอาการ์ดออกหลังจากคัดลอกโปรแกรมลง CPU แล้ว
2. ใช้เป็นโปรแกรมการ์ดเพื่อแทนหน่วยความจำของ CPU โดยฟังก์ชั่นของ CPU ทั้งหมดจะถูกควบคุมผ่านทางโปรแกรม การ์ดแทน โดยการใส่โปรแกรมการ์ดเข้าไปนั้นจะส่งผลให้ภายในหน่วยความจำโหลด ของ CPU ทั้งหมดถูกลบไป (รวมถึงโปรแกรมผู้ใช้) นั่นคือ CPU สั่งงาน โปรแกรมผู้ใช้ผ่านทางโปรแกรมการ์ดแทน
3. ใช้เพื่อเก็บไฟล์ดาต้าล็อกเพราะโปรแกรมการ์ด มีพื้นที่มากกว่าพื้นที่ของ CPU มาก และสามารถใช้ฟังก์ชั่นเว็บเซิร์ฟเวอร์เพื่อทำการโหลดไฟล์ดาต้าล็อกมายังคอมพิวเตอร์ของได้เลยโดยไม่ต้องถอดการ์ดออกมาและยังสามารถใช้การ์ดหน่วยความจำเพื่อทำการอัพเดทเฟิร์มแวร์ได้ด้วย
4. สิ่งสำคัญที่ควรระวังคือ แม้ว่า การ์ดหน่วยความจำของทาง Siemens นี้จะมีรูปร่างเหมือนกับ SD การ์ดทั่วไป แต่ไม่สามารถนำ SD การ์ดทั่วไปมาใช้แทนได้ จำเป็นต้องใช้การ์ดหน่วยความจำของ Siemens เท่านั้น และไม่ควรทำการ format การ์ดหน่วยความจำนี้ด้วยคอมพิวเตอร์ เพราะข้อมูลภายในถูกจัดเรียงมาให้เป็นรูปแบบเฉพาะ การ format อาจทำให้ไม่สามารถใช้การ์ดนี้กับ PLC ได้อีกเลย

ประเภทของข้อมูล

1. Bool คือข้อมูลแบบ บิต
2. Byte คือข้อมูลแบบ 8 บิต
3. Word คือข้อมูลแบบ 16 บิต
4. DWord คือข้อมูลแบบ 32 บิต
5. Integer
   - 5.1. Short Integer
     - 5.1.1.USInt (unsigned 8-บิต integer)
     - 5.1.2.SInt (signed 8-บิต integer)
   - 5.2. Integer
     - 5.2.1.UInt (unsigned 16-บิต integer)
     - 5.2.2.Int (signed 16-บิต integer)
   - 5.3. Double Integer
     - 5.3.1.UDInt (unsigned 32-บิต integer)
     - 5.3.2.DInt (signed 32-บิต integer)
6. Real
   - 6.1. Real คือข้อมูล 32-บิต Real number หรือ Floating-point value

6.2. LReal คือข้อมูล 64-บิต Real number หรือ Floating-point value

7. Date and Time

7.1. Date คือข้อมูล 16-บิต ดาต้า(คล้ายกับ UInt)ที่มีข้อมูลจำนวนของวันตั้งแต่วันที่ 1 มกราคม 1990 โดยข้อมูลที่มากที่สุดคือ 65378 (16#FF62) ซึ่งก็คือ 31 ธันวาคม 2168

7.2. DTL (date and time long) คือข้อมูลจำนวน 12 ไบต์ ที่เป็นข้อมูลของวันที่และเวลา เก็บในรูปของโครงสร้างที่กำหนดไว้ล่วงหน้า

7.2.1. ปี (UInt): 1970 to 2554

7.2.2. เดือน (USInt): 1 to 12

7.2.3. วัน (USInt): 1 to 31

7.2.4. สัปดาห์ (USInt): 1 (Sunday) to 7 (Saturday)

7.2.5. ชั่วโมง (USInt): 0 to 23

7.2.6. นาที (USInt): 0 to 59

7.2.7. วินาที (USInt): 0 to 59

7.2.8. นาโนวินาที (UDInt): 0 to 999,999,999

7.3. Time คือข้อมูล 32-บิต IEC time value (คล้ายกับ DInt) ที่เก็บจำนวนของมิลลิวินาที จาก 0 ถึง 24 วัน 20 ชั่วโมง 31 นาที 23 วินาที 647 มิลลิวินาที) ข้อมูล Time Value นี้สามารถนำไปใช้คำนวณได้และสามารถเป็นค่าลบได้ด้วย

7.4. TOD (time of day) คือข้อมูล 32-บิต (คล้ายกับ DInt) มีที่จำนวนของ มิลลิวินาที ตั้งแต่เที่ยงคืนเป็นต้นไป (0 ถึง 86,399,999)

8. Character and String

8.1. Char คือ 8-บิต single character

8.2. String คือ ข้อมูลตัวอักษรที่แปรผันความยาวได้ถึง 254 ตัวอักษร (กรณีใช้สัญลักษณ์พิเศษ จะใช้สัญลักษณ์ $ นำหน้า เช่น $L คือ Line feed, $R คือ Carriage return, $N คือ Line break หรือก็คือ RL นั่นเอง, $T คือ tab, $P คือ Page feed, $$ คือ $ และ $’ คือ ‘ เป็นต้น)

9. Array and Structure

9.1. Array คือกลุ่มของข้อมูลชนิดเดียวกันหลายๆตัว โดย array สามารถสร้างในบล็อกเครื่องมือแก้ไขอินเตอร์เฟสสำหรับ OB, FC, FB และ DB ไม่สามารถสร้าง array ใน PLC แท็กได้

9.2. Struct คือกลุ่มของข้อมูลหลายๆชนิดที่เอามารวมกลุ่มกัน ดังนั้นจึงใช้ข้อมูลประเภท Struct เพื่อจัดการกับกลุ่มของข้อมูลเสมือนเป็น single ดาต้า unit ได้

ต้องทำการ declare ชื่อและชนิดของข้อมูลของ Struct ดาต้า type ใน ดาต้าบล็อก editor หรือ บล็อก interface editor

9.3. Array และ structure สามารถนำมารวมกันเป็นข้อมูลที่ใหญ่ขึ้นได้ โดยสามารถสร้าง nest ใน struct ได้มากสุดถึง 8 level

10. PLC ประเภทข้อมูล คือ ผู้ใช้กำหนดข้อมูลโครงสร้างที่สร้างขึ้นมาเองมีประโยชน์ในกรณีที่ต้องมีการนำมาใช้ต่ออีกหลายๆครั้ง

11. Pointer

11.1 Pointer คือ การอ้างอิงทางอ้อมให้กับที่อยู่ของแท็กซึ่งตัวมันกินพื้นที่ 6 ไบต์(48 บิต)ในหน่วยความจำ และสามารถรวมข้อมูลดังนี้ในตัวแปรได้ คือ DB (หรือ 0 ถ้าไม่ได้เก็บใน DB) หน่วยความจำพื้นที่ใน CPU และหน่วยความจำที่อยู่

11.2 Any คือ การอ้างอิงทางอ้อมของข้อมูลพื้นที่ และระบุความยาวให้ด้วย point แบบ Any จะกินพื้นที่ 10 ไบต์ ในหน่วยความจำ

11.3 Variant คือ การอ้างอิงทางอ้อมข้อมูล pointer ชนิด variant นี้ไม่ได้กินพื้นที่ของหน่วยความจำ เพราะ variant pointer จะรู้แค่โครงสร้างและองค์ประกอบโครงสร้างของตัวเอง

ถึงแม้ว่าข้อมูลชนิด BCD ตามรูปแบบข้างล่างนี้ไม่ได้กล่าวไว้ในประเภทของข้อมูลแต่ก็สามารถใช้งานได้โดยการใช้คำสั่งการแปลง

- BCD16 คือ BCD แบบ 16-บิต (-999 ถึง 999)
- BCD32 คือ BCD แบบ 32-บิต (-9,999,999 ถึง 9,999,999).

2.5.1 ที่อยู่พื้นที่หน่วยความจำ

1. หน่วยความจำโกลบอล : ได้แก่ อินพุต(I) เอาต์พุต(Q) และหน่วยความจำบิต(M) ซึ่งหน่วยความจำเหล่านี้สามารถเข้าถึงได้จากโค้ดบล็อกทุกตัวไม่มีข้อจำกัด
2. ดาต้าบล็อก (DB) : สามารถใส่ DB เอาไว้ในโปรแกรมผู้ใช้เพื่อเก็บข้อมูลสำหรับโค้ดบล็อก โดยข้อมูลที่เก็บไว้จะยังคงอยู่แม้ว่าการทำงานของแต่ละโค้ดบล็อกจะจบลงไปแล้วก็ตาม
   2.1. โกลบอล DB จะเก็บข้อมูลที่สามารถใช้ได้จากทุกโค้ดบล็อก ในขณะที่อินสแตนซ์ DB จะเก็บข้อมูลสำหรับ FB นั้นๆ
3. หน่วยความจำชั่วคราว : เมื่อไหร่ก็ตามที่โค้ดบล็อกถูกเรียก OS ของ CPU จะทำการจัดสรรหน่วยความจำชั่วคราวหรือหน่วยความจำโลคอล ระหว่างที่บล็อกทำงานและเมื่อโค้ดบล็อกทำงานเสร็จ CPU ก็จะจัดสรรหน่วยความจำโลคอลอีกครั้ง เพื่อนำไปใช้ต่อใน โค้ดบล็อกอื่นๆ

Max. size and number (regardless of the main memory size)		S7 -300/400	S7-1200	S7-1500
DB	Max. size	64 kB	64 kB	64 kB (non-optimized) **10 MB** (optimized CPU1518)
	Max. number	16.000	65.535	65.535
FC/FB	Max. size	64 kB	64 kB	512 kB **3 MB** (optimized CPU1518)
	Max. number	7.999	65.535	65.535
FC / FB / DB	Max. number	4.096 (CPU319) 6.000 (CPU412)	1.024	10.000 (CPU1518)

รูปที่ 2.4 เปรียบเทียบขนาดดาต้าบล็อก

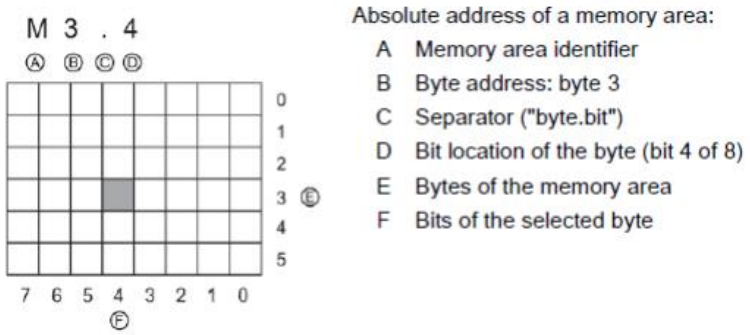
ขนาดของ DB ใน PLC แต่ละรุ่นในแต่ละตำแหน่งหน่วยความจำมีที่อยู่เป็นของตัวเองเฉพาะตัว เช่น I Q หรือ M เป็นต้น ซึ่งในโปรแกรมผู้ใช้ ก็จะใช้ที่อยู่เหล่านี้ในการอ้างตำแหน่งหน่วยความจำด้วยเช่นกัน การอ้างอิงที่อยู่ อินพุต(I)หรือเอาต์พุต(Q) เช่น %I0.3 หรือ %Q1.7 จะเป็นการเข้าถึงตัวภาพกระบวนการ แต่ถ้าหากต้องการเข้าถึงอินพุตหรือเอาต์พุต โดยตรงแบบทันที ให้เพิ่ม “:P” เข้าไปต่อท้ายตัวแปรหรือแท็กที่ต้องการ (เช่น %I0.3:P, %Q1.7:P, หรือ “Stop:P”) ซึ่งคือการทำการสั่ง อินพุต เอาต์พุต นั่นเอง สำหรับรายละเอียดเพิ่มเติม สามารถดูได้ในลำดับต่อไป

Memory area	Description	Force	Retentive
I Process image input	Copied from physical inputs at the beginning of the scan cycle	No	No
I_:P[1] (Physical input)	Immediate read of the physical input points on the CPU, SB, and SM	Yes	No
Q Process image output	Copied to physical outputs at the beginning of the scan cycle	No	No
Q_:P[1] (Physical output)	Immediate write to the physical output points on the CPU, SB, and SM	Yes	No
M Bit memory	Control and data memory	No	Yes (optional)
L Temp memory	Temporary data for a block local to that block	No	No
DB Data block	Data memory and also parameter memory for FBs	No	Yes (optional)

[1] To immediately access (or to force) the physical inputs and physical outputs, append a ":P" to the address or tag (such as I0.3:P, Q1.7:P, or "Stop:P").

รูปที่ 2.5 อธิบายพื้นที่หน่วยความจำ

ตัวอย่างการอ้างอิงที่อยู่แบบบิต คือจะใส่ข้อมูลดังนี้คือ พื้นที่หน่วยความจำ(M) Byte(3) ตามด้วยจุด (.) และสุดท้ายคือบิตที่อยู่ (4) ตัวอย่างข้างล่างคือการใช้งานบิตที่4 ของ M3 นั่นเอง



รูปที่ 2.6 ที่อยู่พื้นที่หน่วยความจำ

การกำหนดค่า I/O ใน CPU และ I/O โมดูล เมื่อทำการเพิ่มตัว CPU หรือโมดูลต่างๆลงไปในซอฟต์แวร์ตัวโปรแกรมจะทำการระบุที่อยู่ ในแต่ละส่วนให้โดยอัตโนมัติ ซึ่งสามารถเปลี่ยน ที่อยู่เองก็ได้เช่นกัน ตัวอย่างเช่น

- STEP7 จะกำหนดดิจิตอลอินพุตและเอาต์พุต กลุ่มละ 8 บิต (1 ไบต์) ไม่ว่าโมดูลนั้นจะใช้ครบทุกบิตหรือไม่ก็ตาม
- STEP7 จะระบุตำแหน่งของอนาล็อกอินพุตและเอาต์พุต กลุ่มละ 2 โดย อนาล็อกแต่ละกลุ่มกินพื้นที่ตัวละ 2 ไบต์ (16 บิต) จากรูปข้างล่าง 64..67 คือ 64-65 และ 66-67

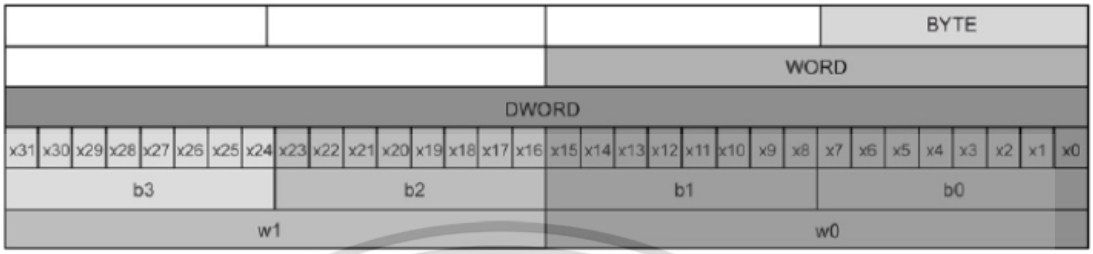
รูปที่ 2.7 ตัวอย่างของ CPU1214C

รูปข้างบนแสดงตัวอย่างของ CPU1214C ซึ่งมือ โมดูล 2SM และ 1 SB ซึ่งจากตัวอย่างนี้สามารถเปลี่ยนที่อยู่ของ DI8x24VDC จาก 8 เป็น 2 ก็ได้ เพราะที่อยู่ 2 ยังไม่มีการใช้งาน โดยตัวซอฟต์แวร์จะช่วยบอกให้ทราบว่าพื้นที่ที่จะเปลี่ยนนั้นระบุผิดขนาดหรือไม่ทับซ้อนกับตัวอื่นหรือไม่

PLC แท็ก หรือ ดาต้าบล็อกแท็กนั้น สามารถเข้าถึงแบบ slice หรือเข้าถึงในบางส่วนในรูปแบบ บิต ไบต์ หรือ word ก็ได้ขึ้นอยู่กับขนาดของตัวมันเอง โดยมีรูปแบบดังนี้

- “<PLC tag name>”.%Xn (bit access)
- “<PLC tag name>”.%Bn (byte access)
- “<PLC tag name>”.%Wn (word access)

- “<Data block name>”.<tag name>.%Xn (bit access)
- “<Data block name>”.<tag name>.%Bn (byte access)
- “<Data block name>”.<tag name>.%Wn (word access)

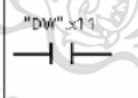


รูปที่ 2.8 ตัวอย่างข้อมูลแบบ slice

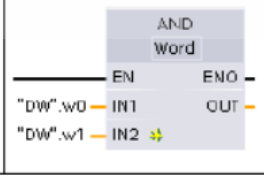
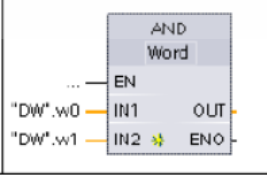
- ชนิดข้อมูลที่ใช้งานแบบ slice ได้คือ Byte, Char, Conn_Any, Date, DInt, DWord, Event_Any, Event_Att, Hw_Any, Hw_Device, HW_Interface, Hw_Io, Hw_Pwm, Hw_SubModule, Int, OB_Any, OB_Att, OB_Cyclic, OB_Delay, OB_WHINT, OB_PCYCLE, OB_STARTUP, OB_TIMEERROR, OB_Tod, Port, Rtm, SInt, Time, Time_Of_Day, UDInt, UInt, USInt และ Word.
- PLC แท็กที่เป็นชนิด Real สามารถเข้าถึงแบบ slice ได้ แต่ดาต้าบล็อกแท็กแบบ Real จะไม่สามารถทำได้

ตัวอย่างการใช้งานข้อมูลแบบ slice โดยสมติว่า “DW” ถูกตั้งเป็นแท็กแบบ DWORD

	LAD	FBD	SCL

Bit access			`IF "DW".x11 THEN`
			`...`
			`END_IF;`

Byte access			`IF "DW".b2 = "DW".b3`
			`THEN`
			`...`
			`END_IF;`

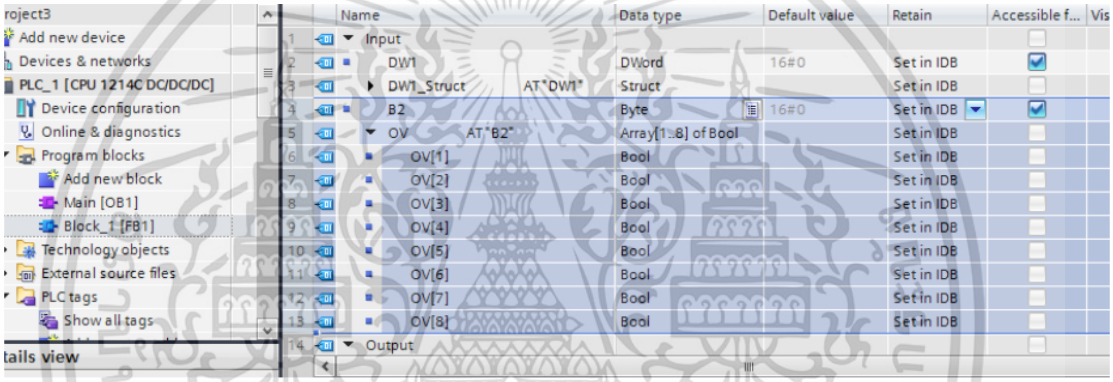
Word access			`out:= "DW".w0 AND` `"DW".w1;`

รูปที่ 2.9 ตัวอย่างการใช้ข้อมูลแบบ slice

การเข้าถึงแท็กด้วยการซ้อนทับ ATAT overlay นอกเหนือจากการเข้าถึงข้อมูลแบบ slice แล้ว ยังมีวิธีการเข้าถึงบางส่วนของแท็กผ่านทาง AT overlay อีกด้วยโดย AT overlay คือ แท็กที่อนุญาตให้เข้าถึงข้อมูลผ่านแท็กที่ได้ประกาศไว้แล้วของตัวบล็อกการเข้าถึงพื้นฐาน ซึ่งข้อดีของการใช้งาน AT overlay คือสามารถประกาศประเภทข้อมูลที่แตกต่างกันได้

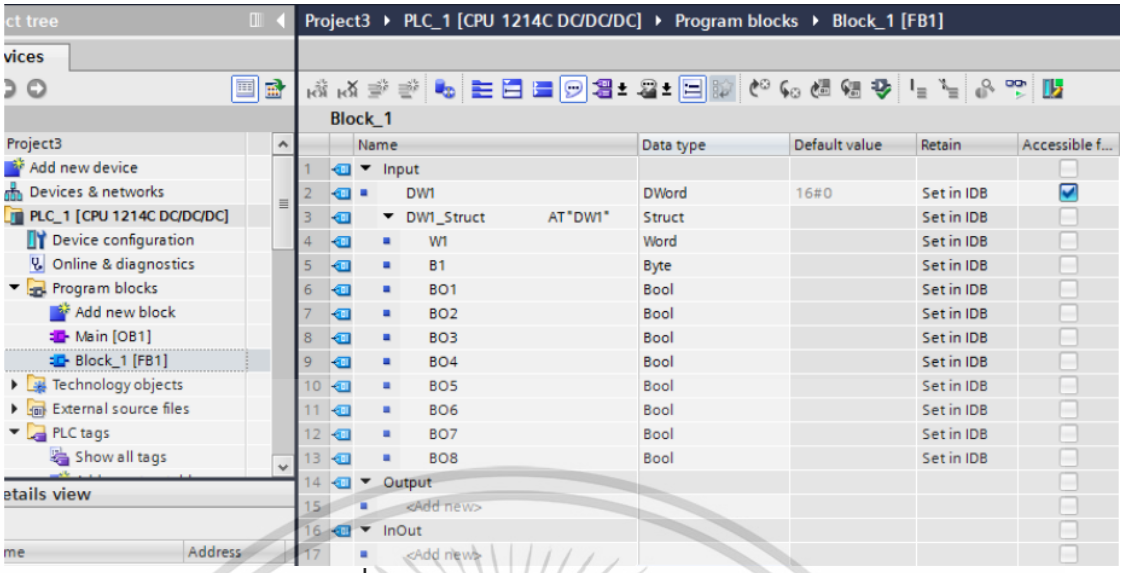
การตั้ง overlay พารามิเตอร์นั้น ให้ประกาศพารามิเตอร์เพิ่มเติมหลังจากตัวแปรหลักที่ต้องการ แล้วใส่ชนิดของประเภทข้อมูลเป็น AT ซึ่งจะทำให้โปรแกรมสร้าง overlay ให้ จากนั้นสามารถกำหนดชนิดข้อมูลของ overlay เป็นข้อมูลย่อยต่อไปได้ หรือจะกำหนดเป็น struc หรือ array ก็ได้ เช่นตัวอย่างข้างล่างนี้

