การพัฒนาระบบจัดการฟาร์มอัจฉริยะ

SMART FARM MANAGEMENT SYSTEM DEVELOPMENT

นายธนาตย์ จอมใจเอกชน

Thanart Jomjaiekahorn

นายเรืองเกียรติ ปิยะณรงค์เดช

Rueangkiat Piyanarongdej

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2565

SMART FARM MANAGEMENT SYSTEM DEVELOPMENT

Thanart Jomjaiekahorn

Rueangkiat Piyanarongdej

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INFORMATION ENGINEERING

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

หัวข้อปริญญานิพนธ์	การพัฒนาระบบจัดการฟาร์มอัจฉริยะ	
รายชื่อนักศึกษา	นายธนาตย์ จอมใจเอกชน	รหัสนักศึกษา 62010388
	นายเรืองเกียรติ ปิยะณรงค์เดช	รหัสนักศึกษา 62010782
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2565	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.ไพศาล สิทธิโยภาสกุล	
	ดร.ธนวิชญ์ อนุวงศ์พินิจ	

ปริญญานิพนธ์ฉบับนี้ ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

.....

(ผศ.ไพศาล สิทธิโยภาสกุล)

.....

(ดร.ธนวิชญ์ อนุวงศ์พินิจ)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	การพัฒนาระบบจัดการฟาร์มอัจฉริยะ	
รายชื่อนักศึกษา	นายธนาตย์ จอมใจเอกชน	รหัสนักศึกษา 62010388
	นายเรืองเกียรติ ปิยะณรงค์เดช	รหัสนักศึกษา 62010782
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2565	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.ไพศาล สิทธิโยภาสกุล	
	ดร.ธนวิชญ์ อนุวงศ์พินิจ	

บทคัดย่อ

ปริญญานิพนธ์นี้เสนอการพัฒนาระบบจัดการฟาร์มอัจฉริยะ โดยใช้อินเทอร์เน็ตเข้ามาในระบบฟาร์ม ซึ่งสามารถนำมาใช้ทดแทนการทำงานในแบบเดิมที่ต้องใช้มนุษย์มากกว่าและยังยุ่งยากในการจัดการ โดยการพัฒนาระบบจัดการฟาร์มอัจฉริยะที่สามารถจัดการอุปกรณ์ต่างๆภายในฟาร์มผ่านการสื่อสารด้วยมาตรฐาน RS-485 Modbus RTU และการสื่อสารผ่าน Modbus TCP/IP และสามารถส่งข้อมูลไปเก็บในฐานข้อมูลและแสดงผลข้อมูลบนคลาวด์ของAmazon Web Services (AWS) โดยผ่านการสื่อสารด้วยมาตรฐาน Message Queuing Telemetry Transport (MQTT) ที่ใช้โครงสร้างการทำงานแบบ Serverless เพื่อใช้ในการทำระบบอัตโนมัติ และพัฒนาเว็บแอปพลิเคชันโดยใช้ Next.js เป็น React Web Framework ในการจัดการและควบคุมข้อมูลในฐานข้อมูล ได้แก่การสร้าง ลบ แก้ไข อ่าน ข้อมูลในฐานข้อมูล การยืนยันตัวตน การค้นหาข้อมูลในหน้าแดชบอร์ด และการควบคุมเปิด/ปิดอุปกรณ์เพื่อส่งคำสั่งกลับไปยังอุปกรณ์เพื่อควบคุม ซึ่งจะทำให้การดูแลจัดการฟาร์มสามารถทำได้ง่ายมากขึ้น เพื่อแก้ปัญหาการจัดการระบบภายในฟาร์มให้มีความสะดวกสบายเพิ่มมากขึ้นและลดความซับซ้อนลง

Thesis Title	Smart Farm Management System Development	
Student	Mr.Thanart Jomjaiekahorn	Student ID. 62010388
	Mr.Rueangkiat Piyanarongdej	Student ID. 62010782
Degree	Bachelor of Engineering	
Program	Information Engineering	
Year	2022	
Thesis Advisor	Asst.Prof.Paisan Sithiyopasakul	
	Dr.Thanavit Anuwongpinit	

ABSTRACT

This thesis proposes the development of an intelligent farm management system using Internet of Things (IoT) technology. The system aims to replace traditional labor-intensive and complex farm management practices by utilizing the Internet to control and monitor various farm devices. The communication between devices within the farm is achieved through the RS-485 Modbus RTU standard and Modbus TCP/IP protocol. Data is collected, stored, and displayed on the cloud using Amazon Web Services (AWS) and the Message Queuing Telemetry Transport (MQTT) protocol, which follows a serverless architecture for automation. The web application is developed using Next.js, a React web framework, to manage and control the database, including operations such as creating, deleting, editing, and reading data. The application also includes user authentication, data search on the dashboard, and device control (on/off) to send commands back to the devices for control purposes. This intelligent farm management system aims to simplify and streamline farm maintenance and management, providing a more convenient and less complex solution for farm owners and operators.

กิตติกรรมประกาศ

ปริญญานิพนธ์เล่มนี้สามารถสำเร็จสมบูรณ์ได้ด้วยดี โดยความอนุเคราะห์และเอาใจใส่ให้คำปรึกษาเป็นอย่างดีมาโดยตลอดจากอาจารย์ที่ปรึกษา ดร.ธนวิชญ์ อนุวงศ์พินิจ และ ผศ.ไพศาล สิทธิโยภาสกุล ที่ได้คอยแนะนำและแก้ปัญหามาโดยตลอดอีกทั้งยังให้ความรู้และประสบการณ์ต่างๆมากมาย จึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาส นี้

ขอขอบคุณ คณาจารย์สาขาวิชาวิศวกรรมสารสนเทศทุกท่าน ที่คอยสนับสนุนในเรื่องของอุปกรณ์ต่างๆที่ใช้ในปริญญานิพนธ์เล่มนี้และให้คำแนะนำในโครงร่างปริญญานิพนธ์ จนทำให้ปริญญานิพนธ์เล่มนี้สำเร็จลงได้

ขอขอบคุณ พี่ๆ เพื่อนๆ และน้องๆ ทุกคนที่คอยเป็นกำลังใจ รวมทุกข์รวมสุข และให้ความช่วยเหลือเก้อกูลตลอดมา

พวกผมมีความซาบซึ้งในความกรุณาของทุกท่านที่ได้กล่าวถึงและผู้ที่ไม่ได้เอ่ยนามในที่นี้ จึงขอกราบขอบพระคุณทุกท่านด้วยความจริงใจ และขอมอบคุณประโยชน์อันที่เกิดมาจากปริญญานิพนธ์เล่มนี้ให้แก่ บิดา มารดา อาจารย์ และผู้เกี่ยวข้องทุกท่านที่ให้การสนับสนุนที่ได้ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ต่างๆ ให้แก่พวกผม

นายธนาตย์ จอมใจเอกชน

นายเรืองเกียรติ ปิยะณรงค์เดช

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1

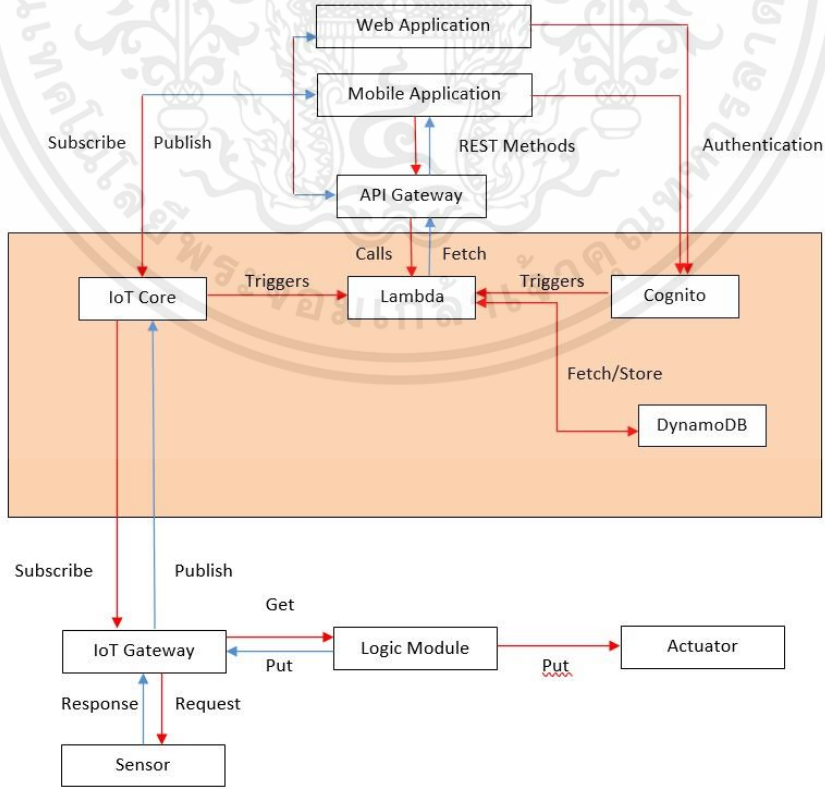
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การทำฟาร์มในแบบเดิม ๆ ในปัจจุบันนั้น มีการสูญเสียเสียทรัพยากรและแรงงานที่มากเกินความจำเป็น อีกทั้งยังมีปัญหาในด้านผลผลิต ที่เกิดจากขาดการวิเคราะห์และจัดการกับข้อมูลเชิงลึก เช่น สภาพอากาศ สภาพดิน สิ่งแวดล้อมที่มีความจำเป็นและสำคัญต่อการเจริญเติบโตของพืช/เลี้ยงสัตว์ เป็นต้น

การนำอินเทอร์เน็ตของทุกสรรพสิ่ง (Internet of things) เข้ามามีบทบาทสำคัญในทุกภาคส่วนของธุรกิจและชีวิตประจำวันของเรา เช่น การสั่งเปิด-ปิดอุปกรณ์ภายในบ้านต่างๆ ผ่านสมาร์ทโฟนหรือ คอมพิวเตอร์ ไม่ว่าจะเป็น หลอดไฟ เครื่องปรับอากาศ โทรทัศน์ เป็นต้น ซึ่งนับได้ว่าอินเทอร์เน็ตของสรรพสิ่งเป็นหนึ่งในเทคโนโลยีที่นำมาใช้เพื่อความสะดวกสบาย ลดภาระในการทำงาน อีกทั้งยังเป็นตัวช่วยที่ทำให้การทำงานง่ายขึ้นมาก

การพัฒนาระบบจัดการฟาร์มอัจฉริยะ โดยเอาระบบอินเทอร์เน็ตของทุกสรรพสิ่งเข้ามาช่วยในการบริหารจัดการดูแลการเพาะปลูก รวมไปถึงกระบวนการผลิตเพื่อนำไปสู่การเกษตรเชิงธุรกิจ และนำเอาเทคโนโลยีสมัยใหม่เข้ามาผสมผสานเข้ากับงานด้านการเกษตรเพื่อช่วยแก้ไขปัญหาต่างๆ โดยได้วางผังระบบดังแสดงในรูปที่ 1 ดังนี้



รูปที่ 1 การทำงานของระบบ

1.2 จุดประสงค์

เพื่อศึกษาการพัฒนาระบบจัดการฟาร์มอัจฉริยะที่สามารถจัดการอุปกรณ์ต่างๆภายในฟาร์ม โดยการใช้ Amazon Web Services ในการจัดการฐานข้อมูลเพื่อการเก็บข้อมูลและแสดงผลข้อมูล เพื่อศึกษาและพัฒนา Web Application ในการใช้ในเพื่อจัดการข้อมูลในฐานข้อมูล เพื่อศึกษาการทำงานของเกตเวย์และอุปกรณ์ต่างๆ ภายในฟาร์ม เพื่อศึกษาวิธีการรับ-ส่งข้อมูลของอุปกรณ์ผ่านการสื่อสารด้วยมาตรฐาน RS485 และการรับ-ส่งข้อมูลระหว่างอุปกรณ์กับฐานข้อมูลผ่านการสื่อสารด้วยมาตรฐาน MQTT เพื่อศึกษาการจัดการบนระบบ Cloud ในการเก็บข้อมูลและแสดงผลข้อมูล เพื่อแก้ปัญหาการจัดการระบบภายในฟาร์มให้มีความสะดวกสบายเพิ่มมากขึ้นและลดความซับซ้อนลง

1.3 ขอบเขตของโครงงาน

1) มีระบบจัดการฟาร์มที่มีอินเตอร์เฟสสามารถใช้งานผ่านทางเว็บแอปพลิเคชันได้ ซึ่งสามารถจัดการสมาชิก จัดการฟาร์ม และจัดการข้อมูลของอุปกรณ์ต่างๆได้
2) สามารถรับ-ส่งข้อมูลระหว่างเกตเวย์กับอุปกรณ์ต่างๆผ่านการสื่อสารด้วยมาตรฐาน RS485 ได้
3) สามารถรับ-ส่งข้อมูลระหว่างเกตเวย์กับฐานข้อมูลผ่านการสื่อสารด้วยมาตรฐาน MQTT ได้
4) มีการจัดการบนระบบ Cloud โดยจัดเก็บข้อมูล และนำมาแสดงผลได้

1.4 ผลที่คาดว่าจะได้รับ

1) ระบบที่พัฒนาขึ้นสามารถนำไปใช้จัดการฟาร์มได้จริง
2) ได้รับความรู้เพิ่มเกี่ยวกับการจัดการระบบฟาร์มบนเว็บ ระบบฐานข้อมูล อุปกรณ์ที่สื่อสารด้วยมาตรฐาน RS485 การทำงานของเกตเวย์ที่สื่อสารด้วยมาตรฐาน MQTT และเทคโนโลยีการจัดการบนระบบ Cloud
3) ระบบที่ทำการศึกษาสามารนำความรู้ไปประยุกต์ใช้ในการจัดการระบบในงานด้านอื่นๆได้
4) ระบบที่พัฒนาขึ้นช่วยสร้างความสะดวกสบายและลดความซับซ้อนในการจัดการฟาร์มให้แก่มนุษย์มากขึ้น

1.5 อุปกรณ์ที่ต้องใช้

ฮาร์ดแวร์

- เครื่องคอมพิวเตอร์สำหรับพัฒนาระบบที่มีการต่อเชื่อมกับเน็ตเวิร์ค จำนวน 2 เครื่อง
- เครื่องเกตเวย์ที่มีการต่อเชื่อมกับเน็ตเวิร์ค (IoT Gateway) จำนวน 1 เครื่อง
- อุปกรณ์วัดความชื้นในดิน จำนวน 1 ชุด
- อุปกรณ์วัดธาตุสารอาหารในดิน จำนวน 1 ชุด
- Switching Power Supply 12 VDC จำนวน 1 ชุด
- เครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ (controller) จำนวน 1 เครื่อง
- พัดลมระบายอากาศ 12 VDC จำนวน 1 ชุด
- ตู้ระบบคอนโทรล จำนวน 1 ชุด

ซอฟต์แวร์

- ใช้ Next.js ซึ่งคือ React Web Framework เป็นการพัฒนาระบบการจัดการฟาร์ม
- ใช้ระบบปฏิบัติการลินุกซ์ ในการจัดการอุปกรณ์เกตเวย์
- ใช้โปรแกรม Modbus Poll ในการจัดการระบบการสื่อสารมาตรฐาน RS485 Modbus RTU ระหว่างเกตเวย์กับอุปกรณ์ต่างๆ
- ใช้โปรแกรม Node-RED ให้ทำงานเป็น Edge Computing รวมถึงการจัดการข้อมูลของอุปกรณ์เซ็นเซอร์ อุปกรณ์สั่งการ คอนโทรลเลอร์ (PLC Siemens LOGO) และข้อมูลจาก AWS Cloud Service โดยมาตรฐานการสื่อสาร MQTT Protocol
- ใช้โปรแกรม WinSCP สำหรับอัพโหลดและดาวน์โหลดไฟล์ผ่าน Protocol FTP จากคอมพิวเตอร์ของผู้ใช้ไปยัง Gateway
- ใช้โปรแกรม Putty สำหรับ Remote Server หรือ SSH (Secure Shell) จากคอมพิวเตอร์ของผู้ใช้ไปยัง Gateway
- ใช้โปรแกรม LOGO! Soft Comfort สำหรับตั้งค่าเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ (PLC Siemens LOGO Controller) ในการควบคุมอุปกรณ์สั่งการ

- ใช้ Services ต่างๆของ AWS ได้แก่

  - AWS Lambda เพื่อใช้เป็นบริการประมวลผลแบบไร้เซิร์ฟเวอร์
  - Amazon DynamoDB เพื่อเป็นฐานข้อมูล
  - Amazon API Gateway เพื่อสร้างRESTful API
  - AWS IoT Core เพื่อช่วยให้สามารถเชื่อมต่ออุปกรณ์กับบริการของ AWS
  - AWS Cognito เพื่อใช้ในการทำ authentication
  - AWS CloudWatch เพื่อในการตรวจสอบทรัพยากรใน AWS
  - AWS IAM ใช้เพื่อกำหนดการเข้าถึง สิทธิต่างๆ
  - AWS CloudFormation เพื่อให้สามารถสร้างกลุ่มของทรัพยากรของ AWS
  - AWS Amplify เพื่อทำงานร่วมกับ Cognito ในการยืนยันตัวตน

1.6 ขั้นตอนการดำเนินงาน

No	การดำเนินงาน	2022							2023		
		มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค
1	เสนอโครงงาน	———									
2	ศึกษาปัญหา ค้นคว้าวิจัยและเก็บรวบรวมข้อมูล	———	———	———							
3	วิเคราะห์และออกแบบระบบ		———	———	———	———					
4	ออกแบบฐานข้อมูล		———	———	———	———					
5	ออกแบบการประมวลผลบนคลาวด์ Amazon Web Services		———	———	———	———					
6	ออกแบบระบบการทำงานของอุปกรณ์			———	———	———					
7	ออกแบบเว็บแอปพลิเคชันและอินเตอร์เฟสผู้ใช้งาน			———	———	———					
8	ดำเนินการทำโครงงาน				———	———	———	———	———		
9	ฐานข้อมูล				———	———	———				
10	การประมวลผลบนคลาวด์ Amazon Web Services					———	———	———			
11	ระบบการทำงานของอุปกรณ์						———	———			
12	เว็บแอปพลิเคชันและอินเตอร์เฟสผู้ใช้งาน							———	———		
13	ทดสอบระบบ หาปัญหาและวิธีการแก้ไขปัญหา					———					

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้

2.1 การพัฒนา Web application

2.1.1 คำจำกัดความของ Web application

2.1.1.1 เว็บไซต์

เว็บไซต์ (website) คือ หน้าเว็บเพจหลายหน้า ซึ่งเชื่อมโยงกันผ่านทางไฮเปอร์ลิงก์ ส่วนใหญ่จัดทำขึ้นเพื่อนำเสนอข้อมูลผ่านคอมพิวเตอร์ ซึ่งต้องเปิดด้วยโปรแกรมเฉพาะทางที่เรียกว่า Web Browser โดยถูกจัดเก็บไว้ในเวิลด์ไวด์เว็บ โดยเว็บไซต์นั้นถูกสร้างขึ้นมาด้วยภาษาทางคอมพิวเตอร์ที่เรียกว่า HTML (Hyper Text Markup Language) อีกทั้งยังได้มีการพัฒนาและนำภาษาอื่นๆเข้ามาร่วมด้วย เพื่อให้มีความสามารถมากขึ้น เช่น PHP, SQL, JavaScript เป็นต้น

2.1.1.2 Application

Application คือ เป็นโปรแกรมที่ช่วยอำนวยความสะดวกในด้านต่างๆ ออกแบบมาสำหรับอุปกรณ์สื่อสารเคลื่อนที่ต่างๆ โดยแอปพลิเคชัน (Application) จะต้องมีสิ่งที่เรียกว่า ส่วนติดต่อกับผู้ใช้ (User Interface หรือ UI) เพื่อเป็นตัวกลางการใช้งานต่างๆ

2.1.1.3 Web Application

Web Application (เว็บแอปพลิเคชัน) คือ Application ที่ถูกเขียนขึ้นมาเพื่อให้ทำงานในBrowserได้โดยตรง ไม่ต้องโหลด Application แบบเต็มๆ ลงเครื่องทำให้ประหยัดทรัพยากรในการประมวลผล ทำให้โหลดหน้าเว็บไซต์ได้เร็วขึ้น อีกทั้งผู้ใช้งานยังสามารถใช้งานผ่านอินเตอร์เน็ตที่ความเร็วต่ำได้

