

การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร M2M - Intelligence Machine Control
3/4 - CPS, ISA95 and Vijeo Citect SCADA <ul style="list-style-type: none"> • ความรู้เบื้องต้นเกี่ยวกับระบบ SCADA • การโปรแกรม PLC ให้ทำงานแบบ SCADA • การโปรแกรมเพื่อสื่อสารข้อมูลผ่าน Modbus TCP • การเชื่อมต่อกันระหว่าง IoTs กับอุปกรณ์ Modbus RTU/ASCII/TCP • คำถามท้ายบทเพื่อทดสอบความเข้าใจ

1/5: -- ความรู้เบื้องต้นเกี่ยวกับระบบ SCADA

<http://mechatronic2day.blogspot.com/2015/03/scada-1.html>
<https://industry4blogs.wordpress.com/2016/11/16/cyberphysical-systems-and-isa95/>

1.1 SCADA คืออะไร

SCADA ย่อมาจากคำว่า Supervisory Control and Data Acquisition คือระบบการส่งข้อมูลในระยะไกลเพื่อใช้ในการตรวจสอบ เก็บข้อมูล และควบคุมกระบวนการผลิตต่างๆ ที่มีหน่วยควบคุมอยู่ห่างไกลกับกระบวนการผลิต โดยจะมีการสื่อสารข้อมูลแบบดิจิทัลผ่านทางระบบเครือข่ายคอมพิวเตอร์ องค์ประกอบหลักของสกาตา ได้แก่ หน่วยติดต่อและปฏิบัติการของผู้ใช้ระดับบน หน่วยควบคุมระยะไกล หน่วยติดต่อระยะไกล และกระบวนการผลิต

ระบบ SCADA เป็นการรวมขบวนการ 2 ขบวนการเข้าด้วยกัน คือ

1. Telemetry System เป็นเทคนิคที่ใช้ในการส่งและรับข้อมูลผ่านสื่อกลาง โดยข้อมูลนั้นสามารถวัดได้ ข้อมูลเหล่านี้จะถูกส่งไปอีกสถานที่หนึ่งโดยผ่านสื่อกลางต่าง ๆ เช่น เคเบิล สายโทรศัพท์ หรือคลื่นวิทยุ
2. Data Acquisition เป็นวิธีการเข้าถึงและควบคุมข้อมูลจากอุปกรณ์ที่ถูกควบคุม หรือถูกตรวจสอบอยู่ โดยที่ข้อมูลที่ได้จะถูกส่งไปให้ระบบ Telemetry System เพื่อทำการส่งต่อไป

SCADA แบ่งออกเป็นสองรูปแบบ คือ

1. Point-to-Point Configuration เป็นการควบคุมที่ใช้หน่วยควบคุมในการควบคุมกระบวนการผลิตเพียงกระบวนการเดียว
2. Point-to-Multipoint Configuration เป็นการควบคุมใช้หน่วยควบคุมเดียวในการควบคุมกระบวนการผลิตหลายกระบวนการ

1.2 ส่วนประกอบของ SCADA

1. Field Instrumentation เป็นส่วนของเครื่องมือหรือเซนเซอร์ที่เชื่อมต่อกับเครื่องจักรหรืออุปกรณ์ที่ถูกควบคุมหรือตรวจสอบ โดยจะเปลี่ยนค่าปริมาณทางฟิสิกส์ ให้เป็นปริมาณทางไฟฟ้า ซึ่งอาจจะอยู่ในรูปของ Analog หรือ Digital
2. Remote Station เป็นส่วนที่ทำการรวบรวมข้อมูลจากเครื่องจักรหรืออุปกรณ์ และส่งไปยังศูนย์กลางระบบ SCADA
3. Communication Network เป็นการส่งข้อมูลดิจิทัลระหว่างสถานที่หนึ่งไปยังสถานที่หนึ่ง โดยผ่านตัวกลางในการติดต่อสื่อสาร เช่น สายเคเบิล คลื่นวิทยุ
4. Central Monitoring Station (CMS) เป็นศูนย์กลางระบบ SCADA โดยรับข้อมูลมาประมวลผลและทำการแสดงกระบวนการบนหน้าจอคอมพิวเตอร์ ประกอบด้วยซอฟต์แวร์ และฮาร์ดแวร์

1.3 ฐานข้อมูลของ SCADA

1. Real-time Database Servers เป็นระบบฐานข้อมูลที่ใช้จัดการและเก็บค่าของกระบวนการ ณ เวลาปัจจุบัน ในขณะใด ๆ ค่า Real-time จะเปลี่ยนแปลงไปตามสภาพของกระบวนการที่เปลี่ยนแปลงไปตามเวลา
2. Historical Database Servers เป็นระบบฐานข้อมูลที่ใช้จัดการและจัดเก็บค่า Historical Data ของกระบวนการเพื่อใช้ในการ Trending, Logging, Statistic และ Report

1.4 มาตรฐาน Protocol ของ SCADA

ปัจจุบัน มี SCADA มาตรฐาน Protocols มากกว่า 200 โปรโตคอลทั่วโลก มาตรฐานโปรโตคอลที่ใช้กันในปัจจุบัน ได้แก่

1. ASCII (American Standard Code for Information Interchange) เป็นโปรโตคอลที่ใช้ในการสื่อสารของคอมพิวเตอร์ที่รู้จักกันอย่างแพร่หลายและเป็นสากล
2. CAP (Compressed ASCII Protocol) เป็น RTU Protocol ที่ดีที่สุด เป็นภาษาที่คนสามารถเข้าใจได้ มีความน่าเชื่อถือ เร็ว และมีความปลอดภัยสูง
3. Modbus เป็น point-to-point PLC protocol ที่ใช้กันทุกแห่งทุกหน แต่มีข้อเสียคือ เป็นภาษาที่คนไม่สามารถอ่านเข้าใจได้
4. Modbus X พัฒนามาจาก Modbus ทำให้สามารถอ่านและสามารถสร้างจำนวนบวกและลบได้
5. IEEE 32 bit Signal Format Floating Point เป็นมาตรฐานของโรงงานอุตสาหกรรม สำหรับส่งตัวเลข 32 บิต ด้วยความถูกต้อง

1.5 องค์ประกอบ SCADA

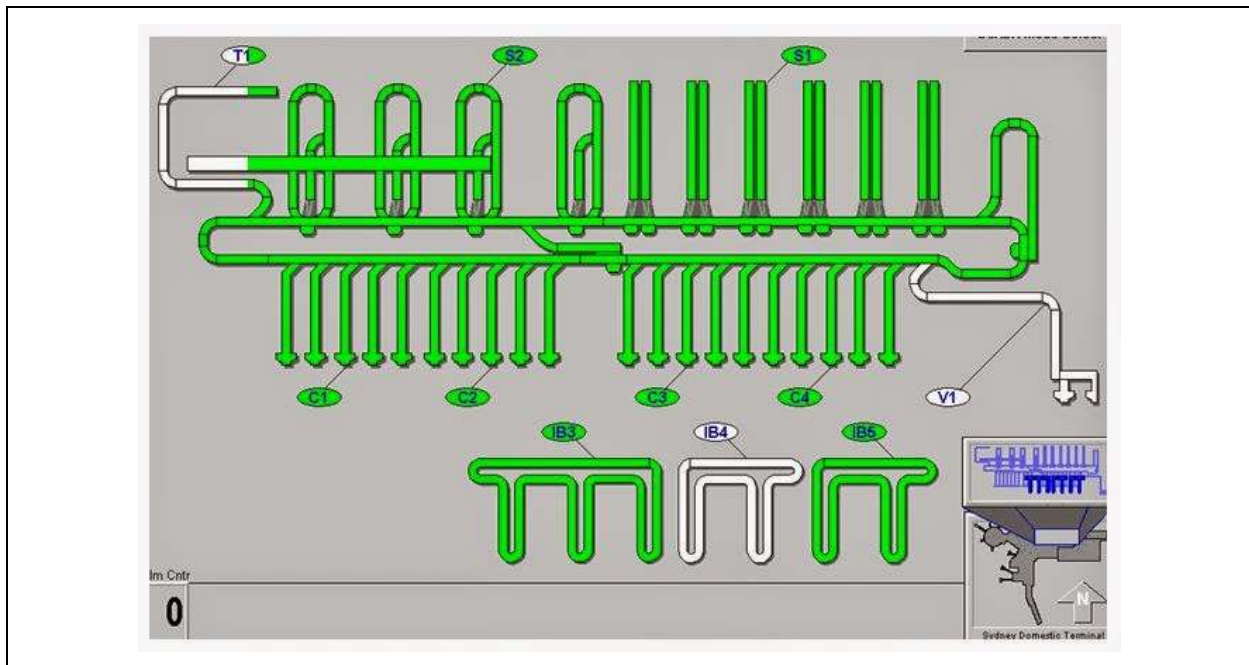
ผู้ใช้งานสามารถตรวจสอบและควบคุมกระบวนการผลิตภายในโรงงานอุตสาหกรรมเป็นระยะทางไกลได้โดย หน่วยติดต่อและปฏิบัติการของผู้ใช้ระดับบนเป็นเครื่องมือปฏิบัติการของผู้ใช้สำหรับตรวจสอบและควบคุม กระบวนการผลิต เชื่อมต่อกับหน่วยควบคุมระยะไกล หน่วยควบคุมระยะไกลติดต่อกับหน่วยติดต่อระยะไกลโดยการสื่อสารข้อมูลแบบ ดิจิตอลทางระบบเครือข่ายคอมพิวเตอร์ และหน่วยติดต่อระยะไกลเป็นเครื่องมือเชื่อมต่อกับกระบวนการผลิต ประกอบด้วย หน่วยรับสัญญาณ และส่งสัญญาณของสัญญาณชนิดแอนะล็อก และสัญญาณชนิดดิจิตอล

1.6 SCADA เหมาะสมกับงานใด

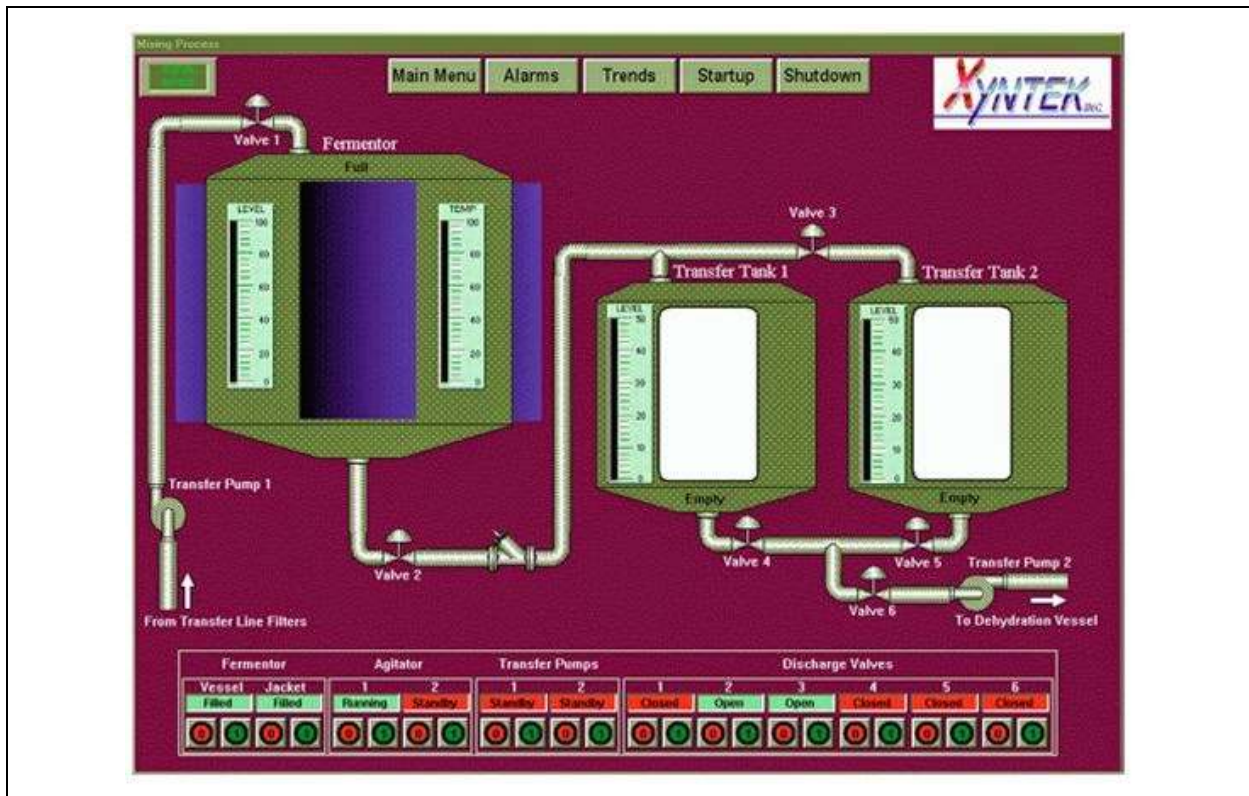
งานการตรวจสอบ การเก็บรวบรวมข้อมูลของกระบวนการผลิต และการบริหารระบบควบคุมกลุ่มโรงงาน อุตสาหกรรมขนาดใหญ่บริเวณกระบวนการผลิตครอบคลุมพื้นที่กว้าง หรือโรงงานอุตสาหกรรมมีกระบวนการผลิต อิสระติดตั้งกระจายทั่วบริเวณพื้นที่การผลิต รวมถึงระบบสาธารณูปโภคต่างๆ

1.7.1. ระบบจ่ายไฟฟ้า

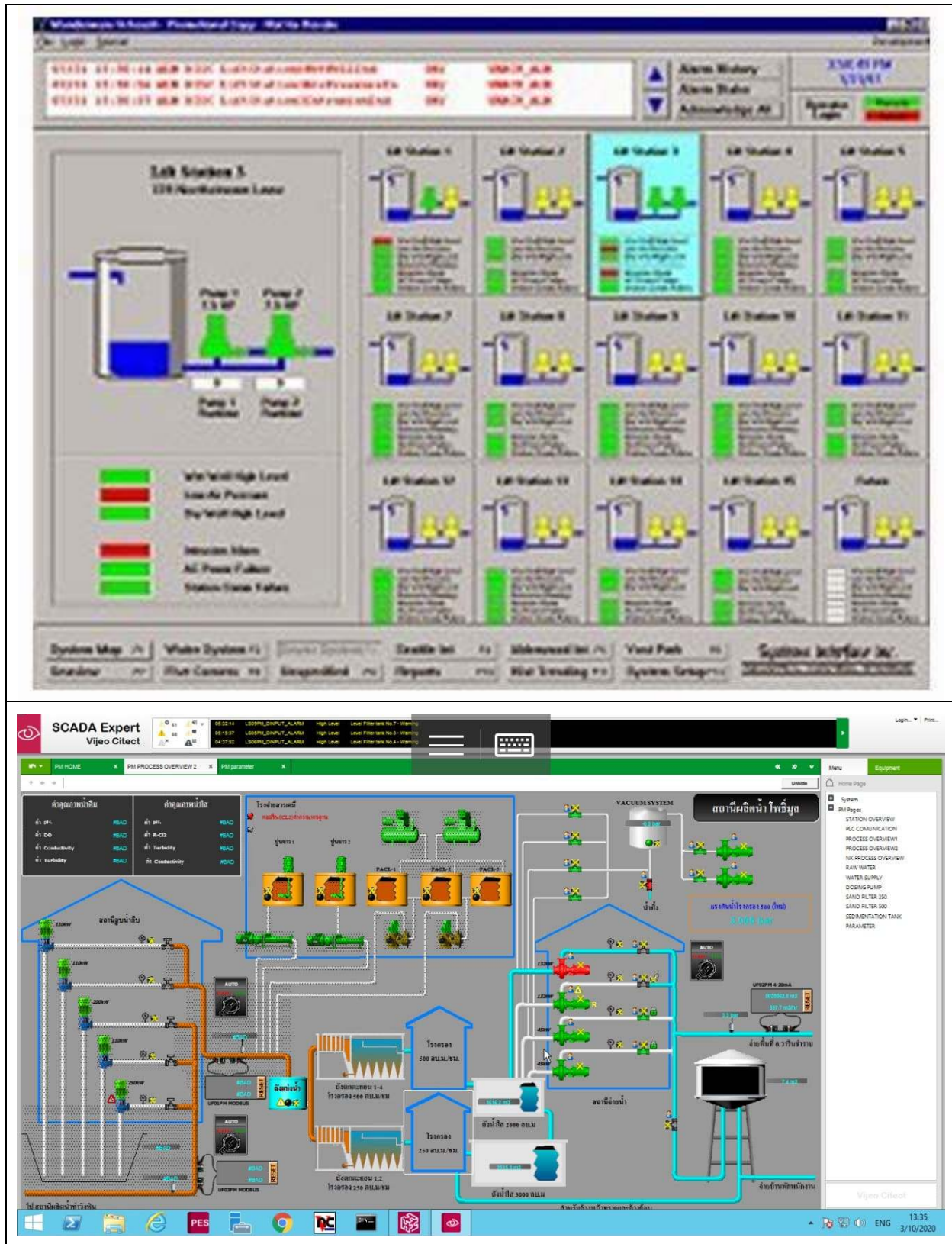
1.7.2. ระบบจ่ายน้ำ



รูปที่ 1.2 แสดงการประยุกต์ใช้ SCADA ในระบบจ่ายน้ำ

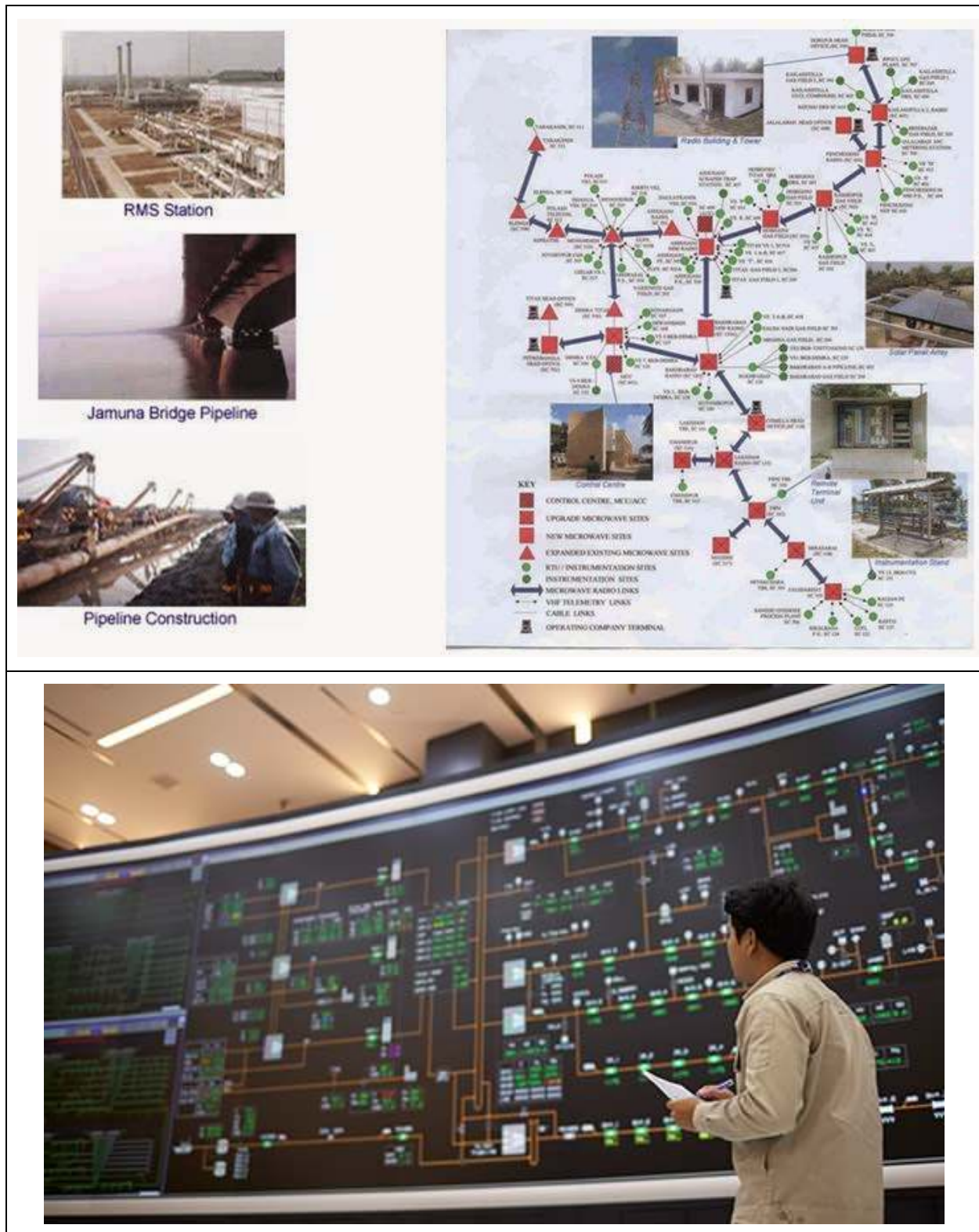


รูปที่ 1.3 แสดงการประยุกต์ใช้ SCADA ในระบบจ่ายน้ำ(ต่อ)

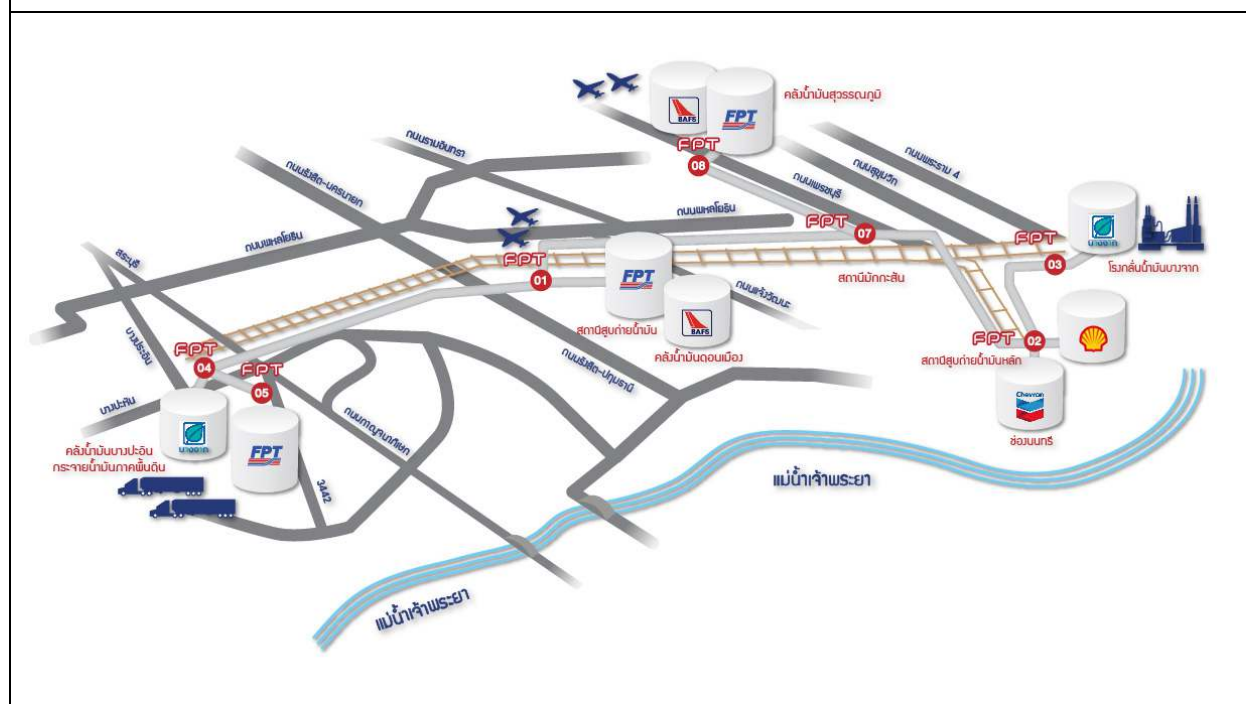


รูปที่ 1.4 แสดงการประยุกต์ใช้ SCADA ในระบบจ่ายน้ำ(ต่อ)

1.7.3. ระบบท่อส่งก๊าซ



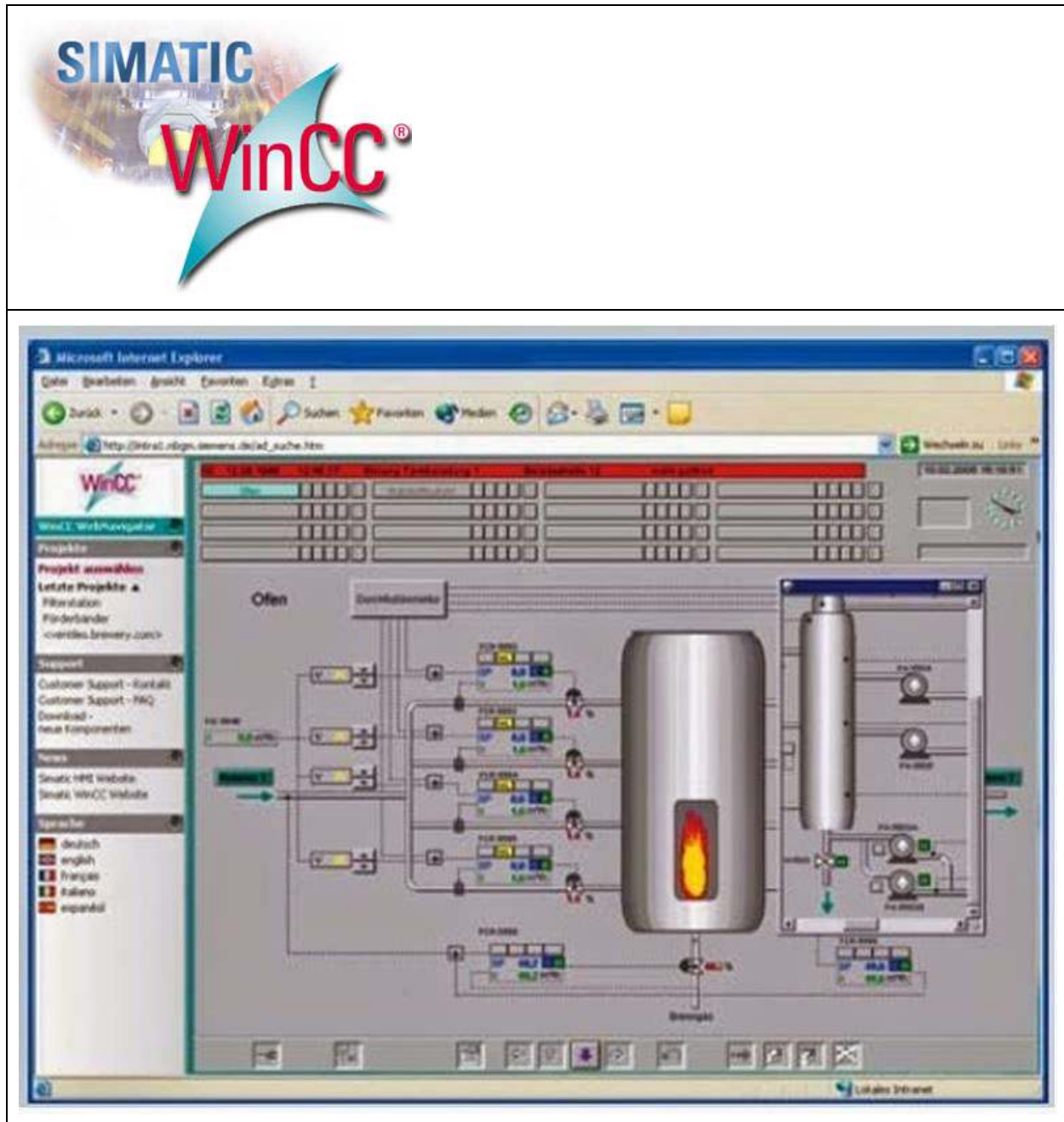
รูปที่ 1.5 แสดงการประยุกต์ใช้ SCADA ในระบบระบบท่อส่งก๊าซ