ตัวอย่างของการกำหนด B1 เป็นไบต์และตัวแปรย่อย overlay ชื่อว่า OV กำหนดเป็น array ของ Bool 8 ตัว



รูปที่ 2.10 ตัวอย่างการกำหนดตัวแปร

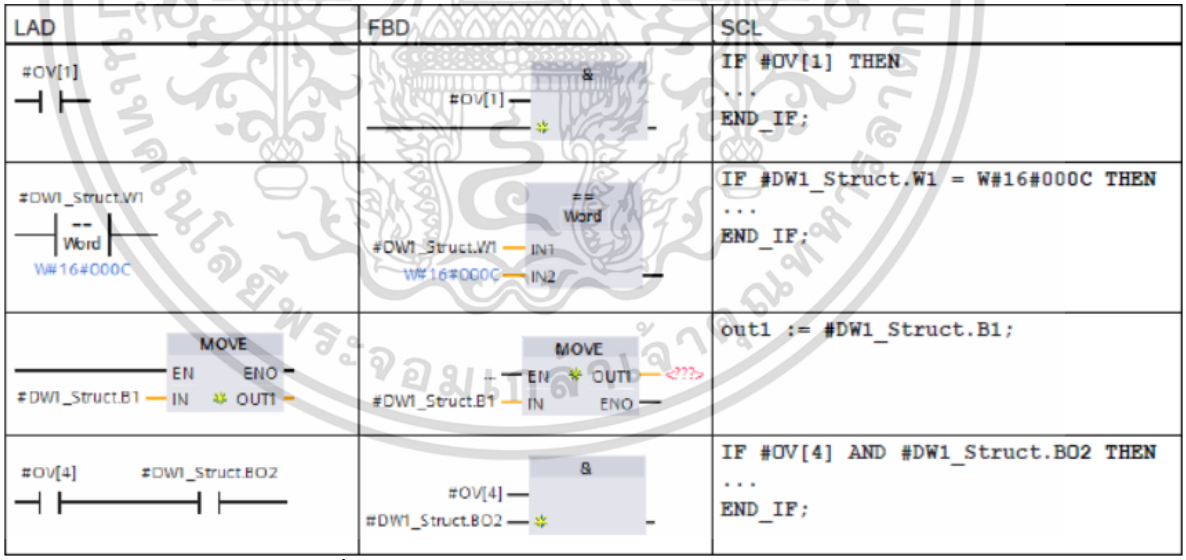
ตัวอย่างของการกำหนด DW1 เป็น DWord และตัวแปรย่อย overlay ชื่อว่า DW1_Struct ที่มีชนิดข้อมูลเป็น Struct ทำให้สามารถกำหนดตัวแปรย่อยใน Struct ได้อีกคือ W1 เป็น Word, B1 เป็น Byte และ BO1-BO8 เป็น Bool ทั้งหมด 8 ตัว ทำให้ชนิดข้อมูลครบ 32 บิต ตามชนิดของ DWord ซึ่งเป็นตัวแปรใหญ่นั่นเอง



รูปที่2.11 ตัวอย่างการกำหนดตัวแปร (ต่อ)

ในกรณีของตัวอย่างข้างบนนี้ DW1 เป็นข้อมูลแบบ 32 บิต ถ้าหากกำหนดตามตัวอย่างข้างต้น แล้วจะทำให้ตัวแปรใน overlay ครบ 32 บิต ตามข้อมูลใน DW1 แต่หากกำหนดข้อมูลใน overlay มากกว่าตัวแปร DW1 เช่น ถ้าเพิ่ม BO9 เข้าไปอีกตัว ก็จะ compile โปรแกรมไม่ผ่านนั่นเอง

ตัวอย่างการใช้งานตัวแปร overlay ในแบบต่างๆ ตามที่ได้ตั้งไว้ข้างต้น



รูปที่ 2.12 ตัวอย่างการใช้งานตัวแปร overlay

กฎของการใช้งาน overlay

1. Overlay แท็กสามารถใช้งานได้ใน FB และ FC แบบ standard access เท่านั้น
2. ไม่สามารถทำ overlay ของข้อมูลประเภท Variant ได้
3. ขนาดของ overlay ต้องน้อยกว่าหรือเท่ากับตัวแปรหลัก (สังเกตว่าน้อยกว่าได้ แต่มากกว่าไม่ได้)
4. ต้องกำหนด AT หลังจากการสร้างตัวแปรนั้นทันที

Pulse เอาต์พุตตัว CPU หรือสัญญาณ (SB) สามารถตั้งค่าให้เป็นตัวจ่าย Pulse เพื่อใช้งาน Pulse ความเร็วสูง เอาต์พุต ฟังก์ชั่น ได้ถึง 4 ช่อง โดยสามาถตั้งค่าเป็น PWM (Pulse-width modulation) หรือเป็น PTO (Pulse-train output) ก็ได้

การใช้งานการเคลื่อนไหวแบบพื้นฐานมักจะใช้ PTO ซึ่งสามารถตั้งค่าแต่ละตัวจ่าย Pulse เพื่อเลือกใช้ PWM หรือ PTO ก็ได้ แต่ไม่สามารถใช้งานทั้งคู่ได้พร้อมกัน Pulse เอาต์พุต ไม่สามารถถูกใช้งานจากคำสั่งอื่นๆ ในโปรแกรมผู้ใช้ได้เมื่อทำการตั้งค่าเอาต์พุตของ CPU หรือ SB ให้เป็นตัวจ่าย Pulse แล้วเอาต์พุตที่อยู่ นั้นๆจะถูกดึงออกจากการใช้งานหน่วยความจำ Q และไม่สามารถใช้เพื่อวัตถุประสงค์อื่นในโปรแกรมผู้ใช้ได้ เช่น หากทำการเขียนค่าเอาต์พุตที่ถูกกำหนดค่าเป็นตัวจ่าย Pulse แล้ว ตัว CPU จะไม่เขียนค่าให้เอาต์พุตอยู่ดี

อย่าจ่ายค่าความถี่ Pulse เกินค่าสูงสุด ค่าความถี่ Pulse สูงสุด ของ ตัวจ่าย Pulse คือ 1 MHz สำหรับ CPU 1217C และมีค่า 100 kHz สำหรับ CPUs 1211C, 1212C, 1214C,และ 1215Cมีค่า 20 kHz สำหรับ SB พื้นฐานหรือ 200 kHz สำหรับ SB ความเร็วสูง เป็นต้น

ตัวจ่าย Pulse ทั้ง 4 จะมี คำสั่ง I/O ที่เป็นค่าเริ่มต้นอยู่ อย่างไรก็ตามสามารถตั้งค่าให้เป็นดิจิตอลเอาต์พุตตัวใดก็ได้ใน CPU หรือ SB

ไม่สามารถตั้งตัวจ่าย Pulse ให้กับ SM หรือกระจาย I/O ได้ เมื่อตั้งค่าให้ใช้งานเกี่ยวกับการเคลื่อนไหวพื้นฐานให้พึงระวังว่าโปรแกรม STEP7 จะไม่เตือนถ้าตั้งให้แกนนั้นๆทำงานที่ ความเร็วหรือความถี่สูงสุด ซึ่งเกินค่าของข้อจำกัดของตัวฮาร์ดแวร์เอง ดังนั้นต้องมั่นใจว่าไม่ได้ทำการสั่งงานเกินค่าความถี่ Pulse สูงสุด ของตัวฮาร์ดแวร์

สามารถเลือกใช้ตัวจ่าย Pulse จาก CPU เอาต์พุต ซึ่งจะมีคำสั่ง I/O เริ่มต้น ตามตารางด้านล่างนี้ (แต่ก็สามารถทำการเปลี่ยนตัวเอาต์พุตเป็นเบอร์อื่นๆ ก็ได้)

PWM นั้นต้องการเพียงแค่เอาต์พุตตัวเดียว ในขณะที่ PTO สามารถเลือกใช้งาน 1 หรือ 2 เอาต์พุตต่อช่องก็ได้ ดังนั้นหากเอาต์พุตนั้นไม่ต้องการใช้งาน Pulse ฟังก์ชั่น ก็สามารถนำไปใช้งานในส่วนอื่นๆได้

Description	Pulse	Direction
PTO1		
Built-in I/O	Q0.0	Q0.1
SB I/O	Q4.0	Q4.1
PWM1		
Built-in outputs	Q0.0	-
SB outputs	Q4.0	-
PTO2		
Built-in I/O	Q0.2	Q0.3
SB I/O	Q4.2	Q4.3
PWM2		
Built-in outputs	Q0.2	-
SB outputs	Q4.2	-
PTO3		
Built-in I/O	Q0.4[1]	Q0.5[1]
SB I/O	Q4.0	Q4.1
PWM3		
Built-in outputs	Q0.4[1]	-
SB outputs	Q4.1	-
PTO4		
Built-in I/O	Q0.6[2]	Q0.7[2]
SB I/O	Q4.2	Q4.3
PWM4		
Built-in outputs	Q0.6[2]	-
SB outputs	Q4.3	-

รูปที่ 2.13 การเลือกใช้ Pulse

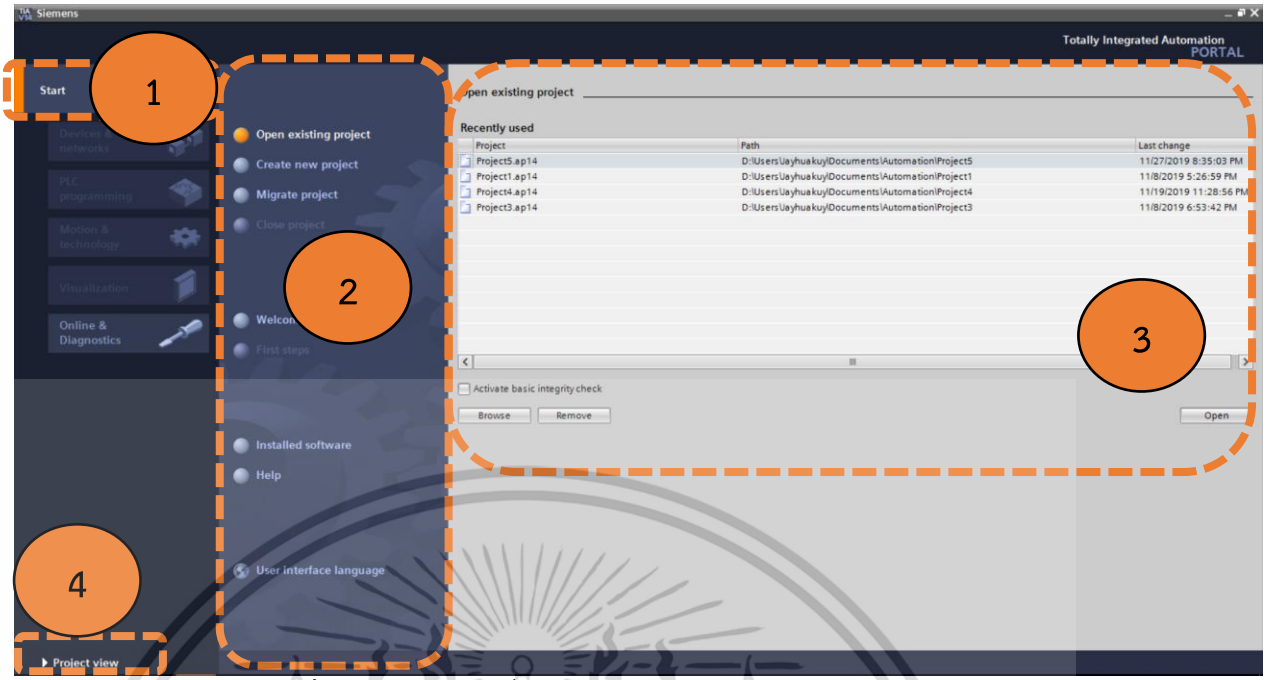
1. CPU 1211C ไม่มี Q0.4, Q0.5, Q0.6, หรือ Q0.7 ดังนั้นไม่สามารถใช้เอาต์พุตเหล่านี้กับ CPU 1211C.
2. CPU 1212C ไม่มี Q0.6 หรือ Q0.7 ดังนั้นไม่สามารถใช้เอาต์พุตเหล่านี้กับ CPU 1212C
3. ตารางนี้ใช้อ้างอิงกับ CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C, และ CPU 1217C

2.6 โปรแกรม TIA Portal V.14

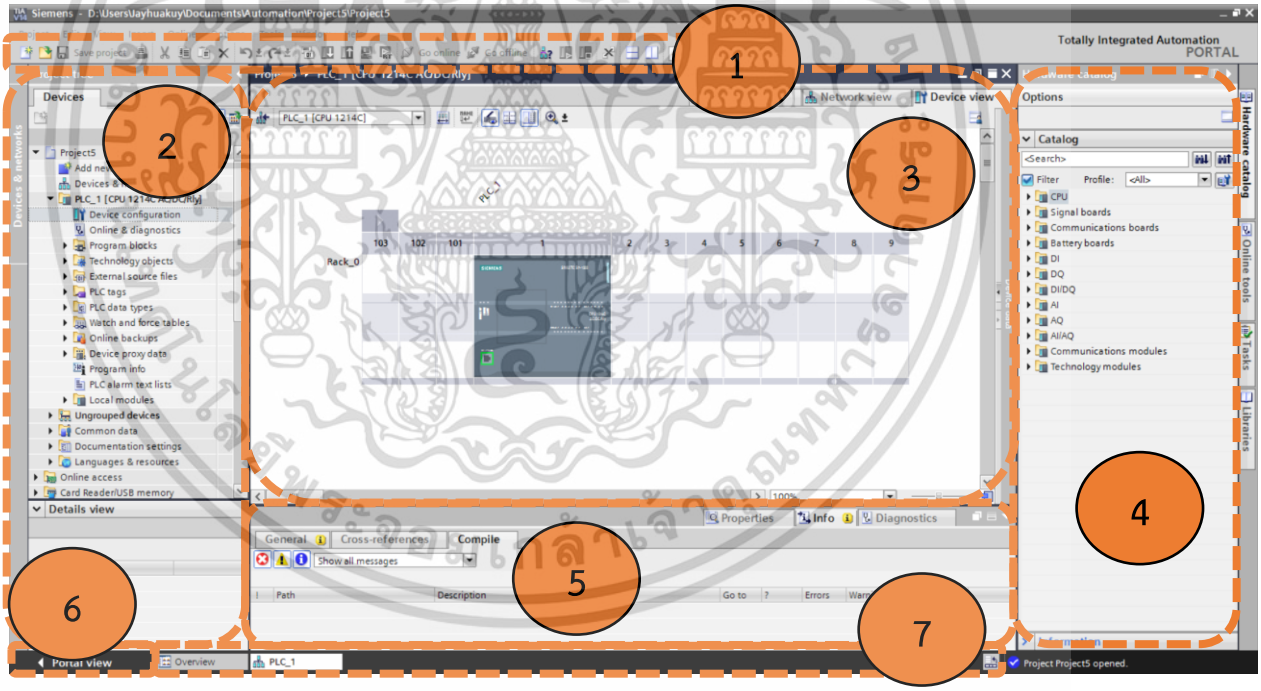
เป็นโปรแกรมที่รวมเอาผลิตภัณฑ์เกือบทั้งหมดของ Siemens มารวมไว้ในโปรแกรมเดียวเพื่อทำให้การออกแบบทางวิศวกรรมทำได้ง่ายยิ่งขึ้น โดยโปรแกรมจะรวมผลิตภัณฑ์ Drive PLC HMI IPC มาไว้ด้วยกันเพื่อง่ายต่อการออกแบบโดยไม่ต้องลงหลายๆโปรแกรม

2.6.1 ส่วนประกอบของโปรแกรม TIA Portal V.14

1. เมนูหลักเพื่อไปยังงานต่างๆ
2. งานที่เลือกมาจากเมนูหลัก (สร้างโปรเจคใหม่ เปิดโปรเจคเก่า ไมเกรทโปรเจค)
3. โปรเจคงานที่ต้องการเลือก
4. เปลี่ยนไปยัง มุมมองแบบโปรเจค(จาก Portal View ไปยัง Project View)



รูปที่ 2.14 หน้าต่างเริ่มต้นของโปรแกรม TIA Portal V.14



รูปที่ 2.15 ส่วนประกอบหน้าต่างการทำงานของโปรแกรม TIA Portal V.14

1. แถบเมนู และแถบเครื่องมือ (Menu bar และ Tool bar)
2. แถบอุปกรณ์ และโปรเจคของอุปกรณ์ (Project Navigator)
3. พื้นที่สำหรับทำงาน (Work Area)
4. อุปกรณ์เสริม และชุดคำสั่ง เพื่อสร้างโปรแกรม (Task Cards)

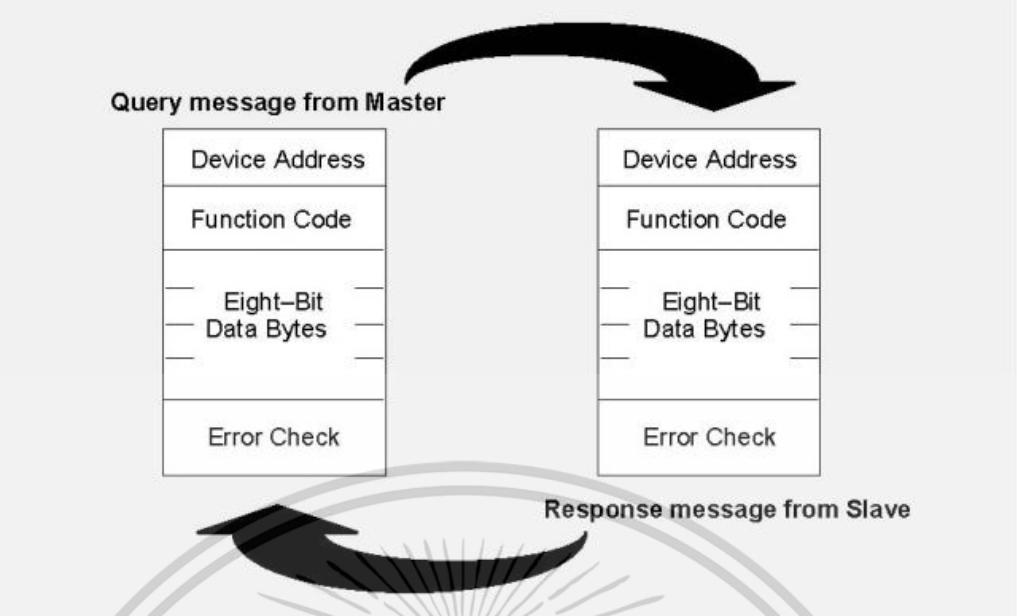
5. หน้าต่างการตั้งค่าของงาน (Inspector Window)
6. เปลี่ยนไปยังมุมมองแบบเมนูหลัก (Change to Portal View)
7. แถบเครื่องมือที่เปิดขึ้นมาเพื่อแก้ไขงาน (Editor bar)

2.7. Modbus

โปรโตคอล Modbus เป็นโปรโตคอลเพื่อสื่อสารข้อมูลอินพุต/เอาต์พุตและรีจีสเตอร์ภายใน PLC ซึ่งถูกคิดค้นโดย Modicon (ปัจจุบันคือบริษัท Schneider Electric) โปรโตคอล Modbus ได้เป็นที่ยอมรับกันอย่างกว้างขวางในการติดต่อสื่อสารที่เป็นแบบเน็ตเวิร์คโปรโตคอล อันเนื่องมาจาก MODBUS เป็นระบบเปิด ไม่มีค่าใช้จ่าย เชื่อมต่อและพัฒนาง่าย พร้อมทั้งยังสามารถนำโปรโตคอลนี้ไปใช้งานในอุปกรณ์อื่นๆ เช่น RTU(Remote Terminal Unit) PLC เป็นต้น นอกจากนี้ Modbus ยังสามารถรองรับและใช้งานร่วมกับแอพพลิเคชั่นจำพวก SCADA และ HMI ซอฟต์แวร์ได้อีกด้วย

โปรโตคอล Modbus เป็นการสื่อสารข้อมูลในลักษณะ Master/Slave ซึ่งเป็นการสื่อสารจากอุปกรณ์แม่ (Master) เครื่องเดียว ส่วนใหญ่มักเป็นซอฟต์แวร์คอมพิวเตอร์หรืออุปกรณ์แสดงผล HMI ไปยังอุปกรณ์ลูก (Slave) ได้หลายๆเครื่อง โดยสามารถกำหนดหมายเลขอุปกรณ์ได้สูงสุด 255 เครื่อง โดยมีลักษณะการส่งข้อมูล 2 แบบ คือ ข้อมูลแบบ ASCII และข้อมูลแบบเลขฐานสอง (Binary) ในโปรโตคอล Modbus ที่สื่อสารข้อมูลแบบ ASCII จะเรียก Modbus ASCII และโปรโตคอล Modbus ที่สื่อสารข้อมูลแบบเลขฐานสอง จะเรียก Modbus RTU ทำให้มีความแตกต่างในการกำหนดค่าพอร์ตสื่อสาร

