NECTEC
a member of NSTDA

เอกสารเผยแพร่

การสื่อสารในงานอุตสาหกรรม ด้วยโพรโทคอล MODBUS

บทความ | ดร.ธีรเชษฐ์ สูรพันธุ์ และ ณฐพล ตันสังวร

ทีมระบบไซเบอร์-กายภาพ (CPS)
หน่วยทรัพยากรด้านการคำนวณและไซเบอร์-กายภาพ (NCCPI)

การสื่อสารในงานอุตสาหกรรมด้วยโพรโทคอล Modbus

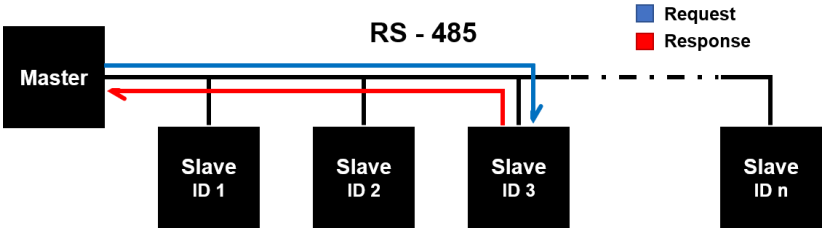
ดร.ธีรเชษฐ์ สูรพันธุ์ และ ณฐพล ตันสังวร
หน่วยทรัพยากรด้านการคำนวณและไซเบอร์-กายภาพ (NCCPI)
ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (เนคเทค)

1. บทนำ

การสื่อสารตามมาตรฐาน Modbus เป็นหนึ่งในมาตรฐานการสื่อสารแบบอนุกรม (Serial Communications protocol) ที่ใช้งานอย่างแพร่หลายในระบบอัตโนมัติอุตสาหกรรม (Industrial Automation Systems : IAS) เพื่อสร้างการเชื่อมโยงข้อมูลระหว่างอุปกรณ์ต่างๆ เช่น อุปกรณ์ควบคุมพีแอลซี (Programmable Logic Controllers : PLC) อุปกรณ์ตรวจวัด (Sensor) อุปกรณ์เครื่องกล อุปกรณ์ขับเร้า (Actuator) หน่วยตรวจวัดระยะไกล (Remote Terminal Unit : RTU) รวมถึงระบบคอมพิวเตอร์ที่ใช้ในการควบคุมและแสดงสถานะของอุปกรณ์ต่างๆ (Supervisory control and Data acquisition : SCADA)

Modbus ถูกพัฒนาขึ้นในปีค.ศ. 1979 โดยบริษัท Modicon (ปัจจุบันคือ Schneider Electric) เป็นโพรโทคอลที่ถูกใช้กันอย่างกว้างขวางในงานอุตสาหกรรมเนื่องจากความง่ายในการใช้งานและมีความน่าเชื่อถือ ในปัจจุบันนี้การสื่อสารสามารถแบ่งได้เป็น 2 ระบบคือ Modbus RTU และ Modbus TCP โดยความแตกต่างอยู่ที่โพรโทคอลการสื่อสารที่ใช้ ในระบบ Modbus RTU จะใช้โพรโทคอลการสื่อสารแบบอนุกรม (Serial-based Protocol) ในขณะที่ระบบ Modbus TCP จะใช้โพรโทคอลการสื่อสารแบบอีเทอร์เน็ต (Ethernet-based Protocol) ซึ่งทั้งสองแบบจะแตกต่างกันตรงที่ความเร็วและระยะทางในการรับส่งข้อมูล โดย Modbus RTU สามารถรับส่งได้ระยะทางสูงสุดถึง 1.2 กิโลเมตร (ที่ความเร็ว 57.6 kbps) ในขณะที่ Modbus TCP สามารถรับส่งได้ที่ความเร็ว สูงสุดถึง 100 Mbps (ที่ระยะทาง 100 เมตร) โดยรายละเอียดมีดังต่อไปนี้

2. Modbus RTU



รูปที่ 1 การสื่อสารแบบอนุกรมด้วย RS – 485 สำหรับ Modbus RTU

Modbus RTU คือ โพรโทคอลที่ใช้การสื่อสารแบบอนุกรม (Serial-based Protocol) ด้วยสถาปัตยกรรมการสื่อสารแบบ Master/Slave หรืออาจกล่าวได้ว่าอุปกรณ์ Slave จะไม่ส่งข้อมูล (Response) กลับมาจนกว่าจะมีการร้องขอ (Request) จากอุปกรณ์ Master ดังรูปที่ 1 Modbus RTU โดยทั่วไปจะใช้การ

สื่อสารในระดับกายภาพ (Physical Layer) แบบ RS-232 หรือ RS-485 ข้อมูลในโพรโทคอล Modbus จะถูกเก็บ 4 รูปแบบ คือ Output coils, Input contacts, Input registers และ Holding registers

โดย 2 แบบแรก Output coils และ Input contacts แต่ละแอดเดรสจะเก็บค่าเพียง 1 บิต หรือมีค่าได้แค่ “0” กับ “1” เปรียบเสมือนค่าการเปิดและปิดของอุปกรณ์รีเลย์และสวิตช์ที่พบได้ในระบบงานอัตโนมัติอุตสาหกรรม

ในขณะที่ 2 แบบหลัง Input registers และ Holding registers สามารถเก็บค่าเป็นตัวเลขได้ถึง 16 บิต เปรียบเสมือนค่าที่มาจากอุปกรณ์ตรวจวัดที่ส่งข้อมูลแบบอนาล็อก (Analog)

การสื่อสารของข้อมูลในระบบ Modbus RTU จะรับส่งเป็นชุดข้อมูล โดยที่ใน 1 ชุดข้อมูลนั้นจะประกอบด้วยส่วน 6 ส่วน ดังแสดงในรูปที่ 2 ซึ่งเริ่มต้นด้วยชุดบิตเริ่มต้น (Start bits) อ้างอิงถึงการเริ่มต้นชุดข้อมูล ตามด้วยค่าตำแหน่งแอดเดรส (Address) ของอุปกรณ์ที่ต้องการสื่อสารด้วย ตามด้วยชุดสำหรับ Function Code และข้อมูลที่ต้องการ (Data) ต่อด้วยชุดข้อมูลตรวจสอบความผิดพลาด (Cyclic Redundancy Check : CRC) และชุดบิตปิดท้าย (End bits) อ้างอิงถึงการสิ้นสุดข้อมูล

Field Name	Bit length	Function
Start	28	At least 3.5 character times of silence (mark condition)
Address	8	Station address
Function	8	Indicates function code eg. read coils/holding registers
Data	n x 8	Data + length will be filled depending on message type
CRC	16	Cyclic Redundancy Check
End	28	At least 3.5 character times of silence between frames

รูปที่ 2 ชุดข้อมูลสำหรับการสื่อสาร Modbus RTU [Ref. 1]

2.1. ฟังก์ชันการทำงานสำหรับ Modbus RTU (Function code)