รูปที่ 1.6 แสดงการประยุกต์ใช้ SCADA ในระบบระบบทอส่งน้ำมัน

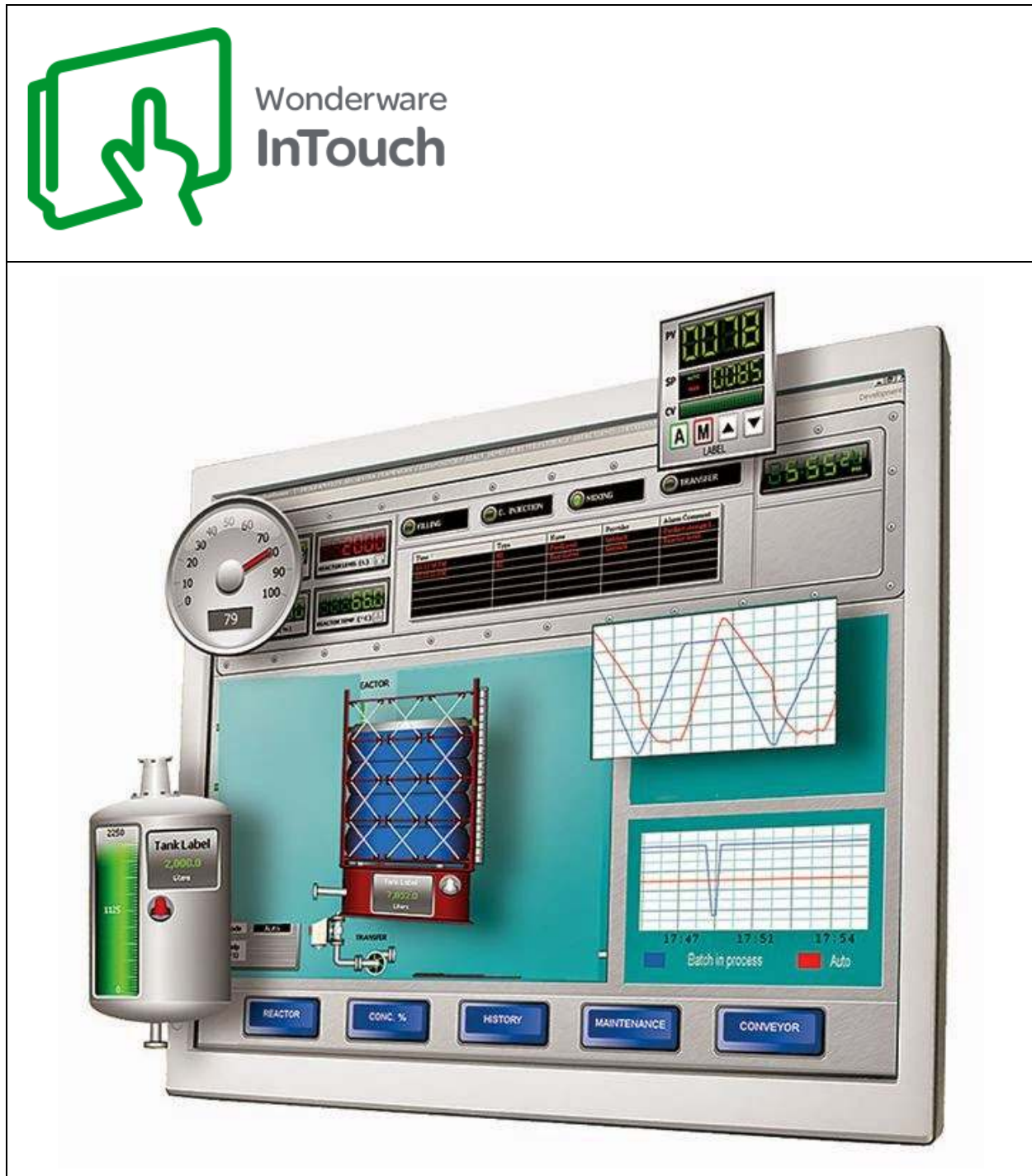
1.8 ตัวอย่างซอฟต์แวร์ระบบ SCADA ที่เป็นที่ใช้งานมากในอุตสาหกรรม

1. Simatic WinCC เป็นโปรแกรม SCADA ของบริษัท Siemens ประเทศเยอรมัน มักจะใช้งานในอุตสาหกรรม การผลิตต่างๆที่มาจากยุโรปที่ใช้อุปกรณ์ควบคุมยี่ห้อ Siemens เช่นอุตสาหกรรมยานยนต์ อุตสาหกรรมปิโตรเคมี อุตสาหกรรมอาหาร ฯลฯ ปัจจุบันออกมาถึง Version 7 แล้ว



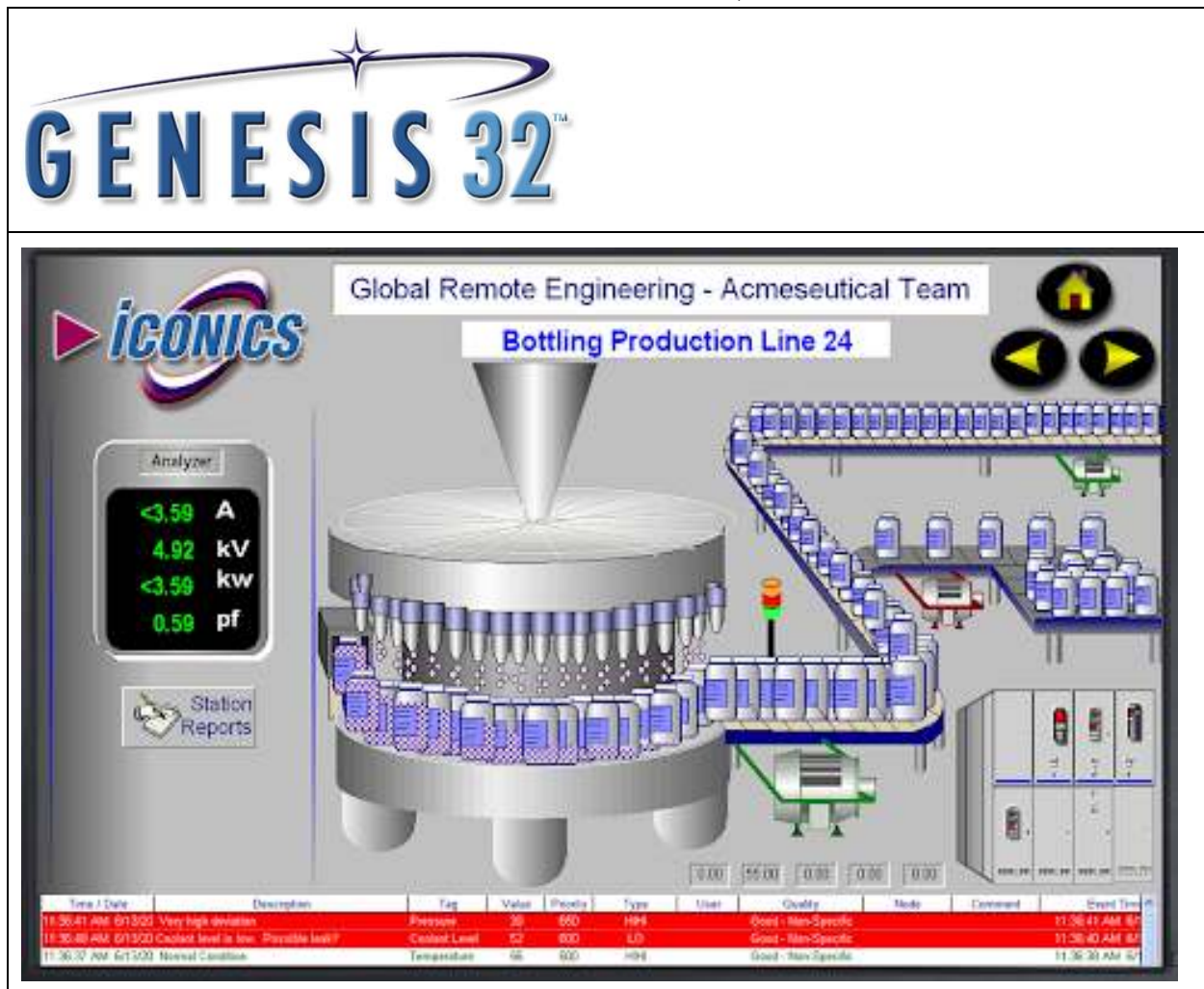
รูปที่ 1.7 แสดง Simatic WinCC

2. **InTouch** เป็นโปรแกรม SCADA ของบริษัท Wonderware ประเทศสหรัฐอเมริกา มักจะใช้งานในอุตสาหกรรมการผลิตต่างๆ โดยเคยมียอดขายเป็นอันดับ 1 ของโลก แต่เนื่องจากเป็นบริษัทที่ผลิตเฉพาะ HMI เท่านั้นเลยทำให้เคยมียอดขายตกลงไป ทำให้บริษัท Wonderware ได้ร่วมกับบริษัท Mitsubishi Automation เพื่อผลิตระบบ SCADA ให้กับระบบอัตโนมัติที่เป็นยี่ห้อของ Mitsubishi



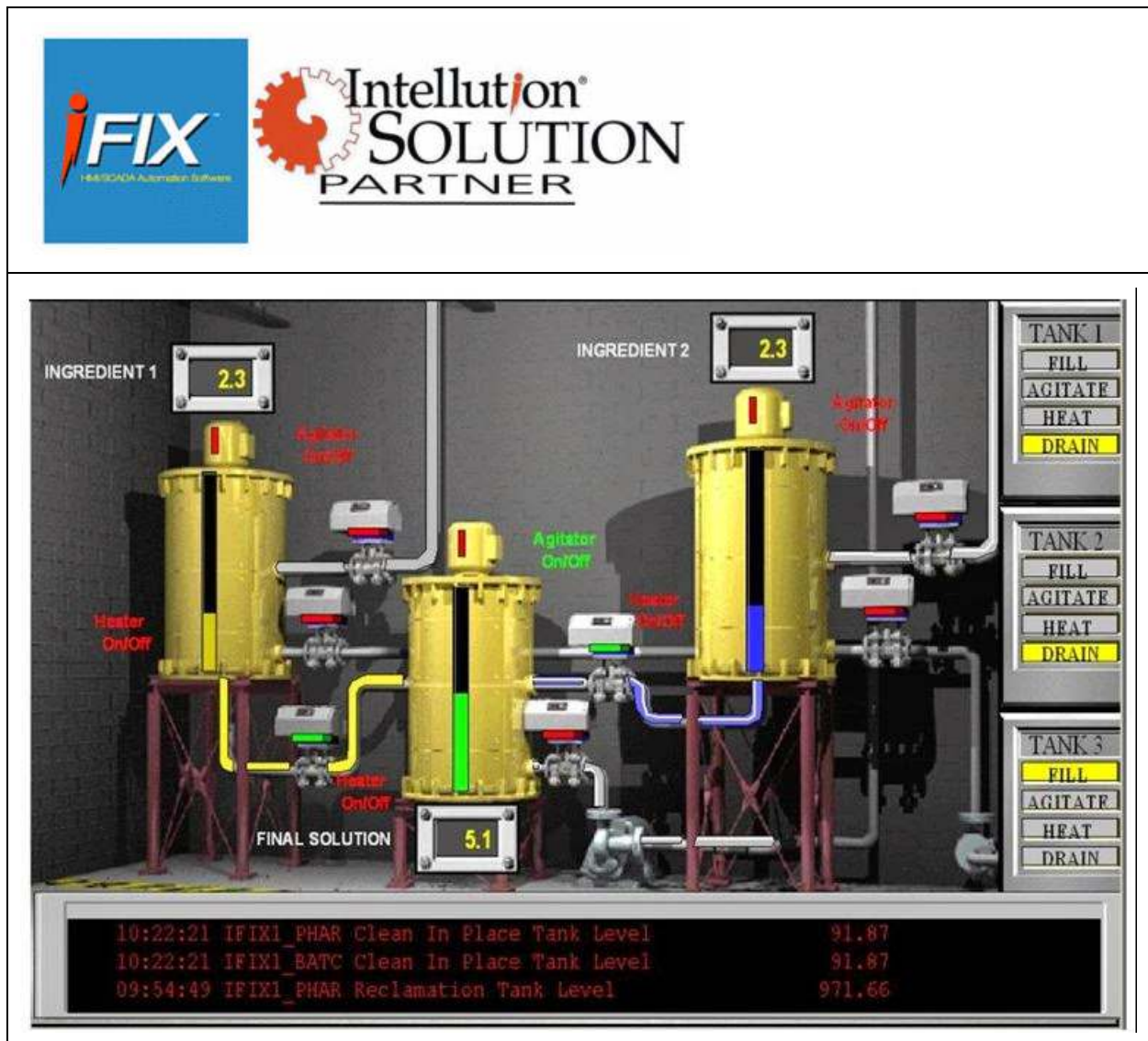
รูปที่ 1.8 แสดง Intouch Wonderware

3. **GENESIS32** ผลิตโดยบริษัท ICONICS ประเทศสหรัฐอเมริกา โดยที่ GENESIS32 ถูกติดตั้งในระบบ SCADA ทั่วโลกกว่า 150,000 ระบบ (ปี 2005) และได้รับรางวัล World Open Award ในโปรเจกต์การขนส่งน้ำมันระหว่างมอสโคว์และไซปรัสซึ่งเป็นโปรเจกต์ที่ใหญ่ที่สุด



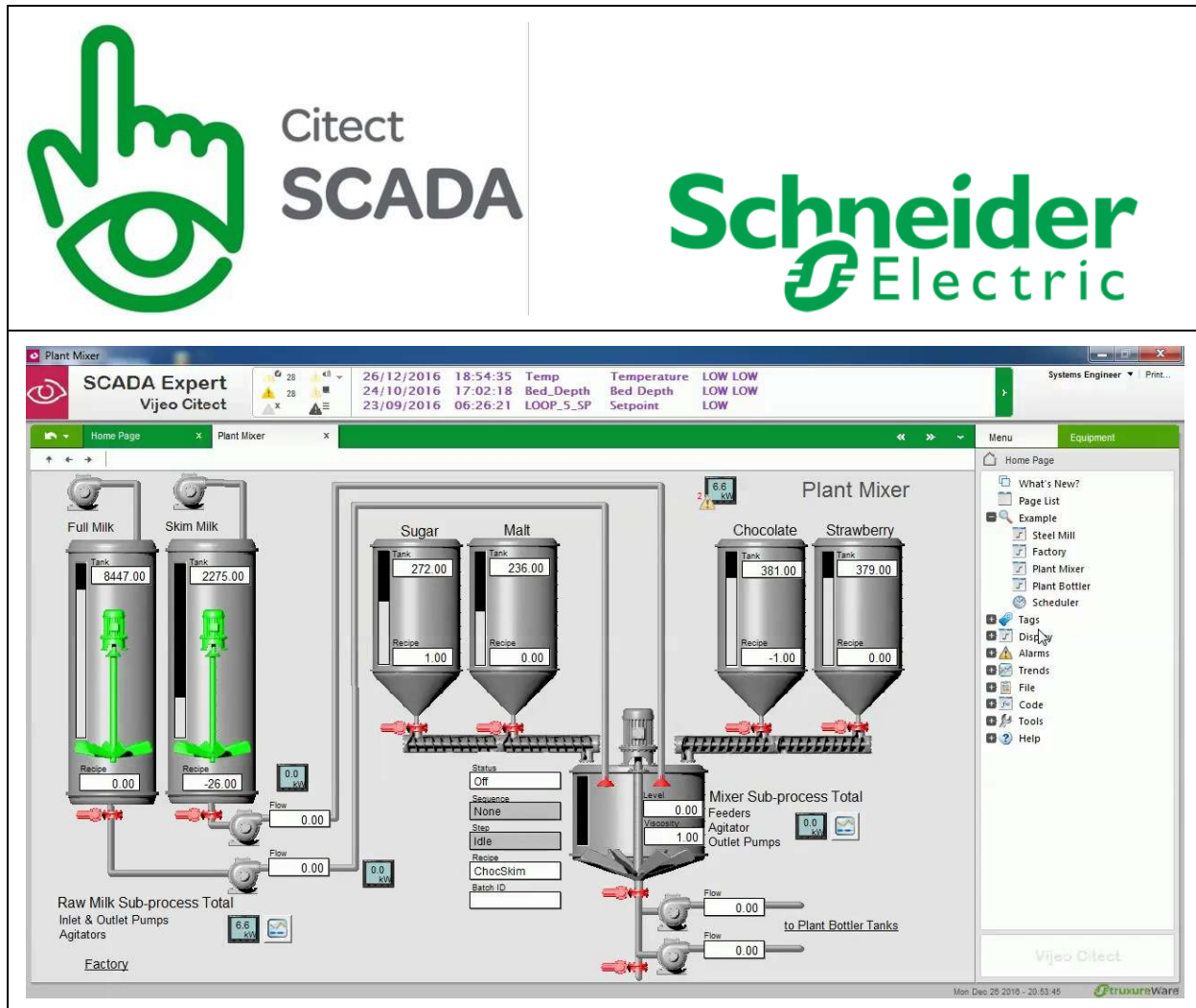
รูปที่ 1.9 แสดง ICONIC Genesis 32

4. Intellutions FIX - iFix Proficiency HMI/SCADA ผลิตโดยบริษัท GE-Fanuc ประเทศสหรัฐอเมริกา เมื่อก่อนมีใช้ในประเทน้อยมาก แต่ปัจจุบันมีแนวโน้มสูงขึ้น เพราะเนื่องจากว่าบริษัท GE (General Electric) เริ่มเข้ามาทำตลาดในประเทศไทย



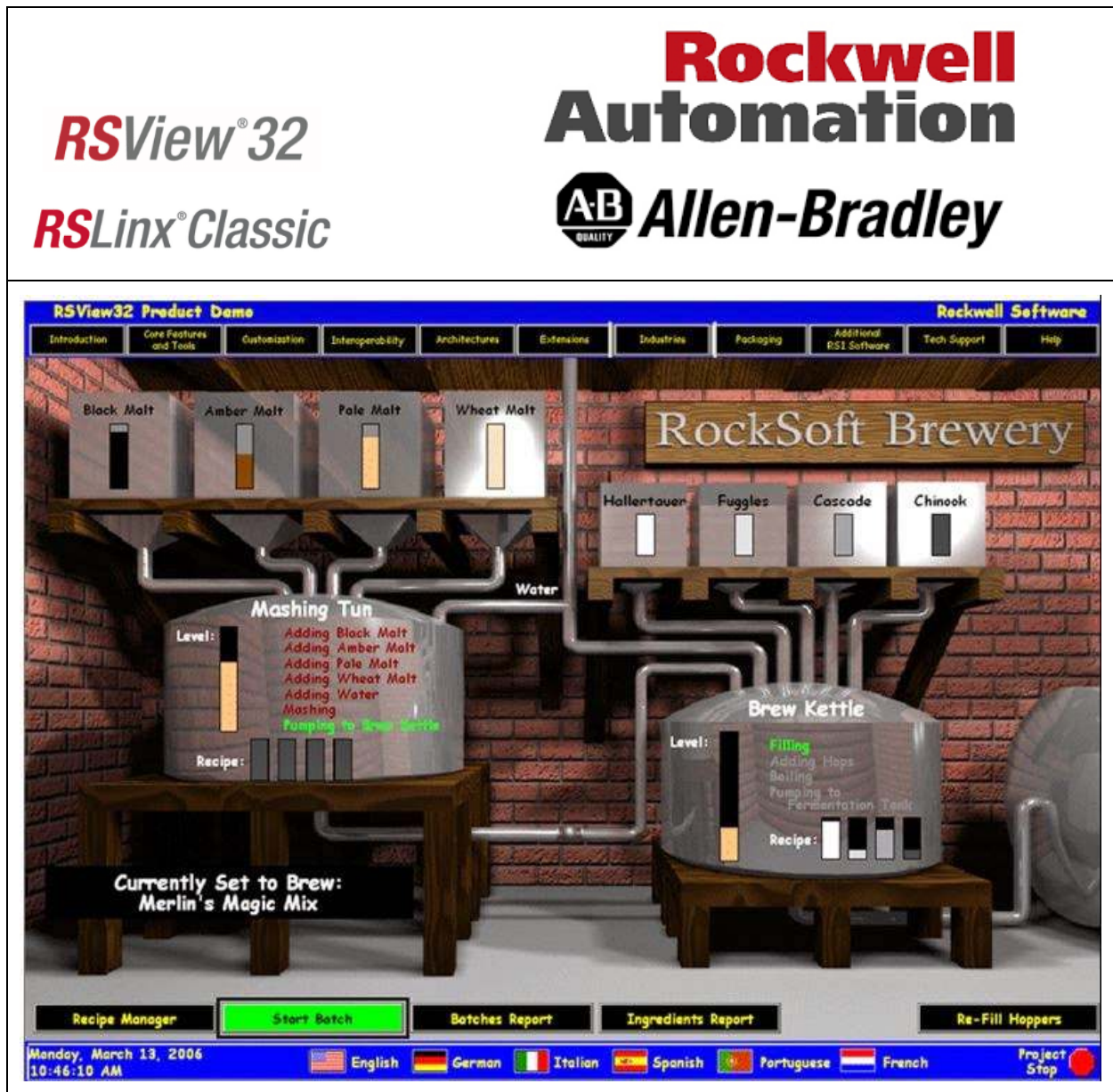
รูปที่ 1.10 แสดง iFix Proficiency HMI/SCADA

5. Citect Scada ผลิตโดยบริษัท Citect แห่งประเทศออสเตรเลีย เป็น SCADA ที่ใช้กันมากที่สุดในทวีปเอเชีย และเคยผลิตระบบ SCADA ต่างๆให้กับ PLC ที่ผลิตจากประเทศญี่ปุ่นเช่น Mitsubishi Automation , Omron เป็นต้น ปัจจุบันได้ถูกบริษัท Schneider Electric จากประเทศฝรั่งเศสซื้อไป เลยเปลี่ยนชื่อเป็น Vijeo Citect มีจุดเด่นคือเป็นระบบ SCADA ที่มีสีสันสวยงาม ใช้งานง่าย



รูปที่ 1.11 แสดง Citect Scada

6. RSview32 ผลิตจากบริษัท Rockwell Automation ซึ่งเป็นบริษัททางด้าน Automation ที่ใหญ่แห่งหนึ่งของโลก โดยที่ RSview32 เป็นระบบ SCADA อีกระบบหนึ่งที่ใช้กันมากในอุตสาหกรรมที่ใช้ระบบ Automation ของ Rockwell



รูปที่ 1.12 แสดง Rsvie 32 SCADA



รูปที่ 1.13 โลโก้ของซอฟต์แวร์ระบบ SCADA ที่เป็นที่ใช้งานมากในอุตสาหกรรม

1.9 ประโยชน์ของการนำระบบ SCADA มาใช้

ในปัจจุบันหลายหน่วยงานได้มีการนำระบบ SCADA นี้มาใช้ เพื่อช่วยในการ Monitor ข้อมูลเพื่อช่วยในการตัดสินใจในการควบคุมระบบ เช่น การไฟฟ้านครหลวง , การประปานครหลวง , ปตท. ฯลฯ ซึ่งเหตุผลที่หลายหน่วยงานเลือกใช้ระบบนี้เพราะเหตุผลต่างๆดังต่อไปนี้

1. การควบคุมจะเป็นไปอย่างต่อเนื่องและครอบคลุมตลอดพื้นที่
2. เพิ่มประสิทธิภาพในการตรวจสอบ และการบำรุงรักษาเครื่องจักรและอุปกรณ์ เนื่องจากสามารถทำการบำรุงรักษา หรือทำ PM ก่อนที่เครื่องจักรจะชำรุด
3. ก่อให้เกิดเสถียรภาพและความน่าเชื่อถือไว้วางใจ
4. ลูกค้าจะได้รับบริการอันรวดเร็ว ทันสมัย แม่นยำ และยุติธรรม
5. การติดตามข้อมูล และการประเมินผลต่างๆ เป็นไปอย่างรวดเร็วและถูกต้อง
6. ประหยัดแรงงานและกำลังคน และทำให้ประหยัดค่าใช้จ่าย

1.10 อะไรคือ ISA-95 และ MOM

1.10.1 เรามาเริ่มกันที่ ISA-95 หรือ มาตรฐาน ANSI/ISA-95

- ANSI/ISA-95 คือ มาตรฐานของ The international society of automation ที่ว่าด้วยเรื่องกำหนดมาตรฐานสำหรับการพัฒนาระบบเชื่อมต่ออัตโนมัติ ที่เชื่อมตั้งแต่การจัดการองค์กร - บนสุด (Enterprise) - ระบบควบคุม (Control Systems) - ลงไปถึงล่างสุดที่เป็น Physical ในสายการผลิต เข้าด้วยกัน
- ANSI/ISA-95 ได้แบ่งโครงสร้างของระบบออกเป็นส่วน ๆ ดังนี้คือ

Level 4	ชั้นบนสุด คือ Enterprise / Business Operations	สมองขององค์กร
Level 3	ถัดลงมา คือ Manufacturing Operations Management หรือ MOM	ตัวจัดการเรื่องราวเก็บข้อมูลมาจากชั้นล่างบริหารจัดการข้อมูล แล้วส่งต่อขึ้นไปให้สมองขององค์กรวางแผน บริหาร จัดการตัดสินใจ สั่งการ
Level 0,1,2	ถือเป็นส่วนล่างสุดที่พูดโดยรวมถึง Industrial Automation	เป็นส่วนของอุปกรณ์ในสายการผลิตและชุดควบคุมอุปกรณ์เหล่านี้

1.10.2 รายละเอียดแต่ละชั้นของ ISA-95

Level 4: ที่นี้จะมี Business Application ต่าง ๆ สำหรับงานในภาพของ Enterprise และ Business Operations ซึ่งก็คือ Business Planning & Logistics - ตรงนี้จะมีชื่อที่ได้ยินก็เช่น

- ERP, APO, APS, CRM
- PLM, Plant/Process Design
- EHS, CMMS, BPM
- BI, DW, SCM, EQMS

Level 3: ที่นี้คือ MOM Applications หรือ Manufacturing Operations Management Application ต่าง ๆ นั่นเอง - มีอะไรบ้างหรือ? ก็เช่น

- WHM, Document Management / Scheduling, Dispatching / MRP, IEM SmartGrid, TIS
- Configure, Model, RTO, APC / Asset Tracking, Mobile, RFID / MES, EWI, Tracking, Batch
- EMI / OI, Historian, Reporting, OEE / W.O., RCM, Equipment Health / Time Track, Training, OTS
- SPC/SQC, HACCP, LIMS

เพิ่มเติมในส่วนของ MOM:

- MOM's Alphabet Soup
- Operations Management

Level 2: เป็นส่วนของ Manufacturing Control ตรงนี้มีอะไรบ้าง ก็อย่างเช่น

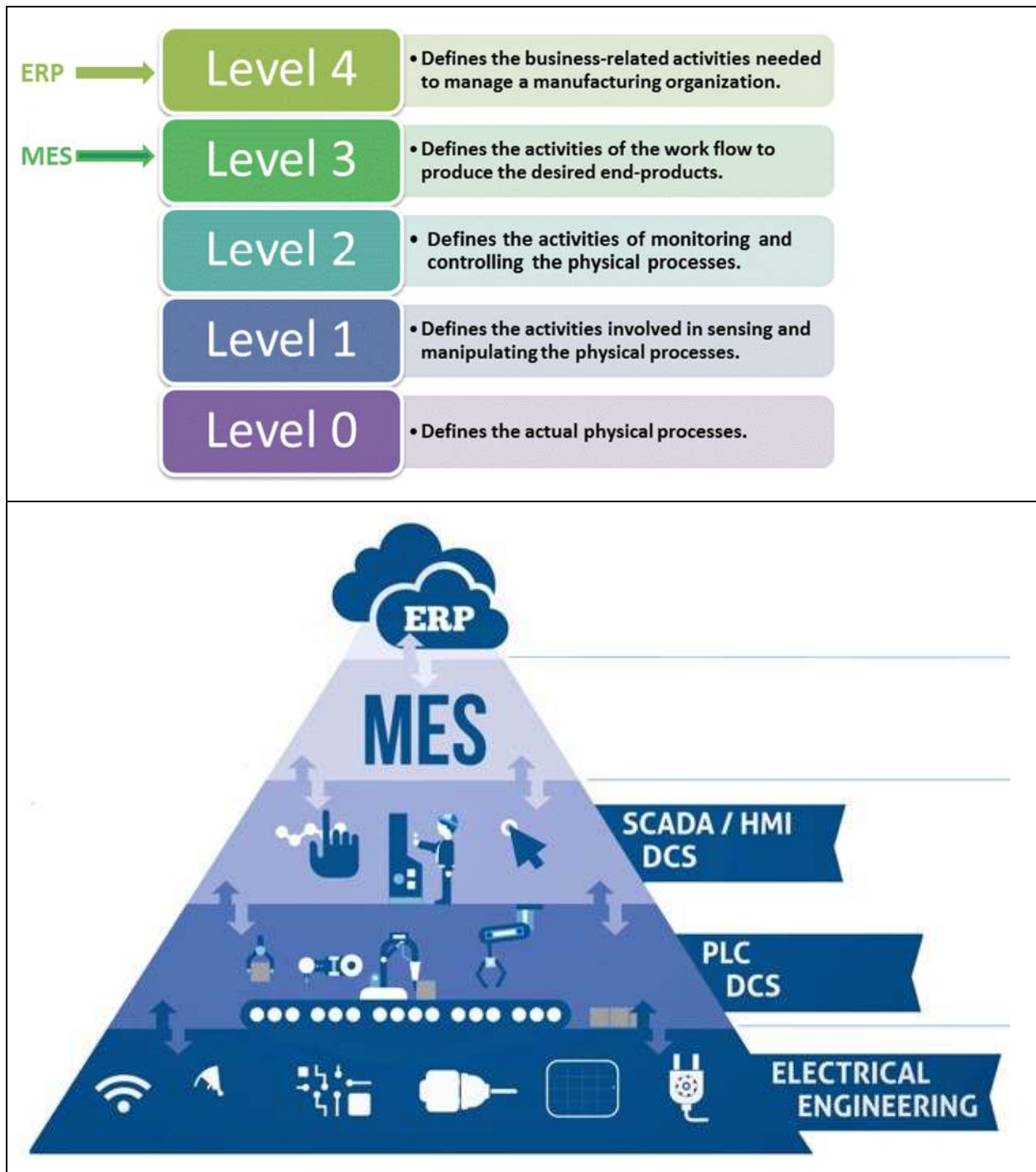
- HMI (Human-Machine Interface)
- SCADA (Supervisory Control and Data Acquisition)
- DCS (Distributed Control System) - ซึ่งลงไปถึง Level 1 ด้วยก็ได้