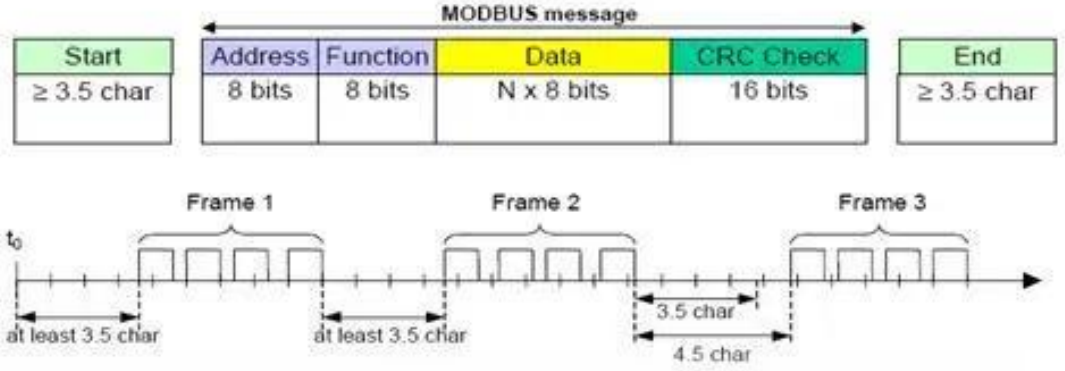
การรับส่งข้อมูลด้วยโปรโตคอล Modbus สามารถเลือกได้ 2 โหมด คือ โหมด ASCII และโหมด RTU ซึ่งทั้ง 2 โหมดนี้มีความแตกต่างกันที่การกำหนดรูปแบบของชุดข้อมูลภายในเฟรม จะเลือกโหมดใดก็ได้แต่มีเงื่อนไขว่า อุปกรณ์ทุกตัวที่ต่อร่วมกันอยู่ในบัสหรือเครือข่ายเดียวกัน จะต้องตั้งให้เลือกใช้โหมดเดียวกันทั้งหมด



รูปที่ 2.16 การติดต่อสื่อสารแบบ Master/Slave

2.7.1 Modbus RTU

เฟรมข้อมูลในโหมด RTU ประกอบด้วยข้อมูลแสดงตำแหน่งแอดเดรส 1 ไบต์, หมายเลขฟังก์ชัน 1 ไบต์, ข้อมูลที่ทำการรับส่งจำนวนมากสุดไม่เกิน 252 ไบต์ และรหัสตรวจสอบความถูกต้องของข้อมูลแบบ CRC (Cyclical Redundancy Checking) ขนาด 2 ไบต์ ค่า CRC นี้เป็นค่าที่คำนวณมาจากข้อมูลทุกไบต์ ไม่รวมบิต Start, Stop และ Parity Check โดยที่ตัว Slave ตัวที่ส่งข้อมูลออกมาจะสร้างรหัส CRC แล้วส่งตามท้ายไบต์ข้อมูลออกมา หลังจากนั้นเมื่อ Master ได้รับเฟรมข้อมูลและถอดข้อมูลออกจากเฟรมแล้วจะทำการคำนวณค่า CRC ตามสูตรเดียวกับ Slave เพื่อทำการเปรียบเทียบค่า CRC ทั้ง 2 ค่าว่าตรงกันหรือไม่ หากไม่ตรงกันแสดงว่าเกิดความผิดพลาดในการรับส่งข้อมูลในโหมด RTU การรับส่งข้อมูล 1 ไบต์ ไม่ว่าจะเป็นข้อมูลส่วนใดภายในเฟรมจะต้องทำการส่งบิตข้อมูลรวม 11 บิต คือ บิต Start 1 บิต, บิตข้อมูล 8 บิต, บิตตรวจสอบ Parity ของข้อมูล 1 บิตและบิต Stop 1 บิต หรือหากเลือกแบบไม่มีบิต Parity ก็จะเป็นแบบ Stop แทน 2 บิต สำหรับการกำหนดให้มีบิต Parity นั้น สามารถเลือกเป็นแบบคู่หรือคี่ก็ได้ และหากต้องการออกแบบให้สอดคล้องกับอุปกรณ์ที่มีใช้กันทั่วไปมากที่สุด ควรเลือกแบบคู่โดยที่สามารถปรับเปลี่ยนเป็นแบบคี่หรือไม่มีการตรวจสอบ Parity ได้ด้วย



รูปที่ 2.17 ลักษณะเฟรมข้อมูลของ MODBUS RTU

รูปที่ 2.18 ลักษณะข้อมูลแต่ละไบต์ของ MODBUS RTU

2.7.2 MODBUS ASCII

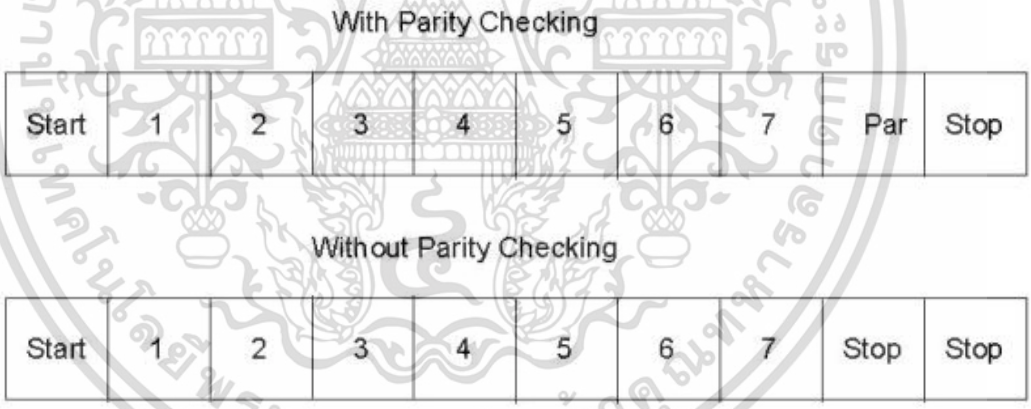
การรับส่งข้อมูลในโหมด ASCII นั้นมีความแตกต่างจากโหมด RTU ตรงที่ในโหมด RTU ข้อมูลที่จะส่งขนาด 1 ไบต์ นำมารวมกับบิตประกอบต่างๆ ก็สามารถส่งออกไปได้เลย แต่สำหรับโหมด ASCII จะมองข้อมูล 1 ไบต์ นั้นออกมาเป็นตัวอักษร 2 ตัว เช่น ค่า 0x5B ซึ่งเป็นเลขฐานสิบหก ก็จะถูกมองเป็นตัวอักษร ‘5’ และตัวอักษร ‘B’ จากนั้นก็จะทำการค้นหารหัส ASCII ของตัวอักษรทั้ง 2 ตัวนั้น ซึ่งได้แก่ 0x35 สำหรับ ‘5’ และ 0x42 สำหรับ ‘B’ แล้วทำการส่งรหัส ASCII ทั้ง 2 ค่านี้ออกไป ซึ่งจะได้ผลเท่ากับการส่งค่า 0x5B ซึ่งเป็นข้อมูลขนาด 1 ไบต์ ในโหมด RTU

จะเห็นได้ว่าการส่งข้อมูลในโหมด ASCII จะต้องทำงานมากกว่าการส่งข้อมูลในโหมด RTU ซึ่งทำให้อัตราเร็วในการสื่อสารมีค่าต่ำกว่า สาเหตุที่เป็นแบบนี้ก็เพราะว่า โหมด ASCII ได้ถูกออกแบบมาสำหรับอุปกรณ์ที่ไม่มีความสามารถในการกำหนดช่วงระยะห่างของเวลาในการส่งเฟรมข้อมูล อย่างเช่นในโหมด RTU ที่อุปกรณ์สามารถกำหนดได้ว่าจะส่งเฟรมข้อมูลแต่ละเฟรมออกมาด้วยเวลาห่างกันเท่าใด และอุปกรณ์ที่รอรับข้อมูลก็ต้องสามารถตรวจจับและแยกแยะได้ว่าเฟรมข้อมูลแต่ละเฟรมที่รับเข้ามานั้น มีระยะเวลาห่างกันภายในช่วงเวลาที่กำหนดหรือไม่ เพื่อทำให้สามารถตรวจ

สอบหาจุดเริ่มต้นและจุดสิ้นสุดของเฟรมข้อมูลแต่ละเฟรมได้ แต่ในความเป็นจริงยังมีอุปกรณ์อีกหลายชนิดที่ไม่มีความสามารถพิเศษนี้ จึงต้องใช้วิธีอื่นที่จะช่วยให้สามารถรับรู้จุดเริ่มต้นและจุดสิ้นสุดของเฟรมข้อมูลได้ นั่นได้แก่โหมด ASCII ซึ่งในโหมดนี้จะเริ่มต้นเฟรมข้อมูลด้วยการส่งรหัส ASCII ที่กำหนดให้หมายถึงจุดเริ่มต้น คือ 0x3A ซึ่งตรงกับตัวอักษร ':' ตามด้วยที่อยู่ของ Slave หมายเลขฟังก์ชัน ข้อมูล รหัสตรวจสอบ RLC และรหัส ASCII 2 ตัว ที่กำหนดให้หมายถึงจุดสิ้นสุด คือ รหัส 0x0D และ 0x0A คือรหัส CR (Carriage Return) และ LF (Line Feed) ตามลำดับ โดยในขณะที่บัสข้อมูลว่างจากการรับส่งข้อมูล อุปกรณ์ทุกตัวจะคอยตรวจสอบข้อมูลในบัสว่ามีการส่งรหัส ASCII ของ ':' ออกมาหรือไม่ ถ้ามีก็จะรับรู้ว่าขณะนี้ได้มีการเริ่มต้นส่งเฟรมข้อมูลออกมาแล้ว ก็จะเข้ากระบวนการรับข้อมูลต่อไป

Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

รูปที่ 2.19 ลักษณะเฟรมข้อมูลของ MODBUS ASCII

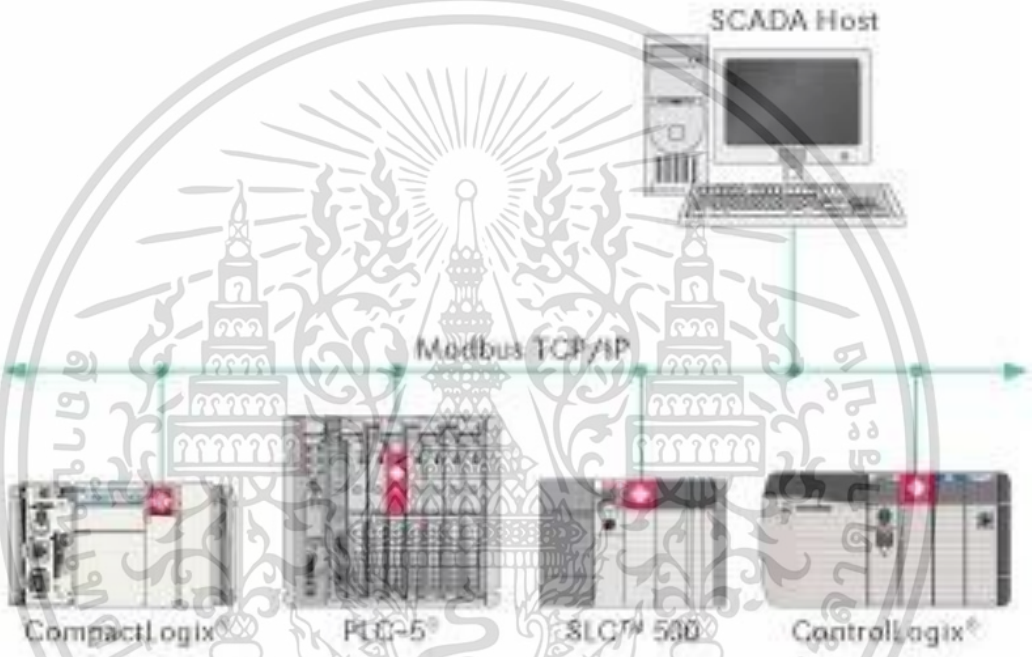


รูปที่ 2.20 ลักษณะข้อมูลแต่ละไบต์ของ MODBUS ASCII

Modbus จะบริการให้อุปกรณ์ติดต่อสื่อสารกันผ่าน Serial Port (RS-232/422/485) แต่ในปัจจุบันได้มีการพัฒนาให้อุปกรณ์สามารถติดต่อสื่อสารกับอุปกรณ์ที่อยู่บนเครือข่ายอีเทอร์เน็ต ซึ่งอุปกรณ์ที่ใช้การสื่อสารแบบ Modbus โปรโตคอล ส่วนใหญ่จะเป็น PLC DCS HMIs อย่างไรก็ตาม Modbus จำเป็นต้องมีอุปกรณ์จำพวก Gateway และ Bridge ในการติดต่อสื่อสารระหว่าง Serial Line กับอีเทอร์เน็ต

2.7.3. Modbus TCP/IP

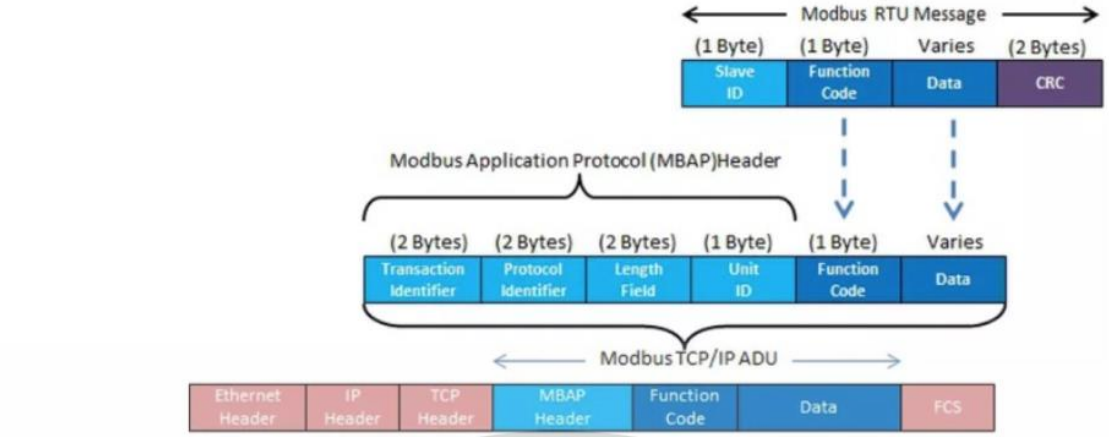
Modbus TCP/IP ถูกพัฒนาขึ้นโดยมีวัตถุประสงค์เพื่อจะนำการสื่อสารแบบอินเทอร์เน็ตมาใช้กับอุปกรณ์จำพวกอีเทอร์เน็ต ระยะในการใช้งานสำหรับการเดินสาย (สายแลน) คือ 100 เมตร โดยสามารถขยายระยะในการสื่อสารได้โดยการใช้อุปกรณ์ Repeater หรือในระบบแลนจะเรียกอุปกรณ์นี้ว่า Hub หรือ Switch ก็จะสามารถลากสายได้อีก 100 เมตร และยังสามารถต่อ Repeater ขยายระยะทางได้โดยไม่จำกัด ในการสื่อสารโดยทั่วไปมีความเร็ว 100,000,000 บิตต่อวินาที (100 Mbps) และเชื่อมต่ออุปกรณ์ได้ไม่จำกัดจำนวน



รูปที่ 2.21 เครือข่าย Modbus TCP/IP

Modbus ASCII/RTU ที่จะติดต่อสื่อสารกับ Mosbus TCP เพื่อให้ใช้งานในเครือข่ายอีเทอร์เน็ตจะใช้ Gateway ติดต่อและแปลงรูปแบบการสื่อสารข้อมูล โดยการสื่อสารของ Modbus RTU/ASCII จะเป็นการสื่อสารผ่านทาง RS-232/422/485 นั้นจะถูก Gateway แปลงให้เป็น Modbus TCP เพื่อใช้ในการติดต่อสื่อสารในเครือข่ายอีเทอร์เน็ตต่อไป

ในทางปฏิบัติ Modbus TCP/IP คือ โปรโตคอล Modbus RTU ที่เชื่อมต่อด้วย TCP โดยทำงานบนอีเทอร์เน็ต โครงสร้างข้อความของ Modbus คือ แอพพลิเคชั่นโปรโตคอลที่จะถูกส่งผ่านไปพร้อมกับ TCP/IP ในทางปฏิบัติ Modbus TCP จะฝังเฟรมข้อมูลของ Modbus RTU ร่วมไปกับเฟรมของ TCP โดยไม่ต้องใช้ Modbus Checksum แต่จะใช้ Checksum ของ TCP แทนดังรูปที่ 2.22



รูปที่ 2.22 การใช้ Checksum ของ TCP แทน

จากรูปที่ 2.22 Modbus แอพพลิเคชั่นโปรโตคอล จะประกอบด้วยข้อมูล 7 ไบต์ ซึ่งจะวางหน้าข้อความของ Modbus RTU โดยมีรายละเอียดดังนี้

1. ตัวระบุ Transaction/invocation (2 ไบต์) : ใช้จับคู่การแลกเปลี่ยนข้อมูลเมื่อมีข้อความหลายๆชุด ถูกส่งออกมาด้วย TCP เดียวกัน ด้วย Client ตัวใดตัวหนึ่ง โดยไม่ต้องรอลำดับการตอบสนอง
2. ตัวระบุโปรโตคอล (2 ไบต์) : ในส่วนจะมีค่าเป็น 0 เสมอ
3. Length (2 ไบต์) : เป็นการระบุจำนวนไบต์ที่รวมจำนวนไบต์ของตัวระบุหน่วย ฟังก์ชั่นโค้ด และพื้นที่ข้อมูล
4. ตัวระบุหน่วย (1 byte) : เป็นการระบุ ID ของเซิร์ฟเวอร์ที่อยู่ในระบบสื่อสาร อาจตั้งเป็น 00 ถึง FF ก็ได้

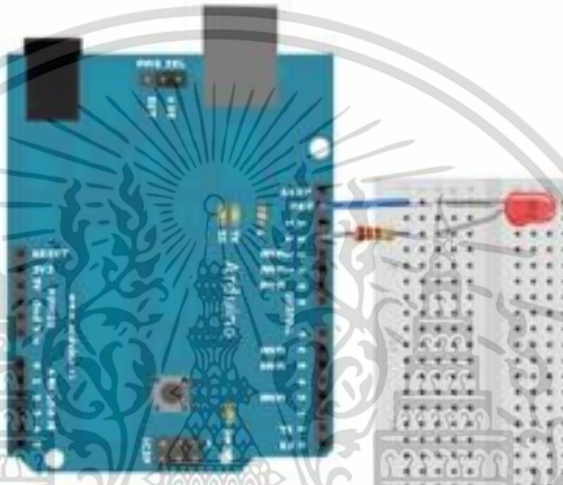
รูปที่ 2.23. การแปลง Modbus Serial เป็น Modbus อีเทอร์เน็ต

2.8 Arduino

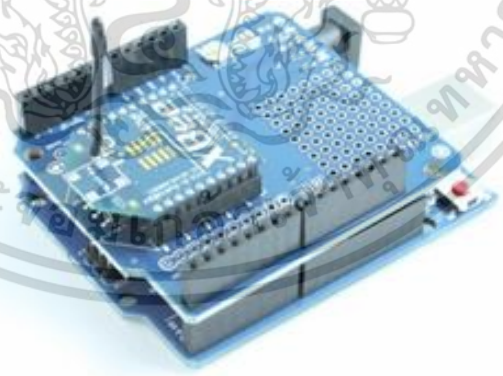
Arduino เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย

ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลงเพิ่มเติมพัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้อีกด้วย

ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่างๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิกส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด (รูปที่ 2.24) หรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆ (รูปที่ 2.25) เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย



รูปที่ 2.24 บอร์ด Arduino ต่อกับ LED



รูปที่ 2.25 บอร์ด Arduino ต่อกับ บอร์ด Xbee Shield

2.8.1 ส่วนที่เป็นฮาร์ดแวร์ (Hardware)

บอร์ดอิเล็กทรอนิกส์ขนาดเล็ก ที่มีไมโครคอนโทรลเลอร์ (MCU) เป็นชิ้นส่วนหลัก ถูกนำมาประกอบร่วมกับอุปกรณ์อิเล็กทรอนิกส์อื่นๆ เพื่อให้ง่ายต่อการใช้งาน หรือที่เรียกกันว่า บอร์ด Arduino โดยบอร์ด Arduino เองก็มีหลายรุ่นให้เลือกใช้ โดยในแต่ละรุ่นอาจมีความแตกต่างกันในเรื่องของขนาดของบอร์ดหรือสเปค เช่น จำนวนของขารับส่งสัญญาณ, แรงดันไฟที่ใช้, ประสิทธิภาพของ MCU เป็นต้น

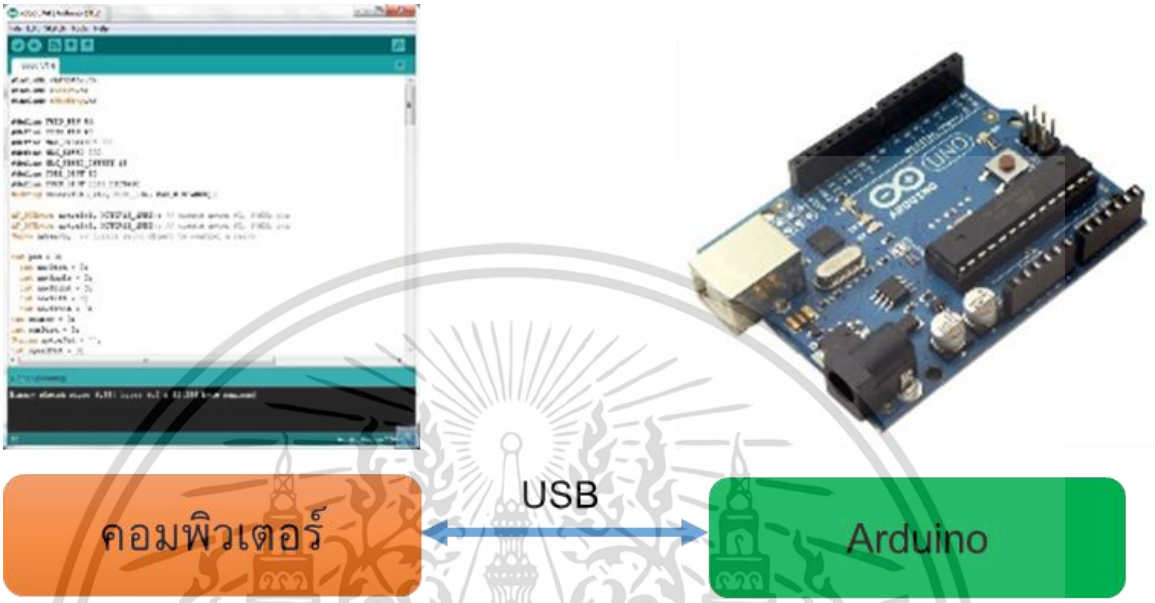
2.8.2 ส่วนที่เป็นซอฟต์แวร์ (Software)

- ภาษา Arduino (ซึ่งจริงๆ แล้วก็คือ ภาษา C/C++) ใช้สำหรับเขียนโปรแกรมควบคุม MCU
- Arduino IDE เป็นเครื่องมือสำหรับเขียนโปรแกรมด้วยภาษา Arduino, Compile โปรแกรม และอัพโหลดโปรแกรมลงบอร์ด