2.1.2 โครงสร้างของ Web application

โครงสร้างของ Web application จะมีส่วนประกอบการทำงานหลักๆ ที่เห็นกันได้ 4 ส่วน

1) Web Application **:** ตัว Web Application ที่ทำหน้าที่เป็นด่านแรกสุดในการรับข้อมูลจากฝั่งผู้ใช้งาน ซึ่งจะมีการสร้างหรือดัดแปลงการใช้งานไปได้หลากหลายทาง
2) Web Browser : คือเครื่องมือในการเปิด Web Application ซึ่งมีหลากหลายตัวเลือก เช่น Google Chrome Firefox หรือ Microsoft Edge เป็นต้น ซึ่งในปัจจุบันสามารถทำงานได้ทั้งบนคอมพิวเตอร์และสมาร์ทโฟน
3) Web Server : ระบบ Server ที่ให้บริการแก่บรรดาเว็บไซต์และเว็บแอปพลิเคชันต่างๆ ทำหน้าที่รับส่งข้อมูลจากฝั่งผู้ใช้งานและฝั่ง Web Application
4) Database : ฐานข้อมูลจากฝั่งผู้ให้บริการ ซึ่งจะทำหน้าที่เก็บข้อมูลที่จำเป็น

เราสามารถแยกส่วนประกอบของการทำงานของ Web application ออกได้เป็น 2 ส่วนหลักๆ คือ

1) เทคโนโลยีฝั่งผู้ใช้งาน (client-side technology) ประกอบไปด้วย 3 ส่วนหลักๆ คือ
   - เว็บเบราว์เซอร์ (web browser) เป็นซอฟต์แวร์ที่ผู้ใช้งานใช้ในการเข้าถึงเว็บแอปพลิเคชัน
   - ส่วนต่อความสามารถเว็บและเบราว์เซอร์ (web plugin และ browser add-on/extension) คือโปรแกรมที่ถูกเขียนให้ทำงานร่วมกับเว็บเบราว์เซอร์
   - ระบบปฏิบัติการ (operating system) คือระบบปฏิบัติการทำหน้าที่ในการจัดการกับทรัพยากรของเครื่องคอมพิวเตอร์
2) เทคโนโลยีฝั่งเซิร์ฟเวอร์ (server–side technology) ประกอบไปด้วย 3 ส่วนหลักๆ คือ
   - เว็บแอปพลิเคชัน (web application) เว็บแอปพลิเคชัน ถือว่าเป็นหัวใจหลักของเว็บไซต์เนื่องจากทำหน้าที่ติดต่อกับผู้ใช้งาน รับและแสดงข้อมูล ประมวลผลข้อมูล จัดการข้อมูลในฐานข้อมูล และอื่นๆ
   - เว็บเซิร์ฟเวอร์ซอฟต์แวร์ (web server software) เป็นโปรแกรมที่ทำงานอยู่บน web server ซึ่งหน้าที่หลักของ web server software คือการประมวลผล HTTP request ที่ได้รับมาและตอบกลับด้วย HTTP response ให้กับผู้ใช้งาน
   - ระบบปฏิบัติการ (operating system) ระบบปฏิบัติการบนฝั่งของเซิร์ฟเวอร์มีหน้าที่ในการจัดการกับทรัพยากรของเครื่องเซิร์ฟเวอร์ เช่น CPU memory และ bandwidth เป็นต้น

โดยมีหลักการทำงานดังนี้

ผู้ใช้งานจะใช้งาน Web Browser เพื่อทำการใช้งาน Web Application ซึ่งตัวเว็บแอปจะทำการดึงข้อมูลที่จำเป็นต้องใช้งานผ่าน Web Server และทาง Web server ก็อาจมีการดึงข้อมูลที่จำเป็นของ Database อีกทีหนึ่ง

2.1.3 เครื่องมือที่ใช้

2.1.3.1 next.js

Next.js คือ Framework ใช้สำหรับการสร้างเว็บไซต์ หรือเว็บแอปพลิเคชัน ที่สามารถใช้งานได้ง่าย มีความยืดหยุ่น สามารถปรับแต่งได้อย่างมาก และเป็น React Web Framework คล้ายๆ กับ Create React App ที่ช่วยให้เราเขียนเว็บได้สะดวกขึ้น เพราะเค้า Setup และ Config อะไรหลายๆ อย่างให้เราเรียบร้อยแล้ว โดยภาษาหลักที่ใช้ในการเขียนคือ JavaScript, html, CSS ซึ่งใน Next เราจะสามารถเขียน html แทรกใน JavaScript หรือ JavaScript แทรกใน html โดยการใช้งานแบบนี้จะเรียกว่า JSX

2.2 การพัฒนาอุปกรณ์เพื่อจัดการฟาร์ม

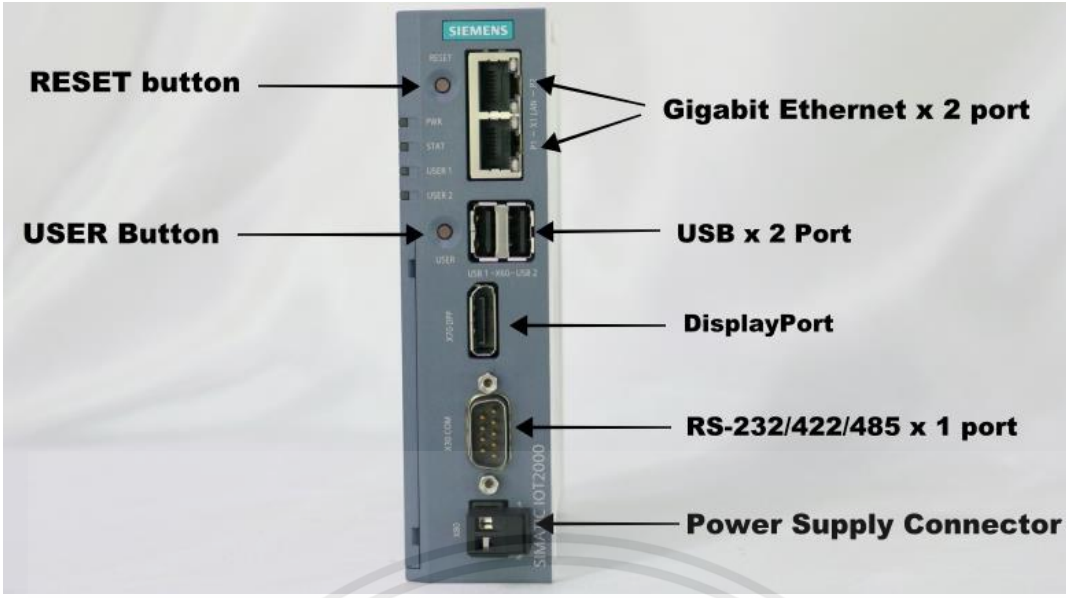
2.2.1 เกตเวย์

เกตเวย์ (Gateway) คือ อุปกรณ์ฮาร์ดแวร์ที่เชื่อมต่อเครือข่ายต่างๆเข้าด้วยกัน โดยมีความสามารถสูงในเชื่อมต่อเครือข่ายที่ใช้โปรโตคอลต่างกัน และใช้สื่อส่งข้อมูลต่างชนิดกันได้อย่างไม่มีขีดจำกัด เช่น การใช้เกตเวย์ในการเชื่อมต่อเครือข่ายที่เป็นคอมพิวเตอร์ประเภทพีซี (PC) เข้ากับคอมพิวเตอร์ประเภทแมคอินทอช (MAC) หรือการเชื่อมต่ออีเทอร์เน็ตแลน (Ethernet LAN) ที่ใช้สายส่งแบบยูทีพี (UTP) เข้ากับโทคเคน ริง แลน (Token Ring LAN) เป็นต้น

2.2.1.1 SIMATIC IoT2050 gateways

SIMATIC IoT2050 เป็น Open industrial IoT gateway ที่เจาะจงไปที่สนับสนุนนักพัฒนารุ่นใหม่ที่ต้องการเข้าใจ IoT (Industrial internet of things) โดยขยายจาก protype พัฒนาด้วย opensource จากบอร์ดอย่าง Arduino, Intel Galileo หรือ Raspberry PI เพื่อนำไปสู่การใช้งานจริง

IoT2050 ไม่ใช่ PLC (Programable Logic controller) มันถูกออกแบบมาเพื่อเป็นสมองของทุกๆ IoT application เช่น เก็บข้อมูลจาก PLC หรือเซนเซอร์จากจุดไหนก็ได้ในโลก โดยต่อฮาร์ดแวร์เพิ่มเติมเป็น Arduino shield ได้ และยังถูกออกแบบมาให้คุยกับอุปกรณ์ตัวอื่นในอุตสาหกรรมผ่าน industrial protocol เช่น MODBUS, PROFINET และใช้โปรโตคอลด้านอินเตอร์เน็ตอย่าง MQTT ได้



รูปที่ 2.1 พอร์ตการเชื่อมต่อของ SIMATIC IoT2050 gateways

ตารางที่ 2.1 เปรียบเทียบความสามารถในการทำงาน SIMATIC IoT gateways ในแต่ละเวอร์ชั่น

	IOT2040	IOT2050 (Basic)	IOT2050 (Advance)
CPU	Intel Quark x1020(Single Core)	TI SOC AM6528 GP Dual Core	TI SOC AM6548 HS Quad Core with High Security possibility
RAM	1 GB DDR3-SDRAM	1 GB RAM DDR4	2 GB RAM DDR4
eMMC	No	No	16GB
OS	Yocto Linux	SIMATIC Industrial OS Based Debian (Excluded)	SIMATIC Industrial OS Based Debian (Pre-installed)
Ethernet interface	2 x 10/100 Mbit/s Ethernet RJ45	2 x Gbit Ethernet RJ45	2 x Gbit Ethernet RJ45
serial interface	2 x RS232/422/485	1 x RS232/422/485	1 x RS232/422/485
USB port	1x USB 2.0, 1x USB client	2 x USB 2.0	2 x USB 2.0
Video Port	No	DisplayPort	DisplayPort
Battery-buffered real-time clock	Yes	No	Yes
Material	Plastic	Stainless Steel, Aluminum, Plastic	Stainless Steel, Aluminum, Plastic
IP degree of protection	IP20	IP20	IP20
Input voltage	DC 9 to 36 V	DC 12 to 24 V	DC 12 to 24 V
Operating Temperature	0 to 50°C	0 to 50°C	0 to 50°C
Program : Arduino sketches	Supported	Not Supported	Not Supported
Mounting	Dinrail, Wallmount (Optional)	Dinrail, Wallmount	Dinrail, Wallmount

2.2.1.2 ระบบปฏิบัติการลีนุกซ์บนเกตเวย์

ระบบปฏิบัติการ (Operating System) คือ กลุ่มโปรแกรมที่จัดระเบียบเพื่อทำหน้าที่ควบคุมการทำงานของระบบ และเสริมการทำงานในส่วนของอุปกรณ์ โดยใช้เป็นตัวเชื่อมโยงระหว่างเครื่องคอมพิวเตอร์และผู้ใช้ รวมถึงการจัดสรรทรัพยากรต่างๆ ในระบบให้มีประสิทธิผลที่ดี

ระบบปฏิบัติการลีนุกซ์ (Linux) คือ ระบบปฏิบัติการเช่นเดียวกับ ดอส ไมโครซอฟต์วินโดวส์หรือยูนิกซ์ โดยลีนุกซ์นั้นจัดว่าเป็นระบบปฏิบัติการยูนิกซ์ประเภทหนึ่ง ความสามารถของตัวระบบปฏิบัติการและโปรแกรมประยุกต์ที่ทำงานบนระบบลีนุกซ์ โดยเฉพาะอย่างยิ่งโปรแกรมในตระกูลของ GNU (GNU's Not UNIX) และสิ่งที่สำคัญที่สุดก็คือระบบลีนุกซ์เป็นระบบปฏิบัติการประเภทฟรีแวร์ (Free Ware) คือไม่เสียค่าใช้จ่ายในการซื้อโปรแกรม ถึงแม้ว่าในขณะนี้ลีนุกซ์ยังไม่สามารถแทนที่ไมโครซอฟต์ วินโดวส์ บนพีซีหรือแมคโอเอส (Mac OS) ได้ทั้งหมดก็ตาม แต่ผู้ใช้จำนวนไม่น้อยที่หันมาใช้และช่วยพัฒนาโปรแกรมประยุกต์บนลีนุกซ์กัน และเรื่องของการดูแลระบบลีนุกซ์นั้น ภายในระบบลีนุกซ์เองมีเครื่องมือช่วยสำหรับดำเนินการให้สะดวกยิ่งขึ้น

2.2.2 อุปกรณ์ต่างๆภายในฟาร์มและเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้

2.2.2.1 อุปกรณ์วัดความชื้นในดิน

SOIL MOISTURE SENSOR OUTPUT RS485 เซ็นเซอร์วัดความชื้นในดิน

รายละเอียดของอุปกรณ์วัดความชื้นในดิน

- ทำมาจากวัสดุ ABS + 316 สแตนเลส
- แหล่งจ่ายไฟ : 12-24V DC
- ช่วงความชื้นของดิน: 0-100%
- ความแม่นยำในการวัดความชื้น: 0 - 53% คือ ± 3%
  53 - 100% คือ ±5%
- เวลาตอบสนอง: น้อยกว่า 1 วินาที

รูปที่ 2.2 อุปกรณ์วัดความชื้นในดิน

2.2.2.2 อุปกรณ์วัดธาตุสารอาหารในดิน

Soil NPK Sensor NPK sensor MODBUS RTU485 เซ็นเซอร์วัดความอุดมสมบูรณ์ของไนโตรเจน, ฟอสฟอรัสและโพแทสเซียมในดินแบบครบจบในโพรบเดียว

รายละเอียดของอุปกรณ์วัดธาตุสารอาหารในดิน

- แหล่งจ่ายไฟ: 5-30VDC
- การใช้พลังงานสูงสุด: ≤0.15W
- อุณหภูมิในการทำงาน: -40 ถึง 80 องศาเซลเซียส
- พารามิเตอร์ NPK
- ช่วง: 1-1999 มก./กก.(มก./ลิตร)
- ความละเอียด: 1 มก./กก.(มก./ลิตร)
- ความแม่นยำ: ±2%FS
- เวลาตอบสนอง: ≤1S
- เกรดการป้องกัน: IP68
- วัสดุโพรบ: สแตนเลส 316
- วัสดุปิดผนึก: อีพอกซีเรซินทนไฟสีดำ
- ความยาวสายเคเบิลเริ่มต้น: 2 เมตร ความยาวสายเคเบิลสามารถปรับแต่งได้
- ขนาด: 45*15*123 มม.

รูปที่ 2.3 อุปกรณ์วัดธาตุสารอาหารในดิน

2.2.2.3 เครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้

SIEMENS Logic Module LOGO! 8 เครื่องควบคุมขนาดเล็กกระทัดรัดที่สามารถโปรแกรมได้ พร้อมฟังก์ชันใหม่ให้คุณเชื่อมต่อได้สะดวกขึ้นกับพอร์ท Ethernet ซึ่งรองรับการสื่อสารผ่านระบบ LAN LOGO!8 ถูกออกแบบมาให้เหมาะกับงานออโตเมชั่นขนาดเล็ก สามารถนำไปประยุกต์ใช้ได้กับหลากหลายงานไม่ว่าจะเป็นงานควบคุมปั๊มน้ำ งานตู้คอนโทรล งานควบคุมระบบไฟในอาคาร บ้านเรือน เป็นต้น

คุณสมบัติเด่น

- หน้าจอแสดงผล พร้อมปุ่มกดอเนกประสงค์ สามารถใช้ตั้งโปรแกรม แก้ไขโปรแกรม ปรับเปลี่ยนตั้งค่าต่างๆได้ง่าย ตามต้องการ เช่น ค่าของเวลา ค่าของจำนวนนับ เป็นต้น ใช้ในการดูตรวจสอบค่าต่างๆ และแสดงข้อความเตือน พร้อมสีของไฟ Backlight ที่เปลี่ยนสีได้ 3 สี (ขาว ,ส้ม ,แดง) ตามต้องการ
- การโปรแกรมสามารถทำได้ง่ายๆ ด้วยคอมพิวเตอร์ ติดตั้งซอฟแวร์ LOGO! Soft COMFORT หรือโปรแกรมผ่านหน้าจอและปุ่มกดบนหน้าปัดเครื่อง
- เชื่อมต่อกับ อุปกรณ์อื่นๆในระบบ SIMATIC S7 และจอแสดงผล TDEได้ง่ายๆด้วยระบบ LAN
- ทำ Web server ได้ง่ายมาก สามารถดูค่าระบบการทำงาน และสั่งการผ่านระบบอินเทอร์เน็ตผ่านทางแท็บเล็ทและโทรศัพท์มือถือได้
- มีให้เลือก ทั้งรุ่นที่มีหน้าจอ และไม่มีหน้าจอ

รูปที่ 2.4 เครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ของ SIEMENS Logic Module LOGO! 8

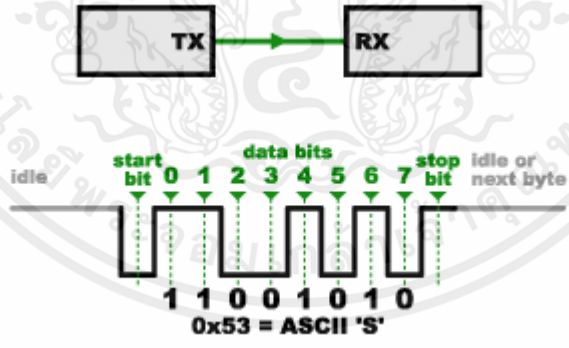
2.2.3 การสื่อสารระหว่างอุปกรณ์ต่างๆกับเกตเวย์

การสื่อสารระหว่างอุปกรณ์ต่างๆกับเกตเวย์เป็นการสื่อสารแบบอนุกรม (Serial Communication) โดยสื่อที่รับ-ส่งข้อมูลที่ละ 1 บิต แทนการสื่อสารแบบขนาน (parallel communication) ที่รับ-ส่งข้อมูลพร้อมกันในทุกๆบิต ซึ่งมีข้อดี คือ ใช้สายที่น้อยกว่าการสื่อสารแบบขนาน แต่มีข้อเสียที่จะใช้เวลามากกว่าการสื่อสารแบบขนาน โดยการสื่อสารมี 3 รูปแบบ คือ

1) Simplex การสื่อสารทางเดียว ต้องเลือกระหว่างการสื่อสารไปหรือกลับ
2) Half-duplex การสื่อสาร 2 ทาง แต่ไม่สามารถสื่อสารพร้อมกันได้ ต้องสลับกันการสื่อสารไปและกลับ
3) Full-duplex การเป็นการสื่อสาร 2 ทางที่สามารถสื่อสารพร้อมกันได้ทั้งไปและกลับ

สำหรับการสื่อสารแบบอนุกรมมี 2 ลักษณะ คือ

1) Synchronous เป็นการรับ-ส่งข้อมูลเป็นบล็อกครั้งละหลายๆไบต์ ใช้สัญญาณนาฬิกาเป็นตัวช่วยในการทำงานของตัวรับ-ส่งให้สอดคล้องกัน
2) Asynchronous เป็นการรับ-ส่งข้อมูลที่ละ 1 ไบต์ 8 บิต โดยรูปแบบ Stream bit มีความหมายดังนี้
   - Start bit บอกจุดเริ่มต้นของข้อมูล มีขนาด 1 บิต
   - Data bit แทนค่าข้อมูล มีขนาดได้ 5 ถึง 8 บิต
   - Parity bit บิตสำหรับใช้ตรวจสอบความผิดพลาดของข้อมูล มีขนาดได้ 0 ถึง 1 บิต
   - Stop bit บิตบอกจุดสิ้นสุดข้อมูล มีขนาดได้ 1, 1.5 และ 2 บิต

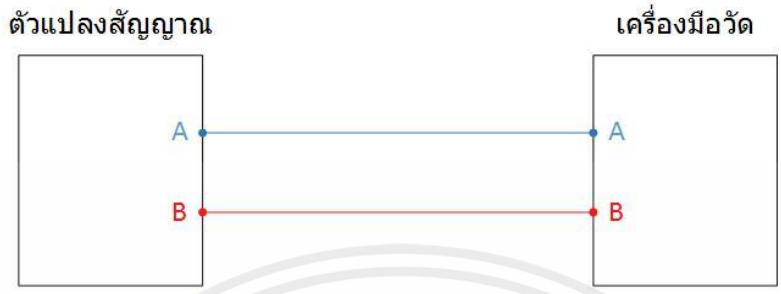


รูปที่ 2.5 ความหมายของ Stream bit

โดยระยะเวลาการส่งข้อมูลและบิตขึ้นอยู่กับการใช้งาน เป็นจำนวน bit/second โดยมีค่ามาตรฐานอยู่หลายช่วง เช่น 2400 4800 9600 เป็นต้น