Level 1: ตรงนี้เป็นส่วนของ Sense & Manipulate ก็ประมาณ IT Automation, I/O Module ต่าง ๆ เช่น

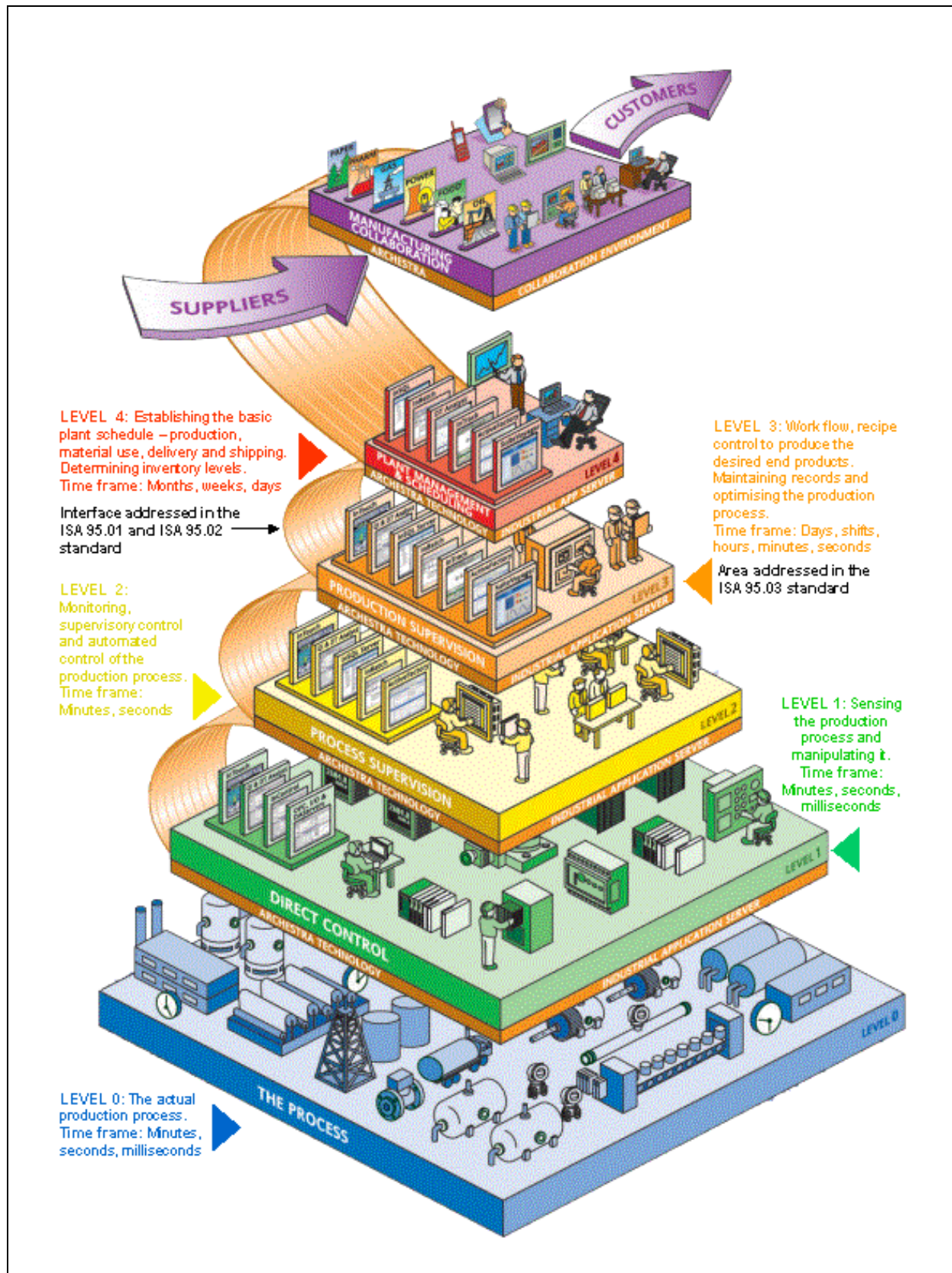
- PAC (Programmable Automation Controller)
- PLC (Programmable Logic Control)
- RTU (Remote Terminal Unit)
- PID Controller (Proportional-Integral-Derivative Controller หรืออีกชื่อคือ Three-Term Controller)

Level 0: ตรงนี้เป็นส่วนของ Production Process ซึ่งก็จะมี Sensor ต่าง ๆ อย่างเช่น ตัววัดอุณหภูมิ, Control Valve และ/หรือ คน ที่เป็นการทำงานจริง ๆ กับ วัตถุดิบ ชิ้นงาน ผลงาน อะไรต่าง ๆ แบบนี้ละ

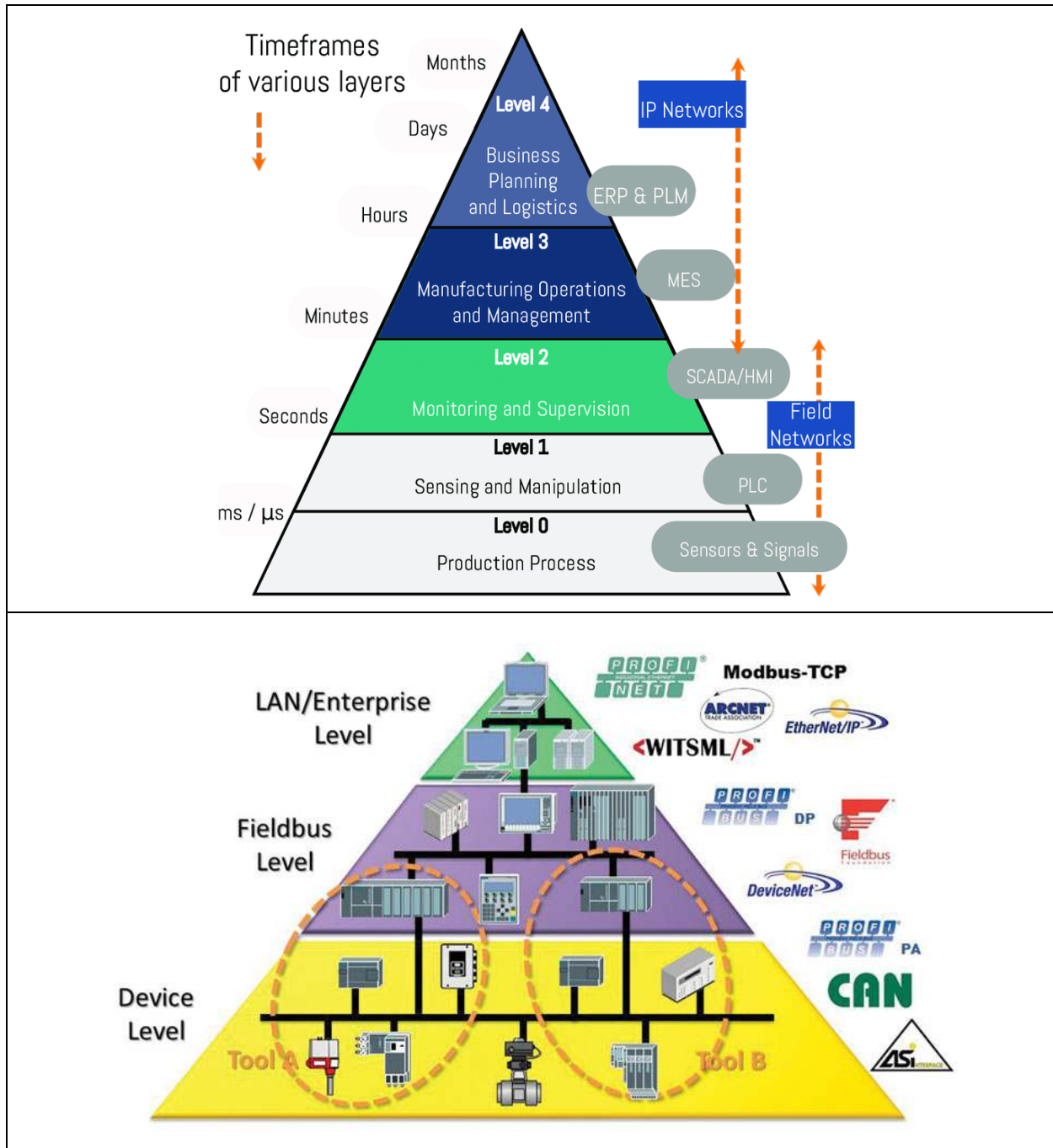
เอาล่ะน่าจะพอไหว ประมาณว่าก็พอรู้อะไร ๆ เยอะขึ้น เห็นภาพได้ชัดเจนขึ้น ได้ยินชื่อ หรือเห็นคำต่าง ๆ มากขึ้น ซึ่งแต่ละชื่อ แต่ละคำ ถ้าเจอก็จะรู้ว่าอะไรอยู่ตรงไหน อยู่ในส่วนของ Level อะไร มีหน้าที่ เป้าหมาย และใช้งานให้เกิดอะไร คราวนี้ที่เหลือก็คือทำความเข้าใจในแต่ละชื่อ และหาข้อมูลในแต่ละส่วนเพิ่มเติม - นีละ Computer Engineering ที่ผสมด้วย Manufacturing และขยับมาถึง Automation Engineering ในที่สุด



รูปที่ 1.14 Automation Pyramid



รูปที่ 1.15 Cyber-Physical Systems and ISA95

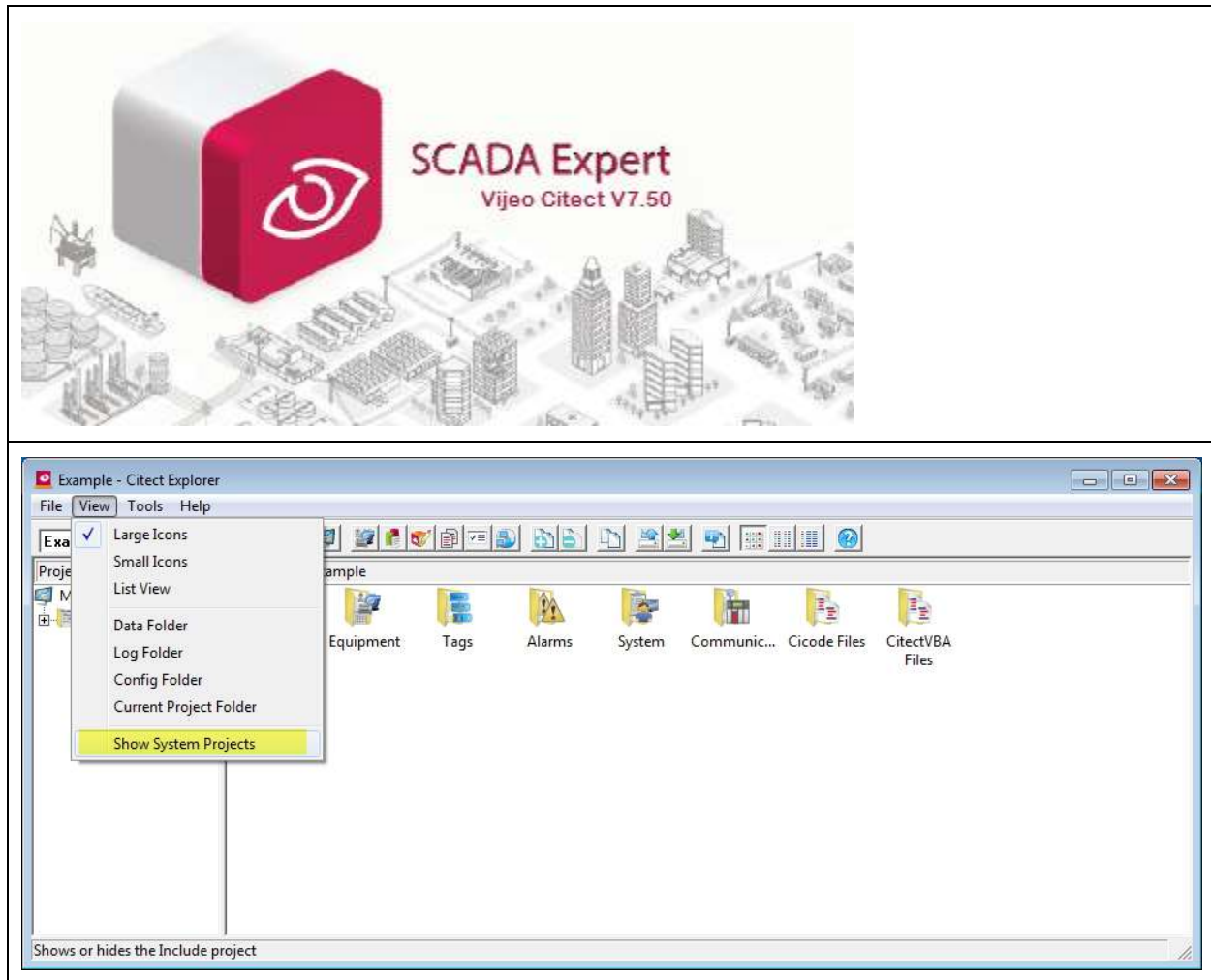


รูปที่ 1.16 Automation Pyramid(ต่อ)

1.11 แนะนำโปรแกรม SCADA ที่ใช้ในหนังสือเล่มนี้

ในใบข้อมูลเล่มนี้จะขอแนะนำ Citect SCADA เนื่องจากว่าเป็นระบบ SCADA ที่ใช้กันอย่างแพร่หลายในอุตสาหกรรมและเป็นซอฟต์แวร์ที่สามารถใช้ได้โดยไม่ต้องมีค่าใช้จ่ายใดๆอีกทั้งสามารถเชื่อมต่อกับ PLC หลายรุ่นหลายยี่ห้อเช่น Siemens , Mitsubishi และ OMRON ซึ่งเป็น PLC ที่ใช้กันมากในอุตสาหกรรมในประเทศไทย

เราสามารถที่จะกำหนดให้โปรแกรมทำการมอนิเตอร์และควบคุมระบบที่เล็กๆจนกระทั่งถึงระบบที่ใหญ่ได้ เพราะว่าปัจจุบันนี้โปรแกรมได้พัฒนาให้มีความยืดหยุ่นสูงสามารถเพิ่มฟังก์ชันต่างๆภายหลังได้ และ Citect SCADA เป็นโปรแกรม SCADA ที่เรียนรู้ได้ง่ายเนื่องจากมีรูปแบบสำเร็จรูปมากมายเช่นพวก Genies, Template และ Wizard ทำให้ลดเวลาอย่างมากในการใช้งานโปรแกรม

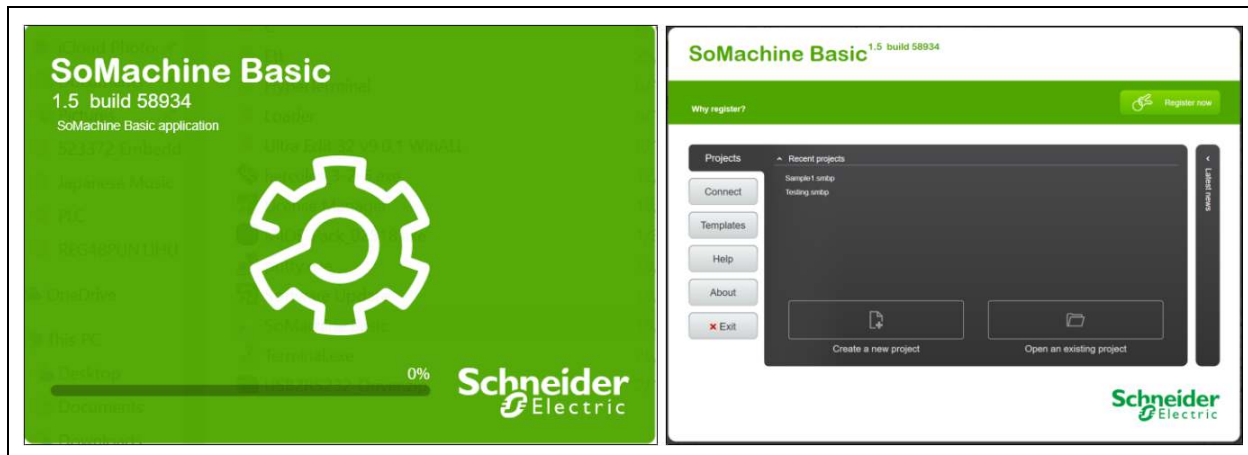


รูปที่ 1.17 หน้าตาของโปรแกรม Citect (Citect Explorer)

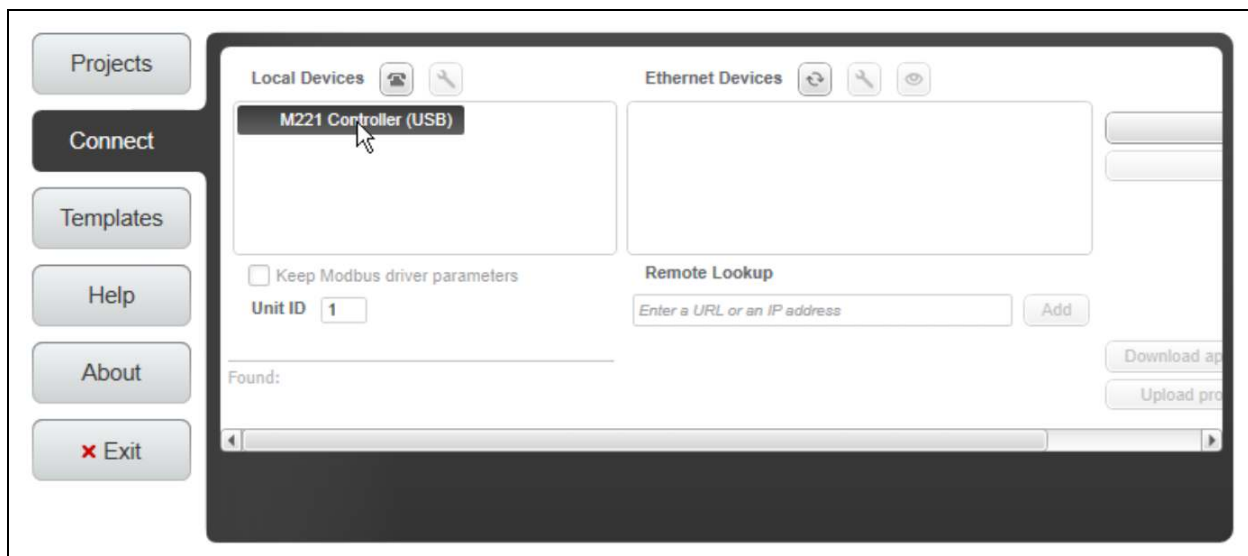
2/5: -- การโปรแกรม PLC ให้ทำงานแบบ SCADA

Test 1/8. SoMachine Basic – Single Input/ Single Output

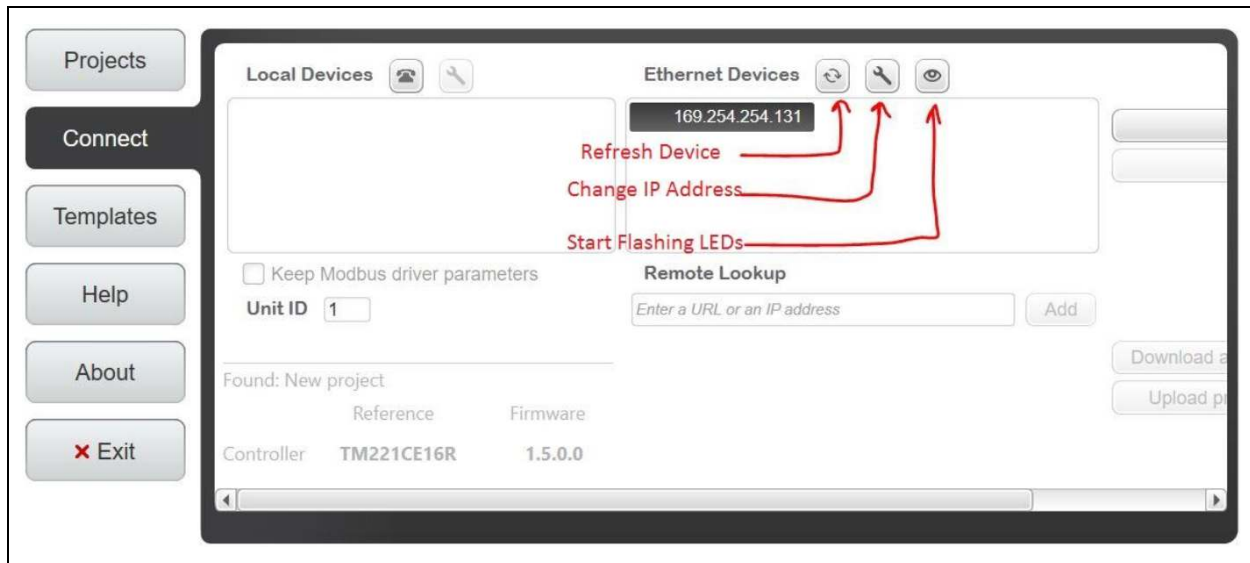
1. โหลดโปรแกรม SoMachine Basic 1.5 จาก <http://www.schneider-electric.us/en/download/document/SOMBASAP15SOFT/>
2. เรียกโปรแกรม SoMachine Basic 1.5



3. ตรวจสอบการเชื่อมต่อระหว่าง PC กับ PLC
กรณีนี้ที่ 1 : ใช้สาย USB Cable

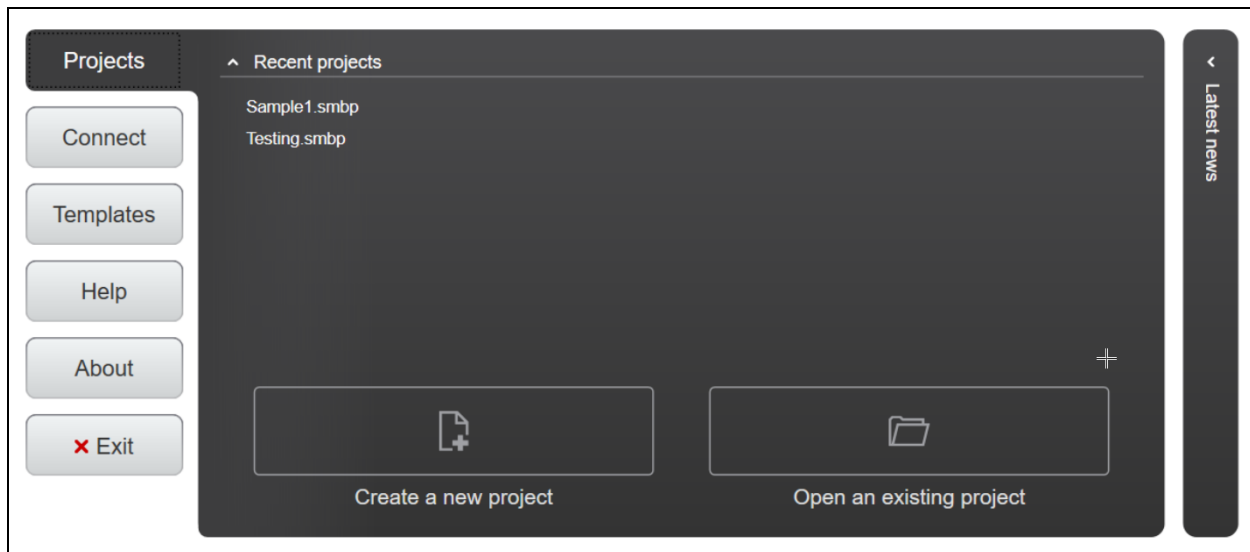


กรณีที่ 2 : ใช้สาย LAN



- Refresh Device > ตรวจสอบค่า IP
- Change IP Address > เพื่อกำหนด IP ให้กับ PLC
- Start Flashing LEDs > เพื่อทดสอบว่า IP ที่คลิกอยู่นี้เป็นตัวไหน เมื่อกด LED บน PLC จะกะพริบ

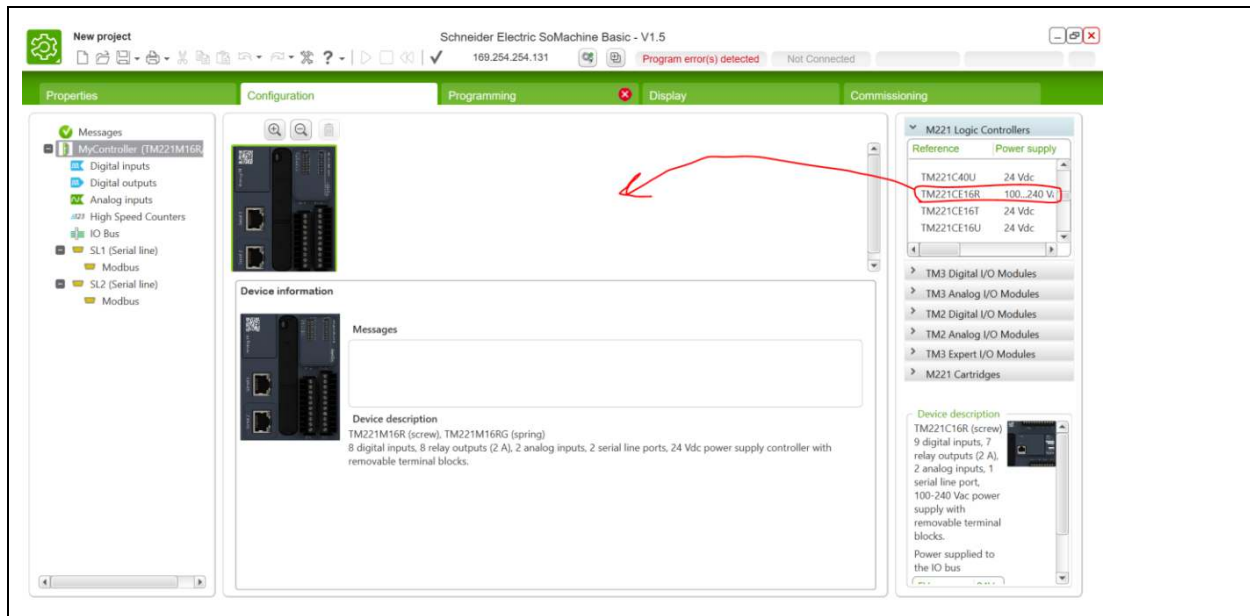
4. Create a new project



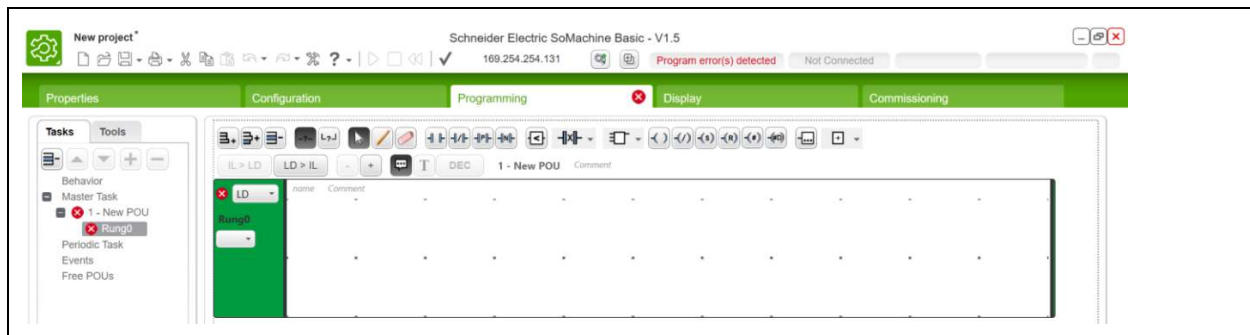
5. เมื่อเข้ามาใน Project จะมี 5 Page หลัก

- Properties รายละเอียดผู้พัฒนา Project
- Configuration กำหนดเกี่ยวกับ PLC ที่ใช้
- Programming การเขียนโปรแกรม
- Display การจำลองการทำงาน
- Commissioning การอัปโหลดโปรแกรมไปยัง PLC

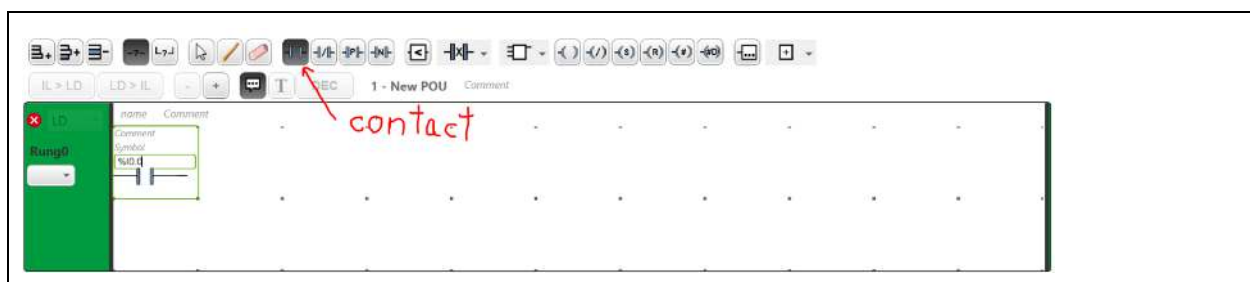
6. ในหน้า Configuration ให้เราลาก TM221CE16R มาใส่



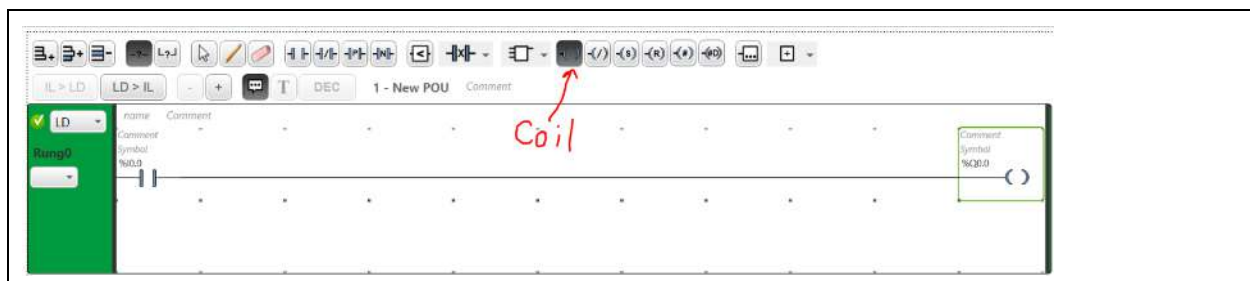
7. ให้เราไปหน้า Programming เพื่อเรียกใช้



- ลาก Contact(F4) เข้ามาใส่ แล้วตั้ง Address เป็น %IO.0



- ลาก Coil(Ctrl+F9) เข้ามาใส่ แล้วตั้ง Address เป็น %Q0.0



8. ไปที่หน้า Commission กดปุ่ม Login กับ IP ของตัวที่ต้องการอัปเดตข้อมูลใส่

The screenshot shows the 'Commission' interface. On the left, under 'Local Devices', there is a 'Unit ID' field with the value '1'. On the right, under 'Ethernet Devices', the IP address '169.254.254.131' is entered. A red arrow points to the 'Login' button. Below the Ethernet Devices section, there is a 'Remote Lookup' section with a text input field 'Enter a URL or an IP address' and an 'Add' button.