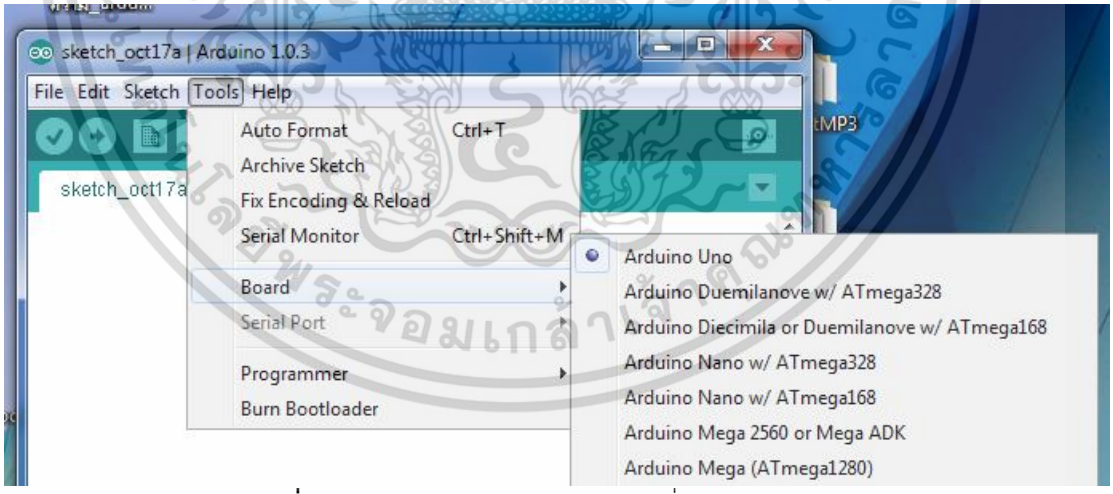
รูปที่ 2.26 Arduino IDE

2.8.3 รูปแบบการเขียนโปรแกรมบน Arduino

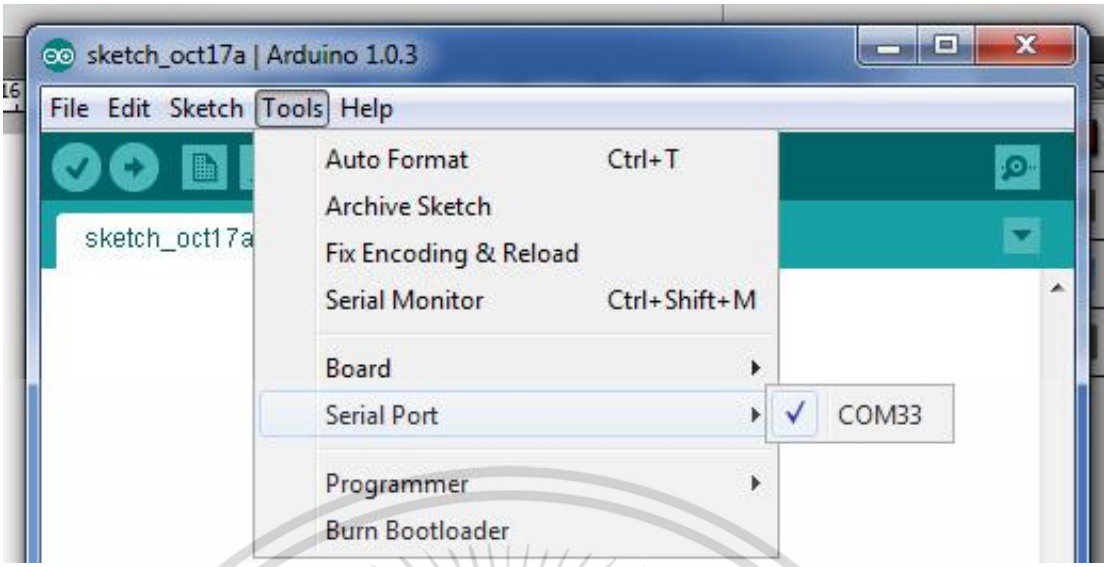
1. เขียนโปรแกรมบนคอมพิวเตอร์ ผ่านทางโปรแกรม ArduinoIDE ซึ่งสามารถดาวน์โหลดได้จาก Arduino.cc/en/main/software
2. หลังจากที่เขียนโค้ดโปรแกรมเรียบร้อยแล้ว ให้ผู้ใช้งานเลือกรุ่นบอร์ด Arduino ที่ใช้ และหมายเลข Com port



รูปที่ 2.27 ตัวอย่างการเชื่อมต่อของคอมพิวเตอร์และ Arduino



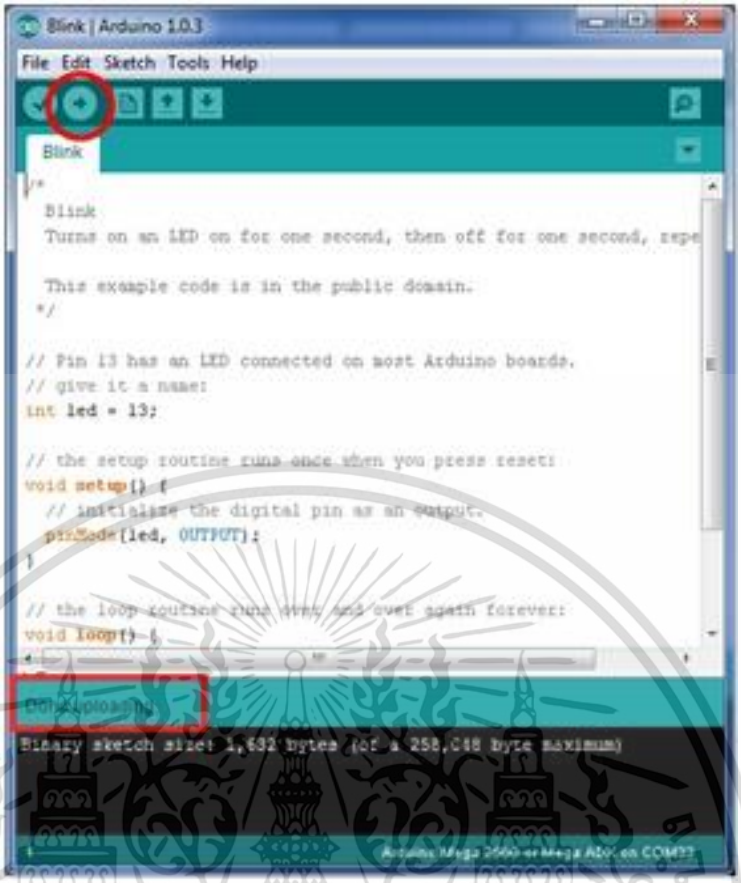
รูปที่ 2.28 เลือกรุ่นบอร์ด Arduino ที่ต้องการอัพโหลด



รูปที่ 2.29 เลือกหมายเลข Comport ของบอร์ด

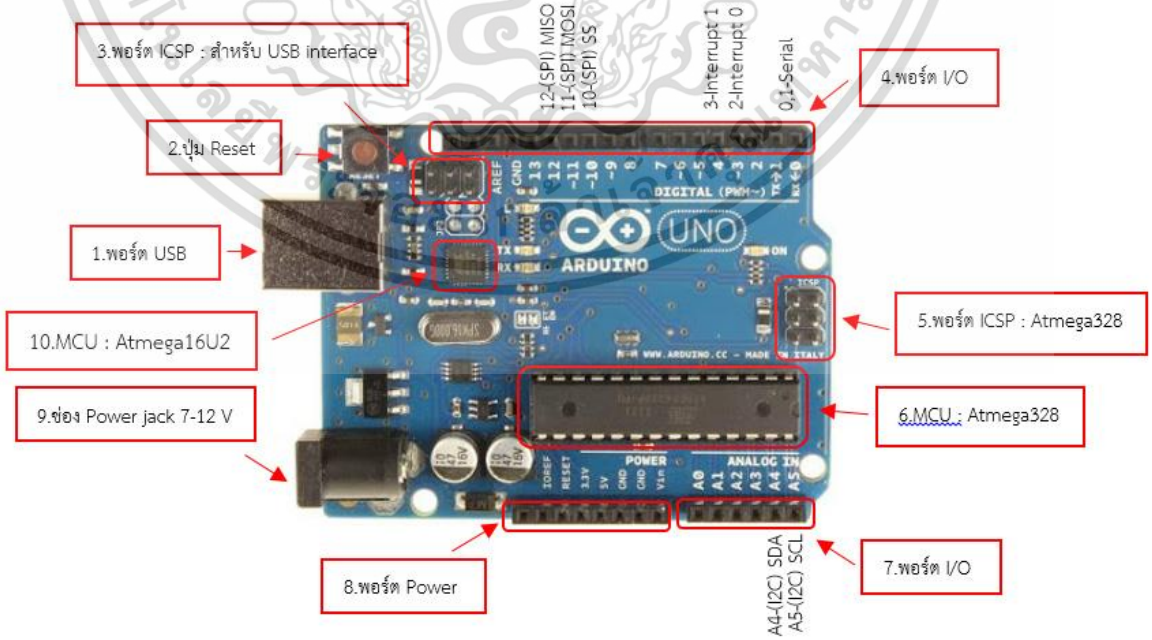
3. กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม จากนั้นกดปุ่มอัพโหลดโค้ด โปรแกรมไปยังบอร์ด Arduino ผ่านทางสาย USB เมื่ออับโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที

รูปที่ 2.30 กดปุ่ม verify เพื่อตรวจสอบความถูกต้อง และCompile โค้ดโปรแกรม



รูปที่ 2.31 อัพโหลดโค้ดโปรแกรม

2.8.4 Layout and Pin out ของบอร์ด Arduino



รูปที่ 2.32 บอร์ด Arduino และ พอร์ตต่างๆ

1. USB Port : ใช้สำหรับต่อกับคอมพิวเตอร์เพื่ออัพโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด
2. Reset Button : เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่
3. ICSP Port ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2
4. I/OPort : ดิจิตอล I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx,Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM
5. ICSP Port : Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader
6. MCU : Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino
7. I/OPort : นอกจากจะเป็นดิจิตอล I/O แล้วยังเปลี่ยนเป็นช่องรับสัญญาณอนาล็อกตั้งแต่ขา A0-A5
8. Power Port : ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, Vin
9. Power Jack : รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V
10. MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ คอมพิวเตอร์ ผ่าน Atmega16U2

2.9 Hub

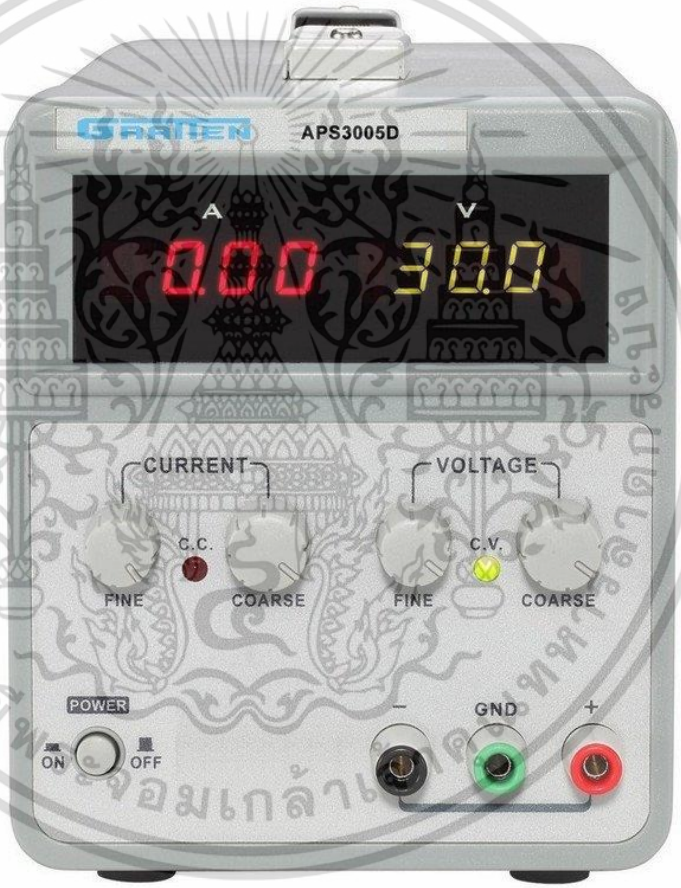
Hub เป็นอุปกรณ์ศูนย์กลางที่เชื่อมต่อคอมพิวเตอร์หรืออุปกรณ์คอมพิวเตอร์ชนิดอื่น เข้าด้วยกัน Hub ในระบบเครือข่าย เป็นอุปกรณ์ที่ใช้สำหรับเชื่อมโยงสัญญาณของอุปกรณ์เครือข่ายเข้าด้วยกัน การจะทำให้คอมพิวเตอร์แต่ละเครื่องคอมพิวเตอร์รู้จักกันหรือส่งข้อมูลถึงกันได้จะต้องผ่านอุปกรณ์ตัวนี้ โดยทั่วไปจะมีลักษณะเหมือนกล่องสีเหลี่ยมแต่แบนมีความสูงประมาณ 1-3 นิ้ว แล้วแต่รุ่นมีช่องเอาไว้เสียบสายแลนแต่ละเส้นที่ลากโยงมาจากคอมพิวเตอร์มีหลายรุ่น เช่น Hub 4 Ports, 8 Ports, 16 Ports, 24 Ports หรือ 48 Ports เป็นต้น

รูปที่ 2.33 Hub

2.10 แหล่งจ่ายไฟ

แหล่งจ่ายไฟ เป็นอุปกรณ์ที่จ่ายพลังงานไฟฟ้าให้กับโหลดไฟฟ้า เป็นคำที่ใช้กันมากที่สุดในการแปลงพลังงานไฟฟ้าจากรูปแบบหนึ่งไปเป็นอีกรูปแบบหนึ่ง แม้ว่ามันจะยังอาจหมายถึงอุปกรณ์ที่แปลงพลังงานรูปแบบหนึ่ง(เช่นพลังงานกล, พลังงานเคมี, พลังงานแสงอาทิตย์)ให้เป็นพลังงานไฟฟ้า แหล่งจ่ายไฟแบบควบคุมได้ (regulated power supply) สามารถควบคุมแรงดันหรือกระแสเอาต์พุตให้มีค่าที่คงที่แน่นอนแม้ว่าโหลดจะมีการเปลี่ยนแปลงหรือมีการเปลี่ยนแปลงที่พลังงานที่อินพุทก็ตาม

แหล่งจ่ายไฟทุกตัวต้องได้รับพลังงานจากแหล่งพลังงานภายนอกเพื่อจ่ายให้โหลดและการบริโภคพลังงานของตัวมันเองในขณะที่ปฏิบัติงาน แหล่งพลังงานภายนอกจะขึ้นอยู่กับการออกแบบ



รูปที่ 2.34 แหล่งจ่ายไฟแบบควบคุมได้

2.11 ตัวต้านทานปรับค่าได้

ตัวต้านทานปรับค่าได้ เป็นตัวต้านทานชนิดหนึ่ง ที่สามารถปรับค่าความต้านทานได้ นิยมใช้งานแบบวงจรแบ่งแรงดัน เช่น ใช้ปรับค่าความดังของเสียง ปรับค่าแรงดันเอาต์พุต เป็นต้น

ตัวต้านทานปรับค่าได้มีตัวถังหลายรูปแบบ เช่นในรูปที่ 2.35 เป็นตัวต้านทานปรับค่าได้แบบวอลุ่ม จะมีอยู่ด้วยกัน 2 ชนิด คือชนิด A และชนิด B

- ตัวต้านทานปรับค่าได้แบบวอลุ่มชนิด A ลักษณะการปรับค่าความต้านจะไม่เป็นเชิงเส้น นิยมใช้ในการปรับค่าความดังของเสียง จะหาซื้อได้ยากกว่าชนิด B
- ตัวต้านทานปรับค่าได้แบบวอลุ่มชนิด B ลักษณะของการปรับค่าความต้านทานจะเป็นแบบเชิงเส้น นิยมใช้งานทั่วไป สามารถหาซื้อได้ง่าย

นอกจากนี้ตัวต้านทานปรับค่าได้แบบวอลุ่ม ยังมีแบบชนิดมีแทปกลาง แบบ 2 ชั้น แบบ 4 ชั้น และแบบมีสวิตซ์ในตัว สำหรับในชุดบทความนี้ยกมาแต่แบบธรรมดา 3 ขาเท่านั้น

ตัวต้านทานปรับค่าได้ มีขาใช้งานอยู่ 3 ขา โดยขากลาง จะเป็นขาที่ต่ออยู่กับแกนเพื่อหมุนเลือกค่าควาต้านทาน ส่วนอีก 2 ขา จะต่ออยู่กับแถบคาบอนที่มีค่าความต้านทาน

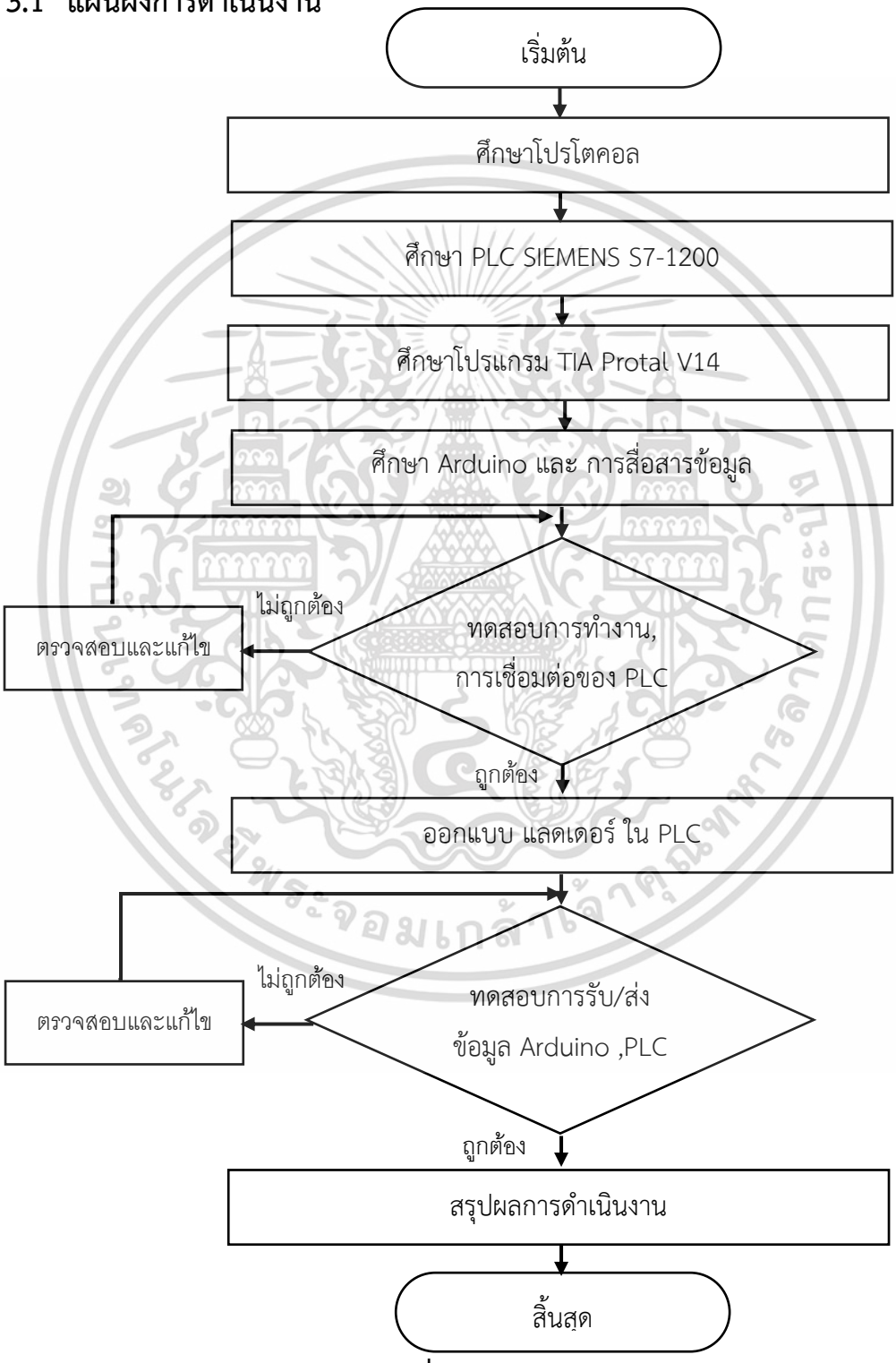
รูปที่ 2.35 ตัวต้านทานปรับค่าได้แบบวอลุ่ม

ตัวต้านทานปรับค่าได้แบบวอลุ่ม จะมีค่าความต้านทานระบุอยู่ที่ตัวถัง โดยค่าความต้านทานที่ระบุ คือค่าที่สามารถปรับได้สูงสุด หากนำมัลติมิเตอร์มาวัดที่ขาทั้ง 2 ด้าน ค่าความต้านทาน จะตรงกับค่าที่ระบุบนตัวถัง และตัวต้านทานปรับค่าได้แบบวอลุ่ม ค่านิยมใช้มากที่สุดคือ 100kΩ แบบชนิด B