ชุดฟังก์ชันการทำงานสามารถแบ่งหน้าที่ต่างๆ ได้ตามรหัส หรือ Function code รายละเอียดแสดงดังรูปที่ 3 โดยหลักๆ แล้วจะมีฟังก์ชันการทำงานอยู่ 2 แบบ คือ การอ่าน (Read) และเขียน (Write) โดยสามารถเลือกที่จะอ่านหรือเขียนข้อมูลไปยัง Coils หรือ Contacts สำหรับข้อมูลแบบดิจิตอล (Digital) หรือ “0” กับ “1” และ Registers สำหรับอ่านหรือเขียนข้อมูลแบบแอนะล็อก โดยมีขนาด 16 บิต หรือ ตั้งแต่ 0000 ถึง FFFF

Function Code (DEC)	Action	Data Type	Object Type
01	Read	Single bit	Output Coils
05	Write Single	Single bit	Output Coils
15	Write Multiple	Single bit	Output Coils
02	Read	Single bit	Input Contacts
04	Read	Word (16bit)	Input Registers
03	Read	Word (16bit)	Holding Registers
06	Write Single	Word (16bit)	Holding Registers
16	Write Multiple	Word (16bit)	Holding Registers

รูปที่ 3 รายละเอียดชุดข้อมูล Function Code [Ref. 2]

2.2. ตำแหน่งแอดเดรสของ Modbus RTU (Address)

ตำแหน่งแอดเดรสใน Modbus RTU จะมีขนาด 16 บิต หรือ 65535 ตำแหน่ง ในแต่ละรูปแบบการทำงาน โดยที่ Output coils (ตำแหน่งแอดเดรสจะเริ่มต้นที่ 000001) Input contacts (ตำแหน่งแอดเดรสจะเริ่มต้นที่ 100001) Input registers (ตำแหน่งแอดเดรสจะเริ่มต้นที่ 300001) และ Holding registers (ตำแหน่งแอดเดรสจะเริ่มต้นที่ 400001) ดังรูปที่ 4

หมายเหตุ สำหรับอุปกรณ์รุ่นเก่าอาจจะมีได้เพียง 9999 ตำแหน่งในแต่ละช่วง

Register Number (DEC)	Register Address (HEX)	Extended Register Number (DEC)	Extended Register Address (HEX)	Type	Object Type
00001-09999	0000 to 270E	000001-065535	0000 to FFFF	Read-Write	Output Coils
10001-19999	0000 to 270E	100001-165535	0000 to FFFF	Read-Only	Input Contacts
30001-39999	0000 to 270E	300001-365535	0000 to FFFF	Read-Only	Input Registers
40001-49999	0000 to 270E	400001-465535	0000 to FFFF	Read-Write	Holding Registers

รูปที่ 4 ตำแหน่งแอดเดรสใน Modbus RTU โดยแบ่งตามรูปแบบการทำงาน

2.3. ชุดข้อมูล (Data)

ในส่วนชุดข้อมูล Data Field นั้นจะถูกแบ่งเป็น 2 ชุด คือ ชุดคำสั่งสำหรับการอ่าน (Read Command) ตามรูปที่ 5 และชุดคำสั่งสำหรับการเขียน (Write Command) ตามรูปที่ 6 โดยชุดคำสั่งทั้ง 2 จะถูกส่งจากอุปกรณ์ที่ทำหน้าที่เป็น Master เท่านั้น เพื่อสั่งการไปยังอุปกรณ์ Slave ที่ต้องการสื่อสาร

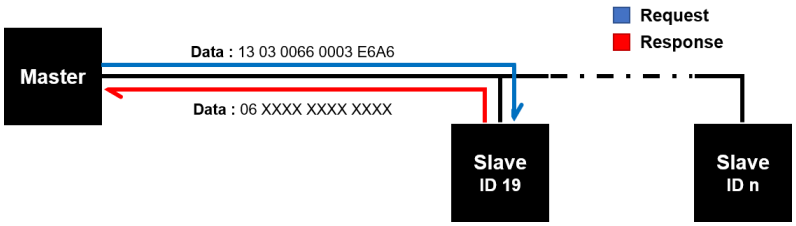
Read Command	
Request Message	start register address (2 bytes) + no. of registers (2 bytes)
Response Message	byte count (1 byte) + data (no. of registers * 2 bytes)

รูปที่ 5 ชุดคำสั่งสำหรับการอ่าน (Read Command)

Write Command	
Request Message	start register address (2 bytes) + no. of registers (2 bytes) + byte count (1 byte) + data (no. of registers * 2 bytes)
Response Message	start register address (2 bytes) + no. of registers (2 bytes)

รูปที่ 6 ชุดคำสั่งสำหรับการเขียน (Write Command)

ตัวอย่าง การอ่านค่าของ Holding register ที่แอดเดรส 40103 ถึง 40105 จากอุปกรณ์ Slave หมายเลข 19 ดังนั้น Frame Message (ไม่รวม Start และ End bits) ที่ถูกส่งไป คือ **13 03 0066 0003 E6A6** โดยที่



รูปที่ 7 การ รับ-ส่ง เฟรมข้อมูล Modbus RTU

ชุดข้อความสำหรับการอ่านค่าจาก Holding register (Request)

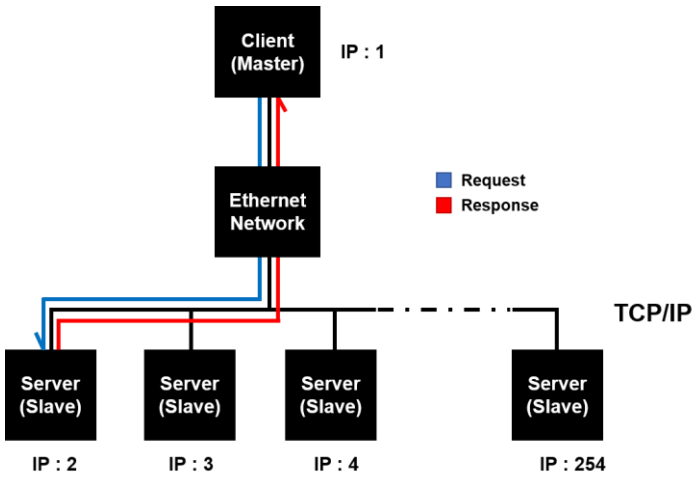
- **13** คือ Station address (19 DEC = 13 HEX)
- **03** คือ Function code (การอ่านค่าที่ Holding registers)
- **0066** คือ Address ของ register ตัวแรก (40103 - 40001 = 102 DEC= 66 HEX)
- **0003** คือ จำนวน Registers ที่ต้องการอ่าน (ทั้งหมด 3 ตัว คือ 40103 ถึง 40105)
- **E6A6** คือ ค่า CRC (Cyclic Redundancy Check) สำหรับเช็คความผิดพลาดของชุดข้อมูล

ชุดข้อความที่ตอบกลับมา (Response)