- เมื่อ Login แล้ว ตัว PLC จะมีข้อความแจ้งเราว่า โปรแกรมของเราเก็บใน PLC เป็นตัวเดียวกันหรือไม่ ตอนนี้เราเลือก ได้ 2 อย่างคือ
 1. PC to Controller(download) > อัปเดตโปรแกรมจาก PC ของเราลงไปใน PLC
 2. Controller to PC(upload) > โหลดโปรแกรมจาก PLC เข้าสู่ PC ของเรา

The screenshot shows the 'Commission' interface after login. A yellow warning box with a triangle icon contains the text: 'PC and controller applications are different' and 'Compare computer and controller applications'. Below this, there is a table showing the project details:

Found: New project		
	Reference	Firmware
Controller	TM221CE16R	1.5.0.0

On the right side, there are buttons for 'Login', 'Logout', 'PC to Controller (download)', 'Controller to PC (upload)', 'Stop controller', and 'Start Controller'.

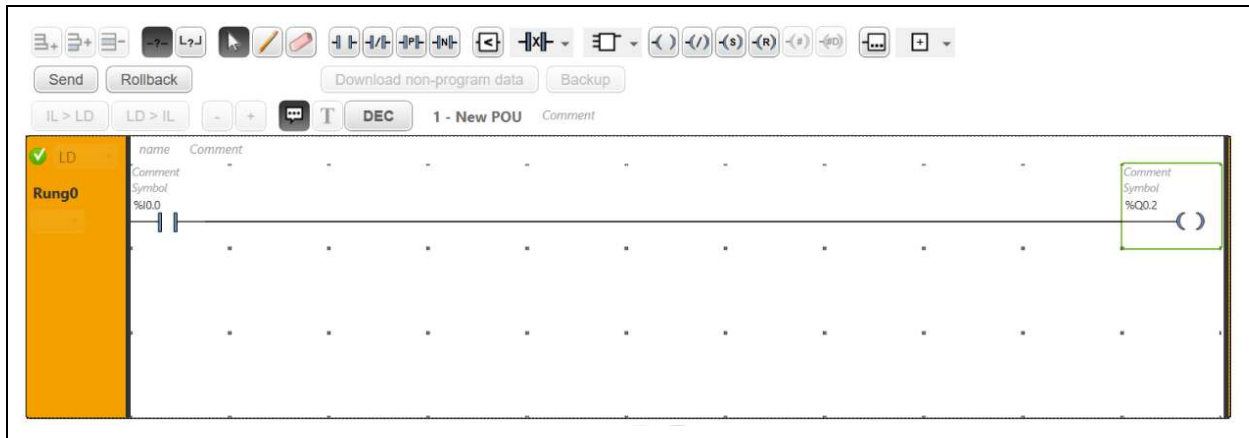
- เมื่อเราอัปเดตจาก PC ลง PLC เสร็จแล้ว จะมีข้อความแจ้งว่า โปรแกรมเป็นตัวเดียวกันแล้ว ถ้าเราต้องการให้ PLC เริ่มทำงานตามโปรแกรมของเราก็กด Start Controller ได้เลย

The screenshot shows the 'Commission' interface after the update. A green success box with a checkmark icon contains the text: 'PC and controller applications are identical' and 'Connection is established'. Below this, there is a table showing the project details:

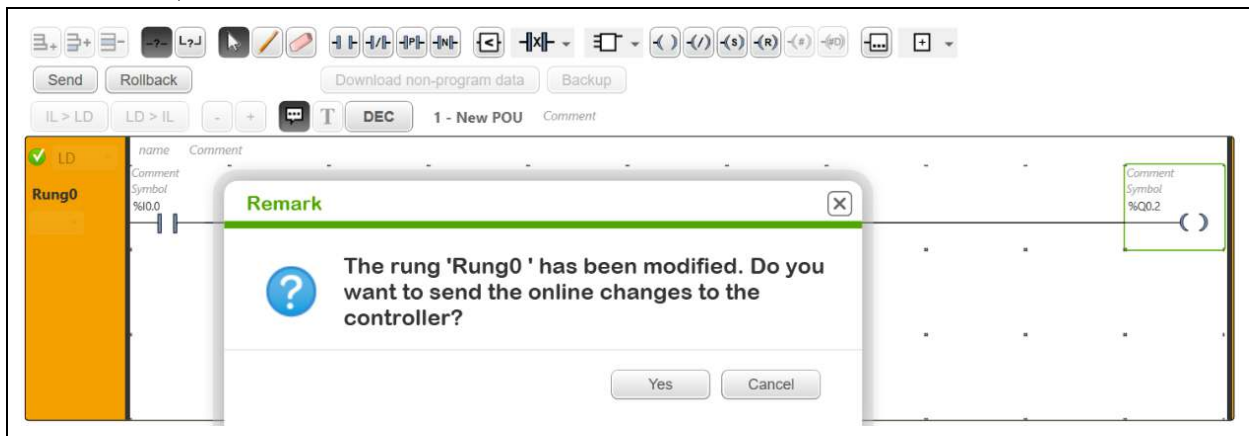
Found: New project		
	Reference	Firmware
Controller	TM221CE16R	1.5.0.0

On the right side, there are buttons for 'PC to Controller (download)', 'Controller to PC (upload)', 'Stop controller', and 'Start Controller'.

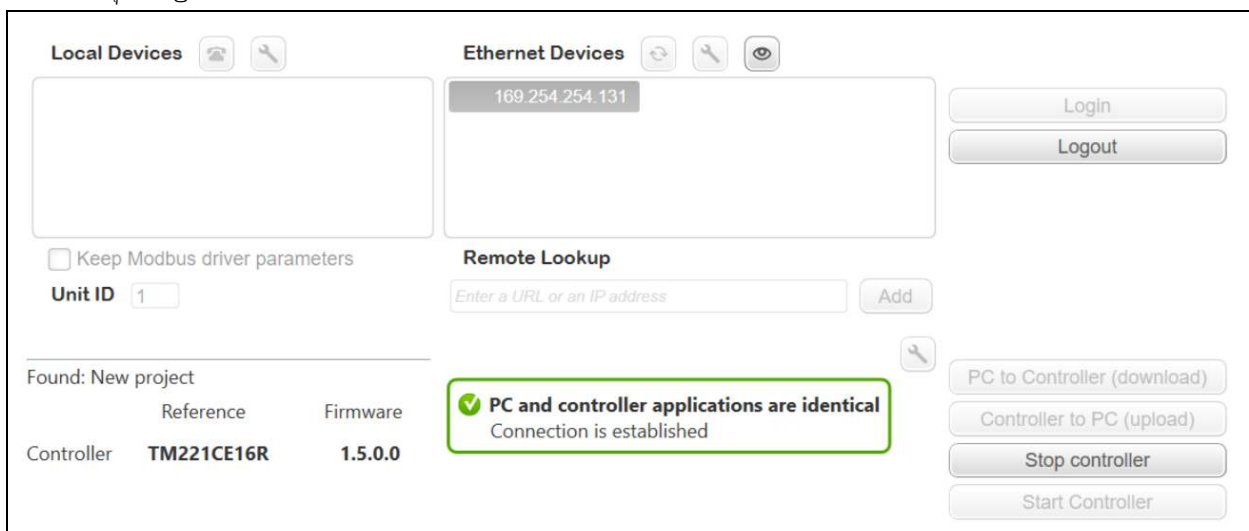
- ที่นี้ เมื่อเราทำการแก้ไขโปรแกรมของเราเช่น เปลี่ยน Address ของ Coil จะขึ้นสีส้มเพื่อเตือนว่ามีการแก้ไขโปรแกรม ภายใน Rung นั้นๆ



- หากเราย้ายไปหน้า Commission ทั้งที่ Rung ยังเป็นสีส้มอยู่ จะขึ้นหน้าต่างแจ้งเตือนเราว่า โปรแกรมมีการปรับปรุง ต้องการอัปเดตไปยัง PLC หรือไม่



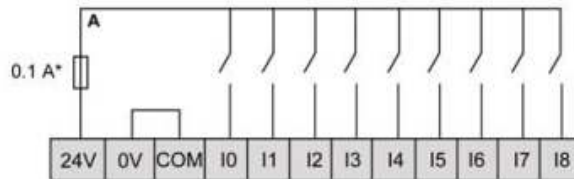
- ปุ่ม Stop Controller > หยุดการทำงานของ PLC ตัวนั้น
- ปุ่ม Logout > จะทำการบันทึกและตัดการเชื่อมต่อกับ PLC ตัวนั้น



9. การต่อวงจร Input, Output ของ M221CE16R

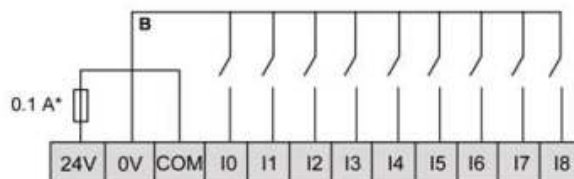
TM221C16R / TM221CE16R Wiring Diagrams

The following figure shows the sink wiring diagram (positive logic) of the inputs to the sensors for TM221C16R and TM221CE16R:



* Type T fuse

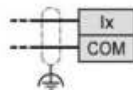
The following figure shows the source wiring diagram (negative logic) of the inputs to the sensors for TM221C16R and TM221CE16R:



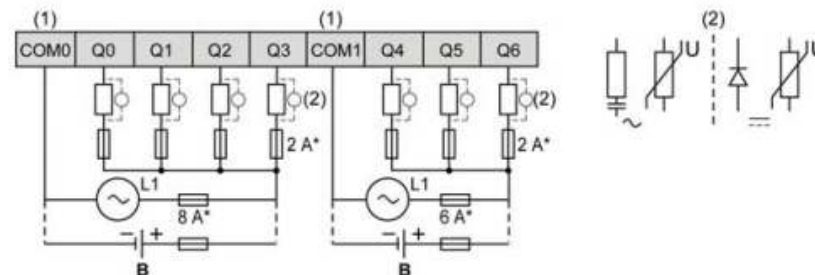
* Type T fuse

NOTE: The TM221C Logic Controller provides a 24 Vdc supply to the inputs.

The following figure shows the connection of the fast inputs:

**Relay Outputs Wiring Diagrams - Positive Logic (Sink)**

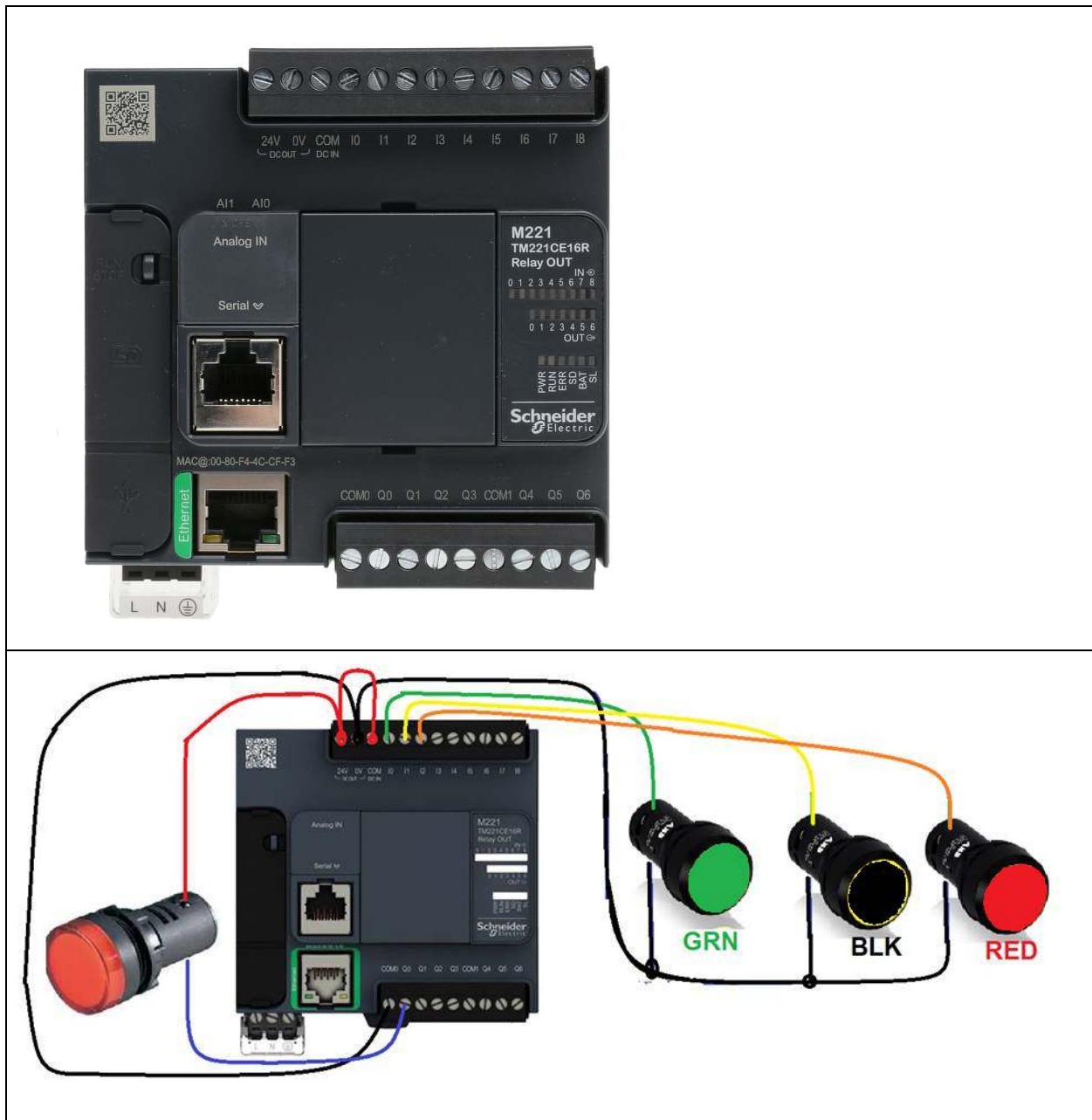
The following figure shows the sink wiring diagram (positive logic) of the outputs of the to the load for the TM221C16R / TM221CE16R:



* Type T fuse

- (1) The COM1 and COM2 terminals are **not** connected internally.
- (2) To improve the life time of the contacts, and to protect from potential inductive load damage, you must connect a free wheeling diode in parallel to each inductive DC load or an RC snubber in parallel of each inductive AC load

10. ทดสอบการทำงาน Input IO ส่งค่าไปยัง Output Q0



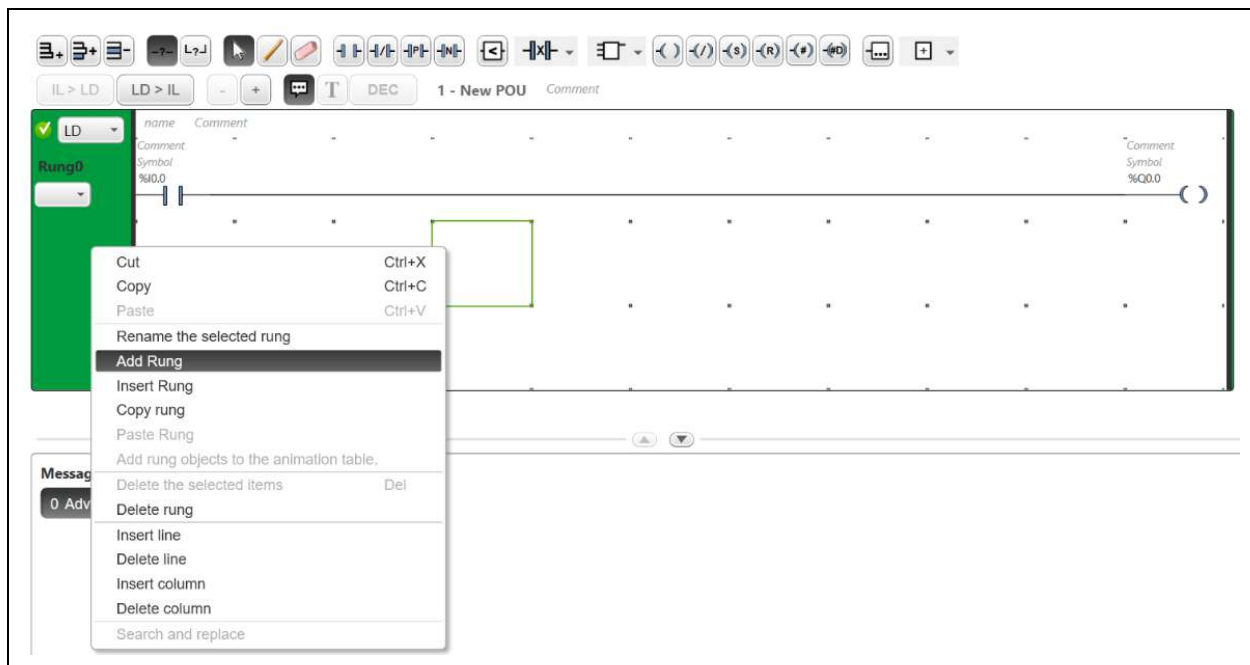
Test 2/8. SoMachine Basic – 4 Input/ 4 Output

11. ต้องการที่จะมี Input 4 ตัว เพื่อไปออก Output 4 ตัว ดังต่อไปนี้

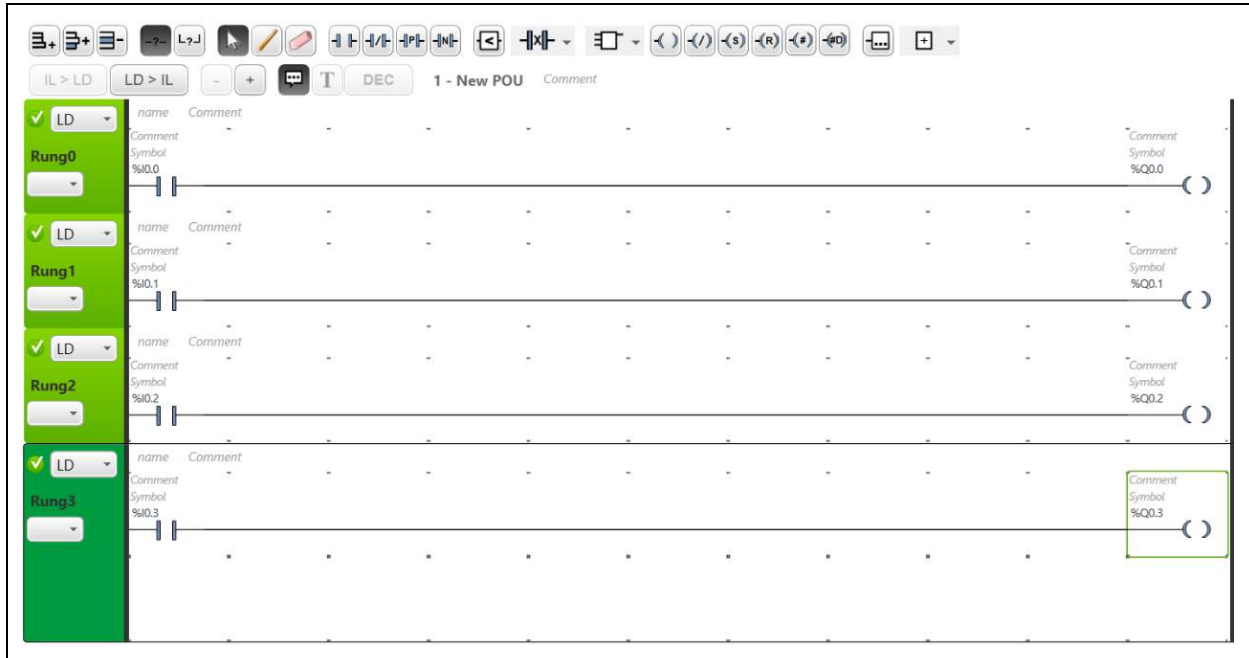
- รับค่า Input ที่ I0.0 ส่งค่า Output ออกที่ Q0.0
- รับค่า Input ที่ I0.1 ส่งค่า Output ออกที่ Q0.1
- รับค่า Input ที่ I0.2 ส่งค่า Output ออกที่ Q0.2
- รับค่า Input ที่ I0.3 ส่งค่า Output ออกที่ Q0.3

(แต่ละ Rung จะไม่สามารถมีสมการได้มากกว่า 1 สมการ ** Hint Add/Insert New Rung)

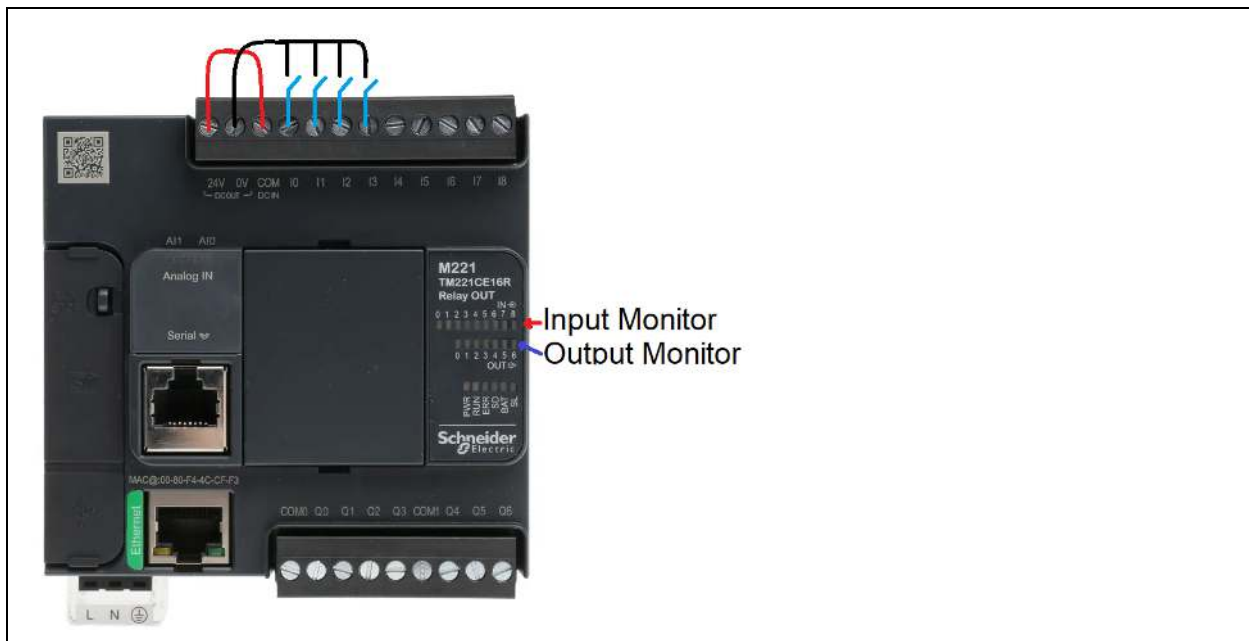
12. จากโปรแกรมเดิม ชั้นแรกเราจะต้อง Add Rung ขึ้นมาก่อน เพราะ เรามีสมการมากกว่า 1 ทำให้ Rung เดียวไม่สามารถทำได้



13. เมื่อเราเพิ่ม Rung ขึ้นมาเป็น 4 อันเพื่อรองรับการทำงานแล้ว ให้เราทำการในหลายๆ Rung ให้สมบูรณ์ คือให้ Input ไปออก Output ที่ต้องการจนครบ ตามรูปตัวอย่าง



14. ทดสอบการทำงาน Input IO ส่งค่าไปยัง Output Q0



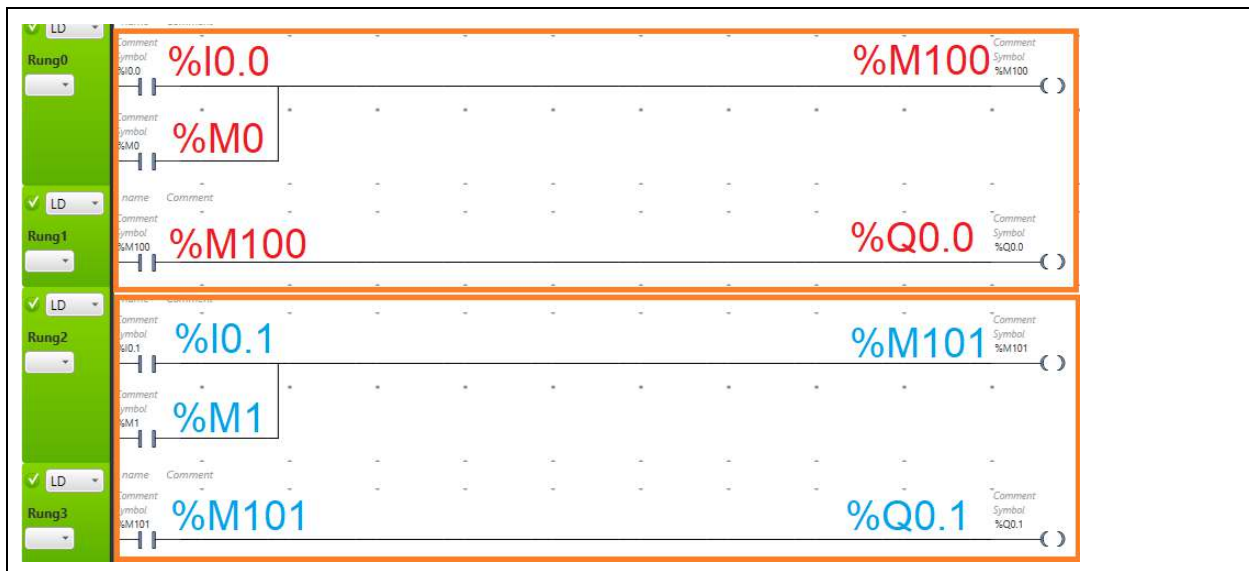
Test 3/8. Vijeo Citect SCADA

15. โหลดโปรแกรม Vijeo Citect 2015 จาก <https://www.citect.schneider-electric.com/scada/citectscada/downloads-updates/product-downloads>

16. แปลงโปรแกรมเดิม ดังนี้

- | | |
|----------------------------|------------------------|
| • รับ Contact %IO.0 or %M0 | ส่งค่าไปยัง Coil %M100 |
| • นำ Contact %M100 | ส่งค่าไปยัง Coil %Q0.0 |
| • รับ Contact %IO.1 or %M1 | ส่งค่าไปยัง Coil %M101 |
| • นำ Contact %M101 | ส่งค่าไปยัง Coil %Q0.1 |
| • รับ Contact %IO.2 or %M2 | ส่งค่าไปยัง Coil %M102 |
| • นำ Contact %M102 | ส่งค่าไปยัง Coil %Q0.2 |
| • รับ Contact %IO.3 or %M3 | ส่งค่าไปยัง Coil %M103 |
| • นำ Contact %M103 | ส่งค่าไปยัง Coil %Q0.3 |