รูปที่ 2.36 การดูค่าความต้านทานของตัวต้านทานปรับค่าได้แบบวอลุ่ม

บทที่ 3
วิธีการดำเนินงาน

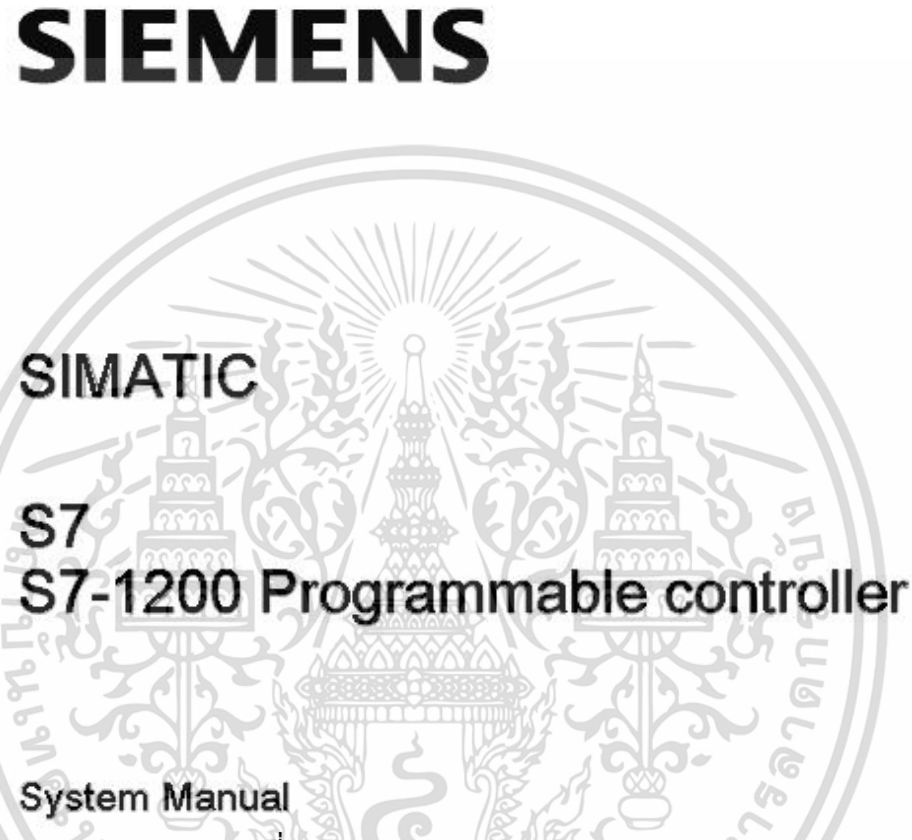
3.1 แผนผังการดำเนินงาน



รูปที่ 3.1 แผนผังการดำเนินงาน

3.2 การศึกษาโปรโตคอล

จากหัวข้อและขอบเขตการทำโครงงานฉบับนี้ เรื่องแรกที่จำเป็นจะต้องศึกษาก็คือส่วนของโปรโตคอล เนื่องจากจะต้องเข้าใจหลักการทำงานของ Modbus TCP/IP เพื่อนำไปต่อยอดกับอุปกรณ์ที่มีอยู่และนำมาใช้ทดลองได้อย่างถูกต้องแม่นยำ



รูปที่ 3.2 PLC S7-1200 Manual Book

3.3 การศึกษา PLC Siemens S7-1200

ในส่วนของ PLC ก็มีความสำคัญและจำเป็นจะต้องศึกษาโครงสร้างหลักการทำงาน ความเหมาะสมในการใช้งานควบคู่กับโปรโตคอล Modbus TCP/IP โดยจะทำการศึกษา PLC ตัวนี้ จากในส่วนของตัว Manual Book

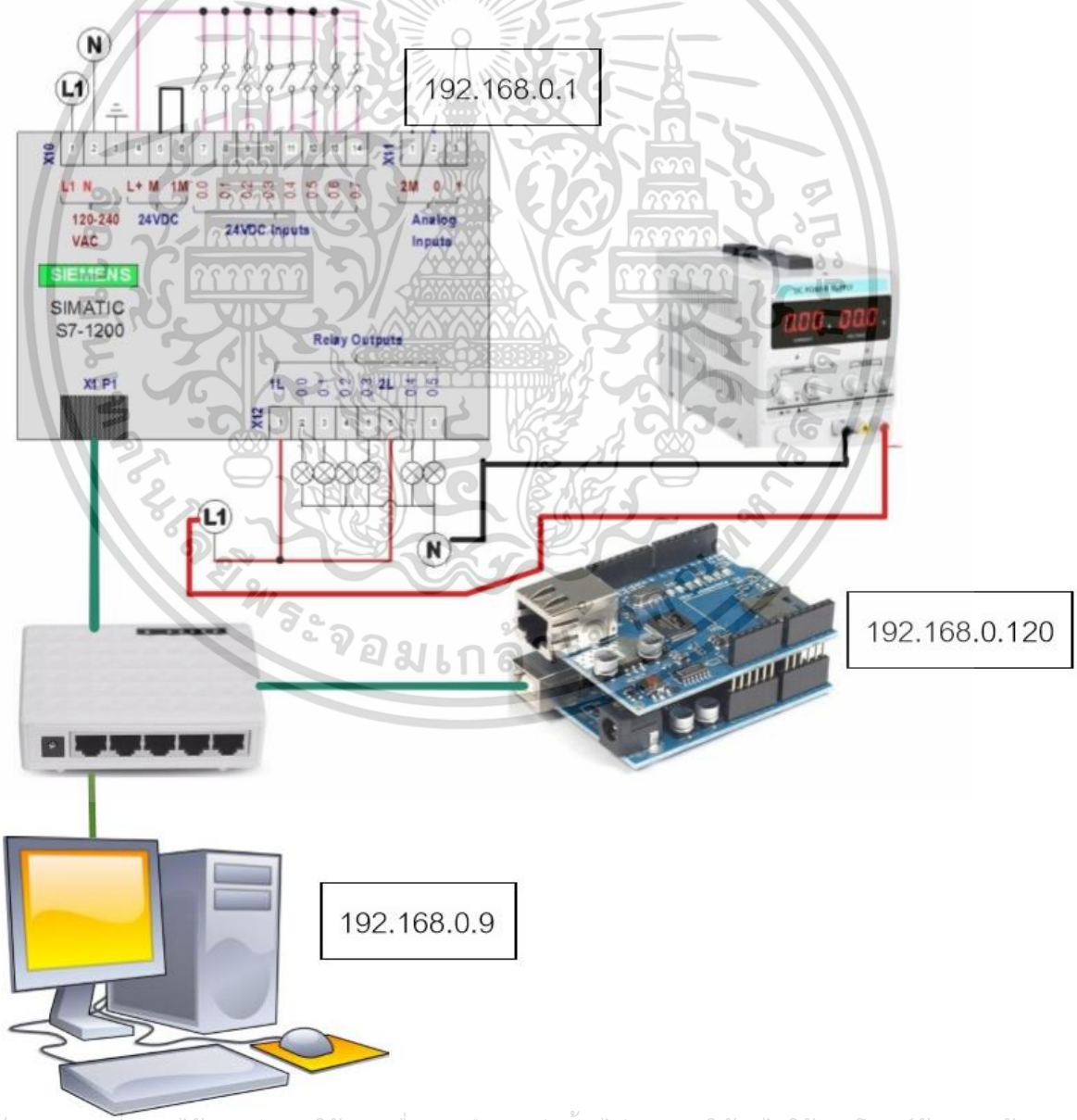
3.4 การศึกษาโปรแกรม TIA Portal V14

เมื่อทำการเลือกใช้งาน PLC ของทางบรัษัท Siemens ก็จะต้องทำการศึกษาโปรแกรมที่นำมาใช้ควบคู่กันในการเขียด แลดเดอร์ไดอะแกรม นั่นก็คือตัว TIA Portal โดยในโปรแกรมจะมีส่วนประกอบสำคัญในการนำมาใช้รับส่งข้อมูลและสั่งการทำงานของตัวโปรโตคอล Modbus TCP/IP

รูปที่ 3.3 TIA Portal V.14

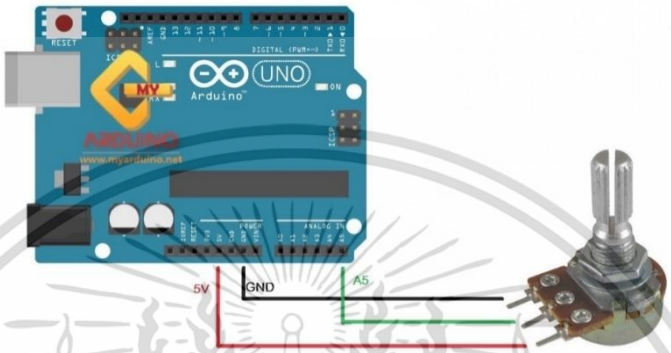
3.5 การเก็บค่าจากกระบวนการ

3.5.1 การต่อสาย



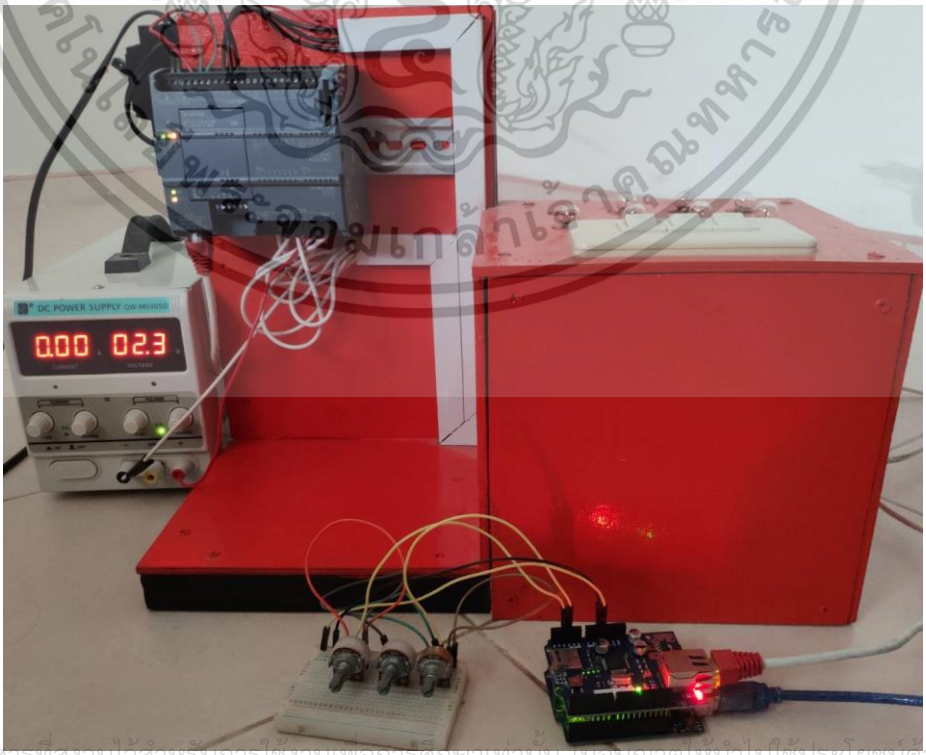
รูปที่ 3.4 การต่อสายทั้งหมด

โดยทำการเชื่อมต่อทั้ง PLC , คอมพิวเตอร์ , Arduino เข้าด้วยกันผ่าน Hub โดยใช้สายอีเทอร์เน็ต ซึ่งการส่งข้อมูลก็จะมีเลขระบุตัวรับ-ส่งชัดเจน คือ IP Address ซึ่งได้ตั้งไว้ให้อยู่ในวงแลนเดียวกันแล้ว และในที่นี้ มี Arduino เป็น Slave และ PLC เป็น Master โดย Arduino จะส่งค่าที่อ่านได้จากตัวต้านทานปรับค่าได้ไปให้ PLC



รูปที่ 3.5 การต่อสาย ARDUINO กับตัวต้านทานปรับค่าได้

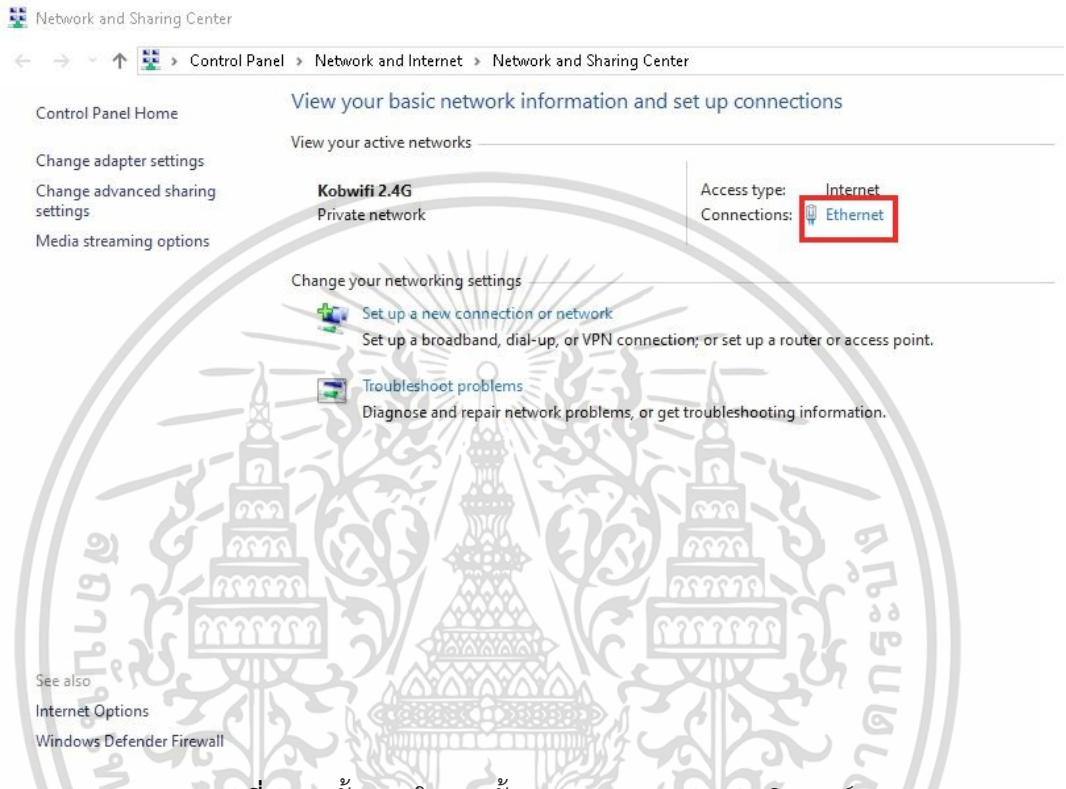
จ่ายไฟ 5V ให้กับตัวต้านทานปรับค่าได้ ในที่นี้ใช้ไฟของ Arduino และ ต่อขา GND ของตัวต้านทานปรับค่าได้กับ GND ของ Arduino และ ต่ออินพุต ไว้ที่ A5 และยังส่งข้อมูลของ PLC ไปให้กล่องทดลองเพื่อแสดงผลการทำงาน



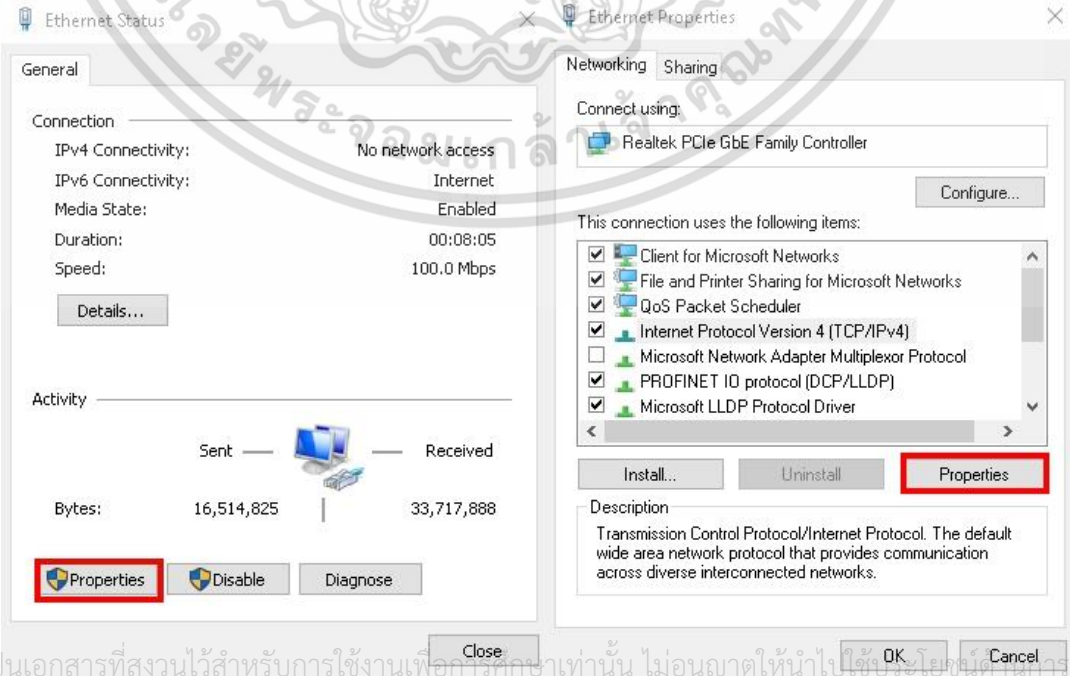
รูปที่ 3.6 การต่ออุปกรณ์ทั้งหมด

3.5.2 การตั้ง IP Address ให้กับอุปกรณ์

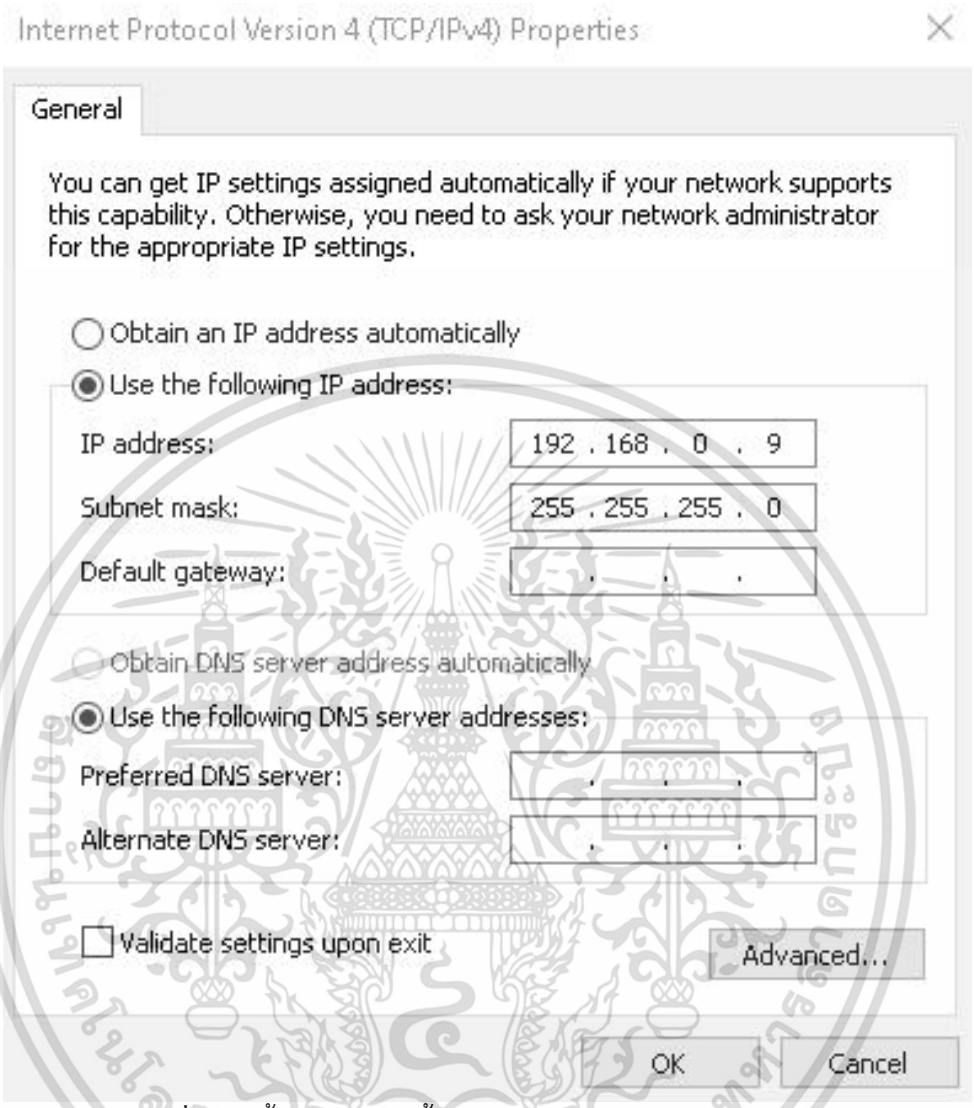
3.5.2.1 การตั้งค่า IP Address ของคอมพิวเตอร์



รูปที่ 3.7 ขั้นตอนในการตั้ง IP Address ของคอมพิวเตอร์

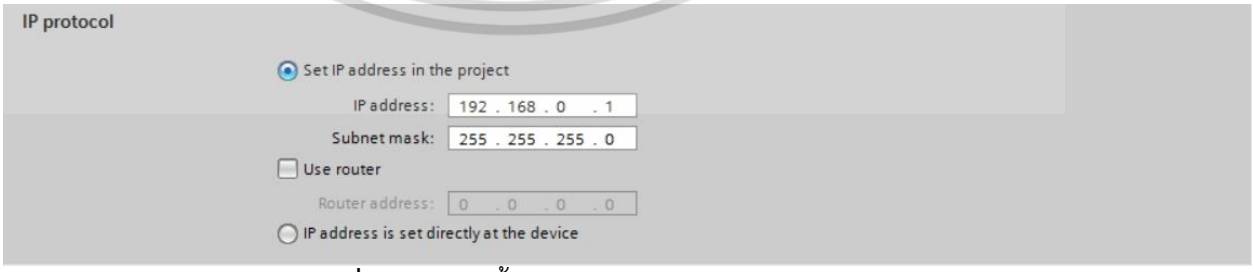


รูปที่ 3.8 ขั้นตอนในการตั้ง IP Address ของคอมพิวเตอร์ (ต่อ)



รูปที่ 3.9 ขั้นตอนในการตั้ง IP Address ของคอมพิวเตอร์ (ต่อ)