2.2.3.1 การสื่อสารด้วยมาตรฐาน RS485

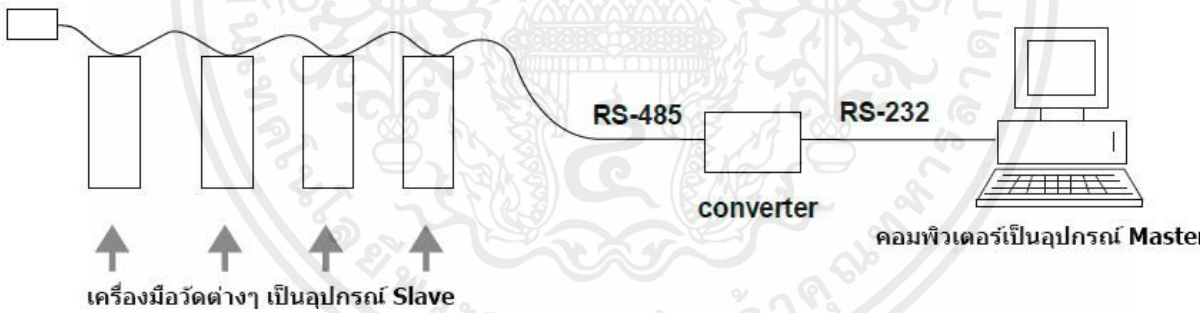
Recommended Standard no. 485 เป็นมาตรฐานการรับ-ส่งข้อมูลแบบ Half-duplex สำหรับการรับ-ส่งข้อมูลจะใช้สายเพียงแค่ 2 เส้นคือ A และ B เป็นตัวบอกสัญญาณดิจิตอล โดยใช้ค่าความต่างของแรงดันระหว่าง A และ B



รูปที่ 2.6 การเชื่อมต่อ RS485 ระหว่างเครื่องมือวัดกับตัวแปลงสัญญาณ

- เมื่อแรงดัน Va – Vb ได้แรงดันน้อยกว่า -200 mV คือสัญญาณดิจิตอล 1
- เมื่อแรงดัน Va – Vb ได้แรงดันมากกว่า +200 mV คือสัญญาณดิจิตอล 0

การทำงานแบบระบบ network จะเชื่อมต่อกันได้มากสุดถึง 32 ตัว โดยมี 1 ตัวทำหน้าที่จัดการลำดับการสื่อสาร เรียกว่า Master ตัวอุกรณ์ที่เหลือ เรียกว่า Slave โดยทุกอุปกรณ์จะมี ID ของตัวเอง โดย master จะทำการส่งคำสั่งพร้อม ID ไปให้ Slave ทุกตัว และถ้าคำสั่งนั้นมี ID ตรงกับตัวเอง slave นั้นจะทำงานตามคำสั่งนั้น



รูปที่ 2.7 การทำงาน RS485 แบบ Network

ข้อดีของ RS485

- สามารถส่งสัญญาณได้ไกลสุดถึง 1200 เมตร และเป็นระยะที่เพียงพอกาใช้ในระบบอุตสาหกรรม
- สามารถเชื่อมต่อเป็นเครือข่าย (Network) แบบ Multipoint ได้สูงสุดถึง 32 ตัว
- เป็นมาตรฐานที่ใช้สายไฟเพียง 2 เส้นในการรับส่งข้อมูล

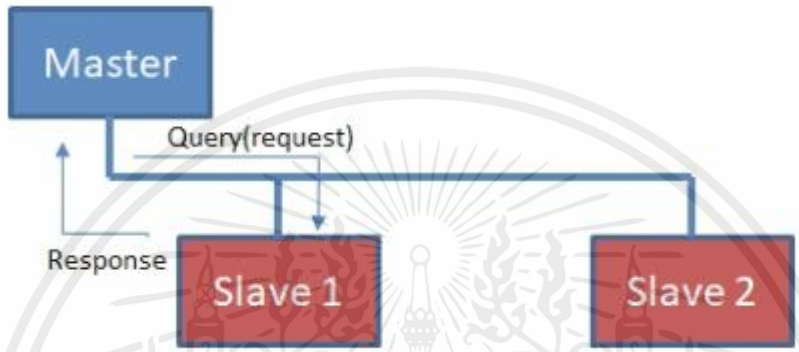
ข้อเสียของ RS485

- ต้องใช้ตัวแปลงสัญญาณในการเชื่อมต่อกับคอมพิวเตอร์หรือเกตเวย์
- ความเร็วในการรับ-ส่งข้อมูลน้อย

2.2.3.2 การสื่อสารด้วยมาตรฐาน Modbus

Modbus คือ โปรโทคอลที่ใช้การสื่อสารแบบอนุกรม ด้วยการสื่อสารแบบ Master-Slave (หรือ Client-Server) จะมีอุปกรณ์เพียงตัวเดียวซึ่งคือ 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

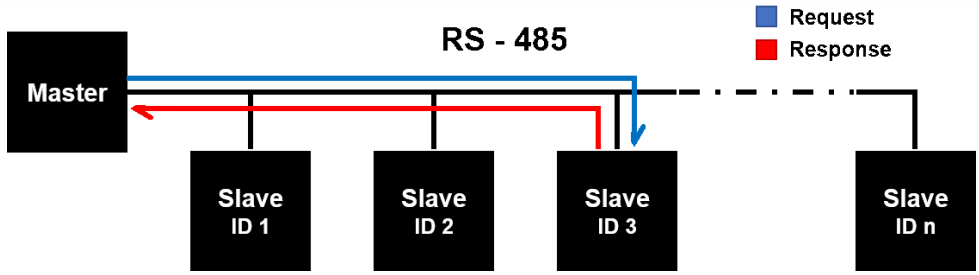


รูปที่ 2.8 การทำงานและการเชื่อมต่อของ Modbus

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 ส่งกลับมาจะประกอบด้วยคำสั่งที่สั่งให้กระทำข้อมูลต่างๆ และ Checksum

2.2.3.2.1 การสื่อสารด้วยมาตรฐาน Modbus RTU

Modbus RTU คือ โปรโทคอลที่ใช้การสื่อสารแบบอนุกรม (Serial-based Protocol) ด้วยสถาปัตยกรรมการสื่อสารแบบ Master/Slave หรืออาจกล่าวได้ว่าอุปกรณ์ Slave จะไม่ส่งข้อมูล (Response) กลับมาจนกว่าจะมีการร้องขอ (Request) จากอุปกรณ์ Master



รูปที่ 2.9 การทำงานและการเชื่อมต่อของ Modbus RTU

Modbus RTU โดยทั่วไปจะใช้กับการสื่อสาร แบบ RS-232 หรือ RS-485 ข้อมูลในโปรโทคอล Modbus จะถูกเก็บ 4 รูปแบบ คือ

1) Output coils ตำแหน่ง address จะเริ่มต้นที่ 000001
2) Input contacts ตำแหน่ง address จะเริ่มต้นที่ 100001
3) Input registers ตำแหน่ง address จะเริ่มต้นที่ 300001
4) Holding registers ตำแหน่ง address จะเริ่มต้นที่ 400001

โดย Output coils และ Input contacts แต่ละ address จะเก็บค่าเพียง 1 บิต หรือมีค่าได้แค่ “0” กับ “1” เปรียบเสมือนค่าการเปิดและปิดของอุปกรณ์รีเลย์และสวิตช์ที่พบได้ในระบบงานอัตโนมัติอุตสาหกรรม

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

ตารางที่ 2.2 ตำแหน่ง address ใน Modbus RTU โดยแบ่งตามรูปแบบการทำงาน

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

โดย Modbus RTU จะรับส่งเป็นชุดข้อมูล โดยที่ใน 1 ชุดข้อมูลนั้นจะประกอบด้วยส่วน 6 ส่วน ตามตารางที่ 2.1

ตารางที่ 2.3 ส่วนประกอบของการรับ-ส่ง Modbus RTU

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

Function code สามารถแบ่งหน้าที่ได้ตามรหัส มีฟังก์ชันการทำงานอยู่ 2 แบบ คือ การอ่าน (Read) และเขียน (Write) ไปยัง Coils, Contacts หรือ Registers ตามตารางที่ 2.3

ตารางที่ 2.4 Function code ของการรับ-ส่ง RS485 Modbus RTU

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

ในส่วนชุดข้อมูล Data Field นั้นจะถูกแบ่งเป็น 2 ชุด ได้แก่

1) ชุดคำสั่งสำหรับการอ่าน (Read Command)

ตารางที่ 2.5 ชุดคำสั่งสำหรับการอ่าน (Read Command)

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)

2) ชุดคำสั่งสำหรับการเขียน (Write Command)

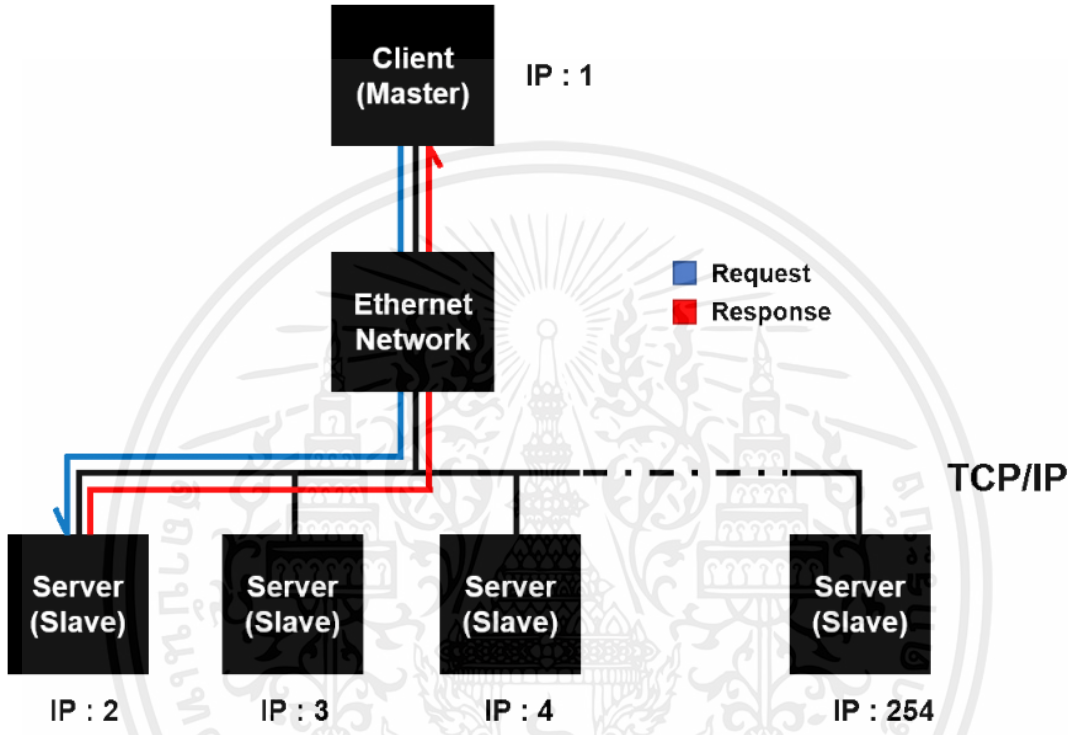
ตารางที่ 2.6 ชุดคำสั่งสำหรับการเขียน (Write Command)

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)

โดยชุดคำสั่งทั้ง 2 จะถูกส่งจากอุปกรณ์ที่ทำหน้าที่เป็น Master เท่านั้น เพื่อสั่งการไปยังอุปกรณ์ Slave ที่ต้องการสื่อสาร

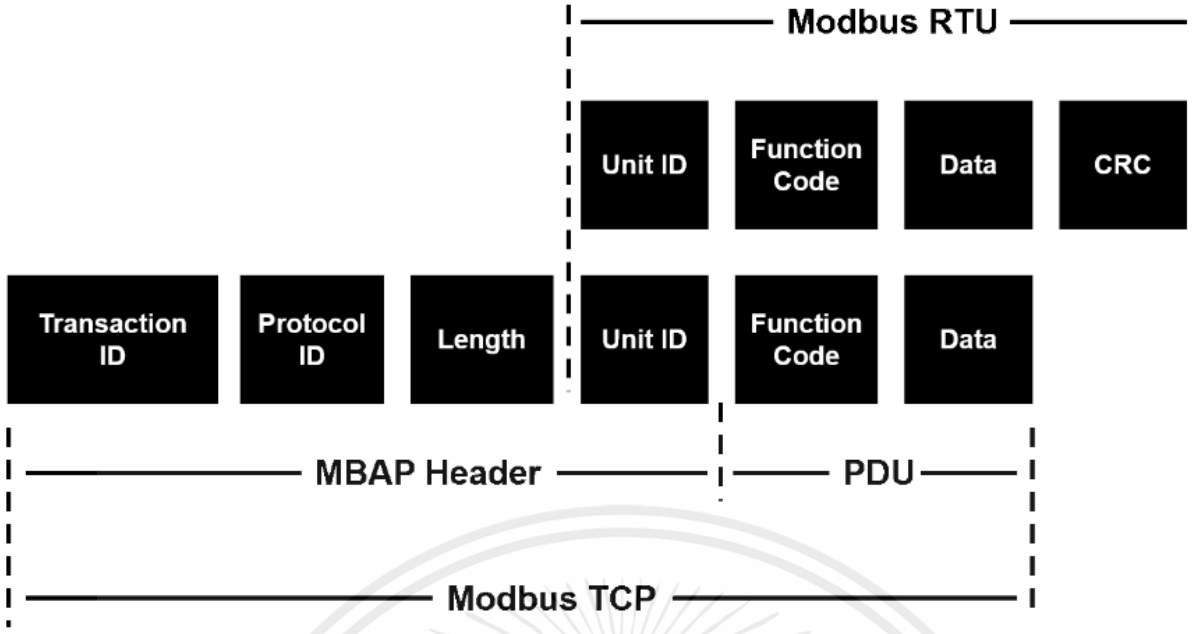
2.2.3.2.2 การสื่อสารด้วยมาตรฐาน Modbus TCP/IP

Modbus TCP คือ โปรโทคอลที่ครอบ Modbus RTU เพื่อใช้การสื่อสารแบบอีเทอร์เน็ต (Ethernet-based protocol) ด้วย TCP/IP (Transmission control protocol) ที่พอร์ต (Port) 502 แทนการใช้การสื่อสารแบบอนุกรม ทำให้อุปกรณ์สามารถสร้างการสื่อสารผ่านเครือข่ายเฉพาะบริเวณ (Local area network: LAN) หรือ เครือข่ายอินเตอร์เน็ต (Internet network) รวมไปถึงการเชื่อมต่อแบบไร้สาย (Wireless) โดยมีอุปกรณ์กระจายสัญญาณ (Router หรือ Access point) เป็นตัวกลางในการเชื่อมต่อดังรูป



รูปที่ 2.10 การทำงานและการเชื่อมต่อของ Modbus TCP/IP

ชุดข้อความใน Modbus TCP (รายละเอียดดังรูปที่ 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 2.11 ส่วนประกอบชุดข้อมูลของ Modbus TCP เทียบกับ Modbus RTU

ตารางที่ 2.6 รายละเอียดของแต่ละ Field ในหนึ่งเฟรมของ Modbus TCP

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.

2.2.4 เครื่องมือที่ใช้ในการพัฒนา

2.2.4.1 node-red

Node-RED เป็นเครื่องมือจัดการและจัดการเหตุการณ์ขึ้นอยู่กับ Node.js แอปพลิเคชัน Node-RED มักทำงานเป็นเว็บเซิร์ฟเวอร์และผู้ใช้สามารถปรับแต่งและจัดการการเชื่อมต่อระหว่างฮาร์ดแวร์ต่างๆและสร้างขั้นตอนการทำงานจากเบราว์เซอร์ของคอมพิวเตอร์เครื่องใดก็ได้ แต่เป็นเรื่องง่ายและทำงานในเว็บเบราว์เซอร์ แอปพลิเคชันเซิร์ฟเวอร์นี้ยังมีความสามารถในการทำงานบนอุปกรณ์ เช่น Gateway

ฟังก์ชันทั้งหมดของ Node-RED ทำงานกับโค้ดที่มีอยู่แล้ว อินเทอร์เฟซผู้ใช้ของ Node-RED ดูเรียบง่ายรูปแบบ Flow Base เหมาะสำหรับงานด้าน IOT สามารถเขียนโปรแกรมบน Web Browser ได้สะดวกลดการเขียน Code โดย Node-Red มี Node Service ต่างๆให้เราเลือกนำมาตั้งค่าและใช้ได้ทันที และเปิดเผยคุณแทบไม่มีปัญหาในการพัฒนาโครงการ IoT ด้วย Node-RED ใน GitHub แสดงใน JSON (JavaScript Object Notation) และสามารถส่งออกไปยังคลิปบอร์ดได้อย่างง่ายดายหรือสามารถนำเข้าสู่ Node-RED หรือแชร์ทางออนไลน์

2.2.4.2 Modbus Poll

Modbus Poll เป็นโปรแกรมสำหรับใช้ติดต่อสื่อสารระหว่างคอมพิวเตอร์ PC กับอุปกรณ์ที่ใช้การสื่อสารผ่าน Modbus RTU ทางพอร์ตสื่อสารแบบ RS485 เป็นซอฟต์แวร์จำลองต้นแบบ Modbus (Modbus master simulator) ที่ออกแบบมาเพื่อช่วยนักพัฒนาอุปกรณ์ Modbus slave หรืออื่น ๆ ที่ต้องการทดสอบและจำลองโปรโตคอล Modbus เป็นหลัก ด้วยอินเทอร์เฟซเอกสารที่หลากหลาย สามารถตรวจสอบ Modbus slaves และพื้นที่ข้อมูลหลาย ๆ ตัวได้ในเวลาเดียวกัน สำหรับแต่ละหน้าต่างคุณเพียงแค่ระบุ Modbus slave ID, function, address, size และ poll rate คุณสามารถอ่านและเขียนรีจิสเตอร์และคอยส์จากหน้าต่างใดก็ได้ หากคุณต้องการเปลี่ยนรีจิสเตอร์เดียว เพียงดับเบิลคลิกที่ค่า หรือคุณสามารถเปลี่ยนรีจิสเตอร์ / คอยส์หลายรายการ มีรูปแบบข้อมูลหลายรูปแบบเช่น float, double และ long พร้อมการสลับลำดับคำ สามารถใช้ทดสอบ อ่าน เขียน ข้อมูลบน Protocol โดยไม่ต้องเสียเวลาเขียนโปรแกรมเอง และฟังก์ชันอื่นๆ อีกมากมาย

2.2.4.3 LOGO! Soft Comfort V8.3

LOGO! Soft Comfort V8.3 เป็นโปรแกรมที่ใช้ในการเขียนโปรแกรมควบคุมเครื่องขนาดเล็กที่สามารถโปรแกรมได้ ที่ใช้งานง่ายและจำลองโปรเจคท์ต่างๆ ตลอดจนความสามารถด้านการจัดการเอกสารซึ่งเป็นที่นิยมของเหล่าผู้ใช้ LOGO! และได้มีการเพิ่มฟีเจอร์ใหม่ๆเข้ามา ประกอบด้วยการทำงานแบบเรียบง่ายในโหมดเครือข่าย การปรับตั้งค่าการสื่อสารโดยอัตโนมัติด้วยการแสดงผลในมุมมองเครือข่าย และความสามารถในการเปิดโปรแกรมพร้อมกันได้ถึง 3 โปรแกรม

นอกจากนี้ ผู้ใช้ยังสามารถลากและวางวัตถุได้อย่างง่ายดายในการถ่ายทอดสัญญาณจากโปรแกรมหนึ่งไปสู่อีกโปรแกรมหนึ่ง Soft Comfort V8.3 ยังช่วยให้การย้ายระบบจากโปรแกรมรุ่นก่อนๆเป็นไปอย่างง่ายดาย

2.2.4.4 WinSCP

WinSCP คือ โปรแกรมสำเร็จรูปที่ใช้สำหรับอัพโหลดและดาวน์โหลดไฟล์ผ่าน Protocol FTP จากคอมพิวเตอร์ของผู้ใช้ไปยัง Server

2.2.4.5 PuTTY

Putty เป็นโปรแกรม Remote Server หรือ SSH (Secure Shell) พูดให้เข้าใจง่ายๆก็คือ เราสามารถใช้โปรแกรมนี้ในการ สั่งงาน Server ด้วย command line โดยส่วนใหญ่แล้วจะใช้เชื่อมต่อไปยัง server ที่เป็น linux รองรับการเชื่อมต่อหลากหลายรูปแบบดังนี้