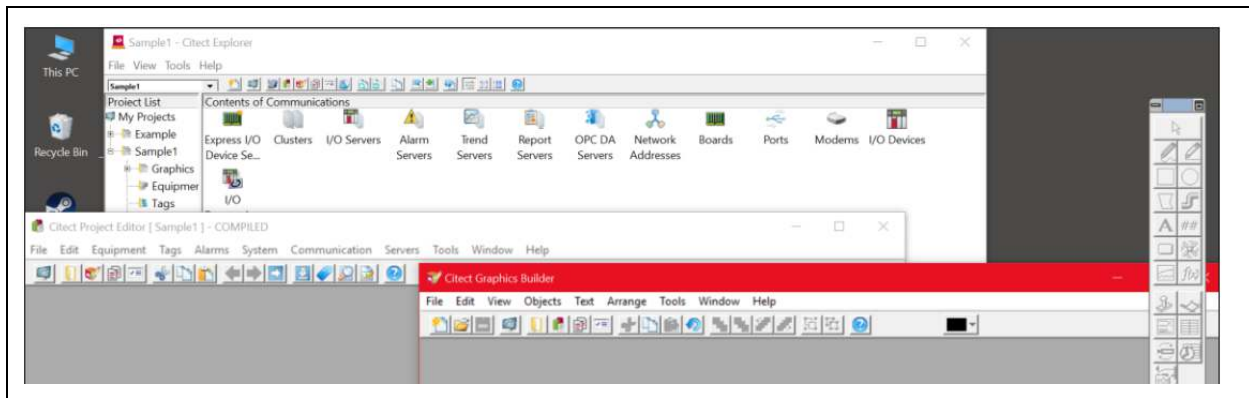
เนื่องจากในการทำงานใน SCADA จะทำผ่าน Memory Register ไม่สามารถนำ Input Output มาแสดงได้



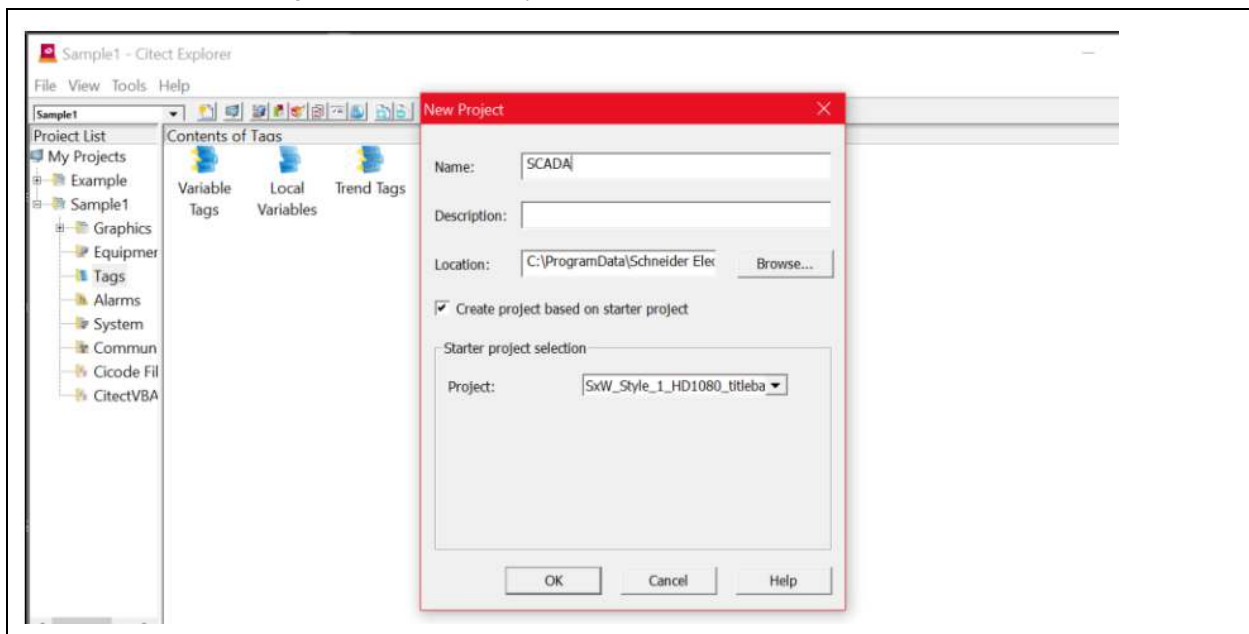
17. ลงโปรแกรมให้เรียบร้อย แล้วเปิดโปรแกรมขึ้นมา



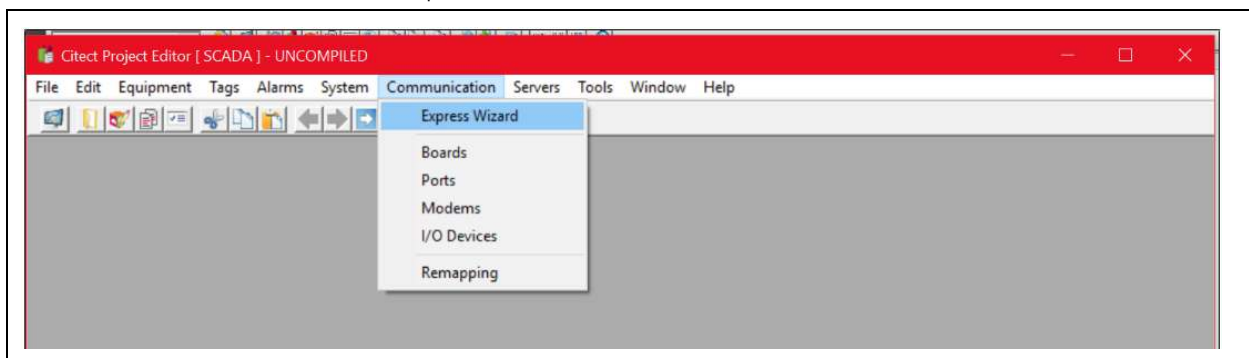
18. จะหน้าต่างดังขึ้นมา 3 หน้า คือ Citect Project Editor, Citect Graphic Builder, Citect Explorer



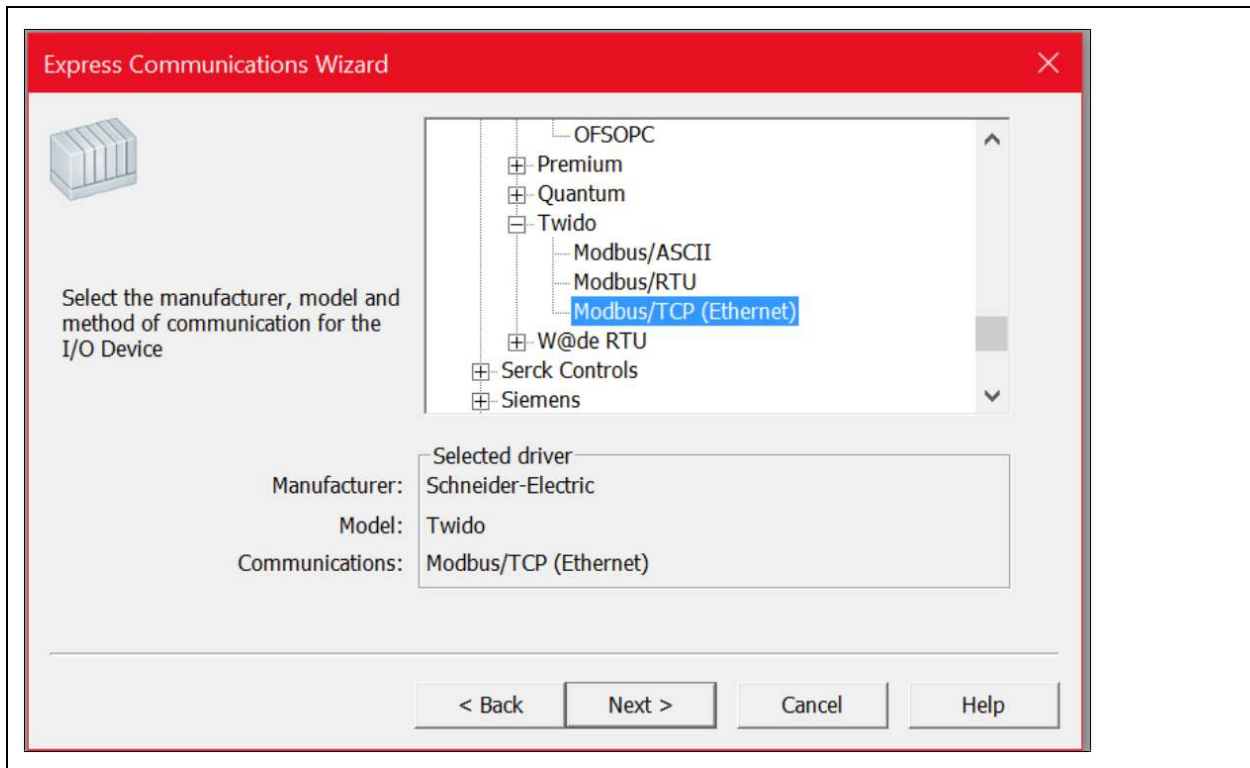
19. เราจะไป New Project ที่หน้า Citect Explorer



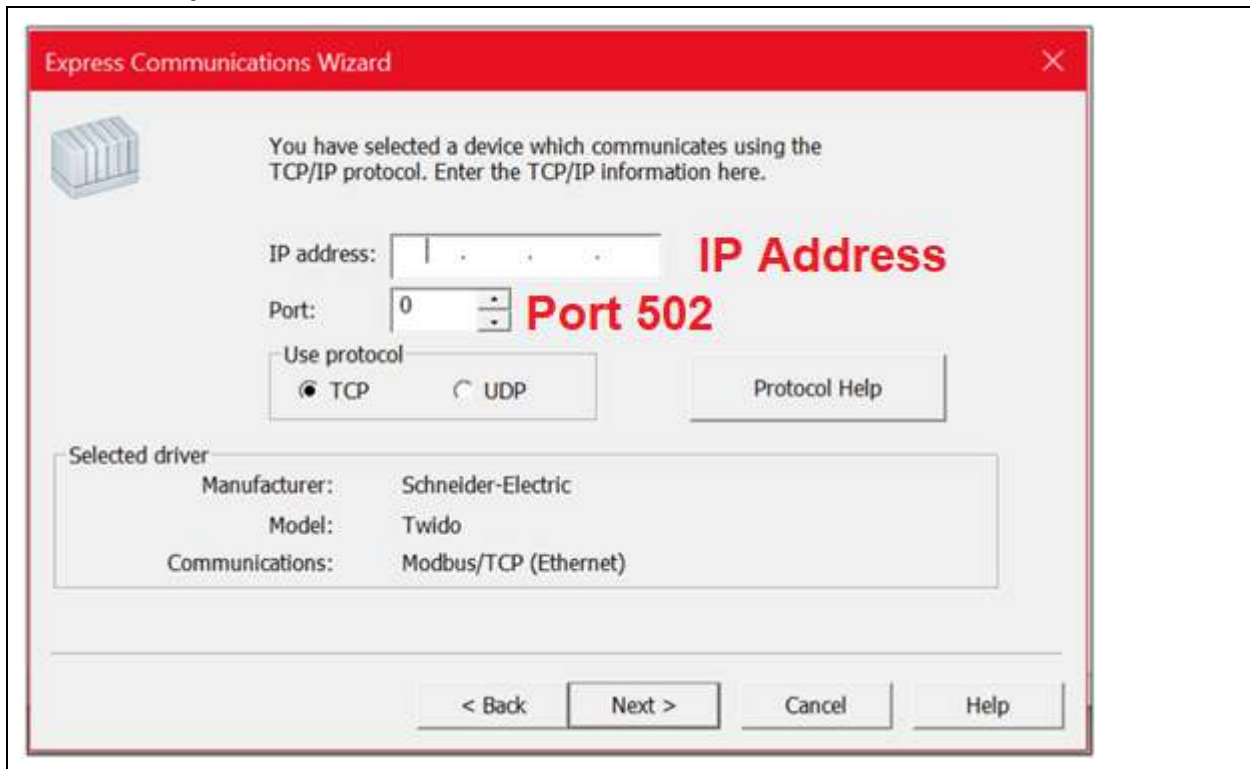
20. หลังจากสร้างโปรเจ็คขึ้นมาแล้วให้ไปที่โปรแกรม Citect Project Editor เพื่อเชื่อมต่อโปรเจ็คเรากับ PLC โดยเลือก Communication > Express Wizard



21. กด Next ไปเรื่อยๆ จนกระทั่งถึงหน้าที่ให้เราเลือกโมเดล ให้เราเลือก Modbus/TCP(Ethernet) IP

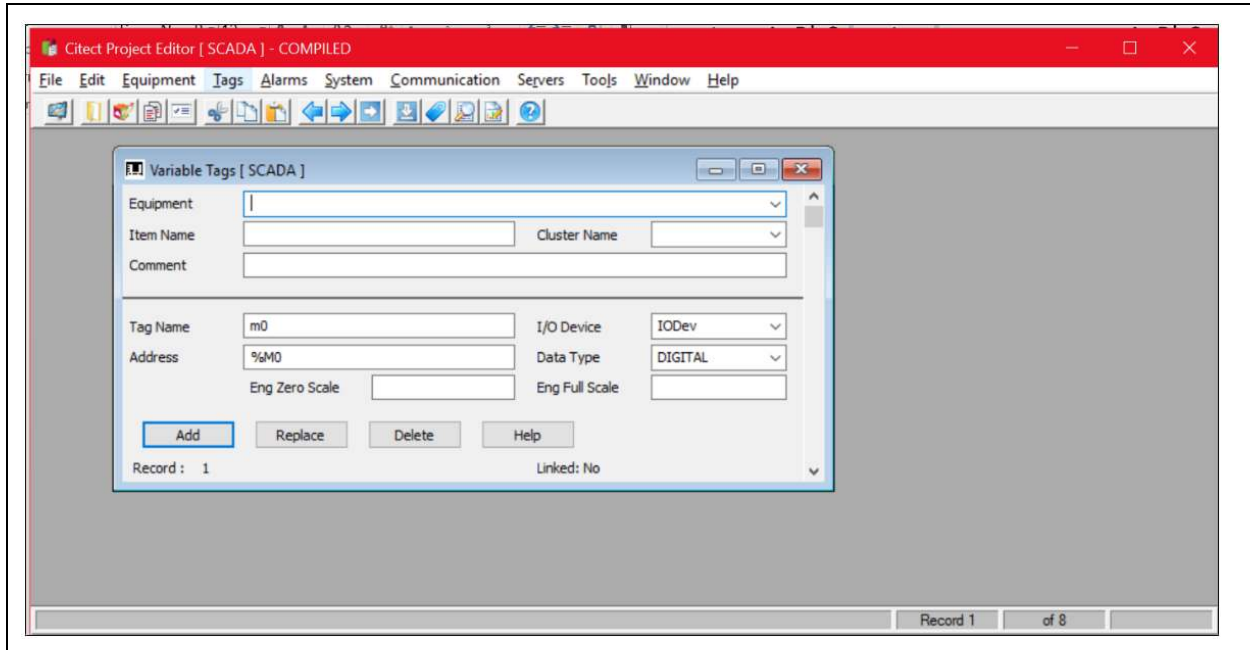


22. หน้าข้อมูลการเชื่อมต่อ IP Address > ใส่ IP Address ของ PLC ที่เราจะเชื่อมต่อ, Port = 502

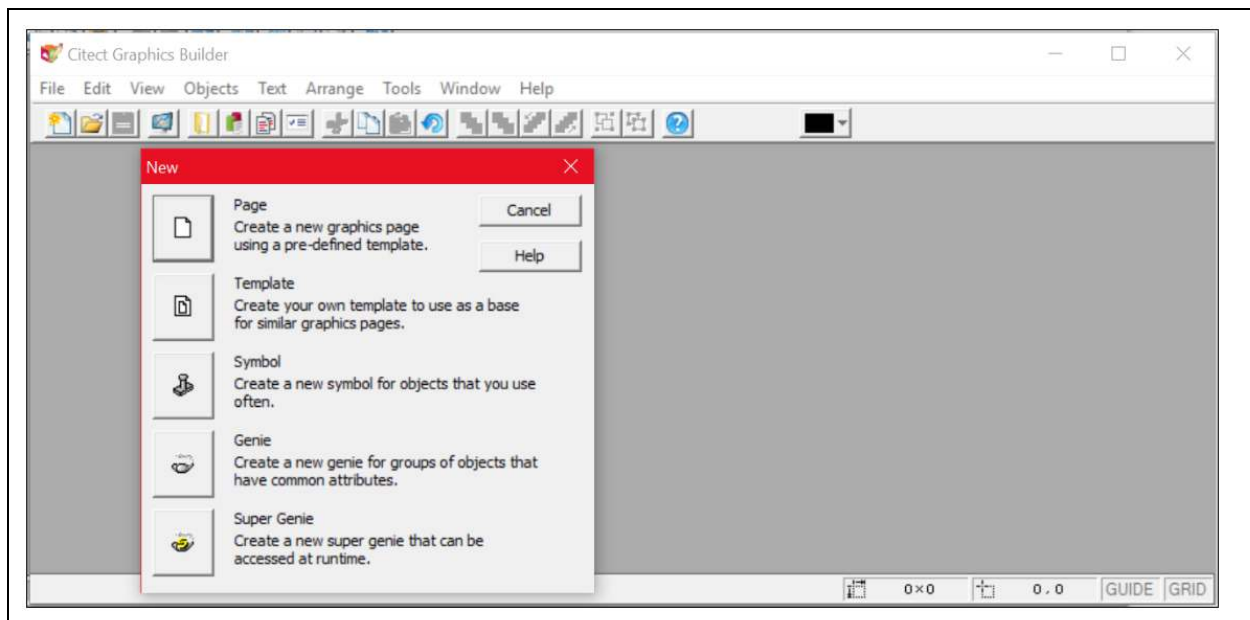


23. เมื่อเสร็จสิ้นจากการเชื่อมต่อกับ PLC แล้ว ให้เราไปตั้งชื่อตัวแปรที่ใช้กับ PLC ทุกตัวที่เราใช้ กรณีนี้ต้องตั้งให้ M0,M1,M2,M3,M100,M101,M102,M103

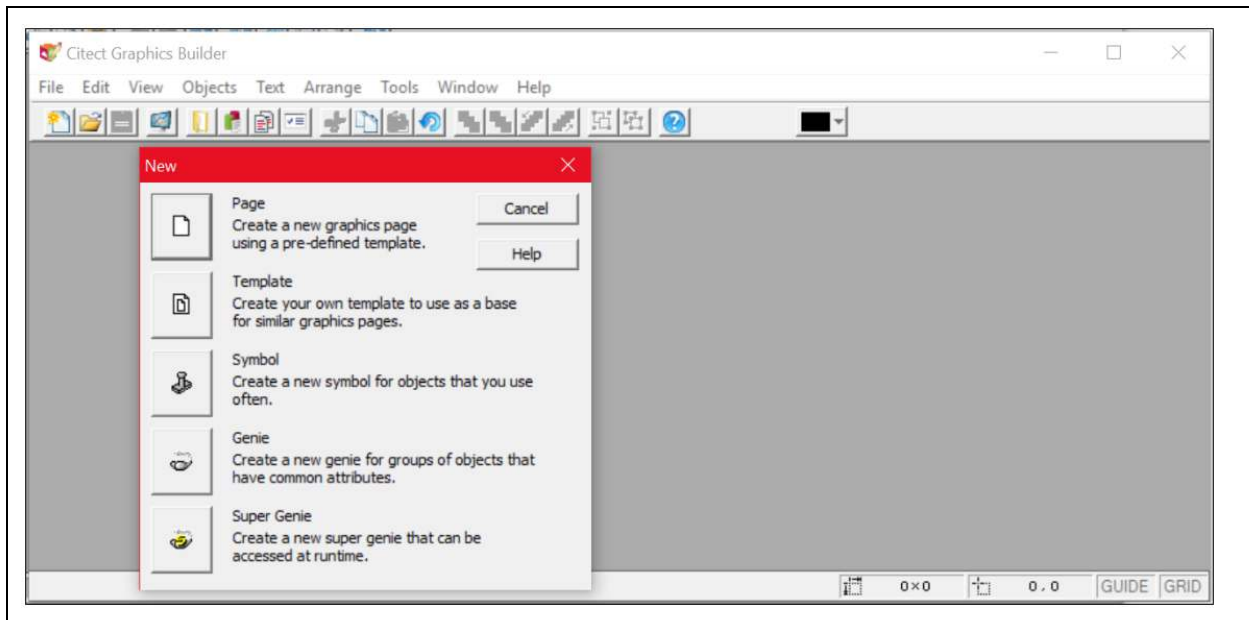
- Tag Name > ชื่อที่จะใช้เรียกค่า Address ตัวนั้น ๆ
- Address > ตำแหน่งที่อยู่ของข้อมูล
- I/O Device > อุปกรณ์
- Data Type > ชนิดของข้อมูลตัวที่กำลังจะตั้ง



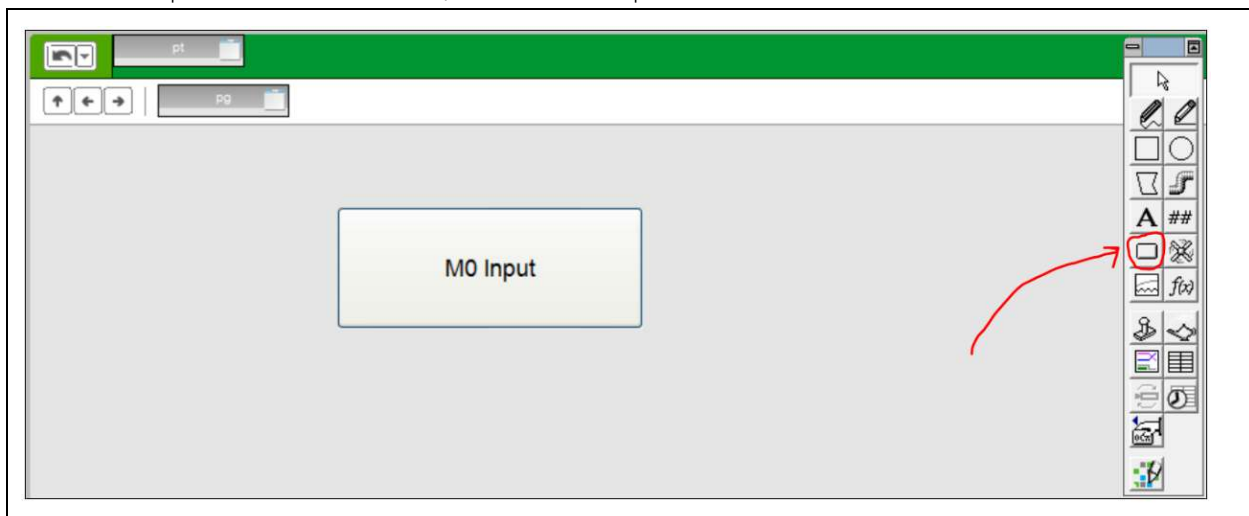
24. ที่หน้าต่าง Citect Graphic Builder จะสร้าง Page ใหม่



25. เลือก Template : **Normal** , Style : **swx_style_1** แล้วกด OK

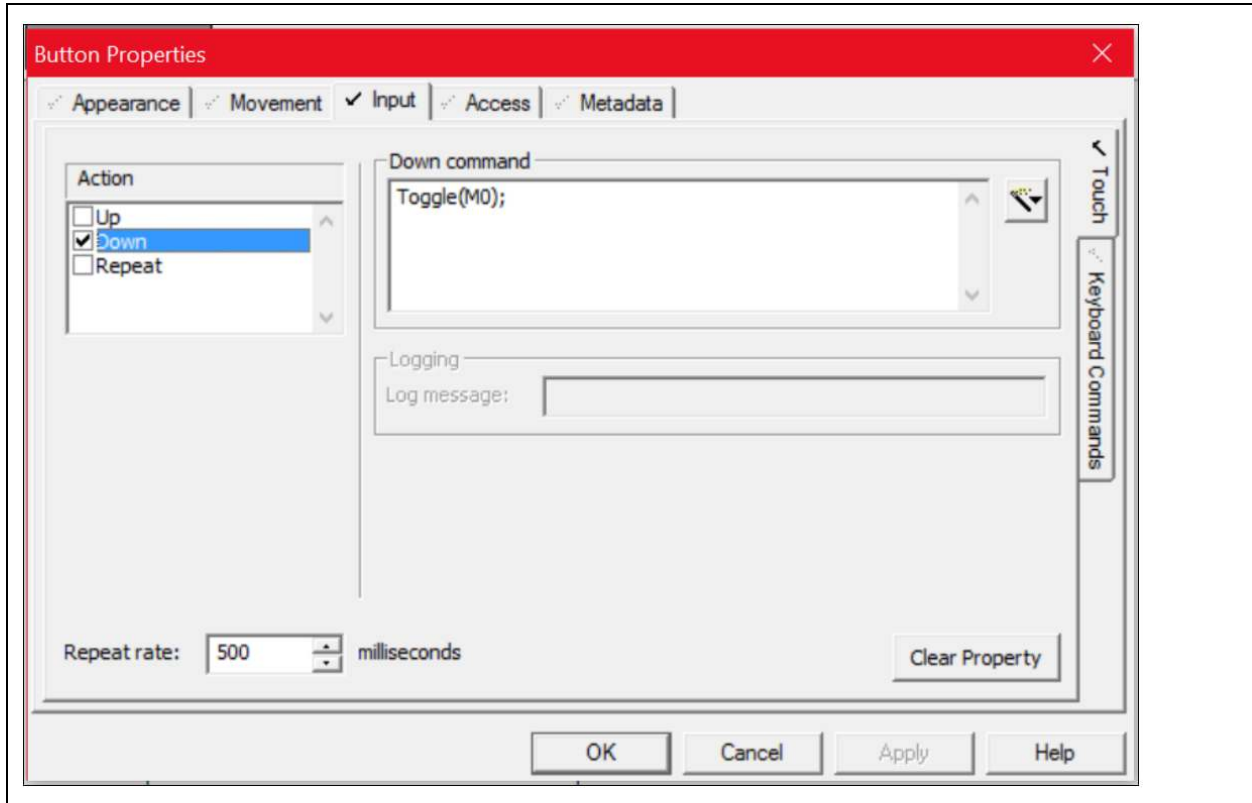


26. สร้างปุ่มกดขึ้นตามจำนวน Input ของเรา (4 ปุ่ม)

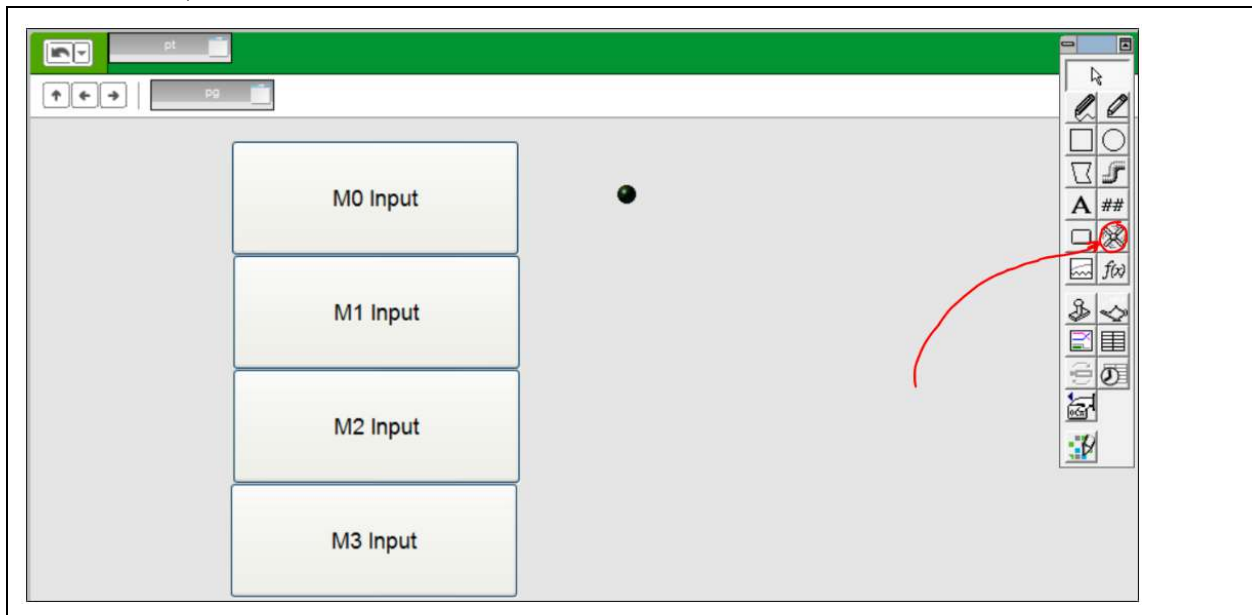


27. เมื่อสร้างครบ ให้คลิกเข้าไปในปุ่มเพื่อแก้ไข Input ตามตัวอย่างคือแก้ไขปุ่ม MO โดยสั่งให้เมื่อมีการกดปุ่มให้กลับค่าที่ อยู่ใน MO