ส่วน Frame message (ไม่รวม Start และ End bits) ที่ตอบกลับมาคือ **06 XXXX XXXX XXXX**

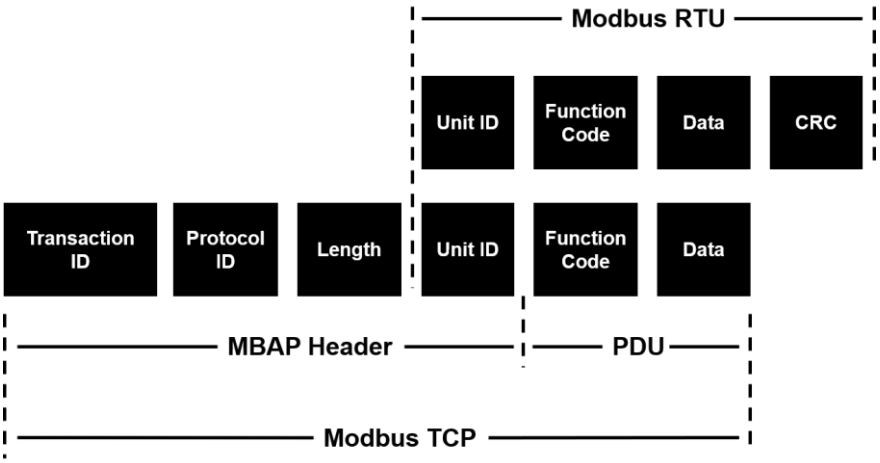
- **06** คือ จำนวน byte ของข้อมูลที่ตอบกลับมา
- **XXXX XXXX XXXX** คือ ข้อมูลของทั้ง 3 ตำแหน่ง ที่ตอบกลับมา (ตำแหน่งละ 2 bytes)

3. Modbus TCP



รูปที่ 8 การสื่อสารแบบอีเทอร์เน็ตสำหรับ Modbus TCP

Modbus TCP คือ โพรโทคอลที่ครอบ Modbus RTU เพื่อใช้การสื่อสารแบบอีเทอร์เน็ต (Ethernet-based protocol) ด้วย TCP/IP (Transmission control protocol) ที่พอร์ต (Port) 502 แทนการใช้การสื่อสารแบบอนุกรม ดังรูปที่ 8 ทำให้อุปกรณ์สามารถสร้างการสื่อสารผ่านเครือข่ายเฉพาะบริเวณ (Local area network : LAN) หรือ เครือข่ายอินเตอร์เน็ต (Internet network) รวมไปถึงการเชื่อมต่อแบบไร้สาย (Wireless) โดยมีอุปกรณ์กระจายสัญญาณ (Router หรือ Access point) เป็นตัวกลางในการเชื่อมต่อ โดยชุดข้อความใน Modbus TCP มีรายละเอียดดังรูปที่ 9 และ 10 เริ่มต้นข้อมูลด้วย Modbus application protocol (MBAP) Header ซึ่งประกอบด้วย Transaction ID, Protocol ID, Length, Unit ID ซึ่งเพิ่มเติมขึ้นมาจาก Modbus RTU ส่วนชุดข้อมูล Function code และ Data จะยังคงเหมือนเดิม ยกเว้นชุดข้อมูล CRC สำหรับเช็คความผิดพลาดจะไม่มี แต่เปลี่ยนไปใช้ของ Ethernet ใน Data link layer แทน

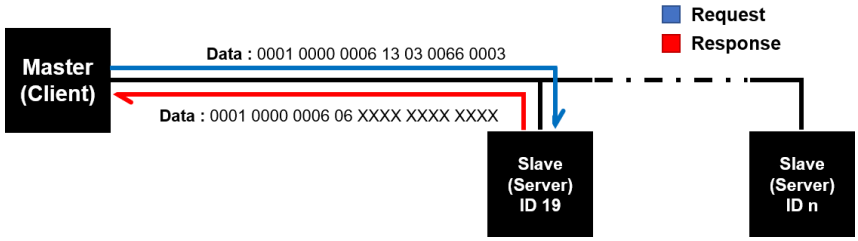


รูปที่ 9 ส่วนประกอบชุดข้อมูลของ Modbus TCP เทียบกับ Modbus RTU [Ref. 2]

Area Name		Area Size	Description
MBAP header (MODBUS® application header)	Transaction ID	2 bytes	Used by the master for matching of the response message from the slave.
	Protocol ID	2 bytes	Indicates the protocol of the PDU (protocol data unit). Stores 0 in the case of MODBUS® /TCP.
	Message length	2 bytes	Stores the message size in byte unit. The message length after this field is stored. (See the above figure.)
	Module ID	1 byte	Used to specify the slave connected to the other line, e.g. MODBUS® serial protocol.
PDU (Protocol data unit)	Function code	1 byte	The master specifies the processing to be performed for the slave.
	Data	1 to 252 bytes	[When master sends request message to slave] Stores the requested processing. [When slave sends response message to master] Stores the result of processing execution.

รูปที่ 10 รายละเอียดของแต่ละ Field ในหนึ่งเฟรมของ Modbus TCP [Ref. 4]

ตัวอย่าง การอ่านค่าของ holding register # 40103 ถึง 40105 จาก slave หมายเลข 19 เหมือนในตัวอย่างของ Modbus RTU ที่กล่าวมาก่อนหน้านี้ ในกรณีของ Modbus TCP frame message จะเป็น **0001 0000 0006 13 03 0066 0003**



รูปที่ 11 การ รับ-ส่ง เฟรมข้อมูล Modbus TCP

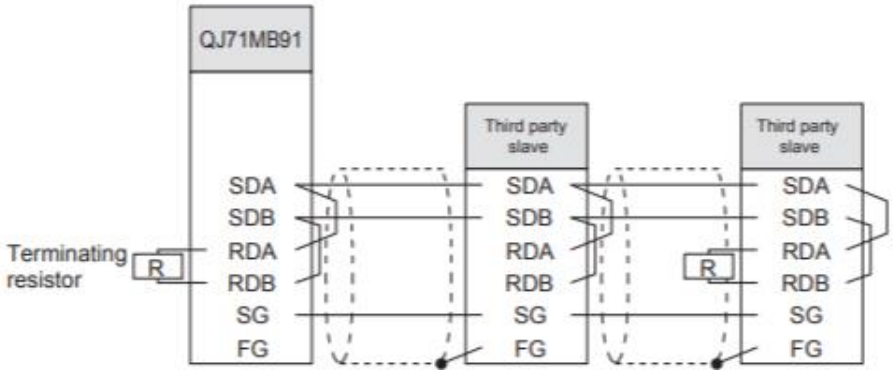
- **0001** คือ Transaction identifier
- **0000** คือ Protocol identifier
- **0006** คือ จำนวน byte ของข้อมูลที่ต่อจากไบต์นี้
- **13** คือ Module identifier หรือ station address (19 ในระบบเลขฐาน 10 จะมีค่าเท่ากับ 13 ในเลขฐาน 16)
- **03** คือ Function code (การอ่านค่าจาก Holding registers)
- **0066** คือ Address ของ register ตัวแรก (40103-40001 = 102 = 66 hex)
- **0003** คือ จำนวน Registers ที่ต้องการอ่าน (ทั้งหมด 3 ตัว คือ 40103 40104 และ 40105)