3.5.2.2 การตั้งค่า IP Address ของ PLC



รูปที่ 3.10 การตั้งค่า IP Address ของ PLC

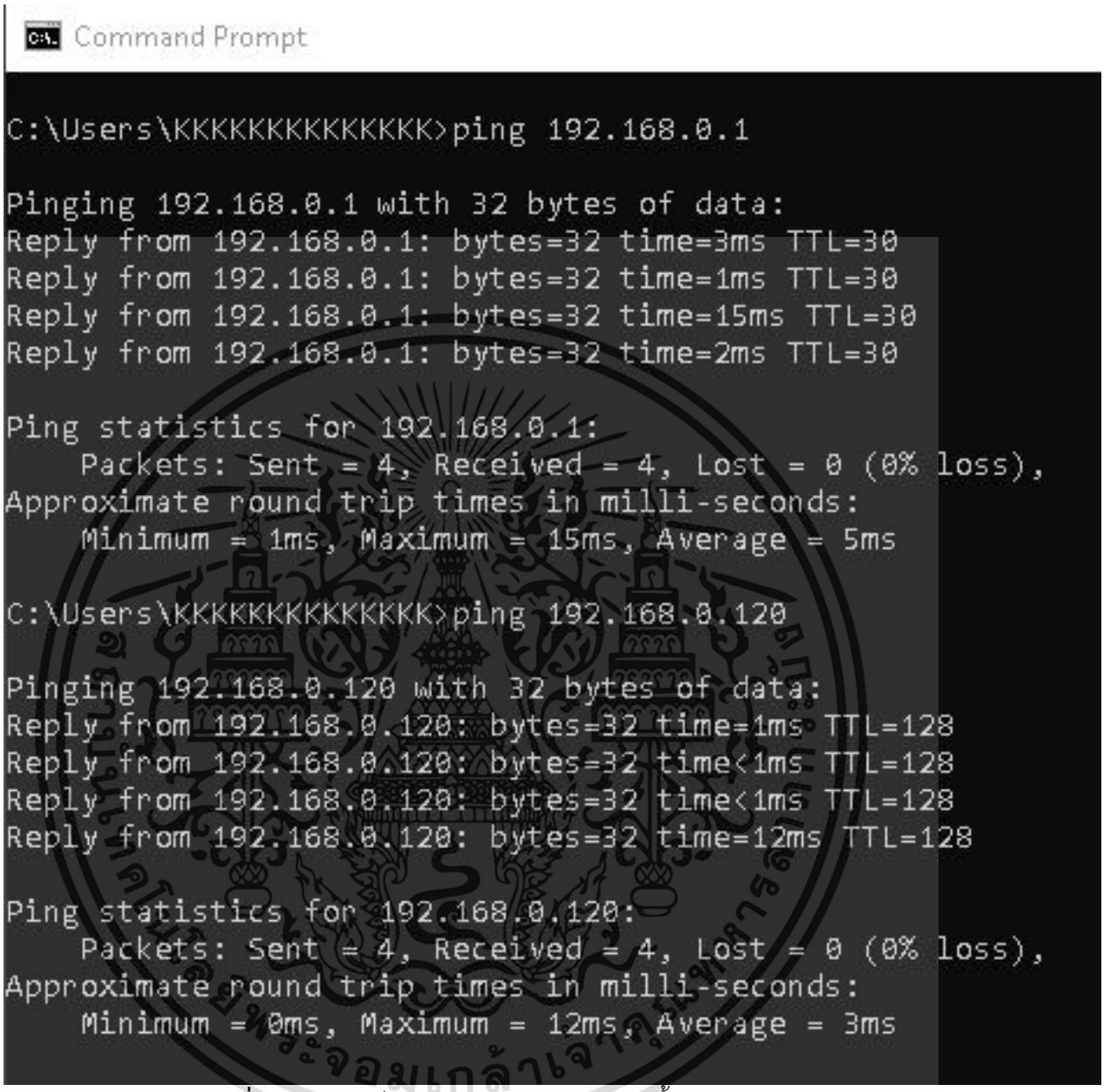
3.5.2.3 การตั้งค่า IP Address ของ Arduino Board

```
#include <SPI.h>
#include <Ethernet.h>
#include <Modbus.h>
#include <ModbusIP.h>

const int SENSOR_IREG=100;
const int sensorPin=A2;

ModbusIP mb;
long ts;

byte mac[]={ 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192,168,0,120);
IPAddress gateway(192,168,0,120);
IPAddress subnet(255,255,255,0);
```

รูปที่ 3.11 การตั้งค่า IP Address ของ Arduino Board

เริ่มจากการตั้งค่า IP Address ของอุปกรณ์ทั้งหมดให้อยู่ในวงแลนเดียวกันเพื่อให้อุปกรณ์ทั้งหมดสามารถสื่อสารถึงกันได้ โดยทำการตั้งค่า IP Address ของคอมพิวเตอร์เป็น 192.168.0.9 , ของ PLC เป็น 192.168.0.1 และ Arduino Board เป็น 192.168.0.120 ดังรูปข้างต้น

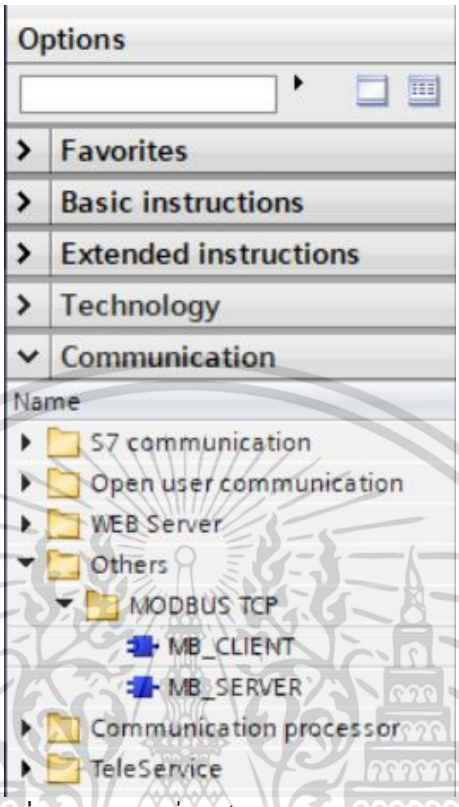
3.5.3 การเช็ค IP Address ของอุปกรณ์ภายในวงแลน



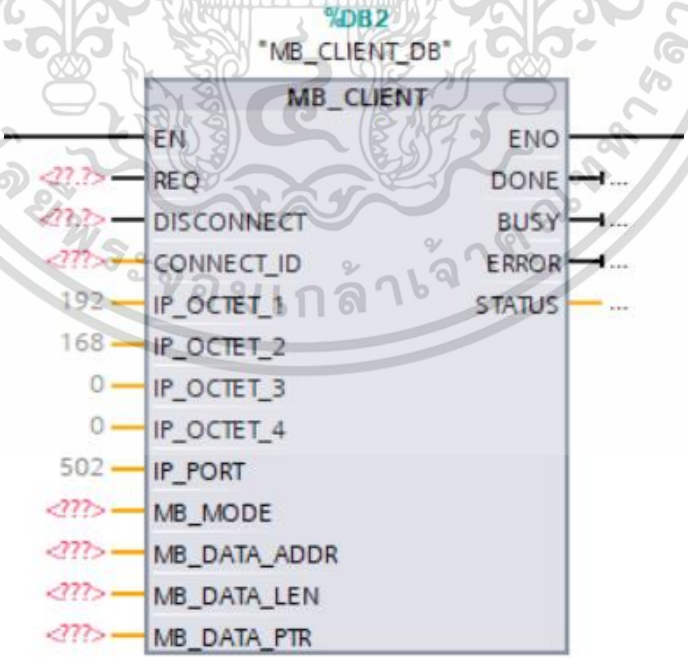
รูปที่ 3.12 การเช็ค IP Address ของอุปกรณ์ทั้งหมดภายในวงแลน

หลังจากที่ได้ทำการตั้งค่า IP Address อุปกรณ์ทั้งหมดแล้วจะทำการตรวจเช็คว่าสามารถส่งข้อมูลจากอุปกรณ์หนึ่งไปยังอีกอุปกรณ์ได้หรือไม่ โดยการตรวจเช็คจะเข้าไปยังโปรแกรมภายในคอมพิวเตอร์ ซึ่งก็คือโปรแกรม Command Prompt แล้วจึงทำการเช็ค IP Address ที่ต้องการตรวจสอบโดยการพิมพ์ ping ตามด้วย IP Address

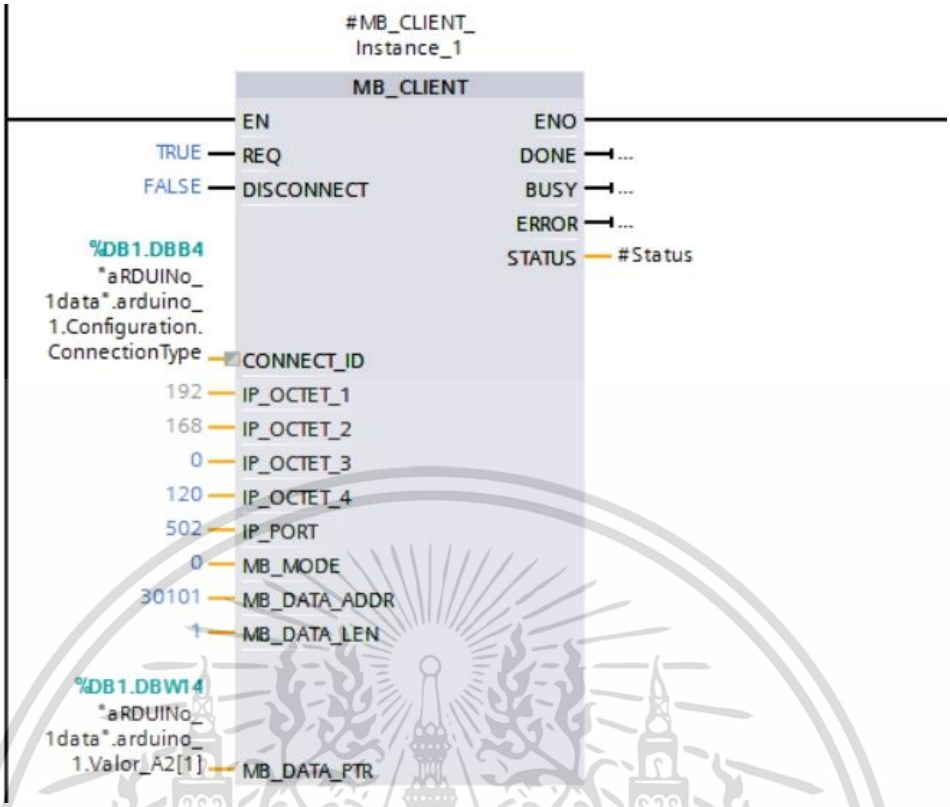
3.5.4 การตั้งค่า PLC เพื่อรับค่าจาก Arduino Board



รูปที่ 3.13 ฟังก์ชั่นบล็อก Communication



รูปที่ 3.14 ฟังก์ชั่นบล็อก MB_CLIENT



รูปที่ 3.15 ฟังก์ชั่นบล็อก MB_CLIENT (ต่อ)

เริ่มจากการสร้างฟังก์ชั่นบล็อก MB_CLIENT และทำการใส่ค่าต่างๆ ดังนี้

REQ : TRUE

DISCINNECT : FALSE

CONNECT_ID : “aRDUINO_1data”.arduino_1.Configguration.ConnectionType

IP_OCTET_1 : 192

IP_OCTET_2 : 168

IP_OCTET_3 : 0

IP_OCTET_4 : 120

IP_PORT : 502

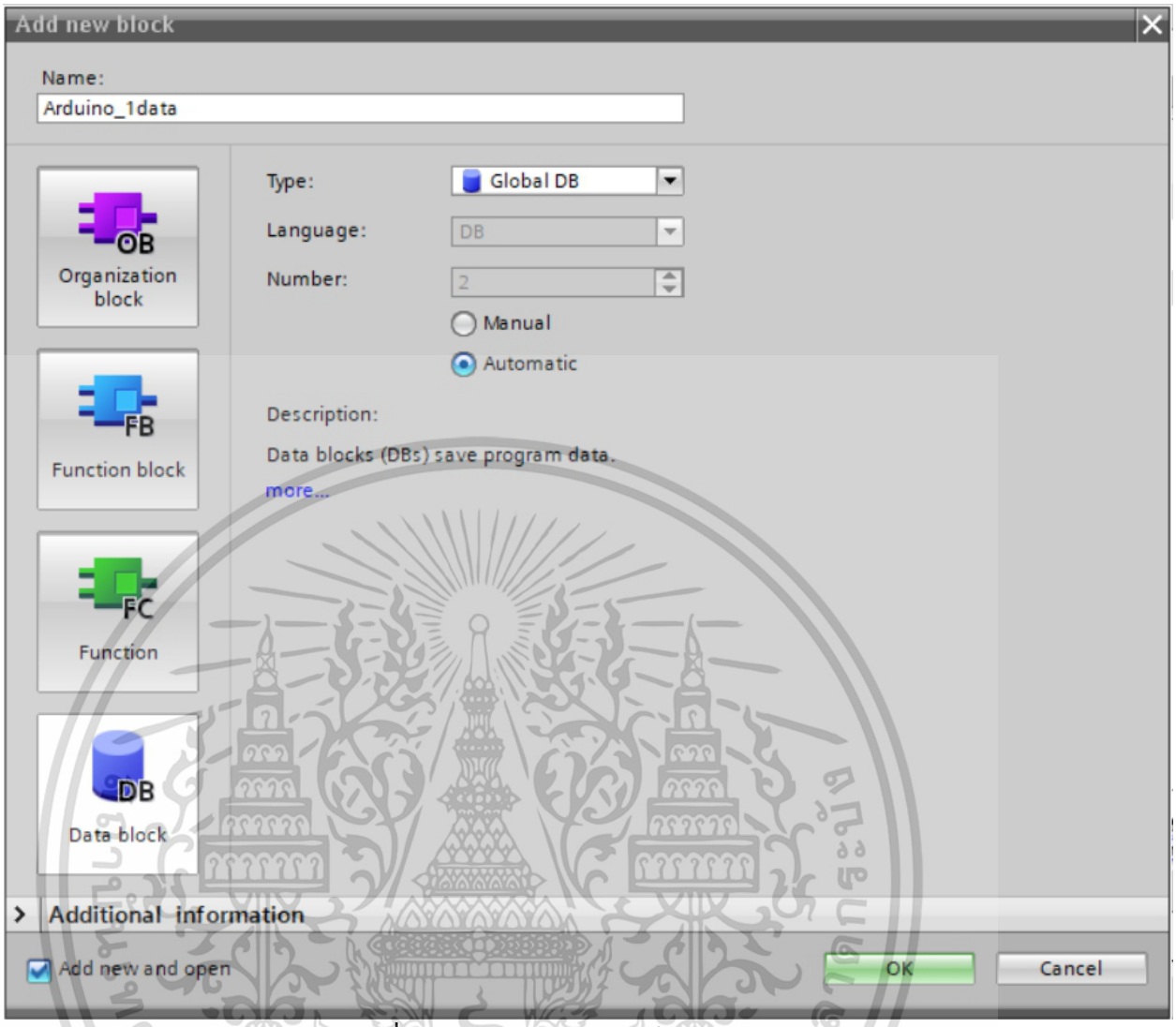
MB_MODE : 0

MB_DATA_ADDR : 30101

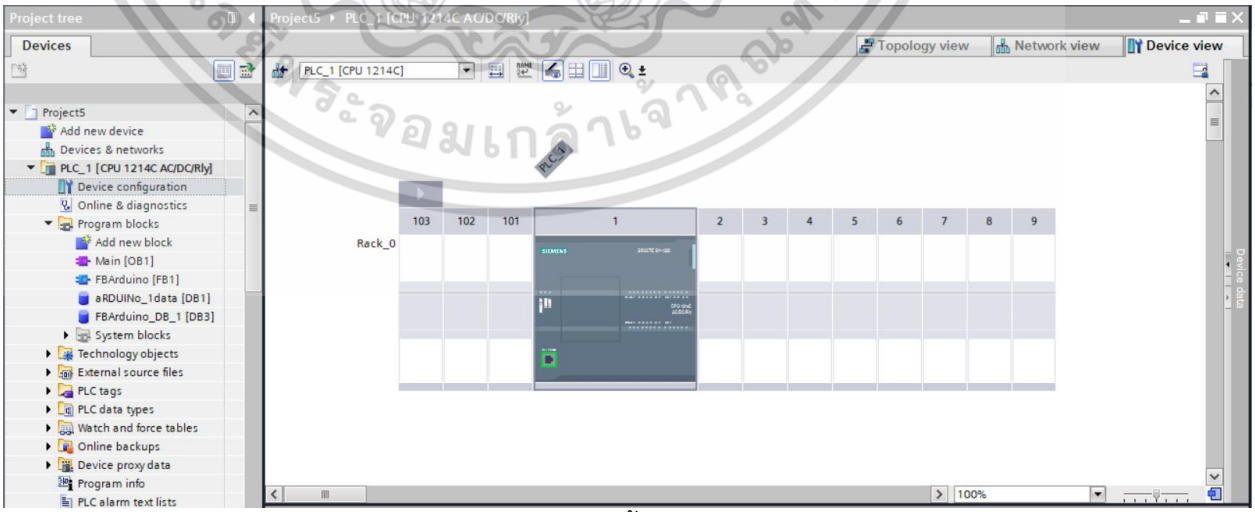
MB_DATA_LEN : 1

MB_DATA_PTR : “aRDUINO_1data”.arduino_1.ValorA2[1]

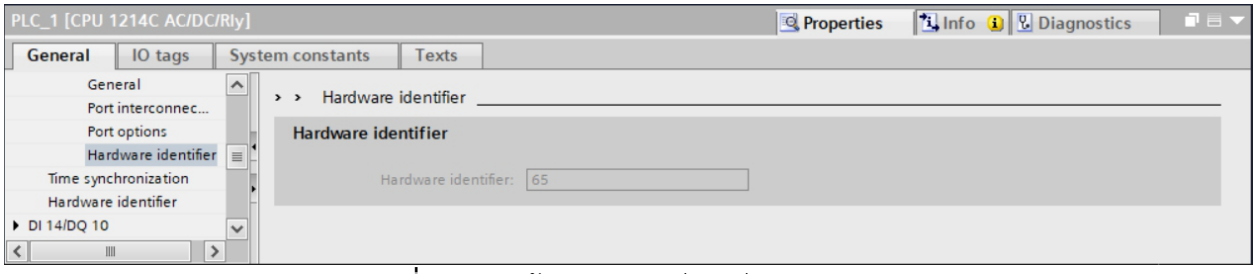
โดยที่ค่า IP_OCTET จะต้องตรงกับ IP Address ของ Arduino Board ต่อจากนั้นทำการระบุตำแหน่งที่เก็บข้อมูลที่รับได้จาก Arduino Board ไว้ที่ “aRDUINO_1data”.arduino_1.ValorA2[1]



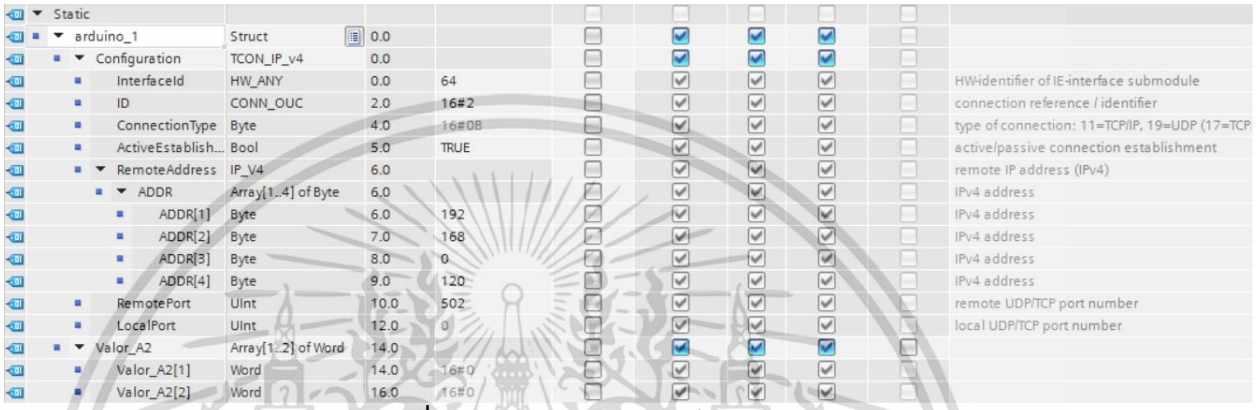
รูปที่ 3.16 การสร้างดาต้าบล็อก



รูปที่ 3.17 หน้าต่างการตั้งค่าอุปกรณ์



รูปที่ 3.18 หน้าต่างระบุฮาร์ดแวร์



รูปที่ 3.19 หน้าต่างดาต้าบล็อก

สร้างดาต้าบล็อกเพื่อเก็บค่าที่รับจาก Arduino Board โดยการสร้างตัวแปร arduino_1 Type Struct ต่อจากนั้น สร้างตัวแปร การตั้งค่า Type TCON_IP_v4 จะได้ตารางดังรูป 3.16 ต่อจากนั้นระบุค่าทั้งหมดดังนี้

InterfaceId : 64

สามารถหาค่า InterfaceId ได้จากการคลิกที่ตัว PLC ที่หน้า การตั้งค่าอุปกรณ์จะมีหน้าต่างแสดงรายละเอียดของ PLC ตัวนี้ที่ด้านล่าง จากนั้นทำการเลือกหัวข้อ ฮาร์ดแวร์ identifier