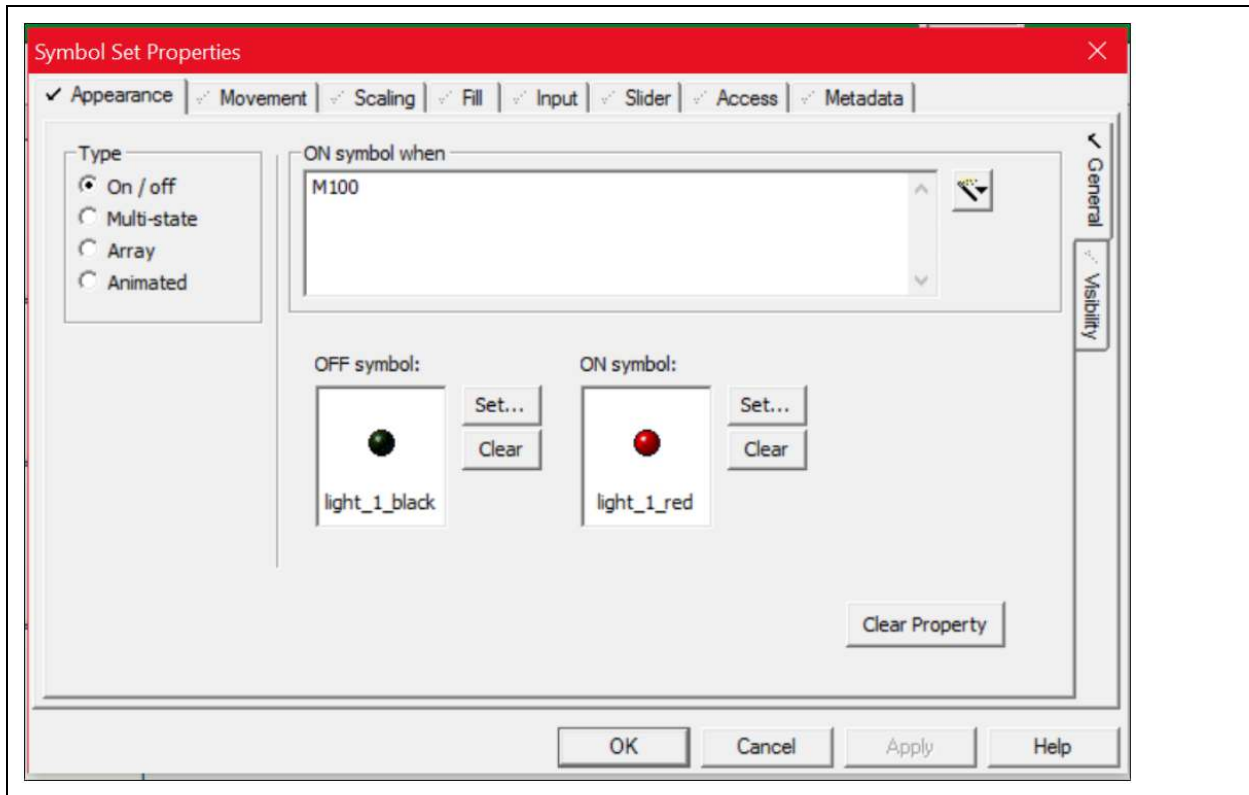
- Action > เกิดขึ้นเมื่อทำอะไรกับปุ่ม
- Command > คำสั่งที่ให้ทำ หลังจากเกิด Action นั้นๆ



28. เมื่อทำปุ่มเสร็จแล้ว ให้เราสร้างหลอดไฟ เพื่อแสดง Output โดยจะสร้างทั้งสี่หลอด

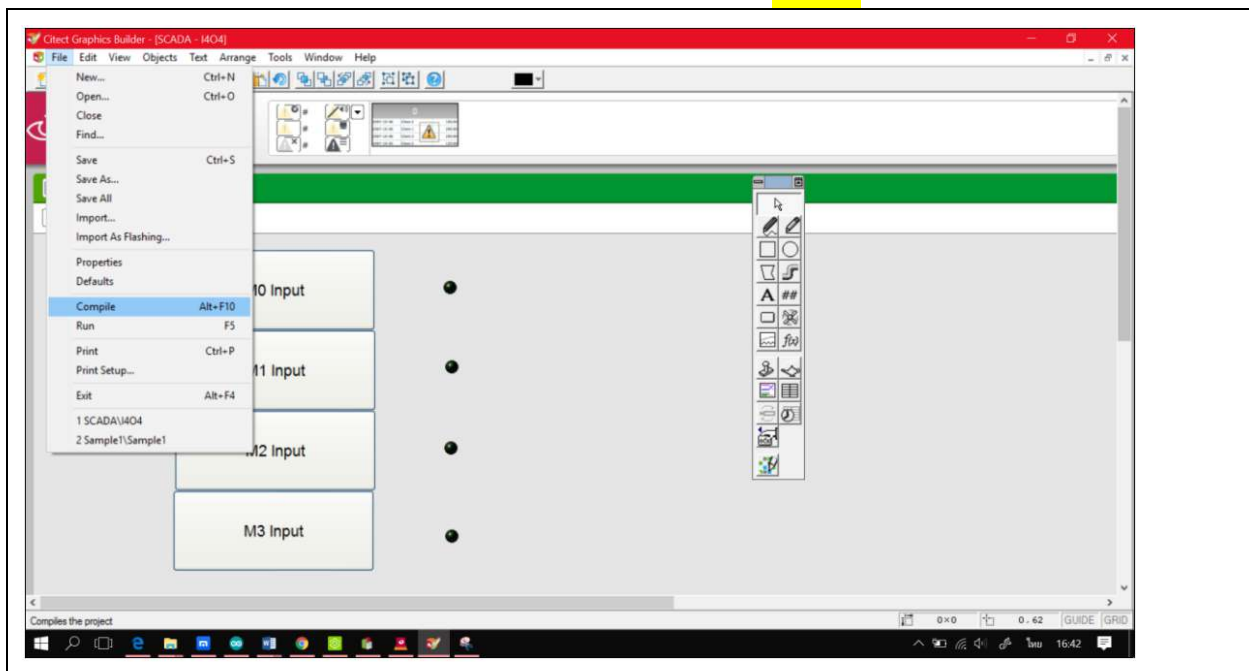


29. เมื่อสร้างครบ ให้คลิกเข้าไปในหลอดไฟเพื่อแก้ไขว่าหลอดไฟยังสว่างเมื่อไหร่ (ON symbol when) เราก็จะใส่ไอลิปจน ครบคือ M100,M101,M102,M103

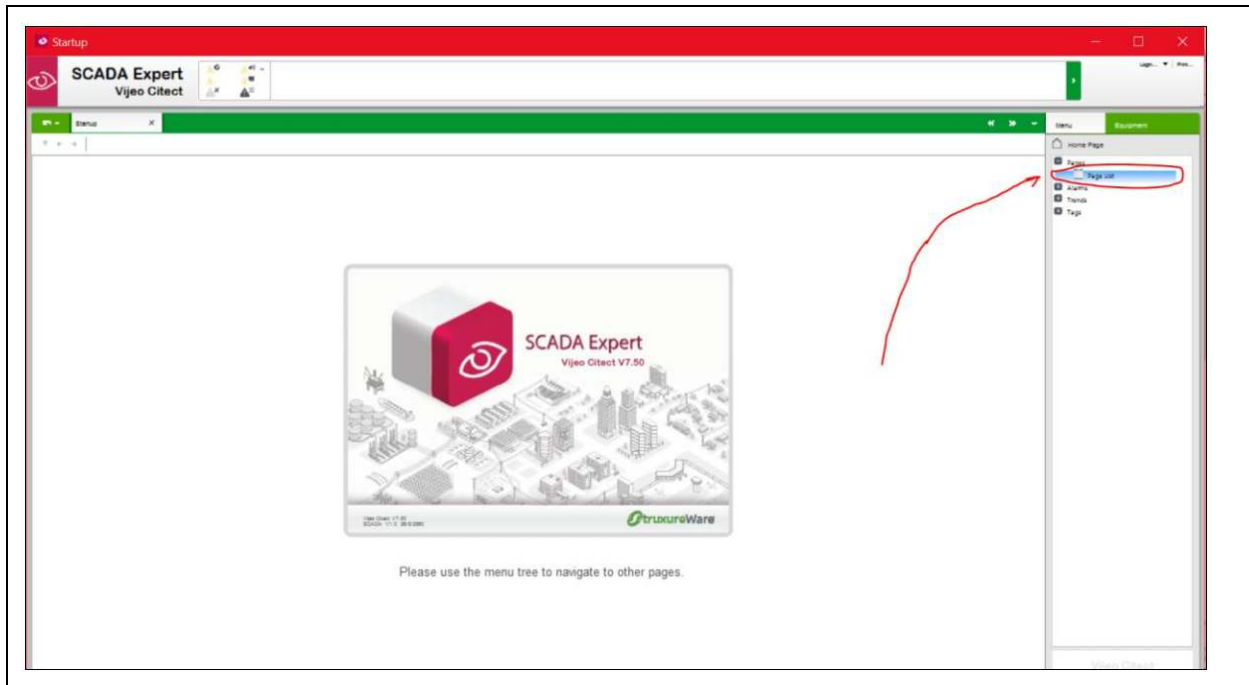


30. เมื่อแก้ไขเสร็จหมดแล้ว ให้ Compile(Alt+F10), เมื่อ Compile เสร็จ ก็สั่ง Run ไปเลย

***การ Compile ครั้งแรก เราจะต้องตั้งชื่อ Page อาจตั้งเป็น **Startup** หากต้องการเรียกแล้วทำงานทันที



31. เมื่อโปรแกรม Startup ถูกเปิด (เป็นแบบ Demo สามารถใช้งานได้ 15 นาทีก่อนที่จะปิดตัวเองลงไป) เราจะเปิดไปยัง Page ของเรา โดยคลิกที่ Page List จะมีชื่อ Page ขึ้นมา เราก็เลือกชื่อ Page ที่เราตั้งเอาไว้ (หากตั้งเป็น Startup จะทำงานทันทีเป็นหน้าแรก)



32. เมื่อกดที่ปุ่ม

	<p> ป้อน IO.0 Input จะทำให้ LED M100 และ QO.0 ทำงาน ป้อน IO.1 Input จะทำให้ LED M101 และ QO.1 ทำงาน ป้อน IO.2 Input จะทำให้ LED M102 และ QO.2 ทำงาน ป้อน IO.3 Input จะทำให้ LED M103 และ QO.3 ทำงาน </p> <p> กดปุ่ม M0 Input จะทำให้ LED M100 และ QO.0 ทำงาน กดปุ่ม M1 Input จะทำให้ LED M101 และ QO.0 ทำงาน กดปุ่ม M2 Input จะทำให้ LED M102 และ QO.0 ทำงาน กดปุ่ม M3 Input จะทำให้ LED M103 และ QO.0 ทำงาน </p>
--	---

33. ทดสอบการทำงานด้วยการป้อน อินพุต IO.0 จะส่งค่าไปยัง M100 ซึ่งจะทำให้ LED ติดที่ Dashboard และส่งค่าไปยัง QO.0 ด้วย และสามารถควบคุม QO.0 ผ่านปุ่ม MO-Input จาก dashboard ได้ด้วย

3/5: -- การโปรแกรมเพื่อสื่อสารข้อมูลผ่าน Modbus TCP

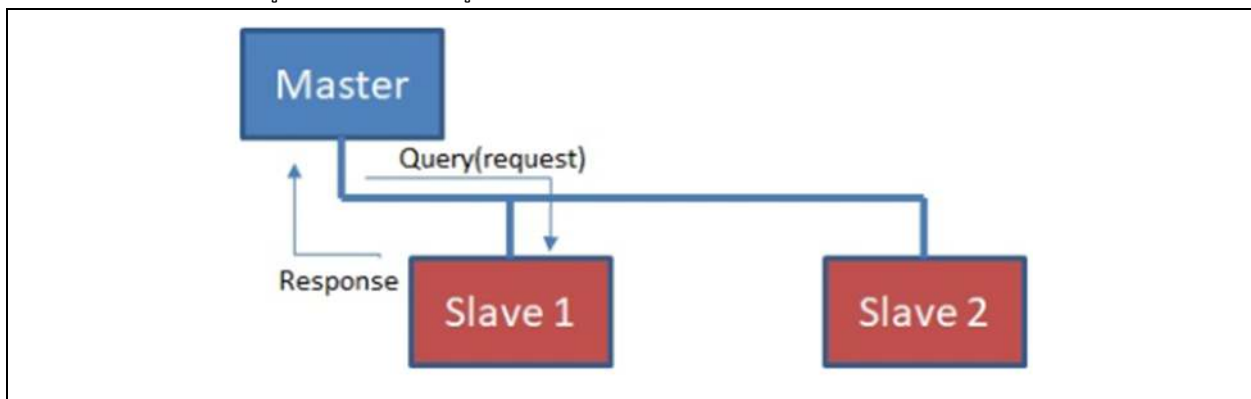
การสื่อสารผ่าน Modbus TCP/IP

<https://sonicautomation.co.th/2019/สื่อสารผ่าน-modbus-tcp-ip/>

โพรโทคอลยอดนิยมที่ใช้กันอย่างกว้างขวางในวงการอุตสาหกรรมคงหนีไม่พ้น Modbus ซึ่งมีทั้ง Modbus RTU และ Modbus TCP ถึงแม้ว่าในปัจจุบันจะมีโพรโทคอลอื่นๆที่มีสมรรถนะที่ดีกว่า Modbus มากมาย แต่ Modbus ก็ยังเป็นโพรโทคอลคลาสสิกและได้รับความนิยมจนถึงทุกวันนี้ ซึ่งโพรโทคอล Modbus ได้รับการพัฒนาขึ้นตั้งแต่ปี พ.ศ. 2522 โดย Modicon Incorporated เพื่อใช้ กับระบบควบคุมอัตโนมัติ สำหรับอุตสาหกรรมและคอนโทรลเลอร์ และมันได้กลายเป็นวิธีการมาตรฐานในอุตสาหกรรมสำหรับการถ่ายโอนข้อมูลแบบ on/off และแบบอนาล็อกตั้งแต่นั้นเป็นต้นมา

อุปกรณ์ที่สื่อสารด้วย Modbus โดยใช้วิธีการ Master-Slave (หรือ Client-Server) จะมีอุปกรณ์เพียงตัวเดียว ซึ่งคือ Master (หรือ Client) สามารถเริ่มต้นการสื่อสารได้เท่านั้น (queries) ส่วนอุปกรณ์ตัวอื่น ๆ จะทำหน้าที่เป็น Slaves (หรือ servers) จะตอบสนองต่อการสื่อสารนั้น โดย Slave จะส่งข้อมูลที่ร้องขอกลับไปยัง Master หรือโดยการดำเนินการบางอย่างตามที่ Master ร้องขอ

Slave อาจเป็นอุปกรณ์ต่อพ่วงใด ๆ (I/O transducer, วาล์ว, Inverter (VFD) หรืออุปกรณ์เครื่องมือวัดอื่นๆ) ซึ่งประมวลผลและส่งข้อมูลไปยัง Master รูปข้างล่างนี้แสดงการสื่อสารระหว่าง Master กับ Slave



Master สามารถติดต่อกับ Slave แต่ละตัวได้ หรือสามารถส่งเป็น Message ถึง Slave ทุกตัวได้ในลักษณะของการ Broadcast และ Slave จะตอบสนองสิ่งที่ Master ต้องการเท่านั้น สิ่งที่ Master ส่งให้จะประกอบด้วย Slave address, function code (คำสั่งหรือสิ่งที่ต้องการให้ทำ), Data และ Checksum ส่วนข้อมูลที่ Slave ส่งกลับ มาจะประกอบด้วยคำสั่งที่สั่งให้กระทำ ข้อมูลต่างๆ และ Checksum

Modbus TCP/IP คือ อะไร

Modbus TCP/IP (Modbus-TCP) คือ โพรโทคอล Modbus RTU ที่เชื่อมต่อกับ TCP โดยทำงานบนสถาปัตยกรรมของ Ethernet โครงสร้าง Message ของ Modbus คือ application protocol ที่จะถูกส่งผ่านไปพร้อมกับ TCP/IP (TCP/IP คือ Transmission Control Protocol และ Internet Protocol ซึ่งเป็นตัวกลางที่ใช้ในการส่ง Message ของ Modbus TCP/IP)

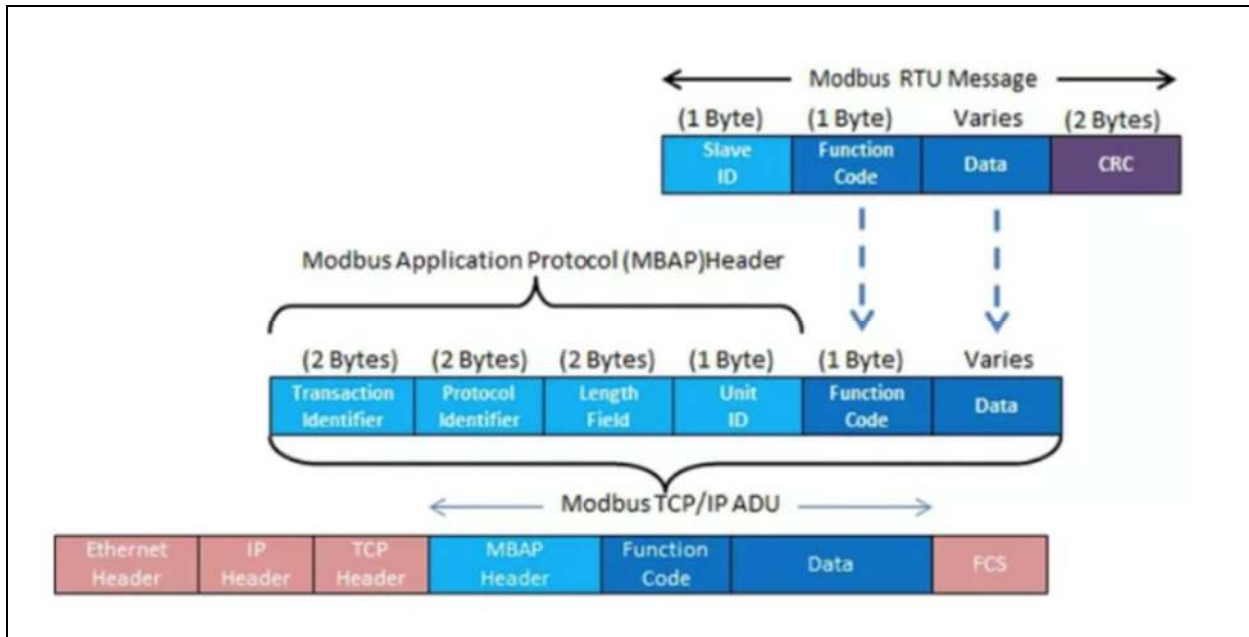
<u>Modbus RTU Data Type</u>	<u>Common name</u>	<u>Starting address</u>
Modbus Coils	Bits, binary values, flags	00001
Digital Inputs	Binary inputs	10001
Analog Inputs	Binary inputs	30001
Modbus Registers	Analog values, variables	40001

ตารางข้างบนนี้แสดงชนิดของ Data และ Address ที่ใช้กับ Modbus RTU ซึ่งถูกใช้งานเช่นเดียวกันใน Modbus TCP/IP

ส่วนตารางข้างล่างนี้แสดงชนิด Function และ Function code ของ Modbus RTU ที่ใช้สำหรับอ่านและเขียนข้อมูลซึ่งเราต้องใส่ไว้ใน Modbus RTU data frame

Function type		Function name	Function code
Bit access	Physical Discrete Inputs	Read Discrete Inputs	2
	Internal Bits or Physical Coils	Read Coils	1
		Write Single Coil	5
		Write Multiple Coils	15
16-bit access	Physical Input Registers	Read Input Registers	4
	Internal Registers or Physical Output Registers	Read Multiple Holding Registers	3
		Write Single Holding Register	6
		Write Multiple Holding Registers	16
		Read/Write Multiple Registers	23
		Mask Write Register	22
		Read FIFO Queue	24

ในทางปฏิบัติ Modbus TCP จะฝัง Modbus RTU data frame รวมไปกับ TCP frame โดยไม่ต้องใช้ Modbus checksum แต่จะใช้ Checksum ของ TCP แทนดังแสดงในรูปต่อไปนี้



จากรูป Modbus Application Protocol (MBAP) จะประกอบด้วยข้อมูล 7 bytes ซึ่งจะวางหน้า Modbus RTU message โดยมีรายละเอียดดังนี้

- Transaction/invocation Identifier (2 Bytes): ใช้จับคู่การแลกเปลี่ยนข้อมูลเมื่อมี Message หลายๆ ชุด ถูกส่งออกมาด้วย TCP เดียวกัน ด้วย Client ตัวใดตัวหนึ่ง โดยไม่ต้องรอลำดับการ Response
- Protocol Identifier (2 bytes): ในส่วนจะมีค่าเป็น 0 เสมอ
- Length (2 bytes): เป็นการระบุจำนวน Byte ที่รวมจำนวน Byte ของ unit identifier, function code, และ data fields
- Unit Identifier (1 byte): เป็นการระบุ ID ของ server ที่อยู่ในระบบสื่อสาร อาจตั้งเป็น 00 ถึง FF ก็ได้

ตัวอย่าง Modbus TCP message

เรามาดูตัวอย่างโปรโตคอล Modbus RTU สำหรับอ่านค่าอนาล็อกเอาต์พุตของ Holding register แอดเดรส 40108 ถึง 40110 จาก Slave หมายเลข 17

11 03 006B 0003 7687

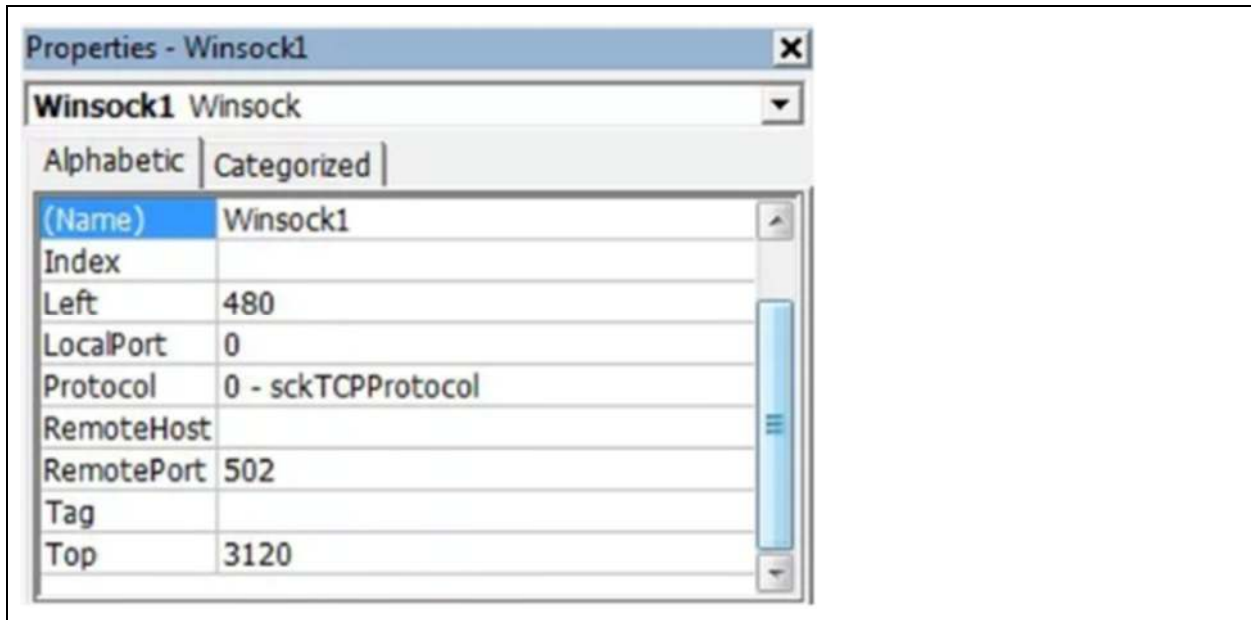
- 11: SlaveID Address (17 = 11 hex)
- 03: Function Code (read Analog Output Holding Registers)
- 006B: Data Address ของ register ตัวแรก (40108-40001 = 107 = 6B hex)
- 0003: จำนวน registers ที่ต้องการอ่าน (อ่าน 3 ตัว 40108 ถึง 40110)
- 7687: เป็น CRC (cyclic redundancy check) สำหรับเช็คความผิดพลาด

เมื่อนำไปรวมกับ MBAP จะได้รายละเอียดโปรโตคอล Modbus TCP ดังนี้ ซึ่ง Frame checksum (CRC) จะถูกตัดออกไป

0001 0000 0006 11 03 006B 0003

- 0001: Transaction Identifier
- 0000: Protocol Identifier
- 0006: Message Length (6 bytes ที่อยู่ตำแหน่งถัดไปจาก Byte นี้)
- 11: Unit Identifier (17 = 11 hex)
- 03: Function Code (อ่านค่า Analog Output Holding Registers)
- 006B: Data Address ของ register ตัวแรก (40108-40001 = 107 = 6B hex)
- 0003: จำนวน registers ที่ต้องการอ่าน (อ่าน 3 ตัว 40108 ถึง 40110)

สมมุติว่าคุณต้องการเขียนโปรแกรม VB เพื่ออ่านข้อมูล Holding register จาก Slave หมายเลข 1 โดย Function code เพื่ออ่านค่าข้อมูลจาก Holding register ซึ่งก็คือ 3 นั้นเอง สามารถดูตัวอย่างการเขียนโปรแกรมได้ดังตัวอย่างข้างล่างนี้ การส่ง Message ผ่าน TCP/IP จะใช้ winsock ของ VB โดยค่า Remote port ที่ตั้งจะเป็น 502 ซึ่งเป็นค่าปกติของ Modbus TCP/IP



ข้างล่างนี้คือตัวอย่างโค้ด Visual Basic สำหรับการสื่อสารในรูปแบบ Modbus TCP

```
Private Sub ReadLW_Click()
'Byte 0 : transaction identifier (upper byte) – copied by server – usually 0
'Byte 1 : transaction identifier (lower byte) – copied by server – usually 0
'Byte 2 : protocol identifier (upper byte) = 0
'Byte 3 : protocol identifier (lower byte) = 0
'Byte 4 : length field (upper byte)
'Byte 5 : length field (lower byte) = number of bytes following
'Byte 6 : unit identifier (slave address)
'Byte 7 : MODBUS function code
'Byte 8 : address of register (upper byte)
'Byte 9 : address of register (lower byte)
'Byte 10 : number of register as need (upper byte)
'Byte 11 : number of register as need (lower byte)
```

```

'The Mod operator returns the remainder of a division. For example, the expression 14 Mod 4 evaluates to 2
'The \ Operator returns the integer quotient of a division. For example, the expression 14 \ 4 evaluates to 3
'The Val function returns number of a string (convert string to number)
Dim StartLow As Byte
Dim StartHigh As Byte
Dim LengthLow As Byte
Dim LengthHigh As Byte
If (Winsock1.State = 7) Then
    StartLow = Val(AdssLW.Text) Mod 256
    StartHigh = Val(AdssLW.Text) \ 256
    LengthLow = Val(NoOfWord.Text) Mod 256
    LengthHigh = Val(NoOfWord.Text) \ 256
    MbusQuery(0) = 0
    MbusQuery(1) = 0
    MbusQuery(2) = 0
    MbusQuery(3) = 0
    MbusQuery(4) = 0
    MbusQuery(5) = 6 '6 Hex :Length field, there are 6 byte count from Unit ID to the last byte
    MbusQuery(6) = 1 '1 Hex :Unit ID (Slave address)
    MbusQuery(7) = 3 '3 Hex :Function code READ multiple holding register
    MbusQuery(8) = StartHigh
    MbusQuery(9) = StartLow
    MbusQuery(10) = LengthHigh
    MbusQuery(11) = LengthLow
    MbusRead = True
    MbusWrite = False
    Winsock1.SendData MbusQuery
    ModbusWait = True
    ModbusTimeOut = 0
    Timer1.Enabled = True
    MbusReadByteOrBit = True
Else
    MsgBox ("Device not connected via TCP/IP")
End If
End

```

จากตัวอย่างโค้ด VB ข้างบนเมื่อเรา Compile แล้ว โปรแกรมที่ได้จะทำงานบนคอมพิวเตอร์ซึ่งจะทำหน้าที่เป็น Master และสามารถใช้โปรแกรมนี้เพื่อเชื่อมต่อกับ Slave ที่รองรับโปรโตคอล Modbus TCP ได้ นอกจากนั้นเราสามารถนำหลักการเดียวกันนี้เพื่อนำไปประยุกต์ใช้กับการเขียนโปรแกรม PLC เพื่อเชื่อมต่อกับอุปกรณ์ที่รองรับ Modbus TCP ได้