- Raw
- Telnet
- Rlogin
- SSH
- Serial

2.3 Internet of Things

ระบบไอโอที (Internet of Things) คือ ระบบที่เชื่อมโยงอุปกรณ์ อิเลกทรอนิกซ์ต่างๆเข้าไว้ด้วยกัน เช่น เซนเซอร์ต่างๆ โดยผ่านเครือข่ายอินเตอร์เน็ต การเชื่อมโยงนี้จะ ทำให้อุปกรณ์ในเครือข่ายสามารถสื่อสารกันได้ และยังทำให้ผู้ใช้สั่งการควบคุมอุปกรณ์ได้ ซึ่งข้อมูลที่เก็บมาได้จากอุปกรณ์จะนำมาเก็บไว้บนพื้นที่ เช่น บนคลาวด์ เป็นต้น และสามารถนำข้อมูลที่เก็บมาได้เหล่านั้นเพื่อไปวิเคราะห์ในภายหลัง การใช้ระบบไอโอทียังช่วยในเรื่องความแม่นยำของข้อมูล และความเป็นปัจจุบัน (Real-time) ทำให้ผู้ใช้สามารถมองเห็นถึงปัญหาได้ตลอดเวลา แต่ด้วยจำนวนข้อมูลที่มีจำนวนมากจึงมีโอกาสเสี่ยงในเรื่องของความปลอดภัยด้วยเช่นกัน

2.3.1 Protocol

โปรโตคอลคือข้อกำหนดในการสื่อสารกันระหว่างอุปกรณ์ต่างๆเพื่อให้เกิดการสื่อสารระหว่างกัน เปรียบเสมือนกับภาษาที่ใช้ในการสื่อสารที่อุปกรณ์สามารถนำไปใช้ในการติดต่อกันได้ซึ่งในโปรโตคอลแต่ละชนิดก็มีจุดประสงค์ที่ทำงานแตกต่างกันไปในโปรเจคนี้ได้เน้นใช้โปรโตคอลสำหรับการแลกเปลี่ยนข้อมูล ได้แก่ MQTT และ HTTP/REST

2.3.1.1 MQTT

MQTT (ย่อมาจาก MQ Telemetry Transport) เป็นโปรโตคอลที่ใช้ในการส่งข้อความ ซึ่งถูกใช้บ่อยในอุปกรณ์ประเภทไอโอที โปรโตคอลนี้ขับเคลื่อนด้วยเหตุการณ์ และ เชื่อมต่ออุปกรณ์ผ่านรูปแบบของการติดตาม-เผยแพร่ (Subscribe-Publish) ผู้ส่งและผู้รับจะสื่อสารกันผ่านหัวข้อที่กำหนดไว้ การเชื่อมต่อระหว่างผู้ใช้งานจะต้องผ่านคนกลางคือ ผู้ให้บริการ MQTT คนกลางนี้จะมีหน้าที่รับข้อความที่เข้ามาจากหัวข้อต่างๆ และกระจายไปยังผู้ที่ติดตามหัวข้อนั้นๆ

2.3.1.2 HTTP/REST

HTTP (Hypertext Transfer Protocol) เป็นโปรโตคอลชั้นแอปพลิเคชันสำหรับการส่ง เอกสารไฮเปอร์มีเดีย เช่น HTML โปรโตคอล HTTP นี้ถูกนำไปใช้ในการสื่อสารกันระหว่างเว็บเบรา เซอร์ของผู้ใช้กับทรัพยากรบนอินเทอเน็ต HTTP เป็นโปรโตคอลที่ไร้สถานะ หมายถึง แต่ละคำขอจะ ไม่ถูกเก็บไว้ในแต่ละครั้งที่ใช้และเป็นอิสระต่อกัน

ตารางที่ 2.5 คำขอที่ใช้ใน HTTP

Methods	Operations	Response
GET	ร้องขอข้อมูลที่แสดงผลในทรัพยากรนั้น	ข้อมูลที่ร้องขอ, Status Code
HEAD	เหมือนกันกับ GET แต่ไม่มีข้อมูล	ความเป็นมาของข้อมูล Status Code
POST	ส่งเอนทิทีไปยังทรัพยากรที่ระบุ ทำให้เกิด การเปลี่ยนแปลงในสถานะ	Status code
PUT	แทนที่การแสดงผลปัจจุบันของเป้าหมาย ด้วย payload ของคำร้องขอ	Status code
DELETE	ลบทรัพยากรที่ระบุ	Status code
CONNECT	สร้างช่องทางไปยังเซิร์ฟเวอร์จากผู้รับด้วยคำร้องขอ	Status code
OPTIONS	อธิบายตัวเลือกช่องทางการสื่อสารของ ทรัพยากรเป้าหมาย	หัวเรื่องที่บ่งชี้ถึงตัวเลือกที่ สามารถปรับใช้กับทรัพยากร เป้าหมายได้
TRACE	ทดสอบเส้นทางการเดินทางของข้อมูลที่จะ เดินทางไปยังทรัพยากรเป้าหมาย	ค่า Max-forward ของ 0 ใน คำร้องขอ
PATCH	นำส่วนที่ถูกปรับเล็กน้อยมาใช้กับทรัพยากร	Status code

REST (Representational State Transfer) เป็นสถาปัตยกรรมซอฟแวร์ที่กำหนดเงื่อนไขการทำงานของ API ว่าต้องทำงานอย่างไร โดยมีหลักดังต่อไปนี้ 1. อินเตอร์เฟสเป็นรูปแบบเดียวกันมีข้อจำกัด 4 ประการคือ คำขอควรระบุทรัพยากร ผู้ใช้มีข้อมูลเพียงพอในทรัพยากรที่จะแก้ไขหรือลบทรัพยากร ผู้ใช้ได้รับข้อมูลเกี่ยวกับวิธีการประมวลผลข้อมูลเพิ่มเติม และผู้ใช้ได้รับข้อมูลเกี่ยวกับทรัพยากรอื่นๆที่จำเป็นสำหรับการทำงานให้สมบูรณ์ 2. ความไร้สถานะ เซิร์ฟเวอร์สามารถดำเนินการตามคำขอได้ทั้งหมดโดยแต่ละคำขอจะไม่ถูกเก็บ

และไม่ขึ้นกับคำขอก่อนหน้า 3. ระบบที่แบ่งออกเป็นชั้น ผู้ใช้สามารถเชื่อมต่อกับตัวกลางอื่นๆได้ ที่ถูกยอมรับโดยทั้งผู้ใช้และเซิร์ฟเวอร์ และยังคงรอการตอบสนองจากเซิร์ฟเวอร์ 4. ความสามารถในการแคช ทรัพยากรจะถูกเก็บไว้เมื่อตอบสนองครั้งแรกทำให้เวลาในการตอบสนองเร็วมากขึ้น

ตารางที่ 2.6 คำขอที่ใช้ใน REST

Methods	Operations	Response
POST	ส่งเอนทิทีไปยังทรัพยากรที่ระบุ ทำให้เกิดการเปลี่ยนแปลงในสถานะ	Status code
GET	ร้องขอข้อมูลที่แสดงผลในทรัพยากรนั้น	ข้อมูลที่ร้องขอ, Status Code
PUT	แทนที่การแสดงผลปัจจุบันของเป้าหมายด้วย payload ของคำร้องขอ	Status code
PATCH	นำส่วนที่ถูกปรับเล็กน้อยมาใช้กับทรัพยากร	Status code
DELETE	ลบทรัพยากรที่ระบุ	Status Code

ตารางที่ 2.7 โค้ดแสดงสถานะ (Status code)

Class	Range
Informational response	100 - 199
Successful response	200 – 299
Redirection response	300 – 399
Client error response	400 – 499
Server error response	500 - 599

2.3.2 Smart farm

สมาร์ทฟาร์ม คือ แนวคิดของระบบการจัดการที่มุ่งเน้นไปที่อุตสาหกรรมการเกษตรโดยการผสมผสานกับเทคโนโลยีขั้นสูงต่างๆ ได้แก่ Big Data, Cloud และระบบไอโอทีซึ่งเป็นแกนกลางหลักของแนวคิดนี้ เพื่อที่จะใช้ในการติดตาม การทำระบบอัตโนมัติ และกระบวนการการวิเคราะห์ต่างๆ

นอกจากนี้ยังช่วยลดทรัพยากรแรงงานให้น้อยลงและทำให้การควบคุมหรือค่าที่ได้จากการติดตามมีความแม่นยำขึ้นด้วย

2.4 Amazon Web Services

คลาวด์ (Cloud) เป็นการนำเครื่องเซิร์ฟเวอร์มาเชื่อมต่อทำงานร่วมกันเป็นกลุ่ม ในรูปแบบ Cluster ทำหน้าที่เป็นที่จัดเก็บข้อมูล การนำไปประมวลผล หรือ อื่นๆ โดยที่ผู้พัฒนาไม่ต้องมีตัวเครื่องเซิร์ฟเวอร์อยู่เลย กล่าวคือ การจำลองเซิร์ฟเวอร์ไว้บนเครื่องเซิร์ฟเวอร์อีกทีหนึ่ง ในงานนี้ ได้เลือกใช้ Firebase ในช่วงทดสอบ และปรับเปลี่ยนมาใช้ AWS

AWS (Amazon Web Services) เป็นแพลตฟอร์มคลาวด์ที่ให้บริการโดย Amazon ที่มีมากกว่า 200 บริการจากดาต้าเซนเตอร์รอบโลก เน้นการให้บริการออนไลน์ที่สามารถขยายระดับได้และมีแนวทางการประมวลผลของคลาวด์ที่คุ้มค่าต่อราคา บริการของ Amazon มีรายประเภท ได้แก่ บริการการประมวลผล ที่เก็บข้อมูล ฐานข้อมูล การทำเครือข่ายและการขนส่งข้อมูล เครื่องมือความปลอดภัย เครื่องมือในการพัฒนา และเครื่องมือในการจัดการ เป็นต้น ในโปรเจคนี้ ทางผู้จัดทำได้เลือกใช้เครื่องมือ ดังต่อไปนี้

· บริการการประมวลผล
    o Lambda / Serverless
· Internet of Things
    o IoT Core
· Front End Web & Mobile
    o API Gateway
· ฐานข้อมูล
    o DynamoDB
· เครื่องมือความปลอดภัย
    o IAM (Identity Access Management)
    o Cognito
· เครื่องการจัดการ
    o CloudWatch
    o CloudFormation

2.4.1 AWS Lambda / Serverless

AWS Lambda คือการรันโค้ดโดยที่ไม่ต้องใช้เซิร์ฟเวอร์มาจัดการ ซึ่งสามารถทำงานได้แบบอัตโนมัติและสามารถขยายระดับการทำงานได้ ทำให้ค่าใช้จ่ายลดลงเหลือแค่ช่วงที่โค้ดถูกรันเท่านั้น โค้ดเหล่านี้ที่เรียกว่า “Lambda Function” หรือ “Serverless Function” ประกอบไปด้วยฟังก์ชันหลายๆฟังก์ชันที่จะถูกเรียกใช้งานผ่านเหตุการณ์ต่างๆ เช่น การเรียกใช้งานผ่าน REST API หรือการอัพเดทข้อมูลในฐานข้อมูล เป็นต้น

2.4.2 AWS IoT Core

AWS IoT Core เป็นบริการใน AWS Cloud ซึ่งทำหน้าที่ในการเชื่อมต่ออุปกรณ์ IoT เข้ากับ Cloud และที่อื่นๆ โดยมีการตั้งค่าการส่งข้อมูลต่างๆด้วยโปรโตคอล เช่น MQTT ทำให้อุปกรณ์ที่เชื่อมต่อสามารถรับข้อมูลหรือส่งผ่านมาด้วยหัวข้อต่างๆได้ แน่นอนว่าการส่งผ่านจะมีการใส่ Certificate ของอุปกรณ์มาก่อนที่จะเชื่อมต่อ ทำให้มั่นใจได้ว่า การรับ-ส่งข้อมูลจะเกิดขึ้นได้เฉพาะระหว่างอุปกรณ์ที่มีการยืนยันแล้วเท่านั้น

2.4.3 API Gateway

API Gateway เป็นบริการที่ทำให้แอปพลิเคชันต่างๆสามารถเข้าถึงข้อมูล หรือการทำงานจากบริการ backend ได้ ซึ่งเป็นการสื่อสารแบบ 2 ทาง โดยมีวิธีการใช้ 2 วิธี ได้แก่ แบบ REST และ WebSocket นอกจากนี้ยังสนับสนุนการทำคอนเทนเนอร์และการทำงานแบบไร้เซิร์ฟเวอร์ (Serverless) API Gateway สามารถรับมือกับการเรียกใช้พร้อมกันได้สูงสุดถึง 10000 ครั้ง ซึ่งรวมถึงการจัดการการอนุญาตให้เข้าถึงข้อมูล CORS (Cross-origin resource sharing) การตรวจดูค่า และ การจัดการเวอร์ชั่น

2.4.4 DynamoDB

DynamoDB เป็นฐานข้อมูลแบบไม่มีความสัมพันธ์ (NoSQL) ที่เก็บข้อมูลในรูปแบบของกุญแจ-ค่า โดยมีจุดเด่นที่ความยืดหยุ่นของรายละเอียดข้อมูลที่ไม่จำกัดจำนวนของรายละเอียด ทำให้ในแต่ละรายละเอียดของข้อมูลสามารมีหลายค่าได้ สามารถปรับขนาดได้อัตโนมัติตามจำนวนของข้อมูล และผู้ใช้ไม่ต้องทำการติดตั้งด้วยตัวเองเลย

2.4.5 IAM (Identity Access Management)

IAM ใช้เพื่อกำหนดการเข้าถึง สิทธิต่างๆของแต่ละผู้ใช้ในแต่ละบทบาท ซึ่งจะถูกนำไปใช้ในการทำระบบการยืนยันตัวตนเพื่อป้องกันการเข้าถึงที่ไม่น่าเชื่อถือ และไม่ให้ข้อมูลรั่วไหล การกำหนดสิทธิของแต่ละบทบาทกลุ่มหรือผู้ใช้จะถูกกำหนดโดยนโยบาย นโยบายเหล่านี้จะเป็นตัวกำหนดการเข้าถึงทรัพยากรและบริการบน AWS ซึ่งจะทำการประเมินผู้ใช้ทุกครั้งที่มีร้องขอการเข้าถึง ผู้ใช้ที่นโยบายเหล่านี้ติดไว้ที่บทบาทของตนจะสามารถเข้าถึงข้อมูลได้

2.4.6 Cognito

Cognito ให้บริการในเรื่องระบบยืนยันตัวตนในการเข้าสู่ระบบหรือการลงทะเบียนเข้าสู่ระบบเพื่อเข้าที่แอปพลิเคชันด้วยการใช้ชื่อผู้ใช้และรหัสผ่าน โดยระบบยังสนับสนุนการเข้าสู่ระบบด้วยวิธีอื่นๆ เช่น เฟซบุ๊ค หรือกูเกิ้ล เป็นต้น องค์ประกอบที่สำคัญของ Cognito มีอยู่ 2 อย่าง คือ user pool เป็นสมุดรายชื่อของผู้ใช้ให้บริการการเข้าสู่ระบบและลงทะเบียน และ identity pool เป็นผู้ให้ credential กับผู้ใช้เพื่อการเข้าถึงขอมูลและบริการต่างๆของ AWS และทำหน้าที่ยืนยันตัวตนของผู้ใช้

2.4.7 CloudWatch

CloudWatch เป็นบริการที่ใช้ในการตรวจสอบทรัพยากรใน AWS Cloud และแอปพลิเคชันที่ทำงานบน AWS การใช้ CloudWatch สามารถเก็บและสะสมไฟล์บันทึกเหตุการณ์ที่เกิดจากการสร้างภายในแอปพลิเคชันได้ซึ่งสามารถใช้ในการแก้ปัญหาถ้าหากแอปพลิเคชันทำงานผิดพลาด

2.4.8 CloudFormation

CloudFormation คือบริการที่ให้ผู้พัฒนาสามารถสร้างกลุ่มของทรัพยากรของ AWS และจากภายนอกมาจัดการให้เป็นกลุ่มเดียวกัน บริการนี้ถูกใช้งานในการติดตั้งปรับใช้ (deploy) ทำให้การติดตั้งสามารถทำได้อย่างรวดเร็ว และไม่มีข้อผิดพลาด และยังสามารถนำรูปแบบที่มีอยู่มาติดตั้งซ้ำได้ในพื้นที่ภูมิภาคอื่นๆ

2.4.9 AWS Amplify

AWS Amplify คือชุดเครื่องมือและบริการที่ใช้ในการพัฒนาแอปพลิเคชันหรือเว็บบน AWS โดยจะรวมไลบรารี องค์ประกอบของ UI และชุดคำสั่ง command line ที่ใช้เพื่อสร้าง backend ของแอปพลิเคชันไว้ นอกจากนี้ยังสามารถเชื่อมต่อกับบริการอื่นๆของ AWS เช่น Cognito, Lambda และ S3 ได้โดยไม่ต้องใช้โค้ดหลายบรรทัด ซึ่งลดเวลาของการติดตั้งและการจัดการบริการของ backend

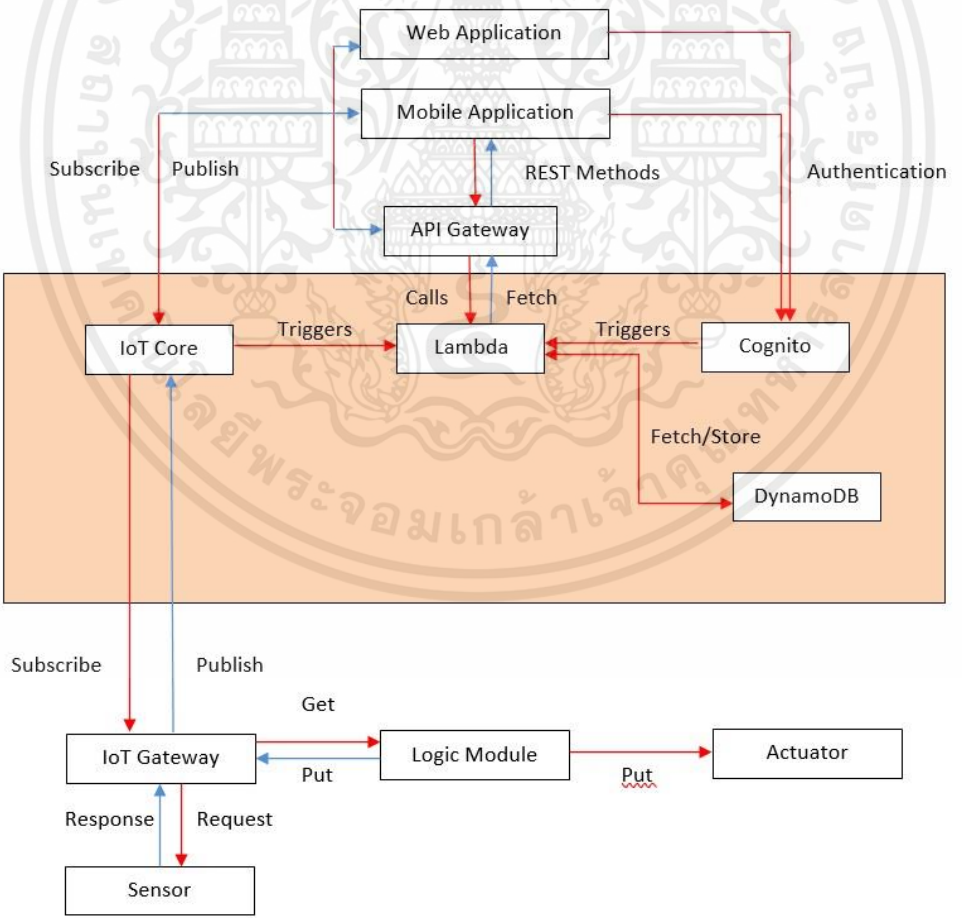
บทที่ 3

โครงสร้างของระบบและการออกแบบ

3.1 การออกแบบภาพรวม

ภาพรวมของระบบที่ออกแบบประกอบไปด้วย 3 องค์ประกอบ คือ แอปพลิเคชันบนสมาร์ทโฟน เว็บแอปพลิเคชัน AWS และ ส่วนของฮาร์ดแวร์ ในโครงงานนี้จะเน้นไปที่ส่วนของเว็บแอปพลิเคชัน AWS และ ส่วนของฮาร์ดแวร์ โดยมีส่วนของการ ทำงานได้แก่ การยืนยันตัวตน การรับ-ส่งข้อมูล การแก้ไขข้อมูลลงในฐานข้อมูล