4. การใช้งาน Modbus RTU และ Modbus TCP

4.1. ตัวอย่างการใช้งาน Modbus RTU ด้วย Mitsubishi Q – Series PLC โมดูล QJ71MB91 อ่านค่าอุณหภูมิจาก PID Controller รุ่น TAIE FY900 ผ่านโปรแกรม GXwork2

4.1.1. การตั้งค่าอุปกรณ์

การใช้งาน Mitsubishi Q - series PLC โมดูล QJ71MB91 ขาสัญญาณที่ใช้สำหรับการสื่อสารบน Physical Layer ด้วยมาตรฐาน RS-485 ซึ่งเป็นการสื่อสารแบบ Half-Duplex หรืออาจกล่าวได้ว่าเป็นการสื่อสารแบบทางเดียว ไม่สามารถรับและส่งข้อมูลพร้อมกันได้ ขาสัญญาณนั้นประกอบด้วย SDA SDB RDA RDB และ SG สามารถแบ่งตามหน้าที่การทำงานได้เป็น 3 กลุ่ม คือ SDA (+) และ SDB (-) สำหรับการส่งข้อมูล (Transmission data) RDA (+) และ RDB (-) สำหรับการรับข้อมูล (Reception data) และ SG สำหรับสายดิน (Signal ground) โดยการต่อสายเพื่อเชื่อมต่อสัญญาณกับอุปกรณ์อื่น มีดังนี้ SDA ต่อกับ RDA และ SDB ต่อกับ RDB ส่วนขา SG ต่อเข้ากับ GND ดังแสดงในรูปที่ 12



รูปที่ 12 Modbus RTU module QJ71MB91 [Ref. 3]

4.1.2. การตั้งค่า GXwork2

วิธีการตั้งค่าการใช้งาน QJ71MB91 ในโปรแกรม GXwork2 [Ref. 3] ขั้นแรกต้องเพิ่มโมดูล QJ71MB91 ที่อยู่ในหมวดของ Intelligent function module หลังจากนั้นเข้าไปที่เมนู Switch setting เพื่อตั้งค่าการสื่อสารและโหมดการใช้งานซึ่งมี 2 โหมด คือ Master Mode และ Slave Mode ดังในรูปที่ 13

Switch Setting 0040:QJ71MB91

Item		CH1	CH2
Mode setting		Master Function /	Master Function
Transmission Setting	MODBUS device assignment parameter starting method	User Setting Parameter	-
	Data bit	8	8
	Parity bit	Exist	Exist
	Even/odd	Odd	Odd
	Stop bit	1	1
	Frame mode	RTU Mode	RTU Mode
	Online change	Enable	Enable
Communication speed setting	Communication speed setting	9600 bps	9600 bps
Station No. setting	Station No. setting	0	0

รูปที่ 13 การตั้งค่า QJ71MB91 ในเมนู Switch setting

นอกจากนี้ยังต้องมีการตั้งค่า Automatic communication parameter สำหรับในกรณีที่เป็น Master mode ดังรูปที่ 14 ซึ่งจะต้องกำหนดค่า Buffer memory ที่จะใช้ในการรับ-ส่งข้อมูลกับอุปกรณ์ Slave และต้องกำหนดว่า Buffer memory นี้จะต้องถูกจับคู่ เข้ากับ Modbus register อันไหนบ้าง ดังตัวอย่างในรูปนั้น Modbus register #138 ถูกจับคู่กับ buffer memory #2000h สำหรับการอ่านค่าจาก Slave และ Modbus register #0 ถูกจับคู่กับ Buffer memory #4000h สำหรับการเขียนค่าไปยัง Slave

Item	CH1	CH2
Automatic communication parameter	Set the automatic communication parameters when using the automatic communication function with the QJ71MB91 operated as a master.	
Automatic Communication Parameter 1	The parameter setting concerning the automatic communication.	
Setting Existence	0:Invalid	1:Valid
Target Station No.	1	1
Request Interval Timer Value	0	50
PLC Response Monitoring Timer Value/Broadcast Delay Value	0	50
Type Specification of The Target MODBUS Device	0000h:No Specification	0500h:Read Holding Registers
Read Setting	The parameter setting concerning reading data from slave.	
Head Buffer Memory Address	0000 h	2000 h
Target MODBUS Device Head Number	0	138
Access Points	0	1
Write Setting	The parameter setting concerning writing data to slave.	
Head Buffer Memory Address	0000 h	0000 h
Target MODBUS Device Head Number	0	0
Access Points	0	0
Automatic Communication Parameter 2	The parameter setting concerning the automatic communication.	
Setting Existence	0:Invalid	1:Valid
Target Station No.	1	1
Request Interval Timer Value	0	50
PLC Response Monitoring Timer Value/Broadcast Delay Value	0	50
Type Specification of The Target MODBUS Device	0000h:No Specification	0005h:Write Holding Registers
Read Setting	The parameter setting concerning reading data from slave.	
Head Buffer Memory Address	0000 h	0000 h
Target MODBUS Device Head Number	0	0
Access Points	0	0
Write Setting	The parameter setting concerning writing data to slave.	
Head Buffer Memory Address	0000 h	4000 h
Target MODBUS Device Head Number	0	0
Access Points	0	1

รูปที่ 14 การตั้งค่า QJ71MB91 ในเมนู automatic communication parameter setting (Master mode)

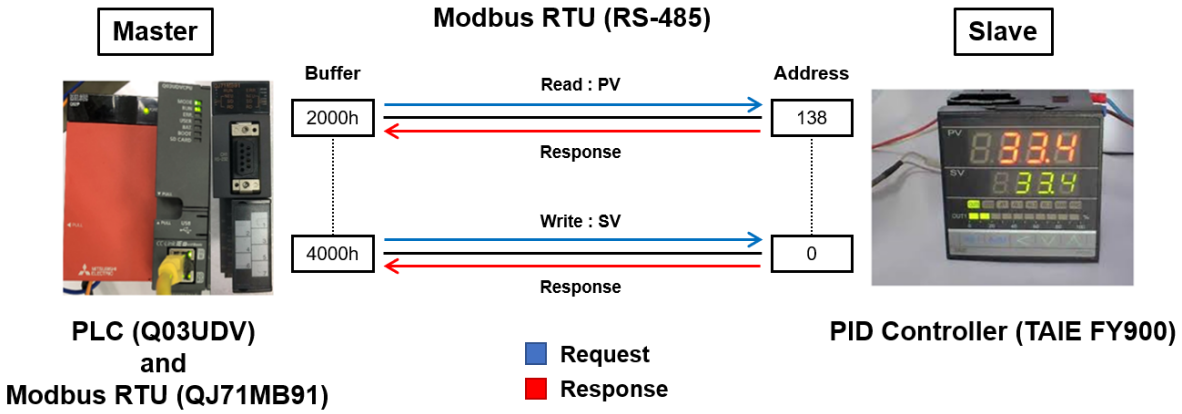
ในกรณีที่ต้องการใช้งาน PLC เป็น Slave mode จำเป็นที่จะต้องตั้งค่าในเมนู Modbus device assignment parameter ดังในรูปที่ 15 ซึ่งจะกำหนดการจับคู่ระหว่างอุปกรณ์ Modbus ทั้ง 4 ประเภทคือ Output coils, Input contacts, Input registers และ Holding registers กับอุปกรณ์ต่างๆ ใน PLC เช่น Output coil #0 ถึง #8192 ถูกจับคู่กับ PLC I/O device Y0 ถึง Y8192 เป็นต้น