Test 4/8. Read from Modbus TCP Device

1. ใช้ HF5111B ในการแปลง ModbusRTU/ASCII เป็น ModbusTCP



Commercial Serial Device Server

- -40~70°C
- Linux Operating System
- International Quality System Certification

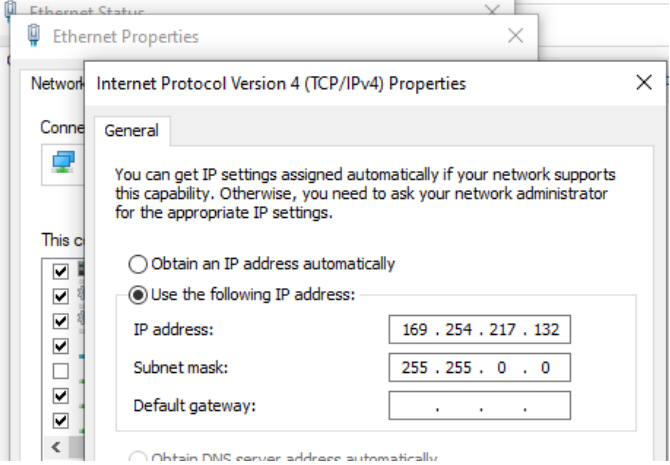
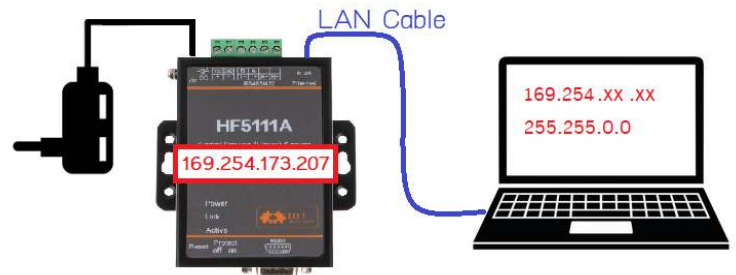
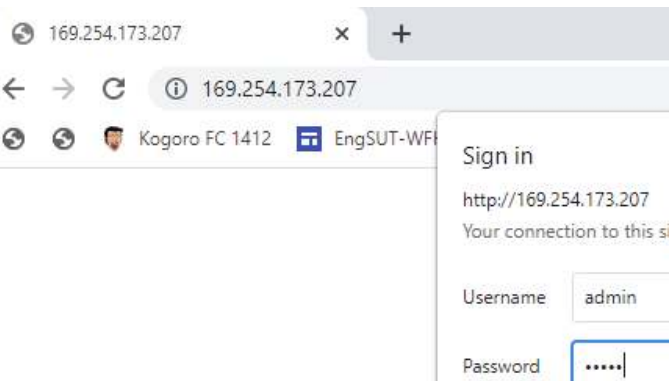
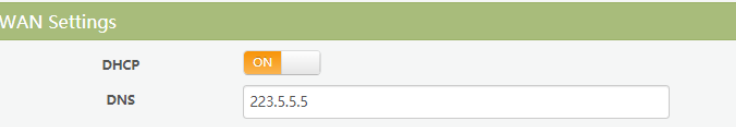
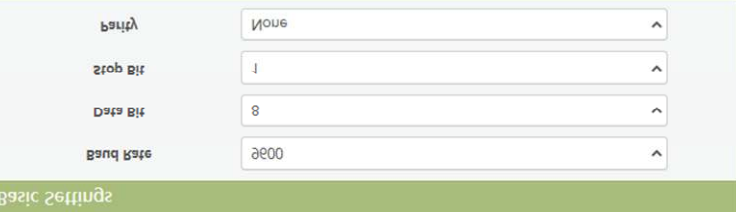
CE FCC RoHS



Overview of Characteristic

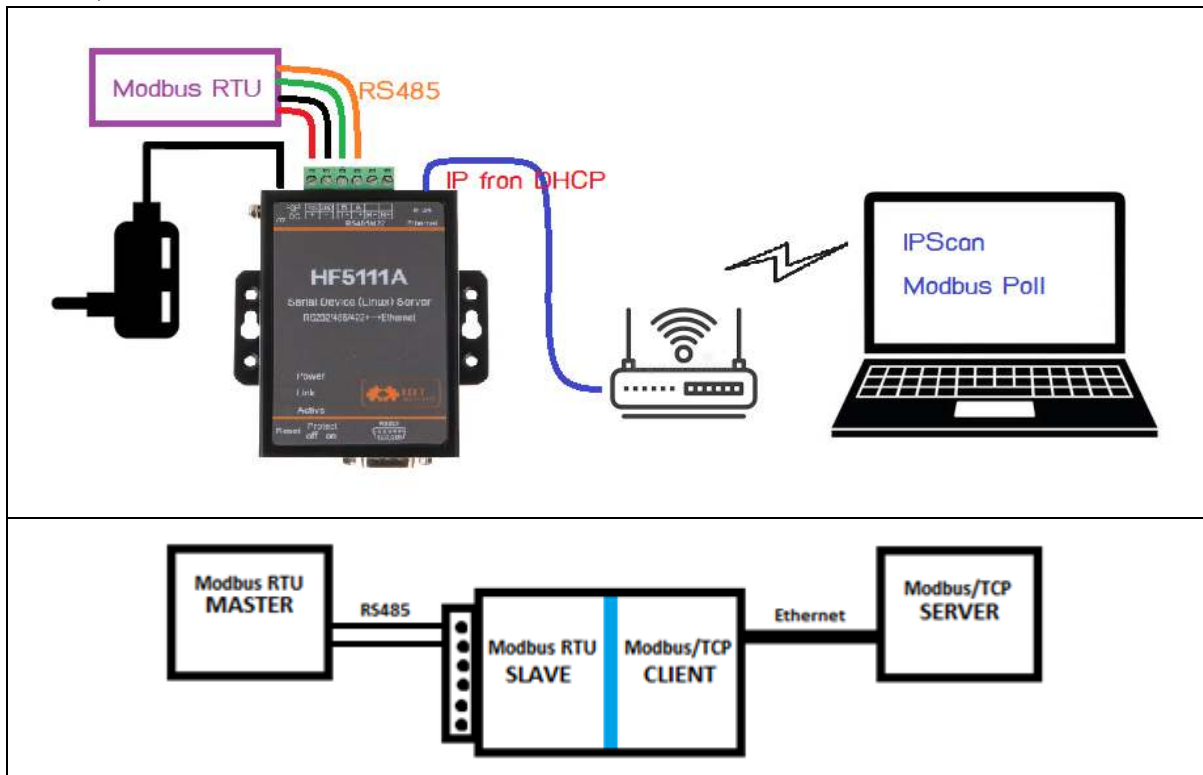
- MIPS MCU with 16MB Flash and 32MB SRAM
- Use Linux Operation System
- Support TCP/IP/Telnet/Modbus TCP Protocol
- Support Serial To 10/100M Ethernet Conversion, Serial Speed Up to 460800 bps
- Support 10/100M Ethernet Auto-Negotiation
- Support Easy Configuration Through a Web Interface
- Support Security Protocol Such as SSL/AES/DES3
- Support Web OTA Wireless Upgrade
- Wide DC Input 5~36VDC or 9~50VDC
- Size: 94 x65x 25 mm (L x W x H)
- <http://www.hi-flying.com/hf5111a>

2. ตั้งค่า HF5111B

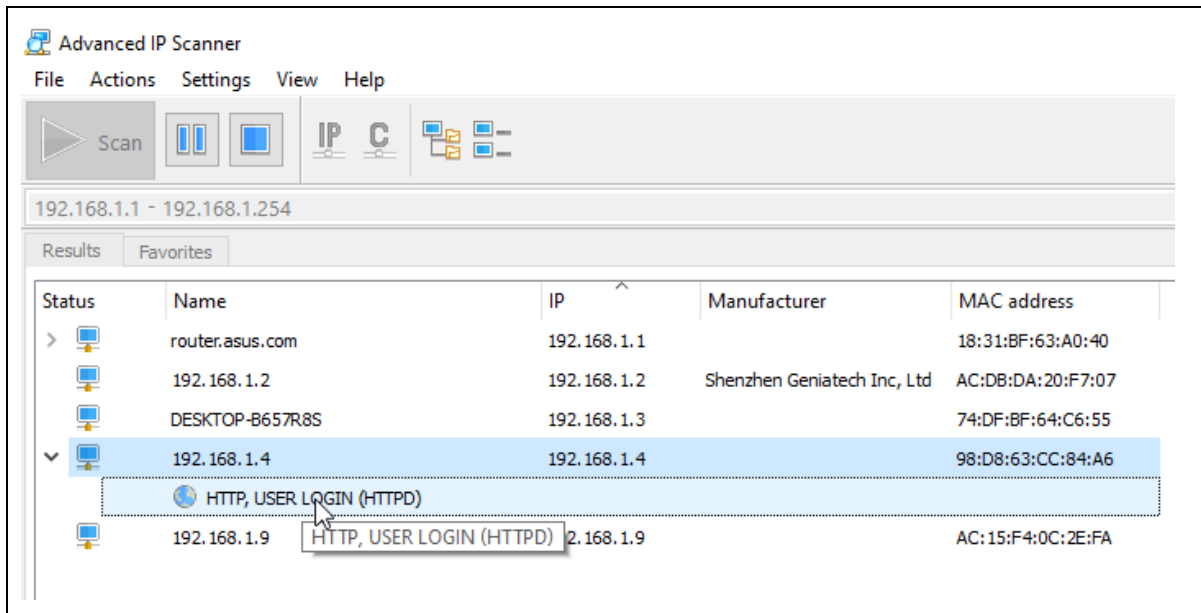
	<p>กำหนด PC IP ที่ต่อทดสอบเป็น</p> <p>169.254.217.xx</p> <p>255.255.0.0</p>
	<p>ต่อ HF5111B เข้ากับ PC โดยตรง</p>
	<p>เรียกอุปกรณ์ HF5111B ที่</p> <p>IP = 169.254.217.207</p> <p>Login = admin</p> <p>Pass = admin</p>
	<p>ตั้งค่าต่างๆ ตามรูป</p> <p>System</p>
	<p>ตั้งค่าต่างๆ ตามรูป</p> <p>Serial</p>

<p>Flow Control Settings</p> <p>Flow Control: Half Duplex</p> <p>Cli Settings</p> <p>Cli: Serial String</p> <p>Serial String: +++</p> <p>Waiting Time: 300</p> <p>Protocol Settings</p> <p>Protocol: Modbus</p>	<p>ตั้งค่าต่างๆ ตามรูป Serial</p>
<p>Basic Settings</p> <p>Name: netp</p> <p>Protocol: Tcp Server</p> <p>Socket Settings</p> <p>Local Port: 502</p> <p>Buffer Size: 512</p> <p>Keep Alive(s): 60</p> <p>Timeout(s): 0</p>	<p>ตั้งค่าต่างๆ ตามรูป Communication</p>

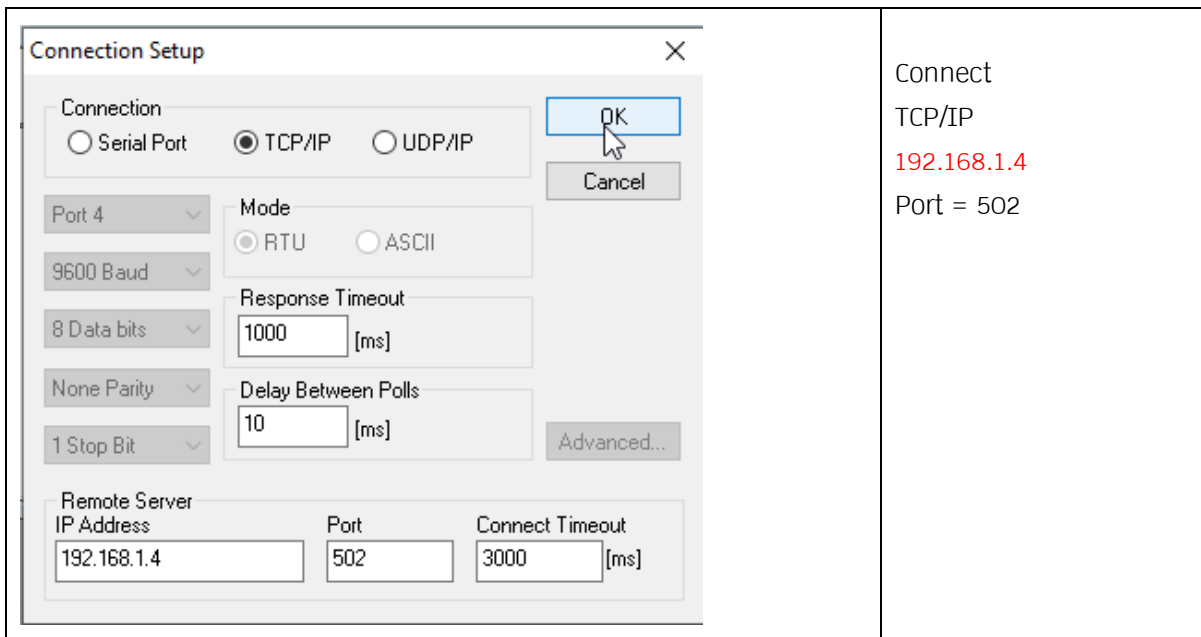
3. ต่ออุปกรณ์ Modbus RTU และ WAN



4. หา IP ด้วยโปรแกรม Advanced IP Scanner



5. ทดสอบด้วย Modbus Poll Software ตามขั้นตอน



Read/Write Definition

Slave ID: OK Cancel Apply

Function:

Address:

Quantity:

Scan Rate: ms

☒ Read/Write Enabled Read/Write Once

View

Rows ☒ 10 ☐ 20 ☐ 50 ☐ 100 ☐ Hide Alias Columns ☐ Address in Cell

Display: ☐ PLC Addresses (Base 1)

Read Holding
From ID = 1
Start 0
2 Word
Unsigned

Mbpol1

Tx = 7: Err = 0: ID = 1: F = 03: SR = 1000ms

	Alias	00000
0		283
1		533
2		
3		
4		
5		

Communication Traffic

Exit Stop Save

```

000000-Tx:00 7A 00 00 00 06 01 03 00 00 00 02
000001-Rx:00 7A 00 00 00 07 01 03 04 01 1C 02 15
000002-Tx:00 7B 00 00 00 06 01 03 00 00 00 02
000003-Rx:00 7B 00 00 00 07 01 03 04 01 1C 02 15
000004-Tx:00 7C 00 00 00 06 01 03 00 00 00 02
000005-Rx:00 7C 00 00 00 07 01 03 04 01 1C 02 15
000006-Tx:00 7D 00 00 00 06 01 03 00 00 00 02
000007-Rx:00 7D 00 00 00 07 01 03 04 01 1B 02 15

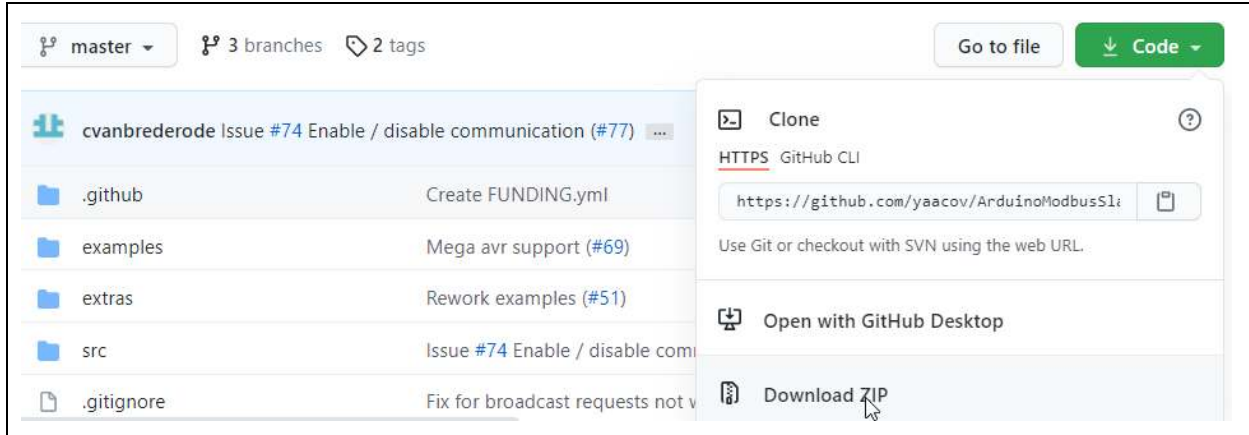
```

IP from DHCP

Device ID = 1

Test 5/8. Modbus TCP Device from ESP32 – Modbus TCP Server

1. Install Lib >> **“ArduinoModbusSlaveTCP-master.zip”**
 - From <https://github.com/yaacov/ArduinoModbusSlave> Download to zip
 - Sketch → Include Lib → Add Zip



2. Load Code

```
// https://github.com/yaacov/ArduinoModbusSlave

#include <WiFi.h>
#include <ModbusSlaveTCP.h>

const char* ssid = "Mue.Home";
const char* pass = "pk1212312121";

#define SLAVE_ID 3

ModbusTCP slave(SLAVE_ID);

void setup() {
  Serial.begin(115200);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  slave.cbVector[CB_WRITE_COIL] = writeDigitlOut;
  slave.cbVector[CB_READ_COILS] = readDigitalIn;
  slave.cbVector[CB_READ_REGISTERS] = readAnalogIn;

  slave.begin();

  Serial.println("");
  Serial.print("Modbus ready, listen on ");
  Serial.print(WiFi.localIP());
  Serial.println(" : 502");
}

void loop() {
  slave.poll();
}
```

```

/**
  Handel Force Single Coil (FC=05)
  set digital output pins (coils) on and off
*/
void writeDigitlOut(uint8_t fc, uint16_t address, uint16_t status) {
  pinMode(address, OUTPUT);
  digitalWrite(address, status);
  Serial.println("digitalWrite(" + String(address) + "," + String(status) + ")");
}

/**
  Handel Read Input Status (FC=02/01)
  write back the values from digital in pins (input status).

  handler functions must return void and take:
    uint8_t fc - function code
    uint16_t address - first register/coil address
    uint16_t length/status - length of data / coil status
*/
void readDigitalIn(uint8_t fc, uint16_t address, uint16_t length) {
  // read digital input
  for (int i = 0; i < length; i++) {
    pinMode(address + i, INPUT_PULLUP);
    int dValue = digitalRead(address + i);
    slave.writeCoilToBuffer(i, dValue);
    Serial.println("digitalRead(" + String(address + i) + ") = " + String(dValue));
  }
}

/**
  Handel Read Input Registers (FC=04/03)
  write back the values from analog in pins (input registers).
*/
void readAnalogIn(uint8_t fc, uint16_t address, uint16_t length) {
  // read analog input
  for (int i = 0; i < length; i++) {
    //int aValue = analogRead(address + i);
    int aValue = (address + i) * 1000 + random(111, 999);
    Serial.println("analogRead(" + String(address + i) + ") = " + String(aValue));
    slave.writeRegisterToBuffer(i, aValue);
  }
}

```

```

load:0x40080400,len:5856
entry 0x400806a8
Connecting to Mue.Home
.....
Modbus ready, listen on 192.168.1.5 : 502

```

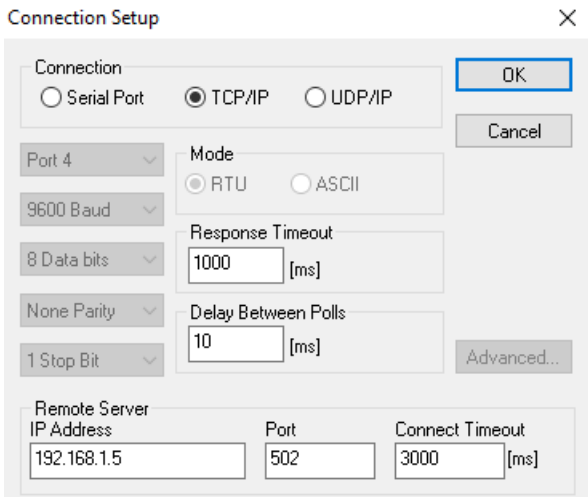
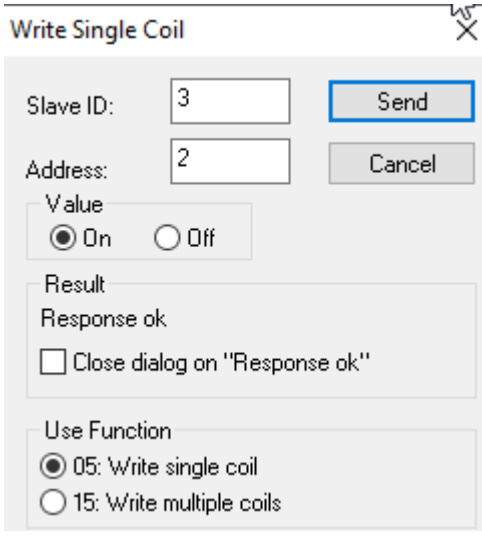
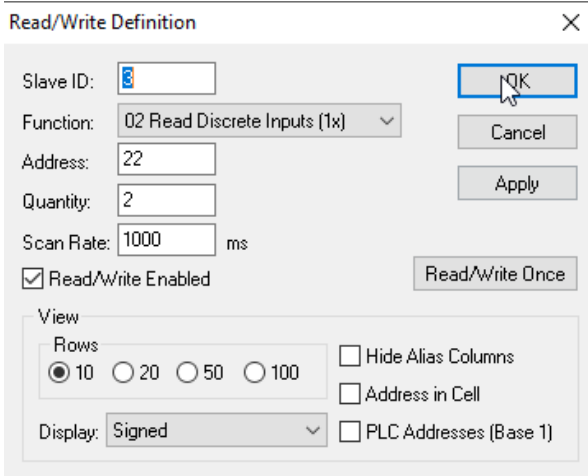
☒ Autoscroll ☐ Show timestamp

Carriage return ▾

115200 baud ▾

Clear output

3. ทดสอบด้วย Modbus Poll Software

 <p>Connection Setup</p> <p>Connection: <input type="radio"/> Serial Port <input checked="" type="radio"/> TCP/IP <input type="radio"/> UDP/IP</p> <p>Port 4 Mode: <input checked="" type="radio"/> RTU <input type="radio"/> ASCII</p> <p>9600 Baud</p> <p>8 Data bits</p> <p>None Parity</p> <p>1 Stop Bit</p> <p>Response Timeout: 1000 [ms]</p> <p>Delay Between Polls: 10 [ms]</p> <p>Advanced...</p> <p>Remote Server</p> <p>IP Address: 192.168.1.5 Port: 502 Connect Timeout: 3000 [ms]</p>	<p>Connection TCP/IP</p> <p>IP = 192.168.1.5</p> <p>Port = 502</p>																											
 <p>Write Single Coil</p> <p>Slave ID: 3</p> <p>Address: 2</p> <p>Value: <input checked="" type="radio"/> On <input type="radio"/> Off</p> <p>Result: Response ok</p> <p><input type="checkbox"/> Close dialog on "Response ok"</p> <p>Use Function: <input checked="" type="radio"/> 05: Write single coil <input type="radio"/> 15: Write multiple coils</p>	<p>Function = 5</p> <p>ID = 3</p> <p>Address = 2 GPIO=2</p> <p>On , Off</p> <p>Send</p>																											
 <p>Read/Write Definition</p> <p>Slave ID: 3</p> <p>Function: 02 Read Discrete Inputs (1x)</p> <p>Address: 22</p> <p>Quantity: 2</p> <p>Scan Rate: 1000 ms</p> <p><input checked="" type="checkbox"/> Read/Write Enabled</p> <p>View: Rows <input checked="" type="radio"/> 10 <input type="radio"/> 20 <input type="radio"/> 50 <input type="radio"/> 100</p> <p>Display: Signed</p> <p><input type="checkbox"/> Hide Alias Columns</p> <p><input type="checkbox"/> Address in Cell</p> <p><input type="checkbox"/> PLC Addresses (Base 1)</p>	<p>ID = 3</p> <p>Function 02</p> <p>Address = 22</p> <p>Number = 2 >> Read Pin 22,23</p> <p>Mbpoll1</p> <p>Tx = 210: Err = 45: ID = 3: F = 02: SR = 1000ms</p> <table border="1"> <thead> <tr> <th></th> <th>Alias</th> <th>00020</th> </tr> </thead> <tbody> <tr><td>0</td><td></td><td></td></tr> <tr><td>1</td><td></td><td></td></tr> <tr><td>2</td><td></td><td>1</td></tr> <tr><td>3</td><td></td><td>0</td></tr> <tr><td>4</td><td></td><td></td></tr> <tr><td>5</td><td></td><td></td></tr> <tr><td>6</td><td></td><td></td></tr> <tr><td>7</td><td></td><td></td></tr> </tbody> </table>		Alias	00020	0			1			2		1	3		0	4			5			6			7		
	Alias	00020																										
0																												
1																												
2		1																										
3		0																										
4																												
5																												
6																												
7																												

Read/Write Definition

Slave ID:

Function: 04 Read Input Registers (3x)

Address: 15

Quantity: 4

Scan Rate: 1000 ms

☒ Read/Write Enabled

View

Rows: ☒ 10 ☐ 20 ☐ 50 ☐ 100

Display: Signed

☐ Hide Alias Columns

☐ Address in Cell

☐ PLC Addresses (Base 1)

Buttons: OK, Cancel, Apply, Read/Write Once

ID = 3

Function = 04

Analog Pin = 15

nPin = 4 >> 15,16,17,18

Tx = 176: Err = 9: ID = 3: F = 04: SP = 1000ms

Alias	00010
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	15901
16	16409
17	17142
18	18784
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	
64	
65	
66	
67	
68	
69	
70	
71	
72	
73	
74	
75	
76	
77	
78	
79	
80	
81	
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	
96	
97	
98	
99	
100	

Communication Traffic

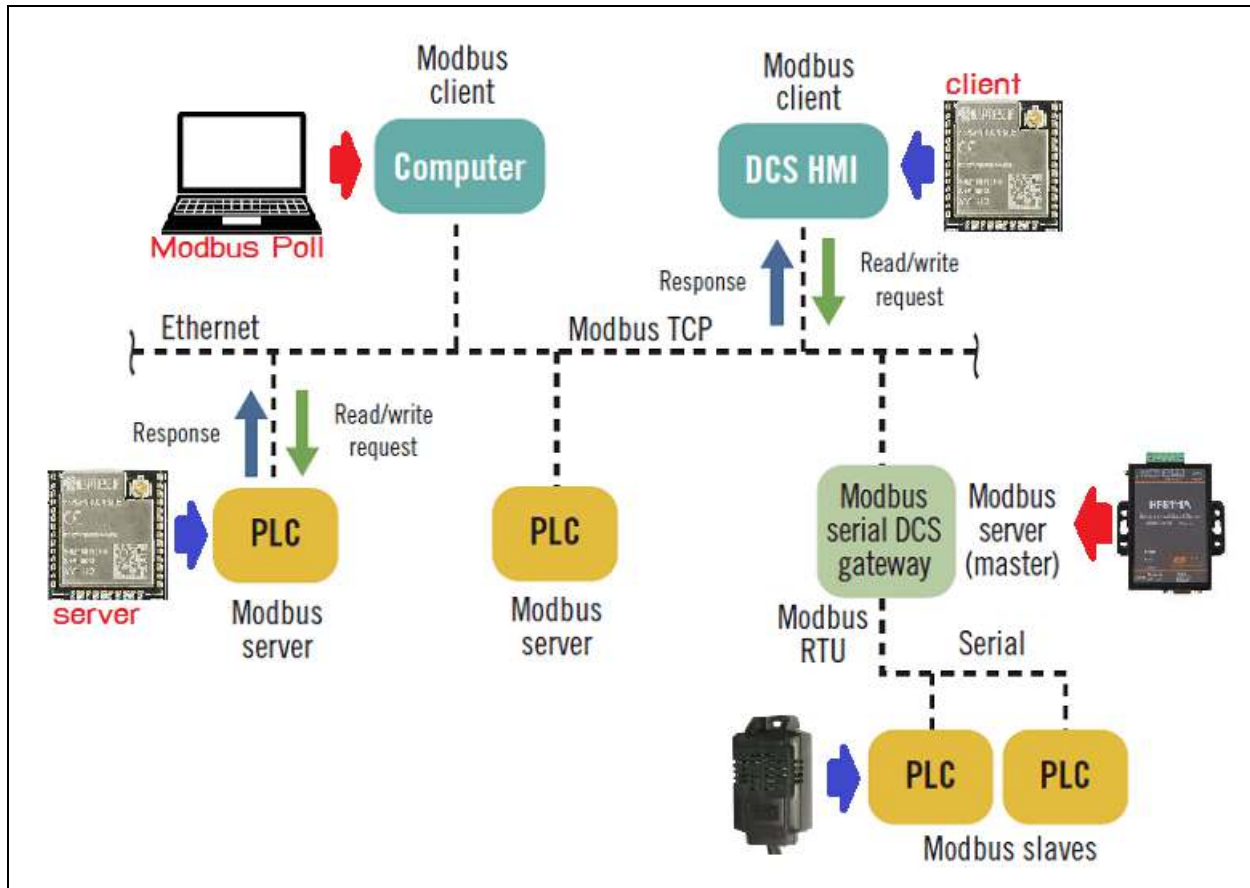
Exit Stop Save

```

001272-Tx:05 C1 00 00 00 06 03 04 00 0F 00 04
001273-Rx:05 C1 00 00 00 08 03 04 00 3C A1 3F 2F
001274-Tx:05 C2 00 00 00 06 03 04 00 0F 00 04
001275-Rx:05 C2 00 00 00 08 03 04 00 3B F4 41 ER
001276-Tx:05 C3 00 00 00 06 03 04 00 0F 00 04
001277-Rx:05 C3 00 00 00 08 03 04 00 3C C5 3F 9C
001278-Tx:05 C4 00 00 00 06 03 04 00 0F 00 04
001279-Rx:05 C4 00 00 00 08 03 04 00 3D 2F 40 27
001280-Tx:05 C5 00 00 00 06 03 04 00 0F 00 04
001281-Rx:05 C5 00 00 00 08 03 04 00 3E 0E 3F C9
001282-Tx:05 C6 00 00 00 06 03 04 00 0F 00 04
001283-Rx:05 C6 00 00 00 08 03 04 00 3D 08 3E P4
001284-Tx:05 C7 00 00 00 06 03 04 00 0F 00 04
001285-Rx:05 C7 00 00 00 08 03 04 00 3E 15 3F C4
001286-Tx:05 C8 00 00 00 06 03 04 00 0F 00 04
001287-Rx:05 C8 00 00 00 08 03 04 00 3C 5B 3F R3
001288-Tx:05 C9 00 00 00 06 03 04 00 0F 00 04
001289-Rx:05 C9 00 00 00 08 03 04 00 3D 1F 3F 34
001290-Tx:05 CA 00 00 00 06 03 04 00 0F 00 04
001291-Rx:05 CA 00 00 00 08 03 04 00 3E 00 41 D3
001292-Tx:05 CB 00 00 00 06 03 04 00 0F 00 04
001293-Rx:05 CB 00 00 00 08 03 04 00 3B 79 3E P2
001294-Tx:05 CC 00 00 00 06 03 04 00 0F 00 04
001295-Rx:05 CC 00 00 00 08 03 04 00 3E 48 40 91