โดยเว็บแอปพลิเคชันเชื่อมต่อบริการต่างๆของ AWS ได้แก่ ใช้ Cognito ในการทำระบบยืนยันตัวตนและเรียกใช้ฟังก์ชัน Lambda ผ่าน API Gateway ซึ่งในแต่ละบริการของ AWS จะเชื่อมต่อกันผ่าน Lambda ซึ่งทำหน้าที่เป็นตัวกลางเปรียบเสมือนเป็น เซิร์ฟเวอร์ โดยบริการ Lambda จะเชื่อมต่อกับ DynamoDB ในการที่จะดึงและแก้ไขข้อมูลในฐานข้อมูล



รูปที่ 3.1 ภาพรวมโครงสร้างของระบบการทำงาน

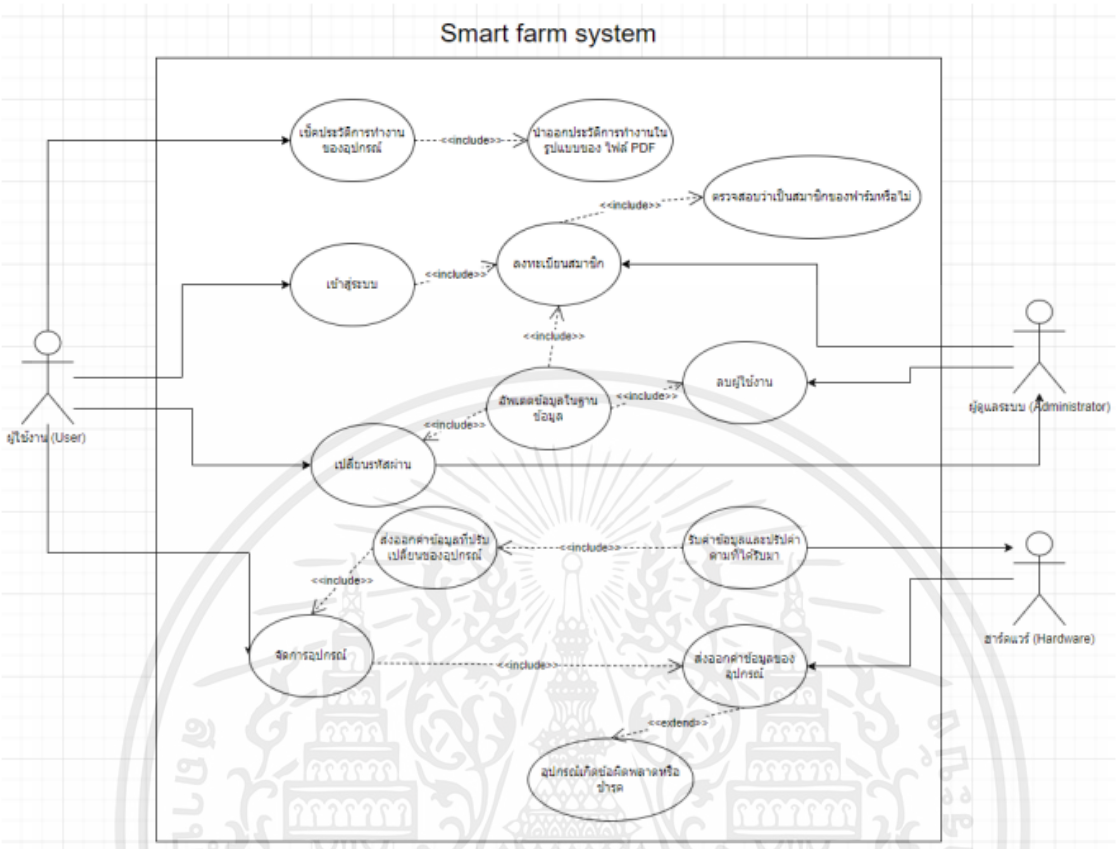
3.2 การออกแบบการใช้งานระบบโดยรวม

การออกแบบแอปพลิเคชัน Smart Farm ประกอบด้วย 3 องค์ประกอบที่ทำงานร่วมกัน ดังต่อไปนี้

1) ผู้ใช้งาน (User) ใช้งานบนแอปพลิเคชันบนสมาร์ทโฟน
2) ผู้ดูแลระบบ (Administrator) ใช้งานบนเว็บแอปพลิเคชัน
3) ฮาร์ดแวร์ (Hardware)

ทางผู้จัดทำใช้ AWS Cloud ในการรับและส่งข้อมูลของทั้ง 3 องค์ประกอบ โดยจะมีหลักการ คือ ผู้ดูแลระบบ (Administrator) ทำหน้าที่ในการเพิ่มหรือลบ ผู้ใช้งาน (User) เข้าไปในฟาร์มโดย การกำหนด บทบาท (role) ให้ หลังจากที่ ผู้ใช้งาน (User) สามารถเข้าใช้งานได้แล้ว จะมีหน้าที่ใน การจัดการ ฮาร์ดแวร์ (Hardware) ต่างๆที่อยู่ภายในฟาร์มที่ได้รับมอบหมายให้ดูแล และ ฮาร์ดแวร์ (Hardware) มีหน้าที่ในการส่งค่าข้อมูลเข้ามาในแอปพลิเคชันเพื่อให้ ผู้ใช้งาน (User) จัดการต่อไป ฟังก์ชันต่างๆที่สามารถใช้งานได้มีดังนี้

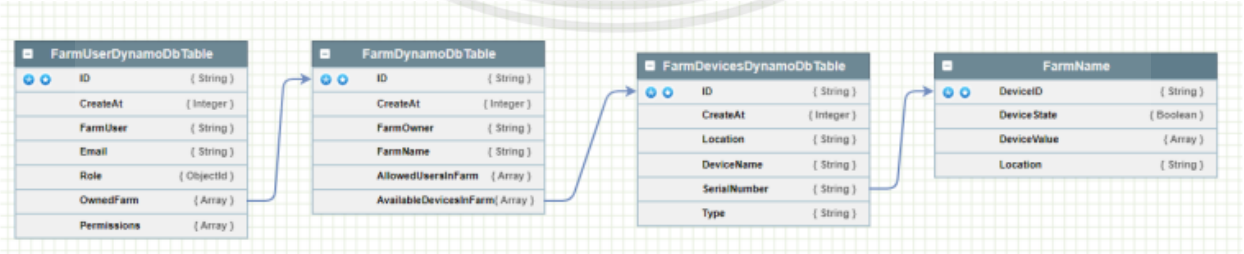
1) ผู้ใช้งาน (User)
   - ดูข้อมูลของ ฮาร์ดแวร์ แสดงผลได้ 2 วิธีคือ รูปแบบตัวเลข และ รูปแบบกราฟ
   - สามารถจัดการค่าข้อมูลต่างๆของ ฮาร์ดแวร์ ผ่านแอปพลิเคชัน
   - สามารถเช็คประวัติการทำงานของ ฮาร์ดแวร์ ว่าเกิดข้อผิดพลาดขึ้นหรือป่าว
   - สามารถนำออกประวัติการทำงาน หรือ ค่าของ ฮาร์ดแวร์ ในรูปแบบของไฟล์ PDF
2) ผู้ดูแลระบบ (Administrator)
   - เพิ่มหรือลบผู้ใช้งานออกจากการใช้แอปพลิเคชันได้
   - กำหนดบทบาทให้กับผู้ใช้งานในการดูแลฟาร์ม
   - จัดการอุปกรณ์และฟาร์มทั้งหมดได้
   - สามารถแก้ไขข้อมูลต่างๆในฐานข้อมูลได้
3) ฮาร์ดแวร์ (Hardware)
   - รับและส่งข้อมูลระหว่างแอปพลิเคชันและเว็บแอปพลิเคชัน



รูปที่ 3.2 แผนภาพกรณีการใช้งานใช้งานระบบโดบรวม

3.3 การออกแบบฐานข้อมูล

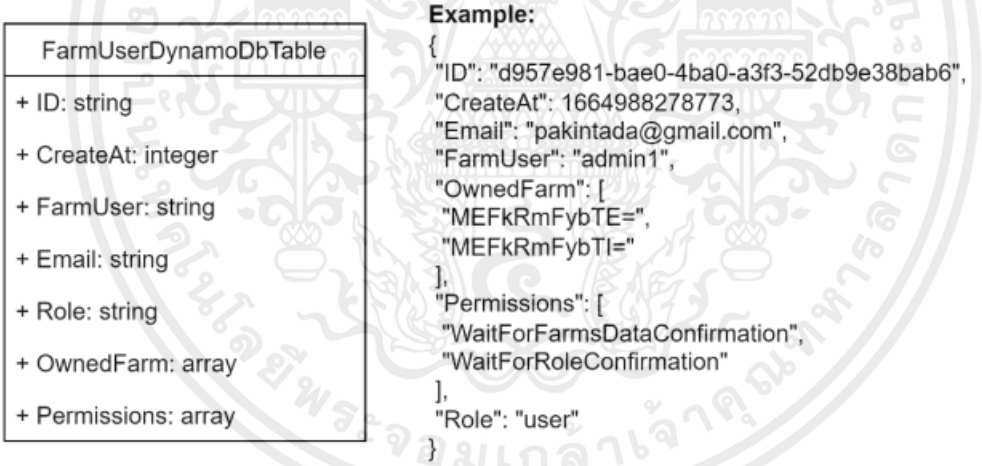
ฐานข้อมูลที่ได้ออกแบบไว้เป็น ฐานข้อมูลแบบ NoSQL ซึ่งไม่ได้มีการเชื่อมโยงกันของข้อมูล ระหว่างตาราง แต่จะเชื่อมโยงกันผ่านการทำงานของฟังก์ชัน ฐานข้อมูลมีอยู่ 2 ส่วนหลักๆ คือ ส่วนเอาไว้เก็บข้อมูลรวมของแต่ละอย่างซึ่งจะไม่เปลี่ยนแปลงบ่อย กับ ส่วนที่อัพเดทข้อมูลแบบตามเวลาจริง



รูปที่ 3.3 การเชื่อมต่อของฐานข้อมูล

ส่วนเอาไว้เก็บข้อมูลรวมของแต่ละอย่างซึ่งจะไม่เปลี่ยนแปลงบ่อยประกอบไปด้วย 3 ตาราง ได้แก่ FarmUser, Farm, FarmDevices

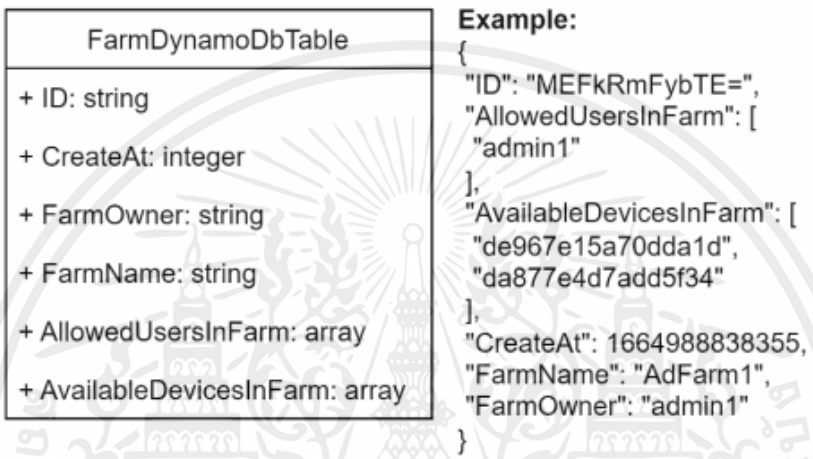
1) FarmUser จะมีกุญแจหลักเป็น ID โดยที่ ID จะถูกสร้างเป็นแบบ UUIDv4 ซึ่งเป็นการสุ่ม และมี ตัวแปรก 6 อย่างคือ
   - CreateAt บันทึกค่าเป็นตัวเลขซึ่งเป็นตัวบอกเวลาที่สร้างและบันทึกผู้ใช้โดยจะอยู่ในลักษณะ ของรูปแบบยูนิกซ์ - FarmUser เป็นชนิด String ใช้ในการบันทึกชื่อของผู้ใช้งาน
   - Email เป็นชนิด String ใช้ในการบันทึกอีเมลล์ของผู้ใช้
   - Role เป็นชนิด String ใช้ในการบ่งบอกถึงระดับของผู้ใช้ว่าอยู่ในระดับใด เช่น user, admin
   - OwnedFarm เป็น array ที่บันทึก String โดยแต่ละค่าจะเป็นค่าที่ถูกเข้ารหัสด้วยฐาน 64 ซึ่ง ค่าเหล่านี้เมื่อถอดรหัสออกมาแล้วจะได้เป็นชื่อของฟาร์มที่ผู้ใช้นี้เป็นเจ้าของอยู่
   - Permissions เป็น array ที่บันทึก String โดยจะบันทึกสิทธิการเข้าถึงของผู้ใช้ เช่น การเข้าถึง ข้อมูลผู้ใช้ การยืนยันระดับของผู้ใช้ เป็นต้น



รูปที่ 3.4 แอททริบิวต์ของตาราง FarmUser และตัวอย่างข้อมูลในรูปแบบ JSON

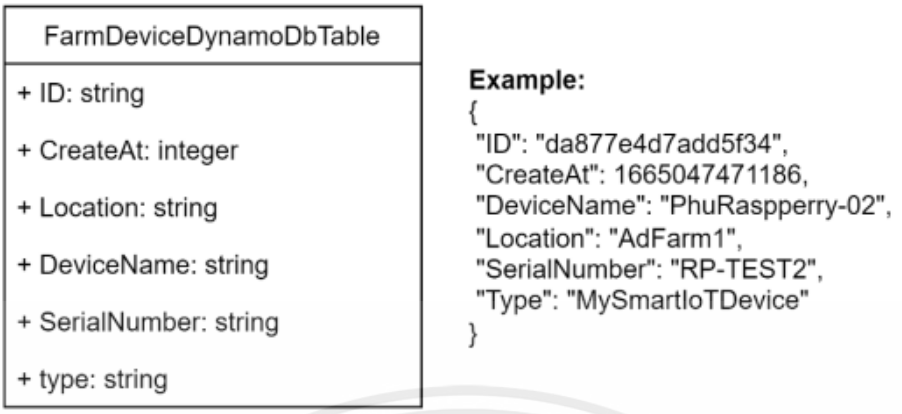
2) Farm จะมีกุญแจหลักเป็น ID โดยที่ ID จะถูกเข้ารหัสด้วยฐาน 64 ซึ่งในการเข้าถึงตารางนี้ได้ สามารถทำได้โดยการนำไอดีมาจาก OwnedFarm ของตาราง FarmUser ในตารางนี้มีตัวแปรอีก 5 ตัวแปรได้แก่
   - CreateAt บันทึกค่าเป็นตัวเลขซึ่งเป็นตัวบอกเวลาที่สร้างและบันทึกผู้ใช้โดยจะอยู่ในลักษณะของรูปแบบยูนิกซ์
   - FarmOwner เป็นชนิด String ใช้เพื่อบ่งบอกว่าใครเป็นเจ้าของฟาร์มนี้

- FarmName เป็นชนิด String บันทึกชื่อของฟาร์ม
- AllowedUsersInFarm เป็น array ที่บันทึก String ซึ่งเป็นชื่อของผู้ใช้ที่มีสิทธิเข้าถึงข้อมูลของฟาร์ม
- AvailableDevicesInFarm เป็น array ที่บันทึก String ของเลขฐาน 16 โดยแต่ละค่าเป็นค่าที่ ถูกสุ่มขึ้นมา โดยค่าเหล่านี้คือกุญแจหลักประจำตัวอุปกรณ์ ซึ่งบ่งบอกถึงอุปกรณ์ที่มีอยู่ในฟาร์มนั้นๆ



รูปที่ 3.5 แอททริบิวต์ของตาราง Farm และตัวอย่างข้อมูลในรูปแบบ JSON

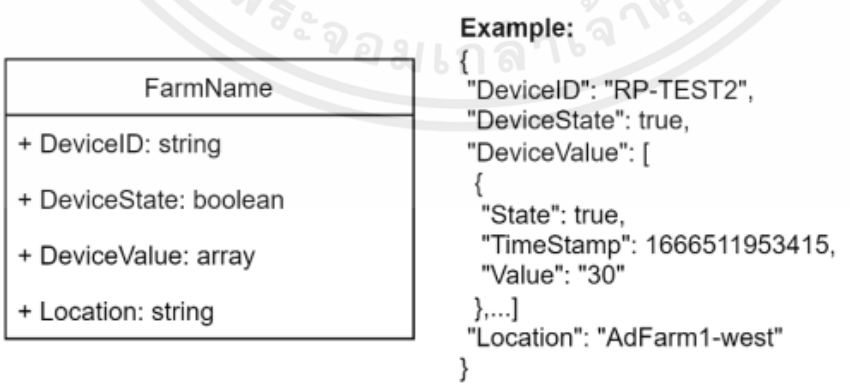
3) FarmDevice มีกุญแจหลักเป็น ID ซึ่งเป็นเลขฐาน 16 ซึ่งใช้ในการหาอุปกรณ์ภายในตารางนี้ เท่านั้นโดยสามารถนำมาจากตัวแปร AvailableDevicesInFarm ในตาราง Farm ซึ่งในตารางนี้มี ตัวแปรอีก 5 ตัวแปร ได้แก่
   - CreateAt บันทึกค่าเป็นตัวเลขซึ่งเป็นตัวบอกเวลาที่สร้างและบันทึกอุปกรณ์ โดยจะอยู่ใน ลักษณะของรูปแบบยูนิกซ์ - Location เป็นประเภท String ซึ่งบอกถึงตำแหน่งที่ตั้งของอุปกรณ์นี้ ณ ปัจจุบัน
   - DeviceName เป็นประเภท String ใช้ในการบันทึกชื่อของอุปกรณ์นั้นๆ
   - SerialNumber เป็นประเภท String ใช้ในการบันทึกหมายเลขประจำตัวของอุปกรณ์
   - Type เป็นประเภท String บันทึกประเภทของอุปกรณ์



รูปที่ 3.6 แอททริบิวต์ของตาราง FarmDevice และตัวอย่างข้อมูลในรูปแบบ JSON

ส่วนที่เปลี่ยนแปลงตามเวลาจริง จะใช้ชื่อตารางเป็นชื่อของฟาร์มนั้นๆ ซึ่งมีจุดประสงค์เพื่อการ บันทึกค่าจากเซนเซอร์ตามเวลาจริง มีกุญแจหลักเป็น DeviceID ซึ่งก็คือ SerialNumber จาก ตาราง FarmDevice และมีตัวแปรอีก 3 ตัวแปรได้แก่

- DeviceState เป็นประเภท boolean ทำหน้าที่บอกสถานะของอุปกรณ์นั้นๆว่าพร้อมในการส่ง ค่าหรือไม่ หากมีค่าเป็นจริง หมายถึงอุปกรณ์นั้นสามารถส่งค่าต่างๆรวมถึงค่าที่วัดได้ แต่ในทาง ตรงกันข้าม หากมีค่าเป็นเท็จ หมายถึงอุปกรณ์นั้นจะส่งค่าต่างๆยกเว้นค่าที่วัดได้
- DeviceValue เป็นประเภท array ซึ่งบันทึก json ที่ส่งมาจากอุปกรณ์โดยแต่ละ json ตะมี ลักษณะดังนี้
  - o {“State”:””, “TimeStamp”:””, “Value”:””}
- Location เป็นประเภท String ซึ่งบอกถึงตำแหน่งที่ตั้งของอุปกรณ์นี้ ณ ปัจจุบัน