ID : 16#2

ActiveEstablished : TRUE

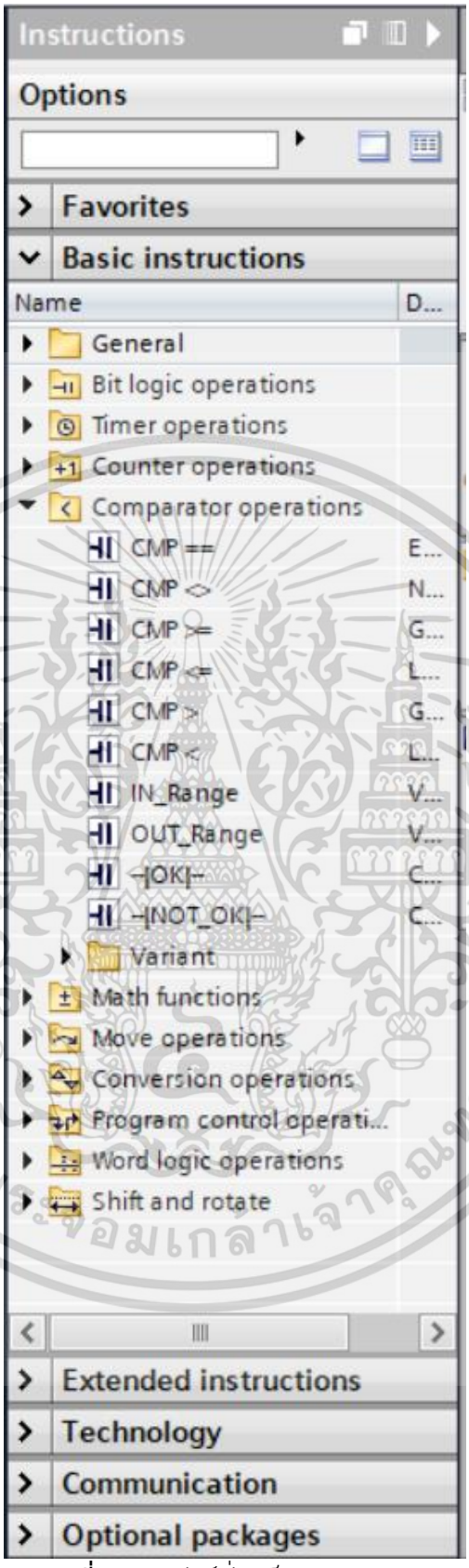
ADDR[1] : 192

ADDR[2] : 168

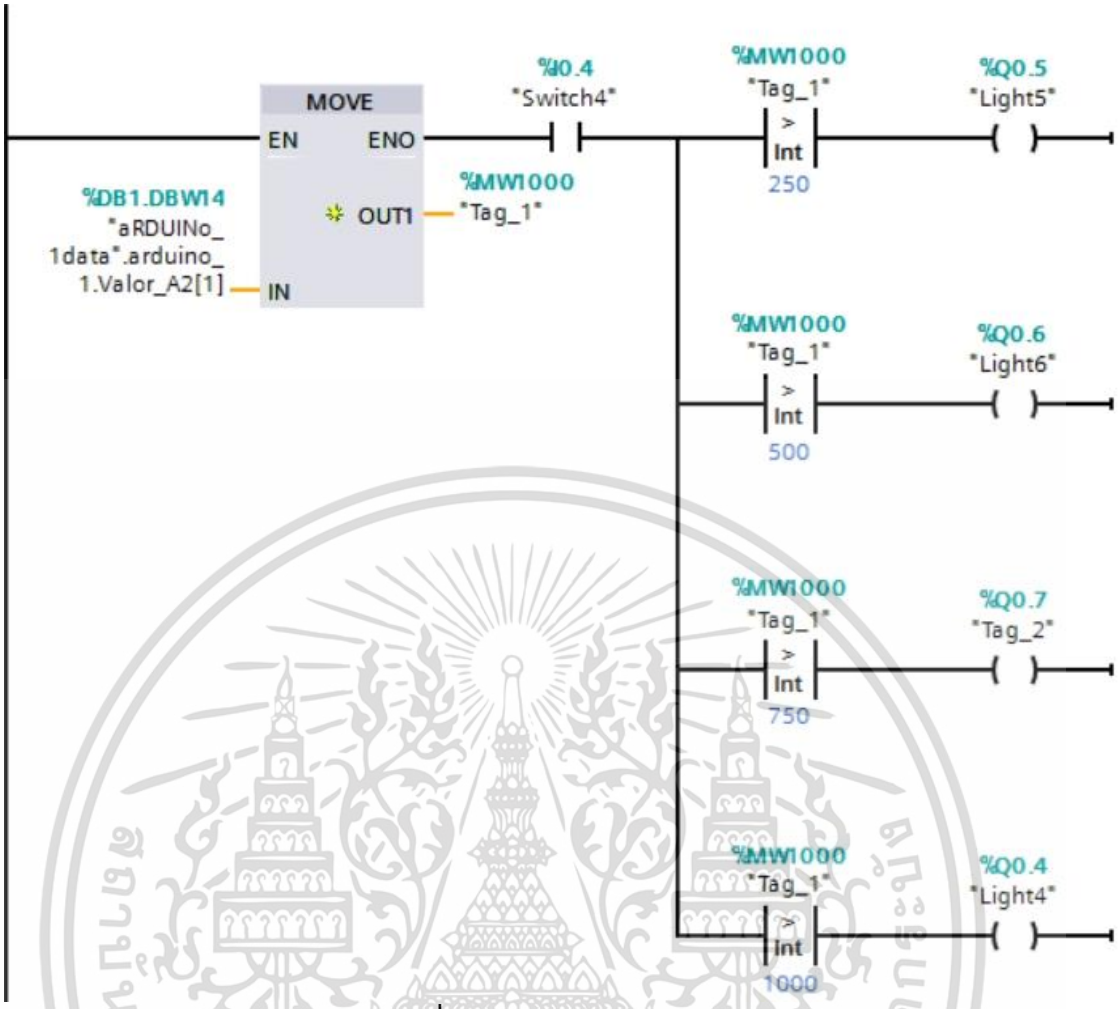
ADDR[3] : 0

ADDR[4] : 120

RemotePort : 502



รูปที่ 3.20 ฟังก์ชั่นบล็อกการทำงาน



รูปที่ 3.21 แลดเดอร์เอาต์พุต

เขียนแลดเดอร์ไดอะแกรม โดยการนำฟังก์ชั่นบล็อกการทำงานมาเขียนเป็นเงื่อนไขเพื่อนำค่าจากดาค้าบล็อกมาแสดงผลออกเป็นการเปิด-ปิดหลอดไฟโดยมีการแบ่งระดับการเปิด-ปิดไว้ตามขนาดของตัวต้านทานที่รับค่าได้จาก Arduino Board

3.5.5 การตั้งค่า Arduino Board รับค่าจาก อินพุต ส่งไปยัง PLC

```
void setup()
{
  Serial.begin(9600);
  Ethernet.begin(mac, ip, gateway, subnet);
  mb.config(mac,ip);
  mb.addIreg(SENSOR_IREG);
  ts=millis();

}

void loop() {
  mb.task();
  if(millis()>ts+2000) {
    ts=millis();
    mb.Ireg(SENSOR_IREG,analogRead(sensorPin));
    Serial.println(analogRead(sensorPin));
  }
}
```

รูปที่ 3.22 โค้ดโปรแกรมสั่งการ Arduino

เริ่มจากการเขียนโปรแกรมเพื่อรับค่าจากตัวต้านทานปรับค่าได้แล้วจึงส่งค่าออกไปผ่าน IP Address ที่ได้ทำการตั้งค่าไว้ก่อนหน้านี้

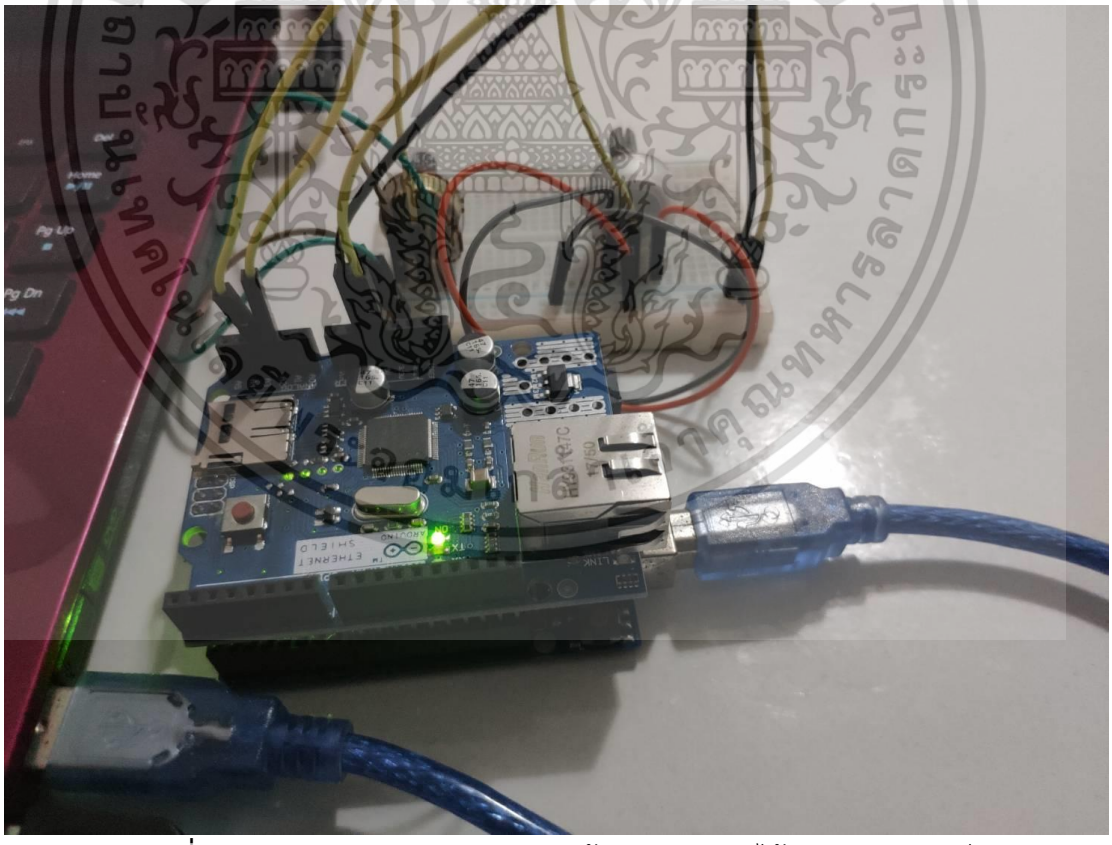
บทที่ 4
ผลการดำเนินโครงงาน

4.1 กล่าวนำ

จากบทที่ 3 ได้มีการกล่าวถึง การต่ออุปกรณ์ ส่วนควบคุม และส่วนแสดงผลต่างๆ ของกระบวนการที่จะนำมาทดลอง เพื่อการปรับปรุงระบบควบคุมและการแสดงผลให้ถูกต้องและแม่นยำ จึงต้องปรับปรุงในส่วนของฮาร์ดแวร์และซอฟต์แวร์ สำหรับเนื้อหาในบทนี้จะกล่าวถึงขั้นตอนการทดลอง และผลการทดสอบเพื่อให้ผลที่ได้จากการทดลองมีความถูกต้องแม่นยำ

4.2 ขั้นตอนการทดลอง

1. ทดลองการรับค่าตัวต้านทานปรับค่าได้ของ Arduino Board การทดลองนี้จะเริ่มทำการทดลองโดยต่อตัวกตัวต้านทานปรับค่ากับคอมพิวเตอร์และทดลองจดบันทึกตรวจสอบค่าที่ได้กับตัวโปรแกรมว่าสามารถอยู่ใน Range ของค่าที่ต้องการโดยประมาณได้หรือไม่

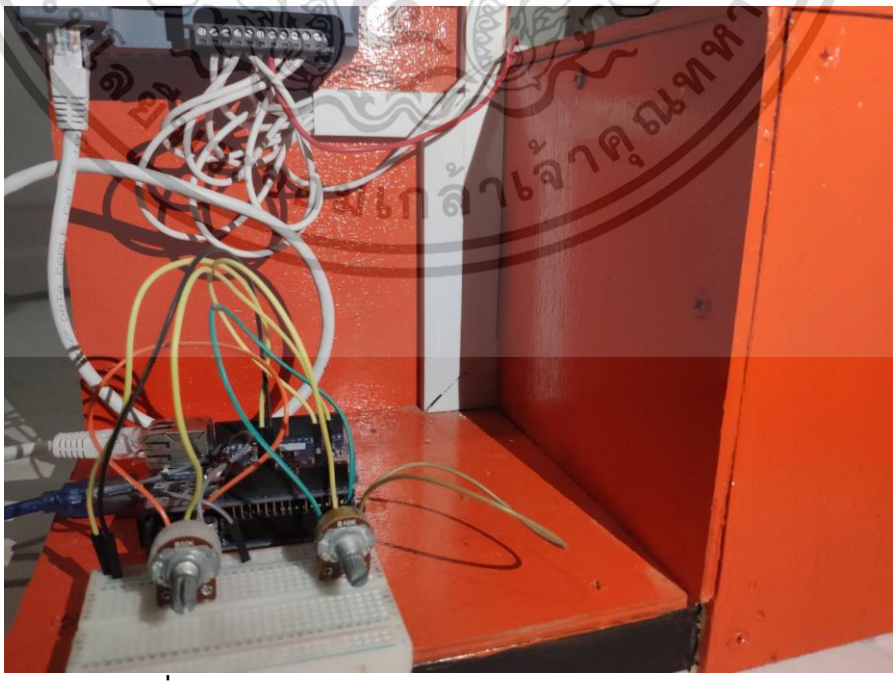


รูปที่ 4.1 ต่อสาย Arduino ระหว่างตัวต้านทานปรับค่าได้และคอมพิวเตอร์

ตารางที่ 4.1 การรับค่าตัวต้านทานปรับค่าได้ของ Arduino Board

ตัวต้านทานปรับค่าได้	ค่าที่แสดงในโปรแกรม Arduino IDE				
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5
0	0	0	0	0	0
100	102	99	102	105	104
200	200	204	204	205	201
300	297	301	303	305	305
400	402	401	402	401	396
500	503	503	500	505	501
600	601	602	602	601	604
700	702	701	701	704	702
800	805	804	799	796	802
900	897	900	902	903	903
1000	1000	1000	1000	1000	1000

2. ทดลองการรับค่าที่ส่งไปยัง PLC โดยการนำค่าที่ได้จาก Arduino ส่งไปยัง PLC ตามที่ได้เขียนไว้ใน แลดเดอร์ไดอะแกรม เพื่อตรวจสอบว่าสามารถรับค่าได้ถูกต้องแม่นยำโดยการทดลองและจดบันทึก จากนั้นเมื่อได้ค่าที่ถูกต้องแม่นยำก็นำผลลัพธ์ที่ได้ไปต่อยอดส่งค่าไปควบคุมการทำงานของไฟ LED เป็นตัวอย่างประกอบให้เห็นการควบคุมและการทำงานของกระบวนการอย่างมีประสิทธิภาพ

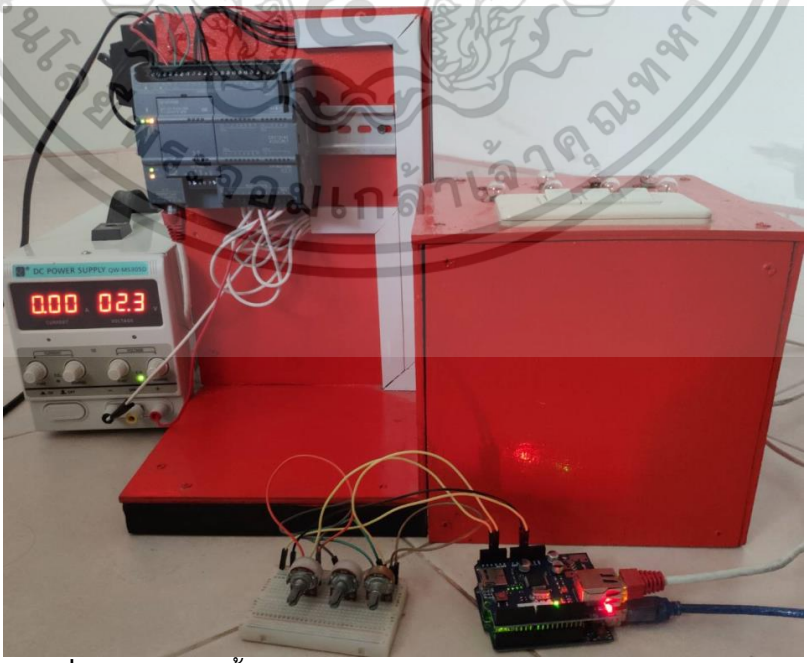


รูปที่ 4.2 ต่อสายแลนจาก Arduino Board เข้ากับตัว PLC

ตารางที่ 4.2 เปรียบเทียบค่าที่ได้รับจากตัวต้านทานปรับค่าได้ส่งไปยัง PLC

ครั้งที่ 1		ครั้งที่ 2		ครั้งที่ 3		ครั้งที่ 4		ครั้งที่ 5	
ค่าจาก Arduino	ค่าที่ PLC	ค่าจาก Arduino	ค่าที่ PLC	ค่าจาก Arduino	ค่าที่ PLC	ค่าจาก Arduino	ค่าที่ PLC	ค่าจาก Arduino	ค่าที่ PLC
0	0	0	0	0	0	0	0	0	0
101	101	101	101	105	105	104	104	102	102
203	203	204	204	198	198	199	199	201	201
296	296	297	297	295	295	302	302	305	305
403	403	401	401	401	401	405	405	396	396
502	502	502	502	499	499	505	505	504	504
605	605	605	605	605	605	601	601	603	603
702	702	703	703	697	697	705	705	703	703
795	795	804	804	803	803	805	805	801	801
902	902	901	901	902	902	901	901	897	897
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

3. ทดลองการแสดงผลของไฟ LED โดยการทดลองผลจากการเขียนแลดเดอร์ไดอะแกรม เพื่อสั่งการให้ไฟ LED ทำงานโดยอิงกับค่าตัวต้านทานปรับค่าได้ ที่ได้รับมาจากตัว PLC และตั้งระบบสวิตช์เปิดปิดเพื่อจำลองเป็นคัทเอาท์ตัดไฟในกระบวนการ



รูปที่ 4.3 ต่อสายทั้งหมดรวมกันระหว่าง Arduino PLC และไฟ LED

ตารางที่ 4.3 การแสดงผลของหลอดไฟ

ช่วงของค่าตัวต้านทานปรับค่าได้ที่ทดลอง	ค่าจาก Arduino	ค่าที่ PLC	หากทำการเปิด Switch จะมีหลอดไฟติด	หากทำการปิด Switch จะมีหลอดไฟติด
0	0	0	0 ดวง	0 ดวง
100	99	99	0 ดวง	0 ดวง
200	201	201	0 ดวง	0 ดวง
300	305	305	1 ดวง	0 ดวง
400	402	402	1 ดวง	0 ดวง
500	505	505	2 ดวง	0 ดวง
600	596	596	2 ดวง	0 ดวง
700	697	697	2 ดวง	0 ดวง
800	803	803	3 ดวง	0 ดวง
900	904	904	3 ดวง	0 ดวง
1000	1000	1000	4 ดวง	0 ดวง

4.3 ผลการทดลอง

จากการทดลองได้ตรวจสอบค่าที่ได้มา 3 ค่า คือ ค่าตัวต้านทานปรับค่าได้ที่ Arduino อ่านได้ ค่าที่ Arduino ส่งให้ แลดเดอร์ไดอะแกรม และ การแสดงผลของหลอดไฟในตัวกล่อง พบว่าทุกอย่างออกมาตรงกัน ซึ่งหมายถึง โค้ด ของ Arduino , การตั้งค่าการเชื่อมต่อของอุปกรณ์ทั้ง 4 ชิ้น , การต่อสายไฟในส่วน ฮาร์ดแวร์ , การเขียน แลดเดอร์ไดอะแกรม สำหรับ Modbus TCP/IP ทั้งหมดนั้นถูกต้องจึงได้ผลลัพธ์ทั้งหมด ออกมาตามที่ดำเนินการไว้ ยกตัวอย่างเช่น หาก Arduino อ่านตัวต้านทานปรับค่าได้ 500 ก็จะส่งข้อมูลต่อไปในคอมพิวเตอร์ และ PLC ซึ่งค่าในคอมพิวเตอร์ที่อ่านได้จากโปรแกรม Arduino ก็อ่านได้ 500 จริง รวมทั้งค่าใน แลดเดอร์ไดอะแกรม ก็แสดงค่าออกมา 500 และหากตั้งค่าให้ ค่า 500 ทำให้ไฟติด 2 ดวง (%Q0.5 , %Q0.6) พบว่าทั้งไฟใน PLC ในส่วนของเอาต์พุต ก็ติด 2 ดวง(%Q0.5 , %Q0.6)และ หลอดไฟในกล่องทดลอง ก็ติด 2 ดวง (Light5 , Light6) ด้วยเช่นกัน แต่หากทำการ ปิด Switch4 , %Q0.4 ก็จะทำให้ไฟทั้งหมดดับลง

บทที่ 5
ผลสรุปและข้อเสนอแนะ

5.1 ผลสรุป

จากการดำเนินการศึกษาและทดลองในส่วนของ โปรโตคอลชนิด Modbus TCP/IP รวมถึงตัว PLC S7-1200 และโปรแกรม TIA Portal V.14 ที่จำเป็นต้องใช้เกี่ยวเนื่องกันเพื่อทำการสั่งการทำงาน ควบคุมและแสดงผลจากค่าที่ได้รับจากตัวต้านทานปรับค่าได้ซึ่งได้จำลองให้เป็นเหมือน เซนเซอร์ที่รับค่าและส่งไปยังส่วนควบคุมซึ่งก็คือ PLC โดยใช้ตัวโปรโตคอล Modbus TCP/IP ในการสื่อสาร และนำค่าที่ได้ไปต่อยอดควบคุมไฟ LED ให้แสดงผลเหมือนการ เปิด/ปิด วาล์วหรือมอเตอร์ จากที่กล่าวมาข้างต้นทำให้เข้าใจหลักการทำงานและหลักการดำเนินงาน เพื่อนำไปประยุกต์ใช้งาน จึงนับว่ามีประโยชน์อย่างมากเมื่อต้องการศึกษาหลักการทำงานและการนำไปใช้งานจริง ของ Modbus TCP/IP และการเขียนคำสั่ง แลดเดอร์ไดอะแกรม ของบริษัท Siemens ที่จำเป็นต้องใช้โปรแกรม TIA Portal V.14 ซึ่งเมื่อเข้าใจถึงหลักการทำงาน การติดตั้ง การใช้งาน และที่มาของ Modbus TCP/IP ก็จะสามารถนำความรู้ที่ได้ไปใช้ต่อยอดในอนาคต

5.2 ปัญหาและวิธีการแก้ไขปัญหา

5.2.1 ปัญหาที่พบ

1. ข้อมูลและเอกสารไม่ครบถ้วน ทำให้เกิดความล่าช้าและติดขัด
2. การใช้งานโปรแกรม TIA Portal V.14 ต้องใช้เวลาในการศึกษาและทดลองใช้ค่อนข้างมาก
3. ไม่มีความรู้ในการเขียนแลดเดอร์ไดอะแกรม ในส่วนของ Modbus TCP/IP และนำมาประยุกต์ใช้ให้ถูกต้องกับรุ่นของตัว PLC S7-1200
4. ไม่มีความรู้ในการเขียนโค้ดเพื่อกำหนด Address ในส่วนของ Arduino

5.2.2 วิธีการแก้ไขปัญหา

1. ข้อมูลบางอย่างศึกษาโดยการทดลองใช้จริงด้วยตนเอง
2. เมื่อติดปัญหาในการใช้งานก็นำปัญหาที่พบไปปรึกษาอาจารย์ เพื่อให้อาจารย์ชี้ช่องทางการแก้ปัญหาทำให้ร่นระยะเวลาในการศึกษาและทดลองใช้
3. ศึกษาเพิ่มเติมจาก Manual book ของ PLC S7-1200 และทดลองเขียน
4. ศึกษาเพิ่มเติมจาก https://github.com