Item	Coil	Input	Input Register	Holding Register
MODBUS Device Assignment Parameter	Set the parameter to associate MODBUS device with programmable controller CPU device memory with the QJ71MB91 acting as a slave.			
Assignment 1				
Device	Y0	X0	R0	D0
Head MODBUS Device Number	0	0	0	0
Assignment Points	8192	8192	1000	12288
Assignment 2				
Device	M0			SD0
Head MODBUS Device Number	8192	0	0	20480
Assignment Points	8192	0	0	2048
Assignment 3				
Device	SM0			H5000
Head MODBUS Device Number	20480	0	0	22528
Assignment Points	2048	0	0	4096
Assignment 4				
Device	L0			W0
Head MODBUS Device Number	22528	0	0	30720
Assignment Points	8192	0	0	8192
Assignment 5				
Device	B0			SW0
Head MODBUS Device Number	30720	0	0	40960
Assignment Points	8192	0	0	2048
Assignment 6				
Device	F0			TN0
Head MODBUS Device Number	38912	0	0	53248
Assignment Points	2048	0	0	2048

รูปที่ 15 การตั้งค่า QJ71MB91 ในเมนู Modbus device assignment parameter (Slave mode)

4.1.3. ตัวอย่างการใช้งาน Modbus RTU

รูปที่ 16 แสดงตัวอย่างการใช้งาน Modbus RTU โดยที่ PLC ทำหน้าที่เป็น Modbus Master เพื่ออ่านและตั้งค่าอุณหภูมิของ PID controller จากคู่มือการใช้งานของ PID controller [Ref 6] การอ่านค่า Present value (PV) ถูกกำหนดไว้ที่ Holding register หมายเลข 138 ส่วนการเขียนค่าเพื่อตั้งค่า Set Value (SV) นั้นถูกกำหนดไว้ที่ holding register หมายเลข 0 ดังนั้นในส่วนของ PLC เราก็สามารถกำหนด Buffer memory เช่น #2000h สำหรับการอ่านค่าจาก register# 138 และ #4000h สำหรับการเขียนค่าไปยัง register #0 เป็นต้น



รูปที่ 16 ตัวอย่างการใช้งาน Modbus RTU

4.2. ตัวอย่างการใช้งาน Modbus TCP ด้วย Mitsubishi Q – Series PLC โมดูล QJ71MT91 ผ่านโปรแกรม GXwork2

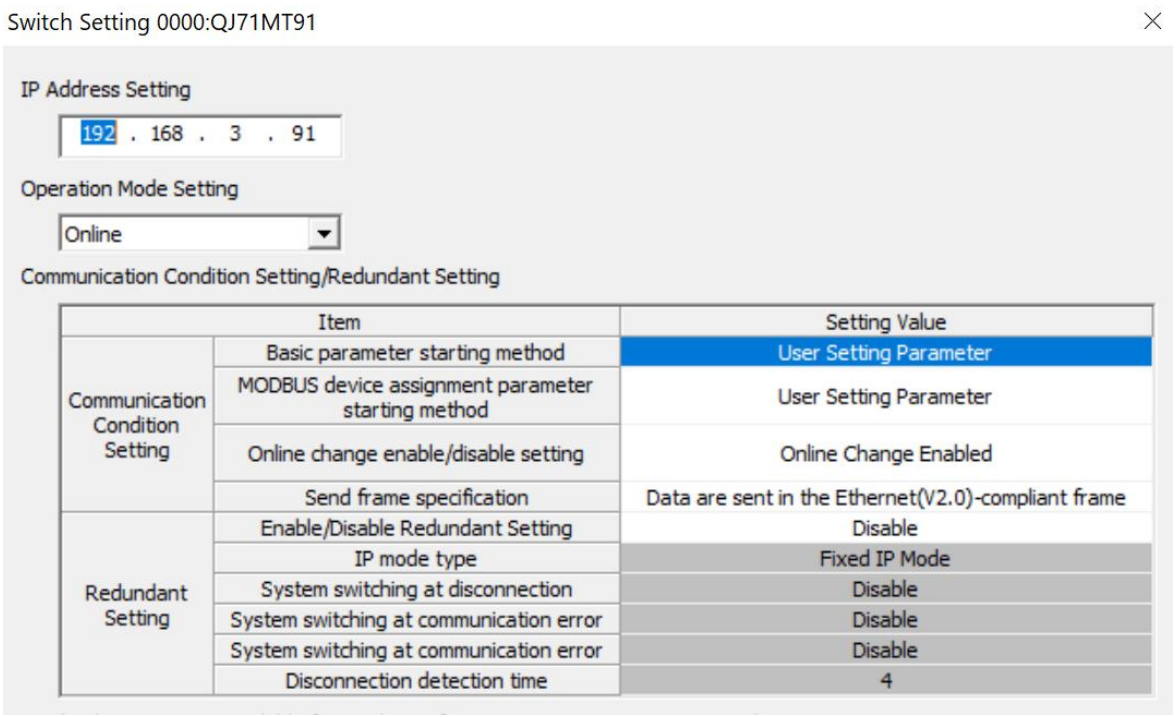
4.2.1. การตั้งค่าอุปกรณ์

การใช้งานโมดูลการสื่อสาร Modbus TCP สำหรับ Mitsubishi Q-series PLC โมดูล QJ71MT91 แสดงดังรูปที่ 17 โมดูลนี้รองรับทั้งโหมด Master และ Slave โดยที่ในโหมด Master สามารถสื่อสารได้สูงสุด 64 Slave ในเวลาเดียวกัน ส่วนในโหมด Slave สามารถรับข้อมูล Request message ได้ถึง 64 ข้อความในเวลาเดียวกัน และมีช่องการสื่อสารแบบ Ethernet 1 ช่อง

รูปที่ 17 Mitsubishi Q-series PLC โมดูล QJ71MT91 [Ref. 4]