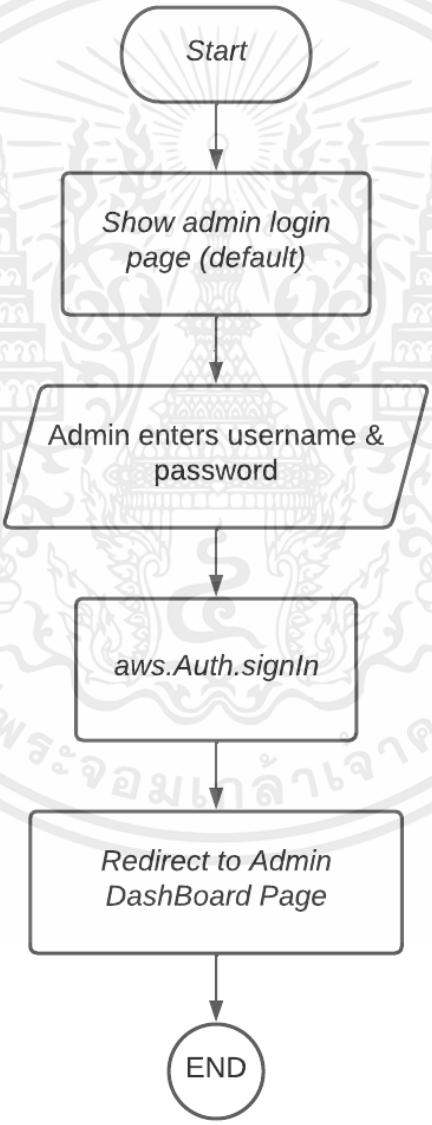


รูปที่ 3.7 แอททริบิวต์ของตาราง FarmName และตัวอย่างข้อมูลในรูปแบบ JSON

3.4 การออกแบบบริการต่างๆ

3.4.1 การเข้าสู่ระบบ

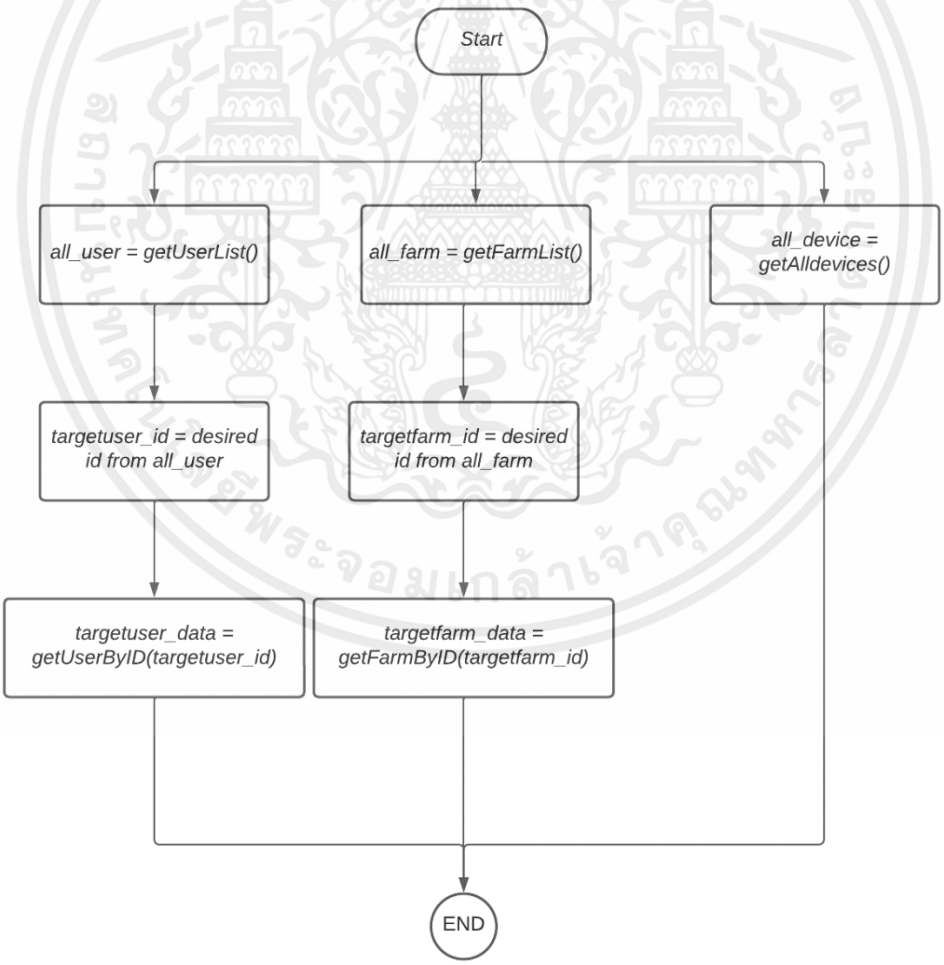
เมื่อผู้ใช้หน้าแอปพลิเคชันมาจะพบกับหน้า log in ซึ่งผู้ใช้จะต้องใส่ username และ password เพื่อเข้าสู่ระบบหลังจากที่ผู้ใช้งานใส่ username และ password ตัวเว็บแอปพลิเคชันจะมาเช็คใน Cognito ว่ามี username และ password ที่ส่งมาอยู่ใน user Pool ไหมถ้ามีก็จะเป็นอันว่าเข้าสู่ระบบเรียบร้อยและไปที่หน้าหลัก



รูปที่ 3.8 ผังงานของการเข้าสู่ระบบ

3.4.2 การแสดงข้อมูลผู้ใช้ ฟาร์ม และอุปกรณ์

ฟังก์ชันที่ใช้ในการดึงข้อมูลมีอยู่ 5 ฟังก์ชัน โดยจะเริ่มจากข้อมูลผู้ใช้จะใช้ getUserList ซึ่งเป็นการดึงข้อมูลในตาราง user ทั้งหมดนำมาแสดงในหน้าหลัก โดยในฟังก์ชันนี้ได้ออกแบบไว้ว่า ให้สามารถดึงได้แค่ 2 รายละเอียดคือ ไอดี UUIDv4 ที่เป็นกุญแจหลัก และชื่อ username ซึ่งหากต้องการดูรายละเอียดเพิ่มเติมของ user นั้นๆจะต้องนำไอดีของ user นั้นไปเรียกฟังก์ชัน API getUserByID เพื่อให้ได้รายละเอียดที่เหลือมา ต่อมาในข้อมูลของอุปกรณ์จะเรียกใช้งาน getAllDevices ซึ่งเป็นการดึงข้อมูลในตาราง Device ทั้งหมดนำมาแสดงในหน้าหลักและสุดท้ายข้อมูลฟาร์มจะใช้ getFarmList ซึ่งเป็นการดึงข้อมูลในตาราง Farm ทั้งหมดนำมาแสดงในหน้าหลัก โดยในฟังก์ชันนี้ได้ออกแบบไว้ว่า ให้สามารถดึงได้แค่ 2 รายละเอียดคือ ไอดี UUIDv4 ที่เป็นกุญแจหลัก และชื่อ FarmName ซึ่งหากต้องการดูรายละเอียดเพิ่มเติมของ farm นั้นๆจะต้องนำไอดีของ farm นั้นไปเรียกฟังก์ชัน getFarmByID เพื่อให้ได้รายละเอียดที่เหลือ



รูปที่ 3.9 การแสดงข้อมูลผู้ใช้ ฟาร์ม และอุปกรณ์

3.4.3 การสร้างข้อมูลในฟาร์มรวม อุปกรณ์และสร้างตารางฟาร์มย่อย

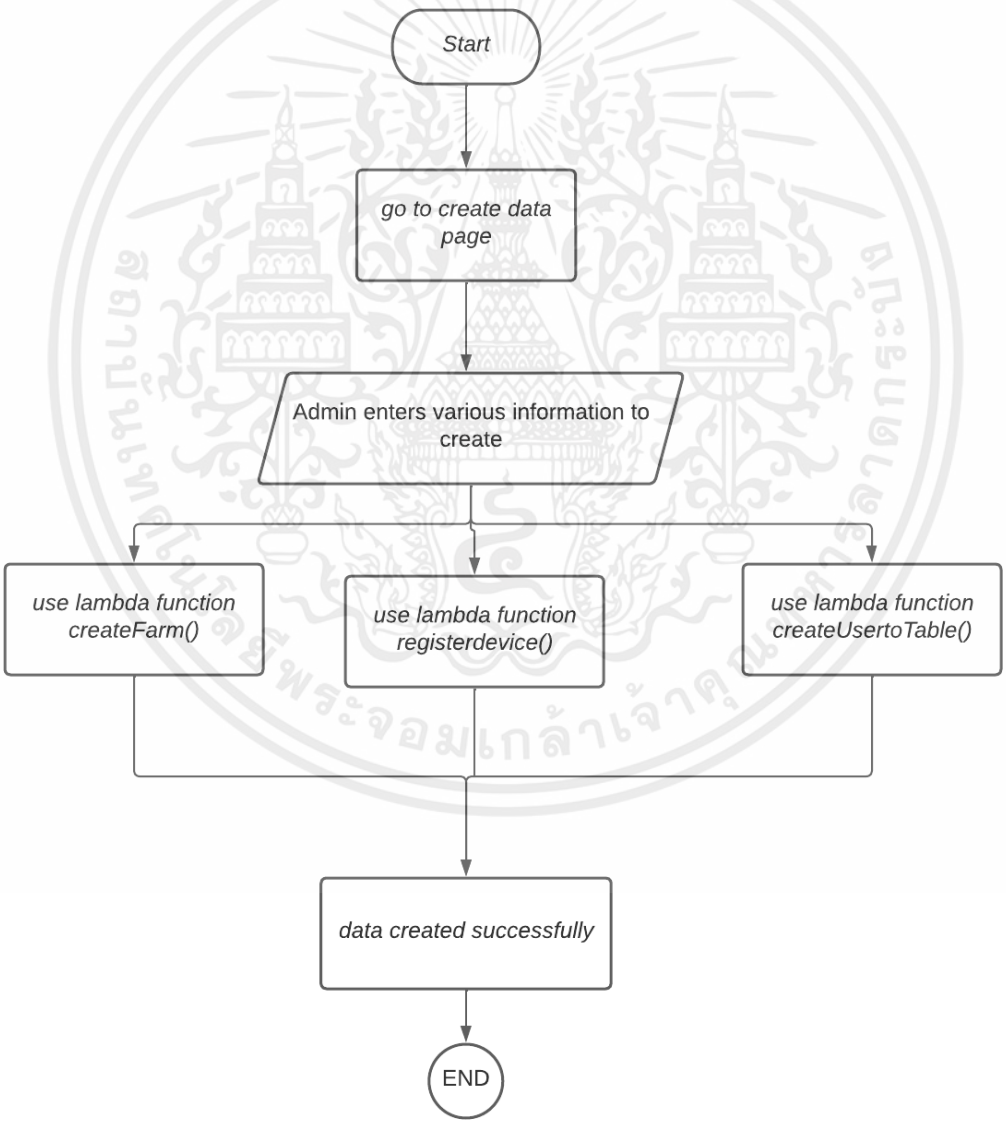
ฟังก์ชันที่ใช้ในการสร้างข้อมูลมีอยู่ 3 ฟังก์ชัน โดยจะเริ่มจากสร้างข้อมูลในตารางฟาร์มรวมซึ่งจะใช้ createFarm ซึ่งจะต้องส่งค่าข้อมูลไปเพื่อสร้างดังนี้

{"FarmName":"", "Owner":"", "AllowedUsers":["",""], "AvailableDevices":["",""]}

ต่อไปเป็นการสร้างข้อมูลในตารางอุปกรณ์ซึ่งจะใช้ registerDevice ซึ่งจะต้องส่งค่าข้อมูลไปเพื่อสร้างดังนี้

{"device":"", "type": "", "serialNumber":"", "farmName":""}

และสุดท้ายการสร้างสร้างตารางฟาร์มย่อยใหม่ซึ่งจะใช้ createUserToTable ซึ่งจะต้องส่งค่าข้อมูลไปเพื่อสร้างดังนี้ {"farmName":""}



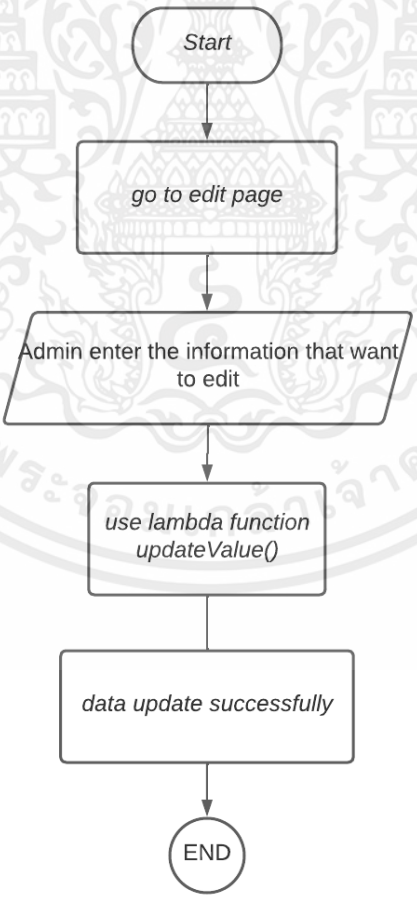
รูปที่ 3.10 การสร้างข้อมูลในฟาร์มรวม อุปกรณ์และสร้างตารางฟาร์มย่อย

3.4.4 การจัดการและแก้ไขข้อมูลต่างๆ

ฟังก์ชันที่ใช้ในการแก้ไขข้อมูลมีอยู่ 1 ฟังก์ชัน โดยจะเรียกใช้ฟังก์ชัน updateValue ซึ่งจะต้องส่งค่าข้อมูลไปเพื่อแก้ไขดังนี้

```
{
  "targetTable": "",
  "updateList": [
    {
      "ID": "",
      "UpdateItems": {
      }
    }
  ]
}
```

โดย targeTable คือตารางที่ต้องการแก้ไข updateList คือตัวข้อมูลที่ต้องการจะแก้ไขโดยมีการใช้ ID เพื่อระบุข้อมูลตัวที่ต้องการจะแก้ไข UpdateItems คือข้อมูลที่การจะแก้ไข ซึ่งข้อมูลทั้งหมดสามารถแก้ไขในหน้าแก้ไขข้อมูลของเว็บแอปพลิเคชัน



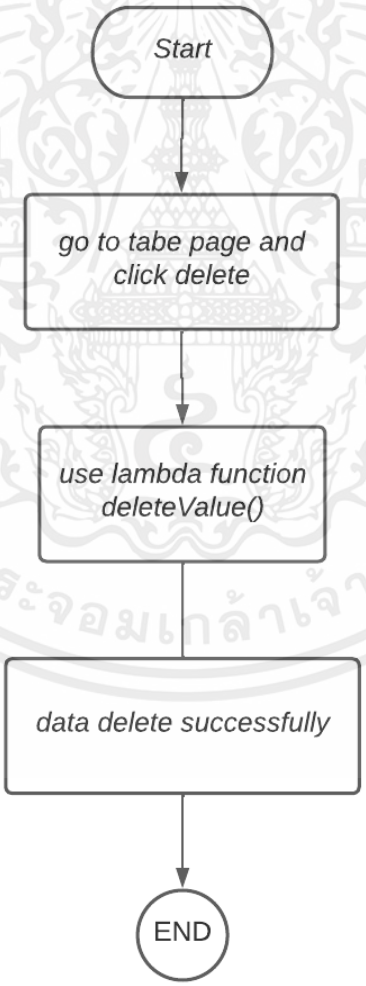
รูปที่ 3.11 ผังงานของการจัดการและแก้ไขข้อมูลต่างๆ

3.4.5 การลบข้อมูลในตารางต่างๆ

ฟังก์ชันที่ใช้ในการลบข้อมูลมีอยู่ 1 ฟังก์ชัน โดยจะเรียกใช้ฟังก์ชัน deleteValue ซึ่งจะต้องส่งค่าข้อมูลไปเพื่อลบดังนี้

```
{
  "targetTable":"",
  "deleteList": [
      {"ID":""}
  ]
}
```

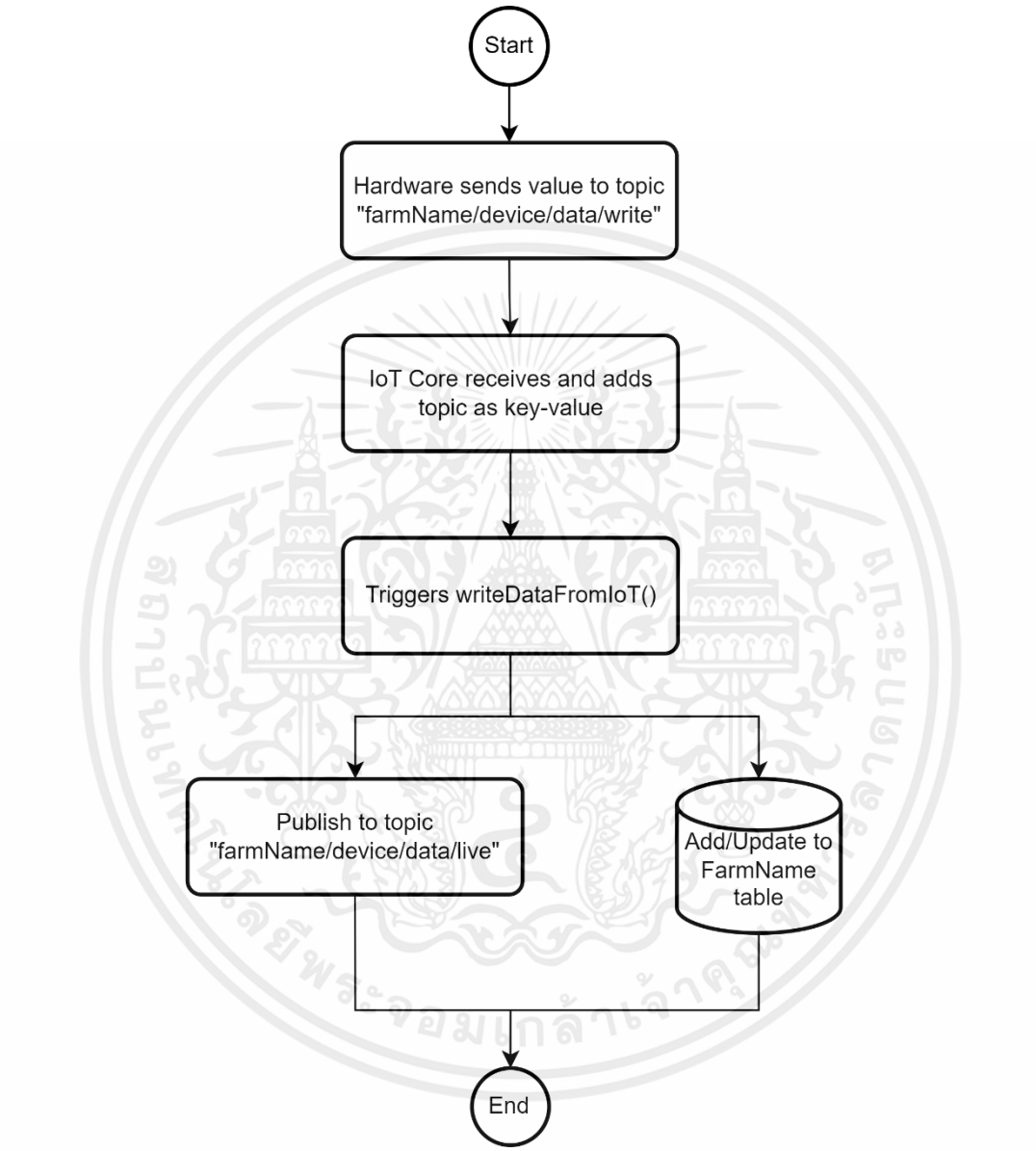
โดย targeTable คือตารางที่ต้องการลบ deleteList คือตัวข้อมูลที่ต้องการจะแก้ไขโดยมีการใช้ ID เพื่อระบุข้อมูลตัวที่ต้องการจะลบ โดยการลบสามารถทำได้ในหน้า Table page ของเว็บแอปพลิเคชัน



รูปที่ 3.12 ผังงานของการลบข้อมูลข้อมูลในตารางต่างๆ

3.4.6 การส่งข้อมูลจากฮาร์ดแวร์และการปล่อยข้อมูลทันทีให้ผู้ติดตาม

ฟังก์ชันนี้ถูกออกแบบมาเพื่อให้ฮาร์ดแวร์สามารถบันทึกหรืออัพเดทข้อมูลลงในฐานข้อมูลได้ในขณะเดียวกัน เมื่อทำสำเร็จก็จะทำการเผยแพร่ข้อมูลนั้นให้ผู้ติดตามทันที



รูปที่ 3.13 ผังงานของการส่งข้อมูลจากฮาร์ดแวร์และการปล่อยข้อมูลทันทีให้ผู้ติดตาม

3.4.7 การควบคุมอุปกรณ์

ฟังก์ชันที่ใช้ในควบคุมอุปกรณ์มีอยู่ 2 ฟังก์ชัน โดยจะเรียกใช้ฟังก์ชัน liveData ซึ่งจะต้องส่งค่าข้อมูลไปเพื่อดูสถานะอุปกรณ์ในขณะนั้นดังนี้

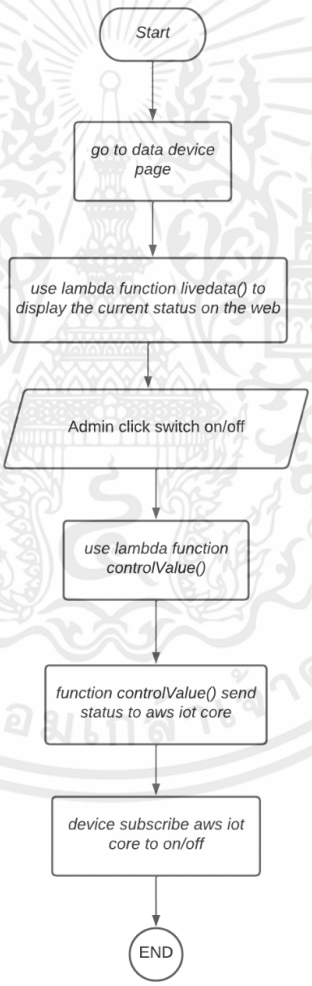
```
{"topic":"Farmtest/MOISTURE01/"}
```