5.3 ข้อเสนอแนะ

งานวิจัยนี้สามารถนำมาใช้ศึกษารายละเอียดหลักการทำงานและการนำ Modbus TCP/IP ไปใช้งานควบคู่กับ PLC Siemens S7-1200 และสามารถใช้การทดลองในหนังสือเล่มนี้ในการฝึกเขียนแลดเดอร์ไดอะแกรม และใช้งานโปรแกรม TIA Portal ให้เข้าใจหลักการทำงานอีกทั้งยังสามารถนำความรู้ที่ได้จากงานวิจัยชิ้นนี้ไปต่อยอดในการควบคุมกระบวนการต่างๆ ที่ใช้ Modbus TCP/IP และ PLC Siemens S7-1200 ด้วยการนำความรู้จากงานชิ้นนี้ไปประยุกต์ใช้ให้เกิดประโยชน์

บรรณานุกรม

[1] andresrmento, “Modbus Library for Arduino”, Nov 11 2015. [Online]
Available : https://github.com/andresarmento/modbus-arduino (Jan 2020)

[2] S7-1200 Programmable controller System Manual, Mar 2014, A5E02486680-AG

[3] Automation 360, “Device configuration”, Jan 19 2017. [Online]
Available : https://automation360blog.wordpress.com/plc/ (Mar 2019)

[4] Myarduino., “การใช้งานตัวต้านทานปรับค่าได้กับ Arduino”, 2562.[ระบบออนไลน์]
แหล่งที่มา : https://www.myarduino.net/article/90/ (มกราคม 2563)

[5] paratrasnet, “S7-1200 Data Sheet”, 2010. [Online]
Available : https://www.paratrasnet.ro/pdf/automatizari-industriale/S7-1200.pdf
(Sep 2019)

[6] sonicautomation, “การสื่อสารผ่าน Modbus TCP/IP”, 4 พฤศจิกายน 2562.
[ระบบออนไลน์] แหล่งที่มา : https://sonicautomation.co.th/2019/ (มกราคม 2563)

[7] riverplus, “Modbus Protocol” , Aug 18 2011. [Online]
Available : https://riverplus.com/2011/08/18/ (Aug 2019)

[8] thaieasyelec, “What is arduino”, Mar 11 2017. [Online]
Available : https://www.thaieasyelec.com/article-wiki/latest-blogs/what-is-arduino-ch1.html (Aug 2019)

ภาคผนวก

Overview

- The compact high-performance CPU
- With 24 integral input/outputs
- Expandable by:
  - 1 signal board (SB)
  - 8 signal modules (SM)
  - max. 3 communication modules (CM)

Design

The compact CPU 1214C has:

- 3 device versions with different power supply and control voltages
- Integrated power supply either as wide-range AC or DC power supply (85 to 264 V AC or 24 V DC)
- Integrated 24 V encoder/load current supply:
  For direct connection of sensors and encoders. With 400 mA, the output current can also be used as load power supply
- 14 integrated digital inputs 24 V DC (current sinking/current sourcing (IEC type 1 current sinking))
- 10 integrated digital outputs, either 24 V DC or relay
- 2 integrated analog inputs 0 to 10 V
- 2 pulse outputs (PTO) with a frequency of up to 100 kHz
- Pulse-width modulated outputs (PWM) with a frequency of up to 100 kHz
- Integrated Ethernet interface (TCP/IP native, ISO-on-TCP)
- 6 fast counters (3 with max. 100 kHz; 3 with max. 30 kHz), with parameterizable enable and reset inputs, can be used simultaneously as up and down counters with 2 separate inputs or for connecting incremental encoders
- Expansion by additional communication interfaces, e.g. RS485 or RS232
- Expansion by analog or digital signals directly on the CPU via signal board (with retention of CPU mounting dimensions)
- Expansion by a wide range of analog and digital input and output signals via signal modules
- Optional memory expansion (SIMATIC Memory Card)
- PID controller with auto-tuning functionality
- Integral real-time clock
- Interrupt inputs:
  For extremely fast response to rising or falling edges of process signals
- Removable terminals on all modules
- Simulator (optional):
  For simulating the integrated inputs and for testing the user program

Device versions

Version	Supply voltage	Input voltage DI	Output voltage DO	Output current
• DC/DC/DC	24 V DC	24 V DC	24 V DC	0.5 A, transistor
• DC/DC/relay	24 V DC	24 V DC	5 ... 30 V DC / 5 ... 250 V AC	2 A; 30 W DC / 200 W AC
• AC/DC/relay	85 ... 264 V AC	24 V DC	5 ... 30 V DC / 5 ... 250 V AC	2 A; 30 W DC / 200 W AC

SIMATIC S7-1200
Central processing units

CPU 1214C

Function

- Comprehensive instruction set:
  A wide range of operations facilitate programming:
  - basic operations such as binary logic operations, result allocation, save, count, create times, load, transfer, compare, shift, rotate, create complement, call subprogram (with local variables)
  - integral communication commands (e.g. USS protocol, Modbus RTU, S7 communication "T-Send/T-Receive" or Freeport)
  - user-friendly functions such as pulse-width modulation, pulse sequence function, arithmetic functions, floating point arithmetic, PID closed-loop control, jump functions, loop functions and code conversions
  - mathematical functions, e.g. SIN, COS, TAN, LN, EXP
- Counting:
  User-friendly counting functions in conjunction with the integrated counters and special commands for high-speed counters open up new application areas for the user
- Interrupt processing:
  - edge-triggered interrupts (activated by rising or falling edges of process signals on interrupt inputs) support a rapid response to process events
  - time-triggered interrupts
  - counter interrupts can be triggered when a setpoint is reached or when the direction of counting changes
  - communication interrupts allow the rapid and easy exchange of information with peripheral devices such as printers or bar code readers
- Password protection
- Test and diagnostics functions:
  Easy-to-use functions support testing and diagnostics, e.g. online/offline diagnostics
- "Forcing" of inputs and outputs during testing and diagnostics:
  Inputs and outputs can be set independently of cycle and thus permanently, for example, to test the user program
- Motion Control in accordance with PLCopen for simple movements
- Library functionality

Programming

The STEP 7 Basic programming package permits complete programming of all S7-1200 controllers and the associated I/O.

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Product version			
associated programming package	STEP 7 Basic V10.5	STEP 7 Basic V10.5	STEP 7 Basic V10.5
Supply voltages			
Rated value			
• 24 V DC		Yes	Yes
• permissible range, lower limit (DC)		20.4 V	20.4 V
• permissible range, upper limit (DC)		28.8 V	28.8 V
• 120 V AC	Yes		
• 230 V AC	Yes		
• permissible range, lower limit (AC)	85 V		
• permissible range, upper limit (AC)	264 V		
• permissible frequency range, lower limit	47 Hz		
• permissible frequency range, upper limit	63 Hz		
Load voltage L+			
• Rated value (DC)	24 V	24 V	24 V
• permissible range, lower limit (DC)	5 V	20.4 V	5 V
• permissible range, upper limit (DC)	250 V	28.8 V	250 V
Current consumption			
Current consumption (rated value)	100 mA at 120 VAC 50 mA at 240 VAC		500 mA; Typical
Current consumption, max.	300 mA at 120 VAC 150 mA at 240 VAC	1.5 A; 24 VDC	1.2 A; 24 VDC
Inrush current, max.	20 A; at 264 V	12 A; at 28.8 V	12 A; at 28.8 V
Current output to backplane bus (DC 5 V), max.	1 600 mA; 5 V DC max. for SM and CM	1 600 mA; 5 V DC max. for SM and CM	1 600 mA; 5 V DC max. for SM and CM
Power loss			
Power loss, typ.	14 W	12 W	12 W
Memory			
Available project memory/user memory	50 Kibyte	50 Kibyte	50 Kibyte

Technical specifications (continued)

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Work memory			
• integrated	50 kbyte	50 kbyte	50 kbyte
• expandable	No	No	No
Load memory			
• integrated	2 Mbyte	2 Mbyte	2 Mbyte
• expandable	24 Mbyte; with SIEMENS Memory Card	24 Mbyte; with SIEMENS Memory Card	24 Mbyte; with SIEMENS Memory Card
Backup			
• present	Yes; entire project maintenance-free in the integral EEPROM	Yes; entire project maintenance-free in the integral EEPROM	Yes; entire project maintenance-free in the integral EEPROM
• without battery	Yes	Yes	Yes
CPU/ blocks			
Number of blocks (total)	DBs, FCs, FBs, counters, timers). Up to 65,535 blocks can be addressed. There is no limit, use of the entire work memory	DBs, FCs, FBs, counters, timers). Up to 65,535 blocks can be addressed. There is no limit, use of the entire work memory	DBs, FCs, FBs, counters, timers). Up to 65,535 blocks can be addressed. There is no limit, use of the entire work memory
OB			
• Number, max.	Limited only by RAM for code	Limited only by RAM for code	Limited only by RAM for code
CPU/ processing times			
for bit operations, min.	0.1 µs; / instruction	0.1 µs; / instruction	0.1 µs; / instruction
for word operations, min.	12 µs; / instruction	12 µs; / instruction	12 µs; / instruction
for floating point arithmetic, min.	18 µs; / instruction	18 µs; / instruction	18 µs; / instruction
Data areas and their retentivity			
retentive data area in total (incl. times, counters, flags), max.	2 048 byte	2 048 byte	2 048 byte
Flag			
• Number, max.	8 kbyte; Size of bit memory address area	8 kbyte; Size of bit memory address area	8 kbyte; Size of bit memory address area
Address area			
I/O address area			
• I/O address area, overall	1024 bytes for inputs / 1024 bytes for outputs	1024 bytes for inputs / 1024 bytes for outputs	1024 bytes for inputs / 1024 bytes for outputs
• overall	1 024 byte	1 024 byte	1 024 byte
• Outputs	1 024 byte	1 024 byte	1 024 byte
Process image			
• Inputs, adjustable	1 kbyte	1 kbyte	1 kbyte
• Outputs, adjustable	1 kbyte	1 kbyte	1 kbyte
Digital channels			
• integrated channels (DI)	14	14	14
• integrated channels (DO)	10	10	10
Analog channels			
• Integrated channels (AI)	2	2	2
• Integrated channels (AO)	0	0	0
Hardware configuration			
Number of modules per system, max.	3 comm. modules, 1 signal board, 8 signal modules	3 comm. modules, 1 signal board, 8 signal modules	3 comm. modules, 1 signal board, 8 signal modules

Technical specifications (continued)

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Time of day			
Clock			
• Hardware clock (real-time clock)	Yes	Yes	Yes
• Backup time	240 h; Typical	240 h; Typical	240 h; Typical
• Deviation per day, max.	60 s/month at 25°C	60 s/month at 25°C	60 s/month at 25°C
Test commissioning functions			
Status/control			
• Status/control variable	Yes	Yes	Yes
• Variables	Inputs/outputs, memory bits, DB, distributed I/Os, timers, counters	Inputs/outputs, memory bits, DB, distributed I/Os, timers, counters	Inputs/outputs, memory bits, DB, distributed I/Os, timers, counters
Forcing			
• Forcing	Yes	Yes	Yes
Communication functions			
S7 communication			
• supported	Yes	Yes	Yes
• as server	Yes	Yes	Yes
Open IE communication			
• TCP/IP	Yes	Yes	Yes
• ISO-on-TCP (RFC1006)	Yes	Yes	Yes
Number of connections			
• overall	15; dynamically	15; dynamically	15; dynamically
1st interface			
Type of interface	PROFINET	PROFINET	PROFINET
Physics	Ethernet	Ethernet	Ethernet
Isolated	Yes	Yes	Yes
automatic detection of transmission speed	Yes	Yes	Yes
Autonegotiation	Yes	Yes	Yes
Autocrossover	Yes	Yes	Yes
CPU/ programming			
Configuration software			
• STEP 7	STEP 7 Basic V10.5	STEP 7 Basic V10.5	STEP 7 Basic V10.5
Programming language			
• LAD	Yes	Yes	Yes
• FBD	Yes	Yes	Yes
Cycle time monitoring			
• can be set	Yes	Yes	Yes
Digital inputs			
Number of digital inputs	14; Integrated	14; Integrated	14; Integrated
• of which, inputs usable for technological functions	6; HSC (High Speed Counting)	6; HSC (High Speed Counting)	6; HSC (High Speed Counting)
m/p-reading	Yes	Yes	Yes
Number of simultaneously controllable inputs			
• Mounting position			
- Concurrently controllable inputs, up to 40 °C	14	14	14

Technical specifications (continued)

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Input voltage			
• Rated value, DC	24 V	24 V	24 V
• for signal "0"	5 V DC at 1 mA	5 V DC at 1 mA	5 V DC at 1 mA
• for signal "1"	15 V DC at 2.5 mA	15 V DC at 2.5 mA	15 V DC at 2.5 mA
Input current			
• for signal "1", typ.	1 mA	1 mA	1 mA
Input delay (for rated value of input voltage)			
• for standard inputs			
- parameterizable	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms, selectable in groups of four	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms, selectable in groups of four	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms, selectable in groups of four
- at "0" to "1", min.	0.2 ms	0.2 ms	0.2 ms
- at "0" to "1", max.	12.8 ms	12.8 ms	12.8 ms
• for interrupt inputs			
- parameterizable	Yes	Yes	Yes
• for counter/technological functions			
- parameterizable	Single phase : 3 at 100 kHz, 3 at 30 kHz differential: 3 at 80 kHz, 3 at 30 kHz	Single phase : 3 at 100 kHz, 3 at 30 kHz differential: 3 at 80 kHz, 3 at 30 kHz	Single phase : 3 at 100 kHz, 3 at 30 kHz differential: 3 at 80 kHz, 3 at 30 kHz
Cable length			
• Cable length, shielded, max.	500 m; 50 m for technological functions	500 m; 50 m for technological functions	500 m; 50 m for technological functions
• Cable length unshielded, max.	300 m; For technological functions: No	300 m; For technological functions: No	300 m; For technological functions: No

Digital outputs			
Number of digital outputs	10; Relay	10	10; Relay
• of which high-speed outputs		2; 100 kHz Pulse Train Output	
Short-circuit protection	No; to be provided externally	No; to be provided externally	No; to be provided externally
Limitation of inductive shutdown voltage to		L+ (-48 V)	
Switching capacity of the outputs			
• with resistive load, max.	2 A	0.5 A	2 A
• on lamp load, max.	30 W DC; 200 W AC	5 W	30 W DC; 200 W AC
Output voltage			
• for signal "1", min.		20 V	
Output current			
• for signal "1" rated value		0.5 A	
• for signal "0" residual current, max.		0.1 mA	
Output delay with resistive load			
• 0 to "1", max.	10 ms; max.	1 µs	10 ms; max.
• 1 to "0", max.	10 ms; max.	5 µs	10 ms; max.
Switching frequency			
• of the pulse outputs, with resistive load, max.	1 Hz	100 kHz	1 Hz
Cable length			
• Cable length, shielded, max.	500 m	500 m	500 m
• Cable length unshielded, max.	150 m	150 m	150 m

Technical specifications (continued)

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Relay outputs			
Number of relay outputs	10		10
Number of operating cycles	mechanically 10 million, at rated load voltage 100,000		mechanically 10 million, at rated load voltage 100,000
Analog inputs			
Number of analog inputs	2	2	2
Cable length, shielded, max.	100 m; twisted and shielded	100 m; twisted and shielded	100 m; twisted and shielded
Input ranges			
• Voltage	Yes	Yes	Yes
Input ranges (rated values), voltages			
• 0 to +10 V	Yes	Yes	Yes
• Input resistance (0 to 10 V)	≥100 kohms	≥100 kohms	≥100 kohms
Analog value creation			
Integrations and conversion time/ resolution per channel			
• Resolution with overrange (bit including sign), max.	10 bit	10 bit	10 bit
• Integration time, parameterizable	Yes	Yes	Yes
• Conversion time (per channel)	625 µs	625 µs	625 µs
Formation of analog values (in isochronous mode)			
Cable length			
• Max. cable length, shielded	10 m; twisted	10 m; twisted	10 m; twisted
Encoder supply			
24 V encoder supply			
• 24 V	permissible range: 20.4 to 28.8 V	permissible range: 20.4 to 28.8 V	permissible range: 20.4 to 28.8 V
Encoder			
Connectable encoders			
• 2-wire BEROS	Yes	Yes	Yes
Integrated Functions			
Number of counters	6	6	6
Counter frequency (counter) max.	100 kHz	100 kHz	100 kHz
Frequency meter	Yes	Yes	Yes
controlled positioning	Yes	Yes	Yes
PID controller	Yes	Yes	Yes
Number of alarm inputs	4	4	4
Number of pulse outputs		2	
Limit frequency (pulse)		100 kHz	
Operator control and monitoring			
Display			
• integrated	No	No	No
Galvanic isolation			
Galvanic isolation digital inputs			
• Galvanic isolation digital inputs	500 V AC for 1 minute	500 V AC for 1 minute	500 V AC for 1 minute
• between the channels, in groups of	1	1	1

Technical specifications (continued)

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Galvanic isolation digital outputs			
• Galvanic isolation digital outputs	Relays	Yes	Relays
• between the channels	No	No	No
• between the channels, in groups of	2	2	1
Permissible potential difference			
between different circuits	500 V DC between 24 V DC and 5 V DC	500 V DC between 24 V DC and 5 V DC	500 V DC between 24 V DC and 5 V DC
EMC			
Interference immunity against discharge of static electricity			
• Interference immunity against discharge of static electricity acc. to IEC 61000-4-2	Yes	Yes	Yes
- Test voltage with air discharge	8 kV	8 kV	8 kV
- Test voltage with contact discharge	6 kV	6 kV	6 kV
Interference immunity to cable-borne interference			
• on the supply lines acc. to IEC 61000-4-4	Yes	Yes	Yes
• Interference immunity on signal lines acc. to IEC 61000-4-4	Yes	Yes	Yes
Immunity to surge voltages			
• on the supply lines acc. to IEC 61000-4-5	Yes	Yes	Yes
Immunity to conducted interference, induced by high-frequency fields			
• Interference immunity against high-frequency radiation acc. to IEC 61000-4-6	Yes	Yes	Yes
Emission of radio interference in accordance with EN 55 011			
• Emission of radio interferences acc. to EN 55 011 (limit class A)	Yes; Group 1	Yes; Group 1	Yes; Group 1
• Emission of radio interference acc. to EN 55 011 (limit class B)	Yes	Yes	Yes
Climatic and mechanical conditions for storage and transport			
Climatic conditions for storage and transport			
• Free fall			
- Max. height of fall (in packaging)	0.3 m; five times, in shipping package	0.3 m; five times, in shipping package	0.3 m; five times, in shipping package
• Temperature			
- permissible temperature range	-40 °C ... +70 °C	-40 °C ... +70 °C	-40 °C ... +70 °C
• Relative humidity			
- permissible range (without condensation) at 25 °C	95%	95%	95%

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Mechanical and climatic conditions during operation			
Climatic conditions during operation			
• Temperature			
- permissible temperature range	0 °C ... 55 °C when horizontally mounted 0 °C ... 45 °C when vertically mounted	0 °C ... 55 °C when horizontally mounted 0 °C ... 45 °C when vertically mounted	0 °C ... 55 °C when horizontally mounted 0 °C ... 45 °C when vertically mounted
- permissible temperature change	5 °C ... 55 °C, 3 °C/min	5 °C ... 55 °C, 3 °C/min	5 °C ... 55 °C, 3 °C/min
• Atmospheric pressure acc. to IEC 60068-2-13			
- permissible atmospheric pressure	1080 ... 795 hPa	1080 ... 795 hPa	1080 ... 795 hPa
- permissible operating altitude	-1000m ... 2000m	-1000m ... 2000m	-1000m ... 2000m
• Concentration of pollutants			
- SO_2 at RH < 60% without condensation	< 0.5 ppm	< 0.5 ppm	< 0.5 ppm
- H_2S at RH < 60% without condensation	< 0.1 ppm	< 0.1 ppm	< 0.1 ppm
Environmental requirements			
Operating temperature			
• Min.	0 °C	0 °C	0 °C
• max.	55 °C	55 °C	55 °C
• vertical installation, min.	0 °C	0 °C	0 °C
• vertical installation, max.	45 °C	45 °C	45 °C
• horizontal installation, min.	0 °C	0 °C	0 °C
• horizontal installation, max.	55 °C	55 °C	55 °C
Storage/transport temperature			
• Min.	-40 °C	-40 °C	-40 °C
• max.	70 °C	70 °C	70 °C

Air pressure			
• Operation, min.	795 hPa	795 hPa	795 hPa
• Operation, max.	1 080 hPa	1 080 hPa	1 080 hPa
• Storage/transport, min.	660 hPa	660 hPa	660 hPa
• Storage/transport, max.	1 080 hPa	1 080 hPa	1 080 hPa
Relative humidity			
• Operation, max.	95 %; no condensation	95 %; no condensation	95 %; no condensation
Vibrations			
• Vibrations	2*g* panel mount, 1*g* DIN rail mount	2*g* panel mount, 1*g* DIN rail mount	2*g* panel mount, 1*g* DIN rail mount
• Operation, checked according to IEC 60068-2-6	Yes	Yes	Yes
Shock test			
• checked according to IEC 60068-2-27	Yes; 15 *g*, 11 ms pulse, 6 shocks in each of 3 axes	Yes; 15 *g*, 11 ms pulse, 6 shocks in each of 3 axes	Yes; 15 *g*, 11 ms pulse, 6 shocks in each of 3 axes
Degree of protection			
IP20	Yes	Yes	Yes
Standards, approvals, certificates			
CE mark	Yes	Yes	Yes
C-TICK	Yes	Yes	Yes
cULus	Yes	Yes	Yes
FM approval	Yes	Yes	Yes

Technical specifications (continued)

	6ES7 214-1BE30-0XB0	6ES7 214-1AE30-0XB0	6ES7 214-1HE30-0XB0
Product-type designation	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/DC	CPU 1214C DC/DC/Relay
Dimensions and weight			
Dimensions			
• Width	110 mm	110 mm	110 mm
• Height	100 mm	100 mm	100 mm
• Depth	75 mm	75 mm	75 mm
Weight			
• Weight, approx.	455 g	415 g	435 g