4.2.2. การตั้งค่า GXwork2

วิธีการตั้งค่าการใช้งานโมดูล QJ71MT91 ในโปรแกรม GXwork2 [Ref. 4] ขั้นแรกจำเป็นที่จะต้องเพิ่มโมดูล QJ71MT91 ก่อน ซึ่งอยู่ในหมวดของ Intelligent Function Module หลังจากนั้นเข้าไปที่เมนู Switch setting เพื่อตั้งค่าการสื่อสารและโหมดการใช้งานว่าต้องการที่จะให้โมดูลทำหน้าที่เป็น Master หรือ Slave ดังรูปที่ 18



รูปที่ 18 QJ71MT91 Switch setting

ต่อไปเป็นการตั้งค่า Automatic communication parameter สำหรับในกรณีที่เป็น Master mode ดังรูปที่ 19 จะต้องกำหนดค่า Buffer memory ที่จะใช้ในการรับ-ส่งข้อมูลกับอุปกรณ์ Slave และต้องกำหนดว่า Buffer memory นี้จะต้องถูกจับคู่เข้ากับ Modbus register ที่ต้องการจากอุปกรณ์ Slave เช่นเดียวกับในกรณีของ Modbus RTU

Item	Setting Value
Automatic Communication Parameter	Set the automatic communication parameters when using the automatic communication function with the QJ71MT91 acting as the master.
Automatic Communication Parameter 1	The parameter setting concerning the automatic communication.
Target Station IP Address	192.168.3.2
Module ID	255
Repetition Interval Timer Value	10
Response Monitoring Timer Value	60
Type Specification of The Target MODBUS Device	0505h:Read/Write Holding Registers
Read Setting	The parameter setting concerning reading data from slave.
Head Buffer Memory Address	1000 h
Target MODBUS Device Head Number	16394
Access Points	10
Write Setting	The parameter setting concerning writing data to slave.
Head Buffer Memory Address	3000 h
Target MODBUS Device Head Number	16404
Access Points	10

รูปที่ 19 QJ71MT91 Automatic communication parameter setting (Master mode)

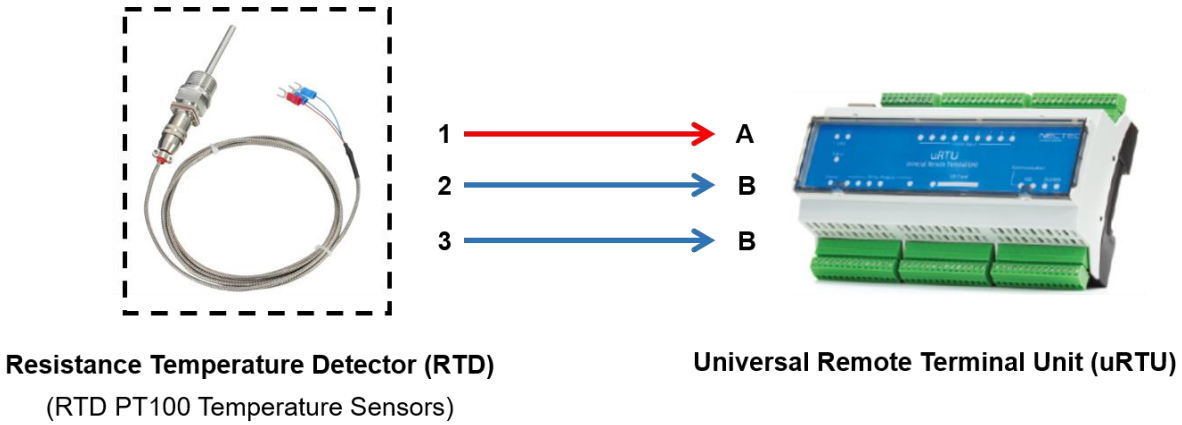
แต่ในกรณีใช้งานเป็น Slave mode ต้องตั้งค่าในเมนู Modbus device assignment parameter ดังในรูปที่ 20 ซึ่งจะกำหนดการจับคู่ระหว่างอุปกรณ์ ทั้ง 4 ประเภทคือ Output coils Input contacts Input registers และ Holding registers กับอุปกรณ์ต่างๆ ใน PLC เช่นเดียวกับในกรณีของ Modbus RTU

Item	Coil	Input	Input Register	Holding Register
MODBUS Device Assignment Parameter	Set the parameter to associate MODBUS device with programmable controller CPU device memory with the QJ71MT91 acting as a slave.			
Assignment 1				
Device	Y0	X0		D0
Start MODBUS Device Start Number	0	0	0	0
Assignment Points	8192	8192	0	12288
Assignment 2				
Device	M0			SD0
Start MODBUS Device Start Number	8192	0	0	20480
Assignment Points	8192	0	0	2048
Assignment 3				
Device	SM0			H5000
Start MODBUS Device Start Number	20480	0	0	22528
Assignment Points	2048	0	0	4096
Assignment 4				
Device	L0			W0
Start MODBUS Device Start				

รูปที่ 20 QJ71MT91 Modbus device assignment parameter (slave mode)

4.3. ตัวอย่างการใช้งาน Modbus TCP ด้วย Universal Remote Terminal Unit (uRTU) อ่านค่าอุณหภูมิจาก RTD RT 100 และส่งข้อมูลไปยัง IoT Platform ผ่าน Node-Red

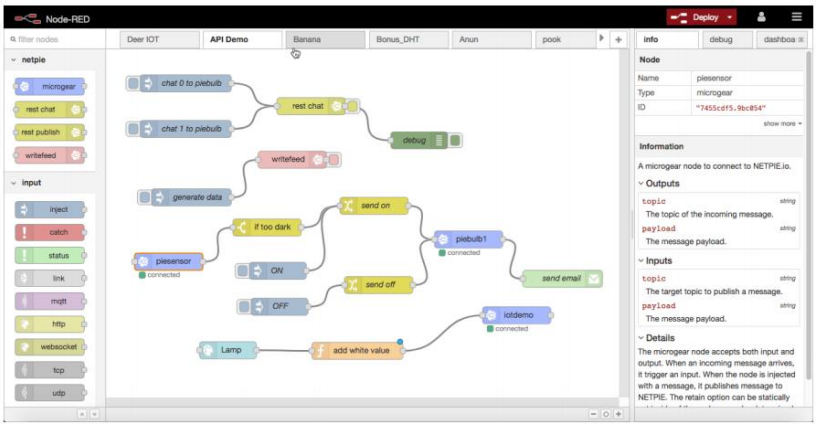
4.3.1. การตั้งค่าอุปกรณ์



รูปที่ 21 การเชื่อมต่ออุปกรณ์ Resistance Temperature Detector (RTD) กับ Universal Remote Terminal Unit (uRTU)