```

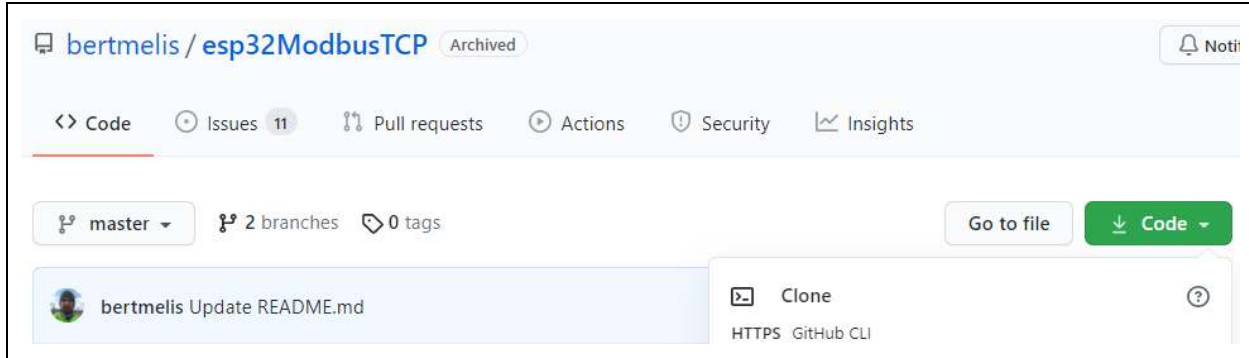
4. ระบบการสื่อสารด้วย Modbus TCP/RTU/ASCII



Test 6/8. Modbus TCP Read from ESP32 – Modbus TCP Slave

1. Install Lib

- From <https://github.com/bertmelis/esp32ModbusTCP> >> Code > Download to zip
- Sketch → Include Lib → Add Zip **esp32ModbusTCP-master.zip**



- From <https://github.com/me-no-dev/AsyncTCP> >> Code > Download to zip
- Sketch → Include Lib → Add Zip **AsyncTCP-master.zip**



2. Load Code

```
// esp32ModbusTCP >> https://github.com/bertmelis/esp32ModbusTCP
// AsyncTCP.h >> https://github.com/me-no-dev/AsyncTCP

#include <Arduino.h>
#include <WiFi.h>
#include <esp32ModbusTCP.h>

char ssid[] = "Mue.Home";
char pass[] = "pk1212312121";

bool WiFiConnected = false;

esp32ModbusTCP sunnyboy(1, {192, 168, 1, 4}, 502);
enum smaType {
  ENUM, // enumeration
  UFIX0, // unsigned 2 Byte, no decimals
  SFIX0, // signed 4 Byte, no decimals
};

struct smaData {
  const char* name;
  uint16_t address;
  uint16_t length;
  smaType type;
  uint16_t packetId;
};

smaData smaRegisters[] = {
  "Temp", 0, 1, UFIX0, 0,
  "Humid", 1, 1, UFIX0, 0
};

uint8_t numberSmaRegisters = sizeof(smaRegisters) / sizeof(smaRegisters[0]);
uint8_t currentSmaRegister = 0;
uint16_t ResultData[3];
```

```

void setup() {
  Serial.begin(115200);
  WiFi.disconnect(true); // delete old config

  sunnyboy.onData([](uint16_t packet, uint8_t slave, esp32Modbus::FunctionCode fc, uint8_t* data, uint16_t len) {
    for (uint8_t i = 0; i < numberSmaRegisters; ++i) {
      if (smaRegisters[i].packetId == packet) {
        smaRegisters[i].packetId = 0;
        switch (smaRegisters[i].type) {
          case ENUM:
          case UFIX0: {
            uint32_t value = 0;          // 2-Byte Data
            value = (data[0] << 8) | (data[1]); // 2-Byte Data
            Serial.printf("%s: %u\n", smaRegisters[i].name, value);
            ResultData[i] = value;
            break;
          }
          case SFIX0: {
            int32_t value = 0;
            value = (data[0] << 24) | (data[1] << 16) | (data[2] << 8) | (data[3]);
            Serial.printf("%s: %i\n", smaRegisters[i].name, value);
            break;
          }
        }
      }
    }
    return;
  });
}

sunnyboy.onError([](uint16_t packet, esp32Modbus::Error e) {
  Serial.printf("Error packet %u: %02x\n", packet, e);
});

delay(1000);

WiFi.onEvent([](WiFiEvent_t event, WiFiEventInfo_t info) {
  Serial.print("WiFi connected. IP: ");
  Serial.println(IPAddress(info.got_ip_info.ip.addr));
  WiFiConnected = true;
}, WiFiEvent_t::SYSTEM_EVENT_STA_GOT_IP);

WiFi.onEvent([](WiFiEvent_t event, WiFiEventInfo_t info) {
  Serial.print("WiFi lost connection. Reason: ");
  Serial.println(info.disconnected.reason);
  WiFi.disconnect();
  WiFiConnected = false;
}, WiFiEvent_t::SYSTEM_EVENT_STA_DISCONNECTED);

WiFi.begin(ssid, pass);
Serial.println();
Serial.println("Connecting to WiFi... ");
}

void loop() {
  static uint32_t lastMillis = 0;
  if ((millis() - lastMillis > 10000 && WiFiConnected)) {
    lastMillis = millis();
    Serial.print("reading registers\n");
    for (uint8_t i = 0; i < numberSmaRegisters; ++i) {
      uint16_t packetId = sunnyboy.readHoldingRegisters(smaRegisters[i].address, smaRegisters[i].length);
      if (packetId > 0) {
        smaRegisters[i].packetId = packetId;
      } else {
        Serial.print("reading error\n");
      }
    }
    Serial.println("Data_1 = " + String(ResultData[0]));
    Serial.println("Data_2 = " + String(ResultData[1]));
  }
}

```

```
Connecting to WiFi...  
WiFi connected. IP: 192.168.1.5  
reading registers  
Data_1 = 0  
Data_2 = 0  
Tempp: 282  
Humid: 596
```

☒ Autoscroll ☐ Show timestamp


Carriage return ▼

115200 baud ▼

4/5: -- การเชื่อมต่อนระหว่าง IoTs กับอุปกรณ์ Modbus RTU/ASCII/TCP

Test 7/8. Modbus RTU to Blynk

1. ทดสอบส่งค่าไปที่ Blynk ด้วยโปรแกรม

	<p>Switch = V0 Switch = V1</p> <p>Gauge = V10 Gauge = V11</p>
<pre>#define BLYNK_PRINT Serial #include <WiFi.h> #include <WiFiClient.h> #include <BlynkSimpleEsp32.h> char auth[] = "YD3FmnLEk5vdhs-BeQIWwrACl8gXNgXK"; char ssid[] = "Mue.Home"; char pass[] = "pk1212312121"; int Value_V0, Value_V1; BLYNK_WRITE(V0) { Value_V0 = param.asInt(); // Get value as integer } BLYNK_WRITE(V1) { Value_V1 = param.asInt(); // Get value as integer } void setup() { Serial.begin(115200); Blynk.begin(auth, ssid, pass); } void loop() { float Tempp = random(0, 1000) / 10.0; float Humid = random(0, 1000) / 10.0; Blynk.virtualWrite(V10, Tempp); Blynk.virtualWrite(V11, Humid); Serial.println("V0=" + String(Value_V0)); Serial.println("V1=" + String(Value_V1)); Serial.println("V10=" + String(Tempp, 1)); Serial.println("V11=" + String(Humid, 1)); Blynk.run(); delay(5000); }</pre>	

2. ทดสอบว่า Modbus RTU ทำงานได้

- ให้แน่ใจว่าติดตั้ง ModbusMaster V 2.0.1 ของ Doc Walker

ModbusMaster
 by Doc Walker 4-20ma@wvfans.net>
Enlighten your Arduino to be a Modbus master. Enables communication with Modbus slaves over RS232/485 (via RTU protocol).
 Requires an RS232/485 transceiver.
[More info](#)

Version 2.0.1 Install

```

#include <ModbusMaster.h>
#define RS485Transmit  HIGH
#define RS485Receive  LOW
#define RS485Control  4  //RS485 Direction control
#define Pin_LEDMonitor  2
#define Slave_Sensor_ID  1
#define Slave_Relay8_ID  3
#define Slave_Ry4In4_ID  5

int state = 0;
float CTemp, Hudmid;
bool DgInput0, DgInput1, DgInput2, DgInput3;

ModbusMaster node_Sensor;
ModbusMaster node_Relay8;
ModbusMaster node_Ry4In4;

void preTransmission() {
  digitalWrite(RS485Control, RS485Transmit);
}

void postTransmission() {
  digitalWrite(RS485Control, RS485Receive);
}

void setup() {
  pinMode(RS485Control, OUTPUT);
  pinMode(Pin_LEDMonitor, OUTPUT);
  Serial.begin(115200);
  Serial2.begin(9600);
  postTransmission();
  node_Sensor.begin(Slave_Sensor_ID, Serial2); // Modbus slave ID=1
  node_Sensor.preTransmission(preTransmission);
  node_Sensor.postTransmission(postTransmission);
  node_Relay8.begin(Slave_Relay8_ID, Serial2); // Modbus slave ID=3
  node_Relay8.preTransmission(preTransmission);
  node_Relay8.postTransmission(postTransmission);
  node_Ry4In4.begin(Slave_Ry4In4_ID, Serial2); // Modbus slave ID=5
  node_Ry4In4.preTransmission(preTransmission);
  node_Ry4In4.postTransmission(postTransmission);
}

void ReadTemperature(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Sensor.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 2 registers starting at 0x0000
  result = node_Sensor.readInputRegisters(0x0000, 2); // From=0, nByte=2
  if (result == node_Sensor.ku8MBSuccess) {
    CTemp = node_Sensor.getResponseBuffer(0x00) / 10.0f;
    Hudmid = node_Sensor.getResponseBuffer(0x01) / 10.0f;
  }
}

```



```

}

void ReadDigitalInput(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Ry4In4.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 4 registers starting at 0x0000
  result = node_Ry4In4.readDiscretInputs(0, 4); // Start=0, nByte=4
  if (result == node_Ry4In4.ku8MBSuccess) {
    int DgTemp = node_Ry4In4.getResponseBuffer(0x00);
    DgInput3 = (DgTemp >> 3) & 1;
    DgInput2 = (DgTemp >> 2) & 1;
    DgInput1 = (DgTemp >> 1) & 1;
    DgInput0 = (DgTemp >> 0) & 1;
  }
}

void RelayControl(int inputCase) {
  int rnMode = inputCase / 10;
  int nRelay = inputCase % 10;
  if (rnMode == 81) node_Relay8.writeSingleRegister(nRelay, 0x0100); // On RelayX
  if (rnMode == 80) node_Relay8.writeSingleRegister(nRelay, 0x0200); // Off RelayX
  if (rnMode == 41) node_Ry4In4.writeSingleRegister(nRelay, 0x0100); // On RelayX
  if (rnMode == 40) node_Ry4In4.writeSingleRegister(nRelay, 0x0000); // Off RelayX
}

void loop() {
  ReadTemperature();
  ReadDigitalInput();
  Serial.print("\n Tempp('C): "); Serial.print(CTempp, 2);
  Serial.print(", Humid(%): "); Serial.print(Hudmid, 2);
  Serial.print(", Sensor[0:3]: "); Serial.print(DgInput3);
  Serial.print("-"); Serial.print(DgInput2);
  Serial.print("-"); Serial.print(DgInput1);
  Serial.print("-"); Serial.print(DgInput0);
  if (Serial.available() > 0) {
    int DataInput = Serial.parseInt();
    Serial.print("\n >> XYZ > X={8,4}Board Name, Y={1,0}On,Off, Z={0-8}RlyID >> ");
    Serial.println(DataInput);
    int Chk = DataInput / 100;
    if ((Chk == 8) || (Chk == 4))
      RelayControl(DataInput); }
  delay(2000);
}

```

```

Tempp('C): 28.70, Humid(%): 57.90, Sensor[0:3]: 1-1-0-0
Tempp('C): 28.70, Humid(%): 58.00, Sensor[0:3]: 1-1-0-0
Tempp('C): 28.70, Humid(%): 58.00, Sensor[0:3]: 1-1-0-0
Tempp('C): 28.70, Humid(%): 58.00, Sensor[0:3]: 1-1-0-0
>> XYZ > X={8,4}Board Name, Y={1,0}On,Off, Z={0-8}RlyID >> 812

```

☒ Autoscroll ☐ Show timestamp

No line ending

115200 baud

Clear output

3. ปรับโปรแกรมและทดสอบ

```

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

#include <ModbusMaster.h>
#define RS485Transmit HIGH
#define RS485Receive LOW
#define RS485Control 4 //RS485 Direction control
#define Pin_LEDMonitor 2
#define Slave_Sensor_ID 1
#define Slave_Relay8_ID 3
#define Slave_Ry4In4_ID 5

char auth[] = "YD3FmnLEk5vdhs-BeQlWwrACl8gXNgXK";
char ssid[] = "Mue.Home";
char pass[] = "pk1212312121";
int Value_V0, Value_V1;

int state = 0;
float CTempp, Hudmid;
bool DgInput0, DgInput1, DgInput2, DgInput3;

ModbusMaster node_Sensor;
ModbusMaster node_Relay8;
ModbusMaster node_Ry4In4;

BLYNK_WRITE(V0) {
  int temp = param.asInt();
  if (temp != Value_V0) {
    Value_V0 = temp;
    RelayControl(801 + temp * 10);
  }
}

BLYNK_WRITE(V1) {
  int temp = param.asInt();
  if (temp != Value_V1) {
    Value_V1 = temp;
    RelayControl(802 + temp * 10);
  }
}

void preTransmission() {
  digitalWrite(RS485Control, RS485Transmit);
}

void postTransmission() {
  digitalWrite(RS485Control, RS485Receive);
}

void setup() {
  pinMode(RS485Control, OUTPUT);
  pinMode(Pin_LEDMonitor, OUTPUT);
  Serial.begin(115200);
  Serial2.begin(9600);
  postTransmission();
  node_Sensor.begin(Slave_Sensor_ID, Serial2); // Modbus slave ID=1
  node_Sensor.preTransmission(preTransmission);
  node_Sensor.postTransmission(postTransmission);
  node_Relay8.begin(Slave_Relay8_ID, Serial2); // Modbus slave ID=3
  node_Relay8.preTransmission(preTransmission);
  node_Relay8.postTransmission(postTransmission);
}

```

```

node_Ry4In4.begin(Slave_Ry4In4_ID, Serial2); // Modbus slave ID=5
node_Ry4In4.preTransmission(preTransmission);
node_Ry4In4.postTransmission(postTransmission);
Blynk.begin(auth, ssid, pass);
}

void ReadTemperature(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Sensor.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 2 registers starting at 0x0000
  result = node_Sensor.readInputRegisters(0x0000, 2); // From=0, nByte=2
  if (result == node_Sensor.ku8MBSuccess) {
    CTempp = node_Sensor.getResponseBuffer(0x00) / 10.0f;
    Hudmid = node_Sensor.getResponseBuffer(0x01) / 10.0f;
  }
}

void ReadDigitalInput(void) {
  uint8_t result;
  // Toggle the coil at address (Manual Load Control)
  result = node_Ry4In4.writeSingleCoil(Slave_Sensor_ID, state);
  state = !state;
  // Read 4 registers starting at 0x0000
  result = node_Ry4In4.readDiscreteInputs(0, 4); // Start=0, nByte=4
  if (result == node_Ry4In4.ku8MBSuccess) {
    int DgTemp = node_Ry4In4.getResponseBuffer(0x00);
    DgInput3 = (DgTemp >> 3) & 1;
    DgInput2 = (DgTemp >> 2) & 1;
    DgInput1 = (DgTemp >> 1) & 1;
    DgInput0 = (DgTemp >> 0) & 1;
  }
}

void RelayControl(int inputCase) {
  int rnMode = inputCase / 10;
  int nRelay = inputCase % 10;
  if (rnMode == 81) node_Relay8.writeSingleRegister(nRelay, 0x0100); // On RelayX
  if (rnMode == 80) node_Relay8.writeSingleRegister(nRelay, 0x0200); // Off RelayX
  if (rnMode == 41) node_Ry4In4.writeSingleRegister(nRelay, 0x0100); // On RelayX
  if (rnMode == 40) node_Ry4In4.writeSingleRegister(nRelay, 0x0000); // Off RelayX
}

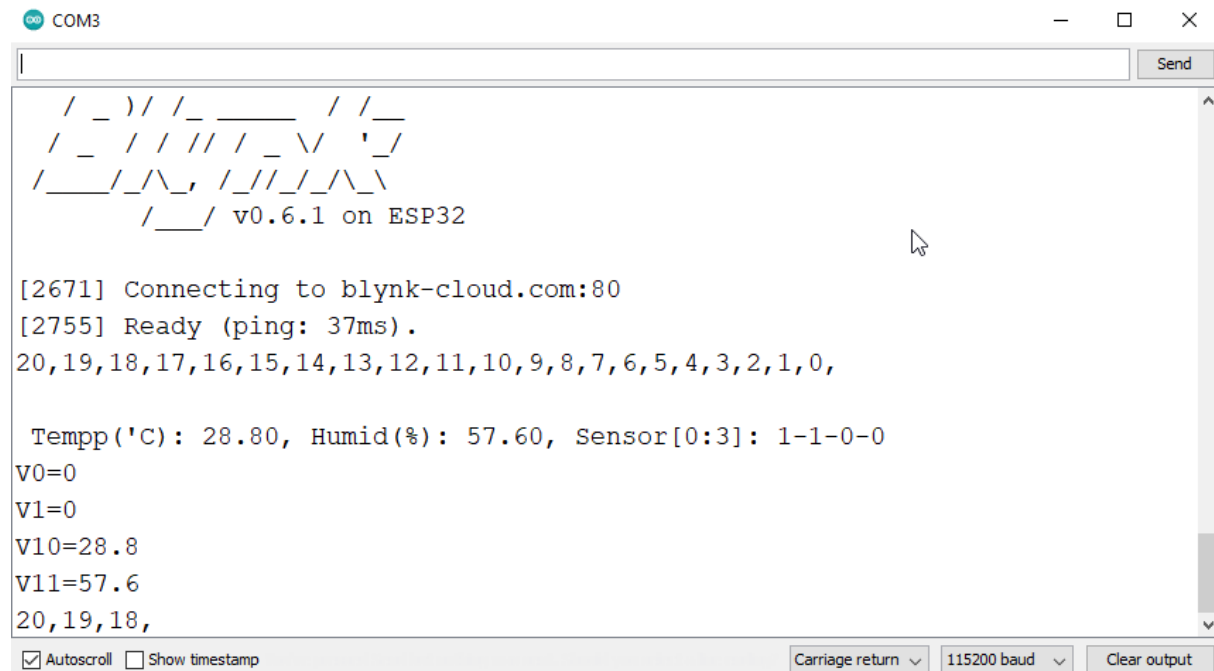
int loopCont = 20;
void loop() {
  if (loopCont < 0) {
    loopCont = 20;
    Serial.println();
    ReadTemperature();
    ReadDigitalInput();
    Serial.print("\n Tempp('C): "); Serial.print(CTempp, 2);
    Serial.print(", Humid(%): "); Serial.print(Hudmid, 2);
    Serial.print(", Sensor[0:3]: "); Serial.print(DgInput3);
    Serial.print("-"); Serial.print(DgInput2);
    Serial.print("-"); Serial.print(DgInput1);
    Serial.print("-"); Serial.print(DgInput0);
    Serial.println();
    if (Serial.available() > 0) {
      int DataInput = Serial.parseInt();
      Serial.print(" >> XYZ > X={8,4}Board Name, Y={1,0}On,Off, Z={0-8}RlyID >> ");
      Serial.println(DataInput);
      int Chk = DataInput / 100;
      if ((Chk == 8) || (Chk == 4))
        RelayControl(DataInput);
    }
  }
}

```

```

Blynk.virtualWrite(V10, CTempp);
Blynk.virtualWrite(V11, Hudmid);
Serial.println("V0=" + String(Value_V0));
Serial.println("V1=" + String(Value_V1));
Serial.println("V10=" + String(CTempp, 1));
Serial.println("V11=" + String(Hudmid, 1));
}
Blynk.run();
Serial.print(String(loopCont-- + ",");
delay(500);
}

```



```

/ _ ) / / _ _ _ _ / / _
/ _ / / / / / _ \ / ' _/
/_ _/_/_\_, /_/_/_/_\
      /_/_/ v0.6.1 on ESP32

[2671] Connecting to blynk-cloud.com:80
[2755] Ready (ping: 37ms).
20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,

  Tempp('C): 28.80, Humid(%): 57.60, Sensor[0:3]: 1-1-0-0
V0=0
V1=0
V10=28.8
V11=57.6
20,19,18,

```

COM3

Send

☒ Autoscroll ☐ Show timestamp

Carriage return 115200 baud Clear output

Test 8/8. Modbus TCP to Blynk

<https://medium.com/mmp-li/ประกอบร่าง-blynk-กับ-node-red-54f67433805b>

1. ปรับโปรแกรมและทดสอบ

```
// esp32ModbusTCP >> https://github.com/bertmelis/esp32ModbusTCP
// AsyncTCP.h >> https://github.com/me-no-dev/AsyncTCP

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Arduino.h>
#include <esp32ModbusTCP.h>

char ssid[] = "Mue.Home";
char pass[] = "pk1212312121";
char auth[] = "YD3FmnLEk5vdhs-BeQIWwrACl8gXNgXK";

bool WiFiConnected = false;
int Value_V0, Value_V1;

esp32ModbusTCP sunnyboy(1, {192, 168, 1, 4}, 502);
enum smaType {
  ENUM, // enumeration
  UFIX0, // unsigned 2 Byte, no decimals
  SFIX0, // signed 4 Byte, no decimals
};

struct smaData {
  const char* name;
  uint16_t address;
  uint16_t length;
  smaType type;
  uint16_t packetId;
};

smaData smaRegisters[] = {
  "Tempp", 0, 1, UFIX0, 0,
  "Humid", 1, 1, UFIX0, 0
};

uint8_t numberSmaRegisters = sizeof(smaRegisters) / sizeof(smaRegisters[0]);
uint8_t currentSmaRegister = 0;
uint16_t ResultData[3];

BLYNK_WRITE(V0) {
  int temp = param.asInt();
  if (temp != Value_V0) {
    Value_V0 = temp;
    RelayControl(801 + temp * 10);
  }
}

BLYNK_WRITE(V1) {
  int temp = param.asInt();
  if (temp != Value_V1) {
    Value_V1 = temp;
    RelayControl(802 + temp * 10);
  }
}
```



```

void RelayControl(int Code) {
  Serial.println("Code is " + String(Code));
}

void setup() {
  Serial.begin(115200);
  WiFi.disconnect(true); // delete old config

  sunnyboy.onData([](uint16_t packet, uint8_t slave, esp32Modbus::FunctionCode fc, uint8_t* data, uint16_t len) {
    for (uint8_t i = 0; i < numberSmaRegisters; ++i) {
      if (smaRegisters[i].packetId == packet) {
        smaRegisters[i].packetId = 0;
        switch (smaRegisters[i].type) {
          case ENUM:
          case UFIX0: {
            uint32_t value = 0;          // 2-Byte Data
            value = (data[0] << 8) | (data[1]); // 2-Byte Data
            Serial.printf("%s: %u\n", smaRegisters[i].name, value);
            ResultData[i] = value;
            break;
          }
          case SFIX0: {
            int32_t value = 0;
            value = (data[0] << 24) | (data[1] << 16) | (data[2] << 8) | (data[3]);
            Serial.printf("%s: %i\n", smaRegisters[i].name, value);
            break;
          }
        }
      }
    }
    return;
  });

  sunnyboy.onError([](uint16_t packet, esp32Modbus::Error e) {
    Serial.printf("Error packet %u: %02x\n", packet, e);
  });

  delay(1000);

  WiFi.onEvent([](WiFiEvent_t event, WiFiEventInfo_t info) {
    Serial.print("WiFi connected. IP: ");
    Serial.println(IPAddress(info.got_ip_info.ip_addr));
    WiFiConnected = true;
  }, WiFiEvent_t::SYSTEM_EVENT_STA_GOT_IP);

  WiFi.onEvent([](WiFiEvent_t event, WiFiEventInfo_t info) {
    Serial.print("WiFi lost connection. Reason: ");
    Serial.println(info.disconnected.reason);
    WiFi.disconnect();
    WiFiConnected = false;
  }, WiFiEvent_t::SYSTEM_EVENT_STA_DISCONNECTED);

  WiFi.begin(ssid, pass);
  Serial.println();
  Serial.println("Connecting to WiFi... ");
}

int loopCont = 20;
void loop() {
  if (loopCont < 0 && WiFiConnected) {
    loopCont = 20;
    Serial.print("\nreading registers\n");
    for (uint8_t i = 0; i < numberSmaRegisters; ++i) {
      uint16_t packetId = sunnyboy.readHoldingRegisters(smaRegisters[i].address, smaRegisters[i].length);
      if (packetId > 0) {
        smaRegisters[i].packetId = packetId;
      }
    }
  }
}

```

```
} else {  
    Serial.print("reading error\n");  
}  
}  
delay(5000);  
  
//Blynk.config(auth);  
float CTempp = ResultData[0] / 10.0;  
float Hudmid = ResultData[1] / 10.0;  
Blynk.virtualWrite(V10, CTempp);  
Blynk.virtualWrite(V11, Hudmid);  
Serial.println("V0=" + String(Value_V0));  
Serial.println("V1=" + String(Value_V1));  
Serial.println("V10=" + String(CTempp, 1));  
Serial.println("V11=" + String(Hudmid, 1));  
}  
Serial.print(String(loopCont--) + ",");  
//Blynk.run();  
delay(500);  
}
```

```
Connecting to WiFi...  
20,19,18,17,16,WiFi connected. IP: 192.168.1.5  
15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,  
reading registers  
Tempp: 276  
Humid: 598  
V0=0  
V1=0  
V10=27.6  
V11=59.8  
20,19,18,17,16,15,14,13,12,11,
```

☒ Autoscroll ☐ Show timestamp

Carriage return ▾

115200 baud ▾

Clear output

การควบคุมเครื่องจักรอัจฉริยะโดยใช้การสื่อสารระหว่างเครื่องจักรกับเครื่องจักร
M2M - Intelligence Machine Control

ชื่อ-สกุล :

5/5: -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_301 – Start SCADA

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
รายละเอียดการทดสอบ
< โปรแกรมทดสอบ >
< ผลการทดสอบ >

Quiz_302 – Modbus TCP Read/Write

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
รายละเอียดการทดสอบ
< โปรแกรมทดสอบ >
< ผลการทดสอบ >

Quiz_303 – Modbus RTU/ASCII/TCP with IoTs

< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
< รูปอุปกรณ์ที่ใช้ทดสอบ ขณะทำการทดสอบ >
รายละเอียดการทดสอบ
< โปรแกรมทดสอบ >
< ผลการทดสอบ >