โดย topic คือให้ระบุชื่อฟาร์มและอุปกรณ์ที่ต้องการจะต้องการดูสถานะ

ต่อมาจะเรียกใช้ฟังก์ชัน controlValue ซึ่งจะต้องส่งค่าข้อมูลไปเพื่อควบคุมอุปกรณ์ดังนี้

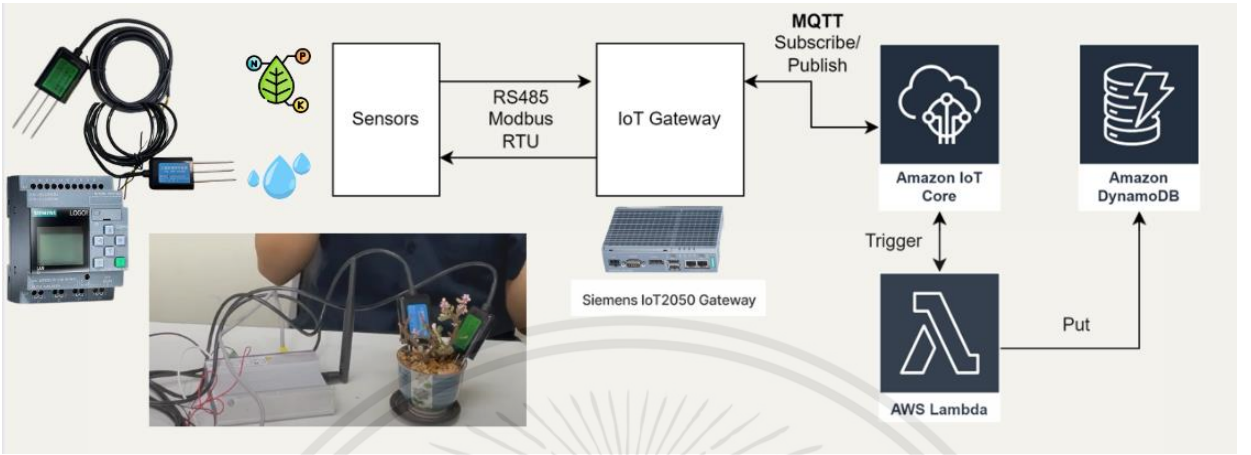
```
{"topic": "Farmname/device",
"controlValue": "false"}
```

โดย topic คือให้ระบุชื่อฟาร์มและอุปกรณ์ที่ต้องการจะควบคุม controlValue คือกำหนดสถานะว่าต้องการเปิด/ปิด ซึ่งจะสามารถควบคุมอุปกรณ์ทั้งหมดในหน้าข้อมูลอุปกรณ์บนเว็บแอปพลิเคชัน



รูปที่ 3.14 ผังงานของการควบคุมอุปกรณ์

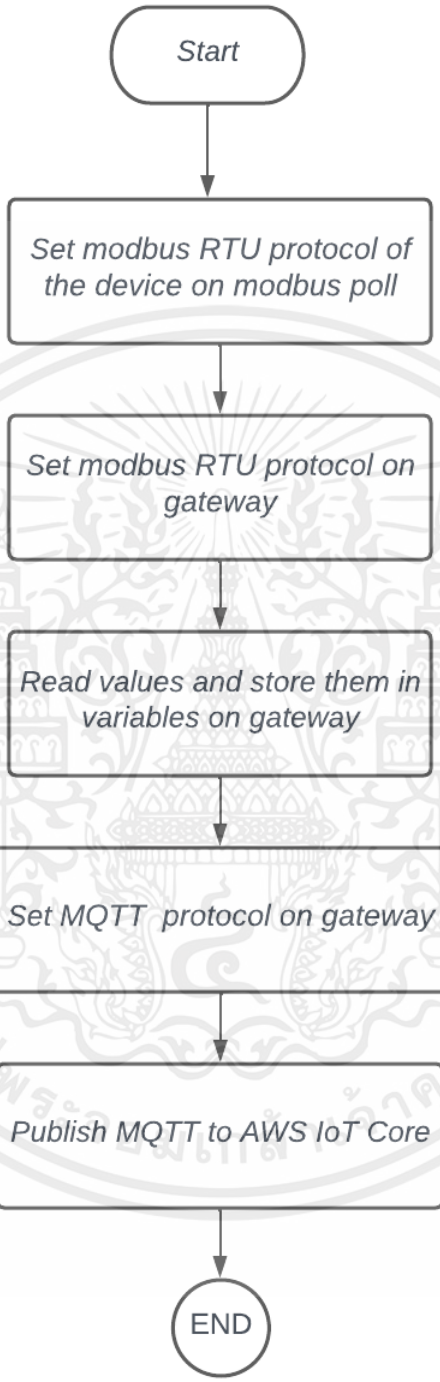
3.5 การออกแบบในส่วนอุปกรณ์เพื่อจัดการฟาร์ม



รูปที่ 3.15 ภาพรวมการออกแบบในส่วนอุปกรณ์เพื่อจัดการฟาร์ม

- ติดตั้งเซ็นเซอร์ (อุณหภูมิ ความชื้น แสง แก๊ส ฯลฯ) และอุปกรณ์สั่งการ (โซลินอยด์วาลว์ มอเตอร์)
- เชื่อมต่อเซ็นเซอร์และอุปกรณ์สั่งการกับคอนโทรลเลอร์ (PLC Siemens LOGO) และ IoT Gateway (IoT2050)
- ตั้งค่า IoT Gateway ด้วย Node-Red ให้ทำงานเป็น Edge Computing รวมถึงการจัดการข้อมูลและ Topic ของ MQTT
- ส่งค่าไป AWS Cloud Service ผ่าน 4G ไปยังกับ AWS IoT Core ด้วย MQTT Protocol และเก็บข้อมูลไปยัง DynamoDB

3.5.1 การอ่านข้อมูลอุปกรณ์ sensor



รูปที่ 3.16 ผังงานของการอ่านข้อมูลอุปกรณ์ sensor

การอ่านข้อมูลอุปกรณ์ sensor จะเริ่มจากการตั้งค่าการสื่อสารระหว่างอุปกรณ์ sensor กับ IoT Gateway โดยใช้มาตรฐานการสื่อสารอนุกรม modbus RTU หลังจากนั้นอ่านค่าที่ได้จากอุปกรณ์ sensor ใน IoT Gateway และเก็บค่าเพื่อรอส่งให้ AWS IoT Core โดยก่อนที่จะส่งให้ AWS IoT Core จะต้องตั้งค่าการสื่อสารระหว่าง IoT Gateway กับ AWS IoT Core โดยใช้มาตรฐานการสื่อสาร MQTT หลังจากนั้น publish MQTTไปยัง AWS IoT Core และบันทึกค่าใน AWS DynamoDB

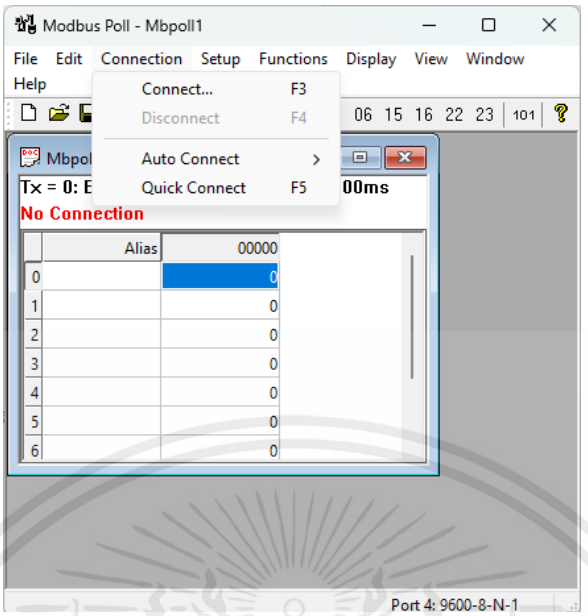
3.5.1.1 การตั้งค่าอุปกรณ์ sensor ด้วย Modbus Poll

3.5.1.1.1 การอ่านค่าอุปกรณ์ sensor ด้วย Modbus Poll

โดยเชื่อมต่ออุปกรณ์ Sensor เข้ากับอุปกรณ์แปลงสัญญาณ RS485 to USB Serial และ Power Supply หลังจากนั้นเชื่อมต่อ USB อุปกรณ์แปลงสัญญาณ RS485 to USB Serial และคอมพิวเตอร์ที่มีโปรแกรมModbus Poll เข้าด้วยกัน

รูปที่ 3.17 อุปกรณ์แปลงสัญญาณ RS485 to USB Serial

หลังจากนั้นการอ่านค่าอุปกรณ์ Sensor ทำได้โดยการใช้โปรแกรม Modbus Poll โดยการที่จะให้สามารถอ่านค่าต้องเชื่อมต่อดังนี้ กดที่แถบ connection และกด connect...



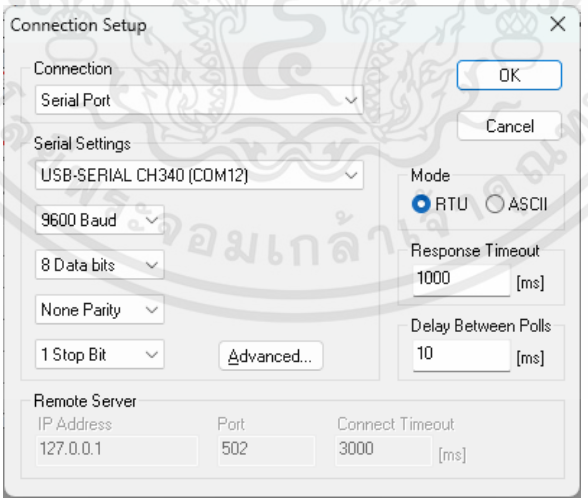
รูปที่ 3.18 การเชื่อมต่ออุปกรณ์ Sensor โดยการใช้โปรแกรม Modbus Poll

หลังจากนั้นจะมีหน้าต่างการตั้งค่า ให้ตั้งค่าตามดังนี้

1) เลือก Serial Port: COM12

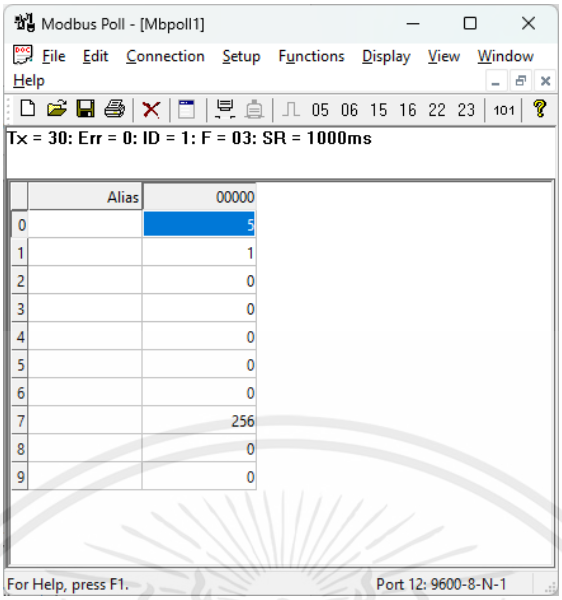
2) Baud Rate: 9600

ตั้งค่าเรียบร้อยกด OK



รูปที่ 3.19 การตั้งค่าอุปกรณ์ Sensor โดยการใช้โปรแกรม Modbus Poll

จะปรากฏค่าที่อ่านได้จากอุปกรณ์ Sensor ตาม data sheet ค่าที่ได้ของอุปกรณ์ Sensor คือ address ที่ 0

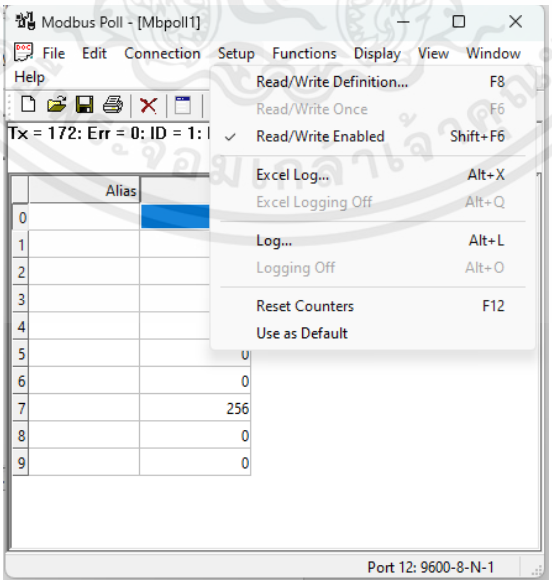


รูปที่ 3.20 การอ่านค่าอุปกรณ์ Sensor โดยการใช้โปรแกรม Modbus Poll

3.5.1.1.2 การเปลี่ยนการตั้งค่าอุปกรณ์ sensor ใหม่ด้วย Modbus Poll

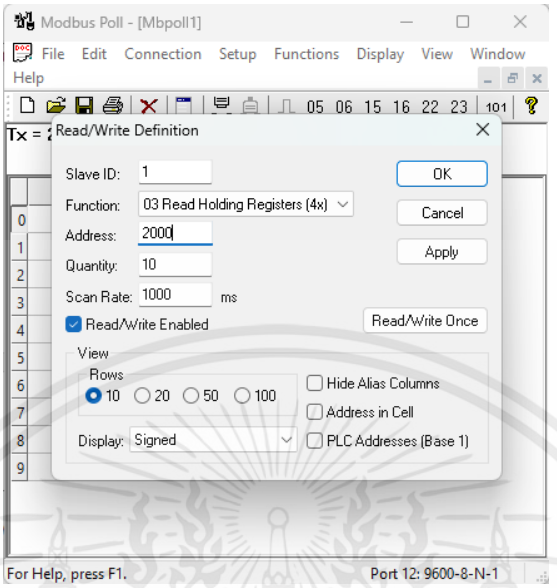
การเปลี่ยนการตั้งค่า Slave ID, Baud rate ของ Sensor ใหม่ เพื่อให้สามารถจัดระเบียบการสื่อสารไม่ให้ชนกันและสามารถทำให้คุยกันได้มากกว่า 1 อุปกรณ์

โดยตั้งค่า slave ID และ baud rate เพื่อง่ายต่อการนำไปใช้งานต่อ โดยการใช้ Modbus Poll ในการตั้งค่าโดยกดที่แถบ Setup แล้วเลือก Read/Write Definition



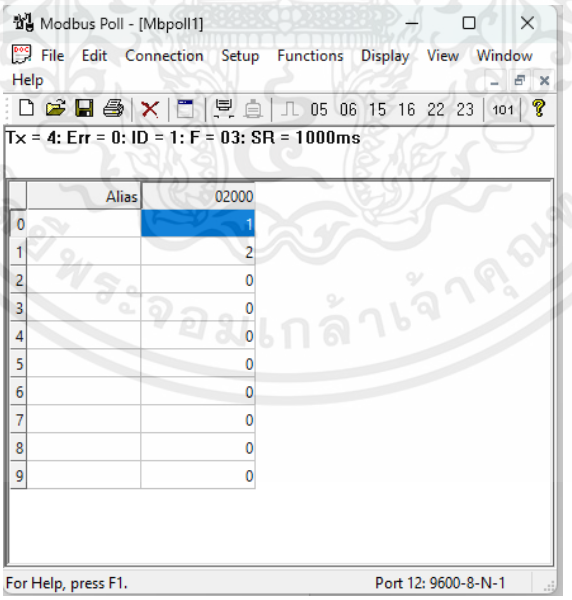
รูปที่ 3.21 การเปลี่ยนการตั้งค่าอุปกรณ์ Sensor โดยการใช้โปรแกรม Modbus Poll

จะปรากฏหน้าต่างการตั้งค่า หน้านี้สามารถแก้ไข slave ID, address ที่จะอ่านได้ ให้เลื่อนไปที่ address 2000



รูปที่ 3.22 การกำหนดการเปลี่ยนตั้งค่าอุปกรณ์ Sensor โดยการใช้โปรแกรม Modbus Poll

Double Click ที่ช่องข้อมูลตรง address 2000 เพื่อตั้งค่า slave ID และ Double Click ที่ช่องข้อมูลตรง address 2001 เพื่อตั้งค่า baud rate

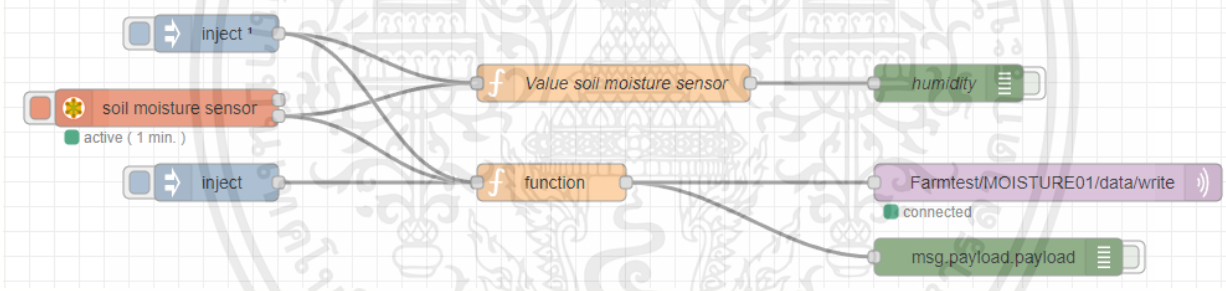


รูปที่ 3.23 ผลที่ได้จากการเปลี่ยนการตั้งค่าอุปกรณ์ Sensor โดยการใช้โปรแกรม Modbus Poll

ตารางที่ 2.6 รายละเอียดการกำหนดค่าต่างๆและข้อมูลที่ต้องใช้ของอุปกรณ์ Sensor

หัวข้อ	อุปกรณ์วัดธาตุสารอาหารในดิน	อุปกรณ์วัดความชื้นในดิน
Baud rate	9600	9600
Slave ID	2	1
Data address	Address 0 (ไนโตรเจน (N): มิลลิกรัม/ลิตร) Address 1 (ฟอสฟอรัส (P): มิลลิกรัม/ลิตร) Address 2 (โพแทสเซียม (K): มิลลิกรัม/ลิตร)	Address 0 (ความชื้นในดิน: %)
Voltage	12VDC	12VDC

3.5.1.2 การตั้งค่า node-red ในเกตเวย์

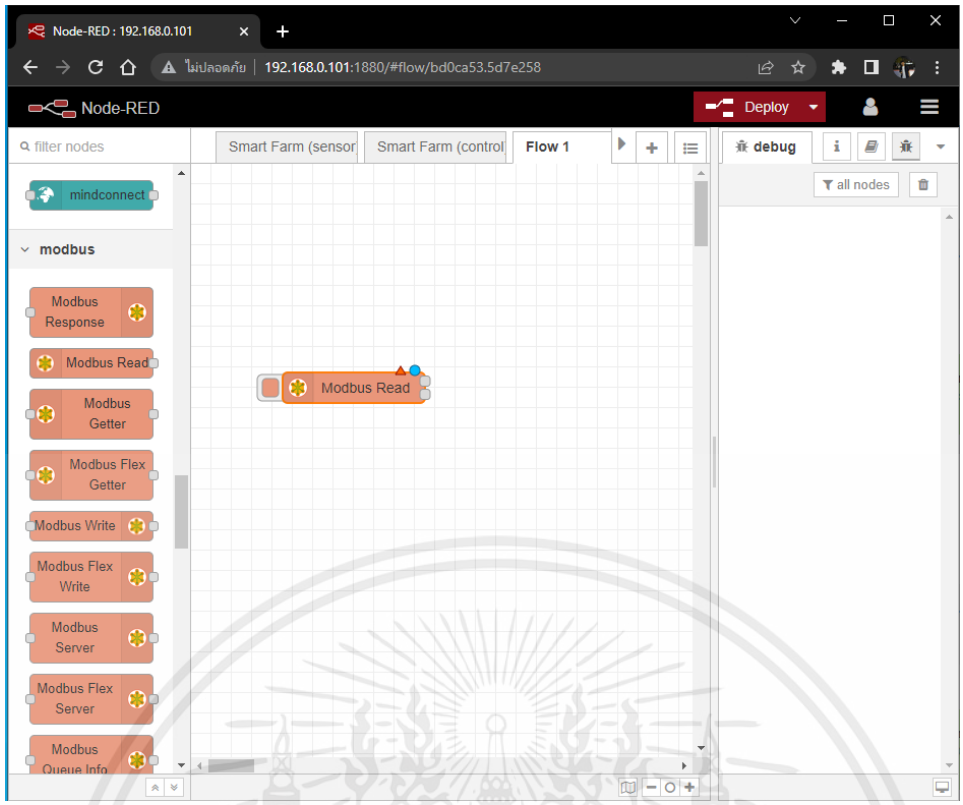


รูปที่ 3.24 ภาพรวมการอ่านข้อมูลอุปกรณ์ sensor บนเกตเวย์

การตั้งค่า node-red ใน IoT Gateway เพื่ออ่านข้อมูลอุปกรณ์ sensor จะเริ่มจากการตั้งค่าการ Modbus Read Node จากนั้นตั้งค่า Function Node เขียนฟังก์ชันเพื่ออ่าน เก็บค่า และการตั้งค่าตัวแปรเพื่อสื่อสารระหว่างอุปกรณ์ IoT Gateway กับ AWS IoT Core โดยใช้มาตรฐานการสื่อสาร MQTT หลังจากนั้น publish MQTT ไปยัง AWS IoT Core โดย MQTT out Node