หน่วยตรวจวัดระยะไกลยูนิเวอร์แซล (Universal Remote Terminal Unit : uRTU) [Ref. 7] ซึ่งพัฒนาโดยทีมวิจัยจาก NECTEC เป็นอุปกรณ์ที่ใช้ในการเปลี่ยนสัญญาณข้อมูลทาง Analog ที่ได้รับจาก Sensor เช่น แรงดัน กระแสไฟฟ้า ความถี่ หรือปริมาณอื่น ๆ ให้เป็นข้อมูล Digital เพื่อนำไปใช้ในการประมวลผล หรือรับข้อมูลจากคอมพิวเตอร์เพื่อนำไปใช้สั่งงานการเปิด-ปิด อุปกรณ์ต่าง ๆ ผ่าน uRTU อีกทั้งยังสามารถเก็บข้อมูลตามเวลาโดยอัตโนมัติ (Data logger) โดย uRTU สามารถติดต่อกับคอมพิวเตอร์ผ่านทาง LAN ด้วย Modbus protocol จากรูปที่ 21 แสดงการเชื่อมต่อสายสัญญาณจากอุปกรณ์ RTD PT100 แบบ 3 สาย ไปยังช่องเชื่อมต่อที่ 1 ของ RTD Card บน uRTU ซึ่งมีทั้งหมด 5 ช่องเชื่อมต่อแต่ละช่องนั้นจะใช้ 2 แอดเดรสในการเก็บข้อมูล แอดเดรสสำหรับอ่านค่าอุปกรณ์ RTD จะอยู่ที่ 30049-30064 โดยขา 1 2 3 ของ RTD ต่อไปยังช่อง A B B ของ uRTU ตามลำดับ

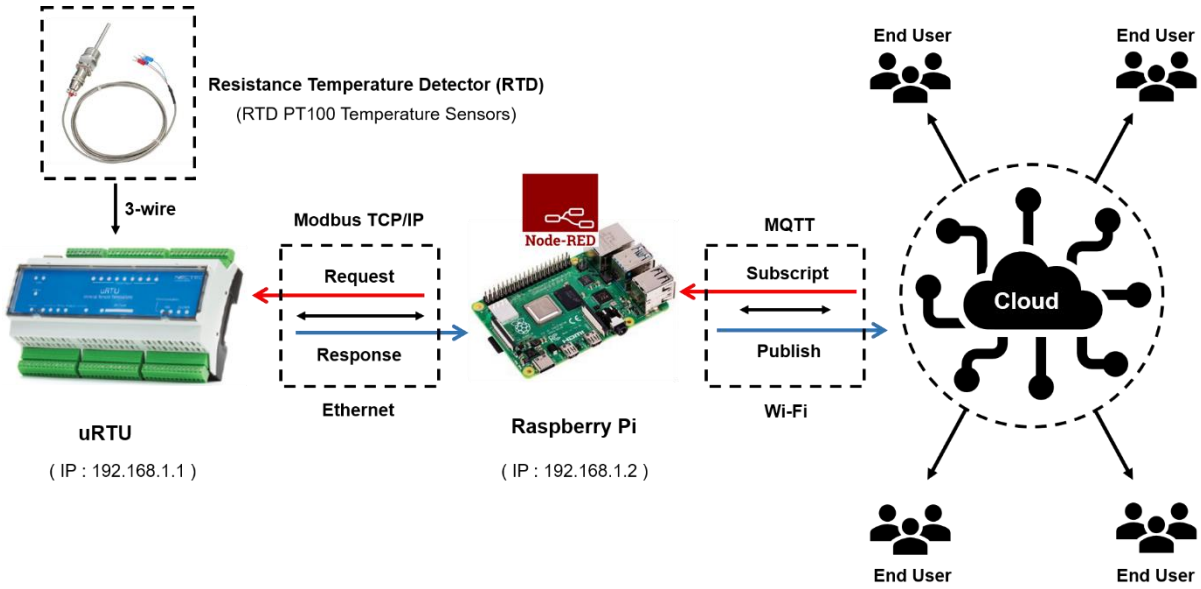
4.3.2. การตั้งค่า Node-Red



รูปที่ 22 โปรแกรม Node-Red

Node-RED เป็นเครื่องมือสำหรับนักพัฒนาโปรแกรมในการเชื่อมต่ออุปกรณ์ฮาร์ดแวร์เข้ากับ APIs (Application Programming Interface) และ Online Services ด้วยวิธีการใหม่ ๆ มีทั้งที่ใช้ได้ฟรีและในส่วนที่ต้องเสียเงิน โดย Node-RED สามารถออกแบบ API ในการรับข้อมูล การคำนวณ การแปลงข้อมูล การเก็บข้อมูล หรือเชื่อมต่อกับบริการอื่น ๆ ได้ตามความต้องการของผู้ใช้ วิธีการติดตั้งสามารถดูได้จาก [Ref 8] โปรแกรม Node-Red สามารถติดตั้งได้หลากหลายแพลตฟอร์ม เช่น คอมพิวเตอร์ส่วนบุคคล Single Board Computer ประเภทต่าง ๆ ในเอกสารนี้เลือกใช้เป็น Raspberry Pi 4 ซึ่งเป็น Single Board Computer ประเภทหนึ่งที่นิยมใช้กันอย่างแพร่หลายสำหรับการสร้างงานต้นแบบ เนื่องจากประสิทธิภาพสูง ราคาไม่สูงมาก และง่ายต่อการใช้งาน

4.3.3. แผนผังการใช้งาน Modbus TCP ด้วย Universal Remote Terminal Unit (uRTU) อ่านค่าอุณหภูมิจาก RTD RT 100 และส่งข้อมูลไปยัง IoT Platform ผ่าน Node-Red



รูปที่ 23 ภาพรวมการใช้งาน Node-Red อ่านค่าอุปกรณ์ RTD จาก uRTU ด้วยโพรโทคอล Modbus TCP

จากรูปที่ 23 แสดงการเชื่อมต่อและการสื่อสารข้อมูลของอุปกรณ์แต่ละอุปกรณ์ โดย Raspberry Pi ทำหน้าที่เป็น Gateway หรือ ตัวกลางในการแปลงเฟรมข้อมูลจากโพรโทคอล Modbus ให้เป็นโพรโทคอล MQTT เพื่อส่งข้อมูลไปยัง IoT Platform ในตัวอย่างนี้เลือกใช้ NETPIE IoT Cloud Platform ที่เปิดให้บริการโดยเนคเทค (รายละเอียดการใช้งานสามารถศึกษาได้จาก [Ref 9]) เพื่อให้ผู้ใช้งานสามารถติดตามผลและควบคุมจากระยะไกล โดยโปรแกรมที่เข้ามาช่วยในการสร้างระบบ IoT สำหรับอุปกรณ์อุตสาหกรรมนี้คือ Node-Red รายละเอียดมีดังนี้

รูปที่ 24 การตั้งค่าพารามิเตอร์ใน Node ต่างๆ บน Node-Red

จากรูปที่ 24 *ส่วนที่ 1* เป็นการใช้งาน Function Node ในการกำหนดเฟรมข้อมูลคำสั่ง Request รูปแบบ JSON ไปยัง uRTD เพื่อให้ได้รับ Response ข้อมูลค่าที่วัดได้จากอุปกรณ์ RTD กลับมา ซึ่งเฟรมข้อมูลดังกล่าวจะประกอบไปด้วย [‘fc’:4] คือ การเลือก Function เป็น Read input registers (ช่วงแอดเดรส 30001-39999) [‘unitid’:1] คือ กำหนด ID ของ uRTU [‘address’:48] คือ การกำหนดแอดเดรสเริ่มต้นที่ต้องการอ่านไว้ที่ 30049 และสุดท้าย [‘quantity’:2] คือ การกำหนดจำนวนแอดเดรสที่ต้องการอ่าน (30049-30050) โดยชุดข้อมูลนี้สามารถปรับเปลี่ยนได้ตามความเหมาะสมของอุปกรณ์ Modbus (FC,UnitId,Address และ Quantity)

ส่วนที่ 2 เป็นการกำหนดค่าพารามิเตอร์สำหรับ Modbus Node โดยเลือกใช้ Modbus Flex Getter เป็น Node การอ่านค่าจากอุปกรณ์ Modbus สำหรับ Type กำหนดเป็น TCP (สามารถเลือกใช้งานได้ทั้งแบบ TCP และ Serial) Host เป็นการตั้งค่า IP ของ uRTU และ Unit Id สำหรับตั้งค่า UnitId ของ uRTD

ส่วนที่ 3 เป็นการกำหนดค่าพารามิเตอร์สำหรับ MQTT node เพื่อส่งข้อมูล (Publish) ที่อ่านได้ไปยัง IoT Platform ต่างๆ ขั้นตอนแรกกำหนด Topic ให้กับข้อมูลที่ต้องการส่งก่อน ในตัวอย่างกำหนดเป็น @msg/rtd/ch1 ในส่วน Connection พารามิเตอร์แรกคือการกำหนด Server หรือ End-Point (ตามตัวอย่างกำหนดเป็นเป็น broker.netpie.io) และพอร์ตสำหรับโพรโทคอล MQTT (ปกติแล้วจะกำหนดเป็น 1883) ต่อมาเป็น Client ID คือการกำหนดชื่อให้อุปกรณ์ซึ่งใช้ในการอ้างอิงสำหรับการสื่อสารด้วยโพรโทคอล MQTT โดยในแต่ละ IoT Platform จะกำหนดต่างกันออกไปตามแต่รูปแบบของ Platform นั้น ๆ สำหรับส่วน

Security เป็นการเพิ่มความปลอดภัยในการส่งข้อมูลไปยัง Platform โดยจะมี 2 พารามิเตอร์ คือ Username และ Password ขึ้นอยู่กับการกำหนดจากทาง Platform

5. ข้อสรุป

Modbus เป็นโพรโทคอลการสื่อสารที่ใช้กันอย่างแพร่หลาย เนื่องจากเป็นโพรโทคอลแบบเปิด มีความน่าเชื่อถือ และง่ายต่อการใช้งาน จากตัวอย่างที่ได้นำเสนอในเบื้องต้นแสดงให้เห็นว่าอุปกรณ์ที่ใช้สำหรับโรงงานอุตสาหกรรมส่วนใหญ่มักจะรองรับโพรโทคอลนี้เป็นหนึ่งในช่องทางการสื่อสาร โดย Modbus ที่ใช้งานกันทั่วไปจะมี 2 รูปแบบ คือ Modbus RTU เป็นการสื่อสารแบบอนุกรม (Serial Level Protocol) ผ่าน RS485 หรือ RS232 เป็นการสื่อสารพื้นฐานที่ใช้ในอุปกรณ์สำหรับโรงงานอุตสาหกรรมทั่วไป ซึ่งมีข้อจำกัดอยู่ที่การสร้างระบบเครือข่ายการสื่อสารข้อมูลระหว่างอุปกรณ์ ทั้งในด้านความหลากลาย และการขยายระบบเพื่อลองรับอุปกรณ์ที่มีเพิ่มมากขึ้น และอีกรูปแบบ คือ Modbus TCP เป็นการสื่อสารแบบอีเทอร์เน็ต (Ethernet physical layer) ซึ่งถูกสร้างขึ้นมาเพื่อแก้ไขข้อจำกัดของ Modbus RTU ให้สามารถจัดการและเข้าถึงอุปกรณ์ได้ง่ายขึ้นผ่าน TCP/IP ในทางกลับกันนั้น Modbus ทั้ง 2 รูปแบบนี้มีข้อดีและข้อเสียต่างกันขึ้นอยู่กับรูปแบบงานที่นำไปใช้ และเพื่อต่อยอดไปยังระบบ IoT อุปกรณ์ Modbus ส่วนใหญ่ในปัจจุบันได้พัฒนาให้สามารถเชื่อมต่อเข้ากับ IoT Platform ที่มีให้เลือกใช้อย่างหลากหลาย ทำให้สามารถจัดการและติดตามผลของอุปกรณ์ต่าง ๆ ในโรงงานจากระยะไกลได้ โดยผ่านเครือข่ายอินเตอร์เน็ตแบบ Real-Time เพื่อนำผลที่ได้ไปวิเคราะห์และปรับปรุงประสิทธิภาพของโรงงานให้เพิ่มมากขึ้น

เอกสารอ้างอิง

[1] Modbus [อินเทอร์เน็ต]. [เข้าถึงเมื่อ 6 ม.ค. 2563]. เข้าถึงได้จาก https://en.wikipedia.org/wiki/Modbus

[2] Modbus [อินเทอร์เน็ต]. [เข้าถึงเมื่อ 6 ม.ค. 2563]. เข้าถึงได้จาก http://www.simplymodbus.ca/

[3] Modbus Interface Module User's Manual [เข้าถึงเมื่อ 6 ม.ค. 2563]. เข้าถึงได้จาก http://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh080578eng/sh080578engj.pdf

[4] Modbus/TCP Interface Module User's Manual [เข้าถึงเมื่อ 6 ม.ค. 2563]. เข้าถึงได้จาก https://eu3a.mitsubishielectric.com/fa/en/dl/1694/158847.pdf

[5] GX Works2 Beginner's Manual [เข้าถึงเมื่อ 6 ม.ค. 2563]. เข้าถึงได้จาก https://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh080787eng/sh080787engr.pdf

[6] FY-Series Digital PID Controller Operation Manual [เข้าถึงเมื่อ 6 ม.ค. 2563]. เข้าถึงได้จาก http://www.golink.com.tw/pdf/FY-Operation_Manual-V200602.pdf

[7] Universal Remote Terminal Unit (uRTU) [เข้าถึงเมื่อ 15 พ.ค. 2563]. เข้าถึงได้จาก https://www.nectec.or.th/ace2019/wp-content/uploads/2019/09/uRTU.pdf

[8] Running on Raspberry Pi [เข้าถึงเมื่อ 15 พ.ค. 2563]. เข้าถึงได้จาก https://nodered.org/docs/getting-started/raspberrypi

[9] NETPIE IoT Cloud-based Platform-as-a-Service [เข้าถึงเมื่อ 15 พ.ค. 2563]. เข้าถึงได้จาก https://netpie.io/