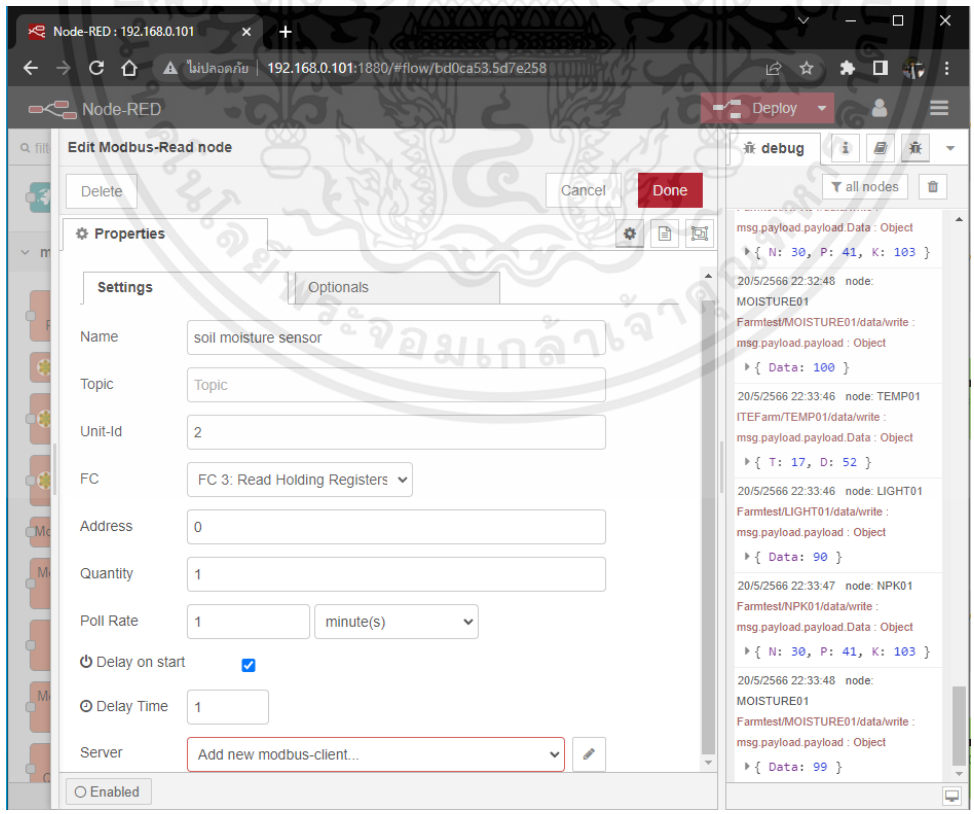
3.5.1.2.1 อ่านข้อมูลอุปกรณ์ sensor ด้วย Modbus RTU Protocol

โดยเชื่อมต่ออุปกรณ์ sensor เข้ากับ IoT Gateway ที่ COM interfaces (RS232/422/485) และ Power Supply หลังจากนั้นการอ่านข้อมูลอุปกรณ์ sensor ด้วย Modbus RTU Protocol โดยตั้งค่าโปรแกรม node-red ใน IoT Gateway ดังนี้



รูปที่ 3.25 Modbus Read Node ในการอ่านข้อมูลอุปกรณ์ sensor

เลือกหา Modbus Read Node ใน Modbus Node ลากมาวางใน Flow แล้ว Double Click ที่ตัว Node

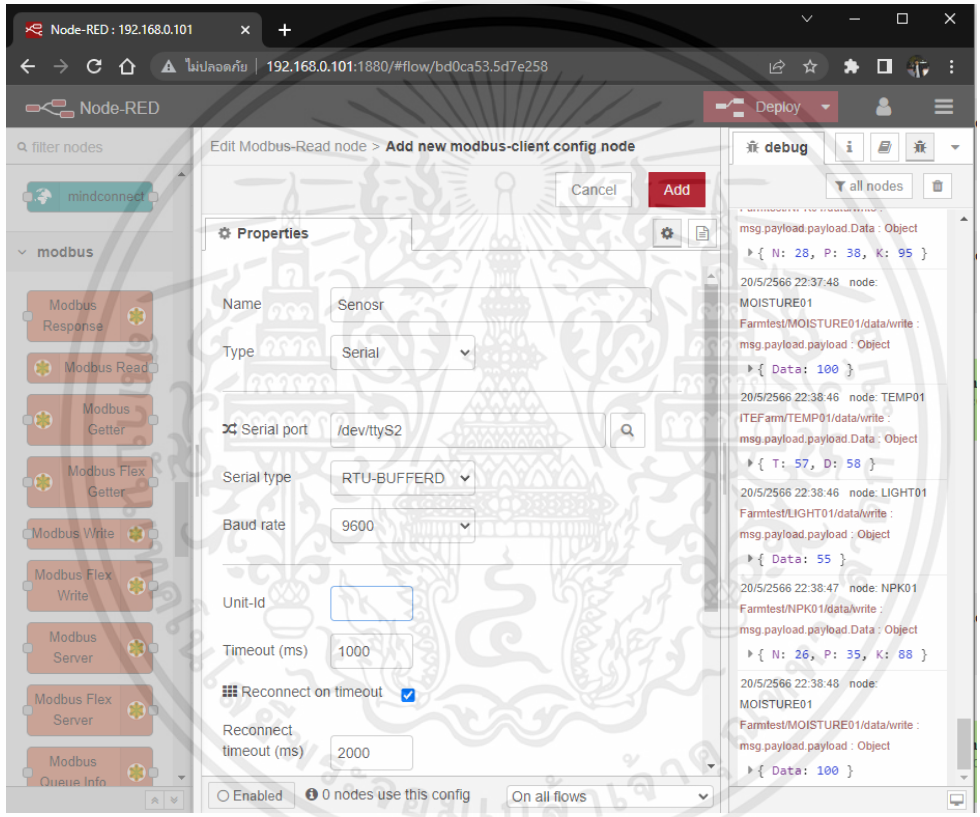


รูปที่ 3.26 การตั้งค่า Modbus Read Node ในการอ่านข้อมูลอุปกรณ์ sensor

หลังจากนั้นจะปรากฏการตั้งค่า Modbus-Read Node ให้ตั้งค่าตามตารางข้อมูลเฉพาะของอุปกรณ์ sensor ดังนี้

1) ตั้งชื่อ: soil moisture sensor
2) Unit-Id/ Slave ID: 2
3) FC 3: Read Holding Registers
4) Address: 0
5) Quantity: 1

หลังจากกดปุ่มแก้ไขตรง Server เพื่อสร้างมาตรฐานการสื่อสารด้วย Modbus RTU Protocol

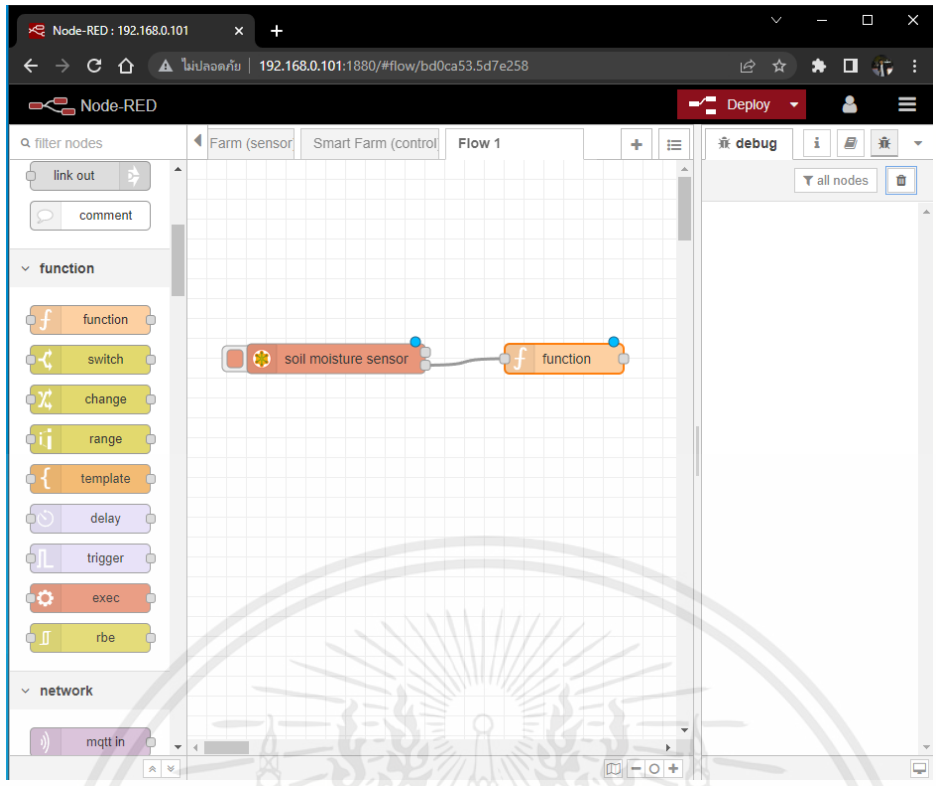


รูปที่ 3.27 การตั้งค่า Modbus-client config node ในการอ่านข้อมูลอุปกรณ์ sensor

หลังจากนั้นจะปรากฏการตั้งค่า Modbus-client config node ให้ตั้งค่าตามดังนี้

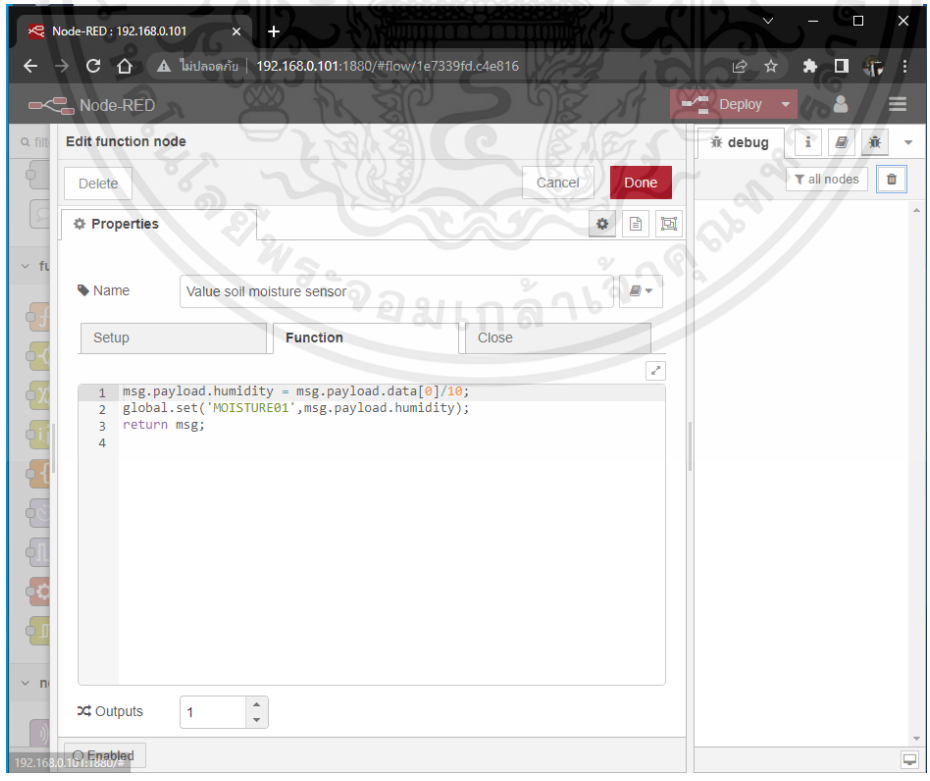
1) ตั้งชื่อ: Sensor
2) Type: Serial
3) Serial port: /dev/ttyS2
4) Serial type: RTU-BUFFERD
5) Baud rate: 9600

หลังจากกดปุ่ม Add



รูปที่ 3.28 การเชื่อมต่อ Function Node กับ Modbus Read Node ในการอ่านข้อมูลอุปกรณ์ sensor

เลือกหา Function Node ใน Function ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง Function Node กับ Modbus-Read Node หลังจากนั้นกด Double Click ที่ตัว Node

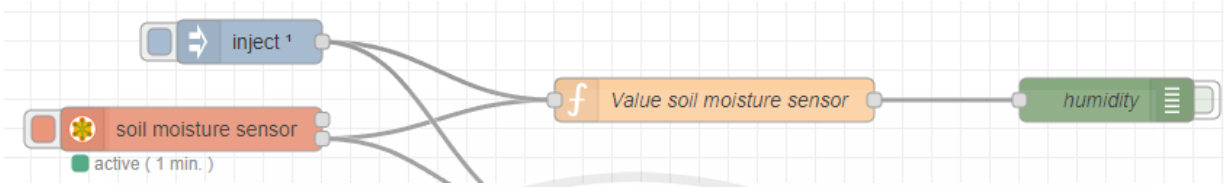


รูปที่ 3.29 การตั้งค่า Function node ในการอ่านข้อมูลอุปกรณ์ sensor

หลังจากนั้นจะปรากฏการตั้งค่า Function node ให้เขียนโปรแกรมตามดังนี้

1) msg.payload.humidity = msg.payload.data[0]/10;
2) global.set('MOISTURE01',msg.payload.humidity);
3) return msg;

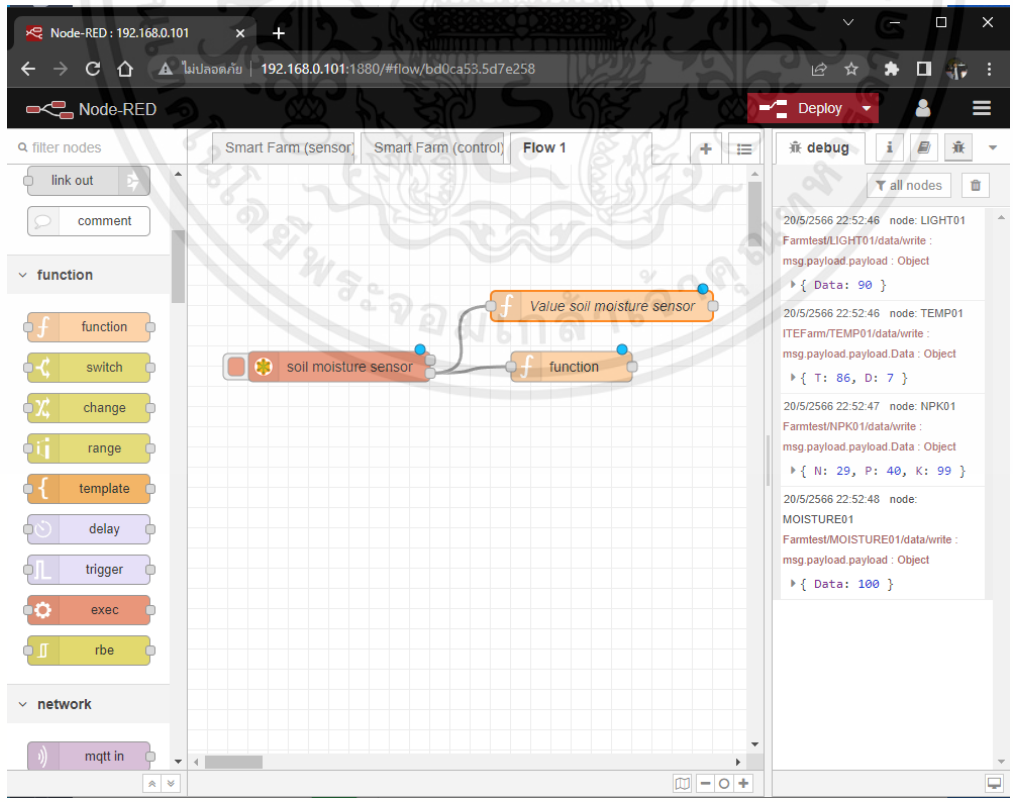
หลังจากกดปุ่ม Done



รูปที่ 3.30 การอ่านข้อมูลอุปกรณ์ sensor บนเกตเวย์

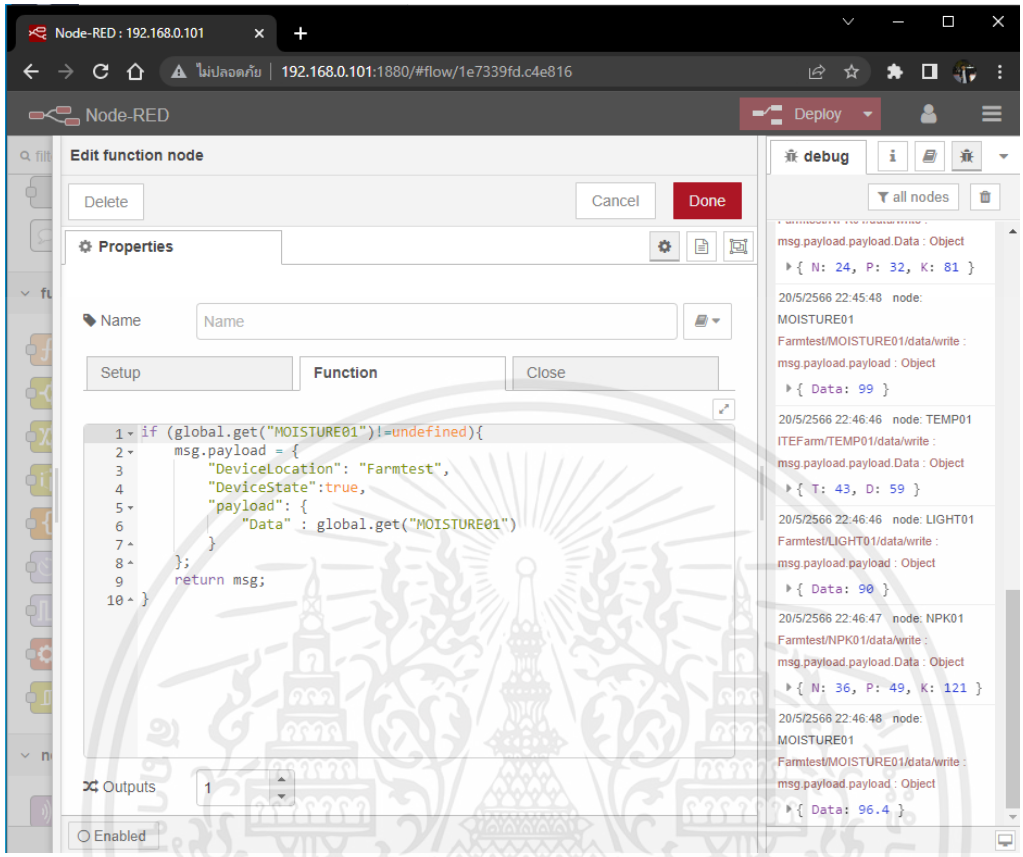
3.5.1.2.2 ส่งข้อมูลอุปกรณ์ sensor ไปยัง AWS IoT Core ด้วย Publish MQTT Protocol

โดยเชื่อมต่ออุปกรณ์ sensor เข้ากับ IoT Gateway ที่ COM interfaces (RS232/422/485) และ Power Supply หลังจากนั้นการอ่านข้อมูลอุปกรณ์ sensor ด้วย Modbus RTU Protocol โดยตั้งค่าโปรแกรม node-red ใน IoT Gateway ดังนี้



รูปที่ 3.31 การเชื่อมต่อ Function Node ในการส่งข้อมูลอุปกรณ์ sensor

เลือกหา Function Node ใน Function ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง Function Node กับ Modbus-Read Node หลังจากนั้นกด Double Click ที่ตัว Node ที่สร้างมาใหม่

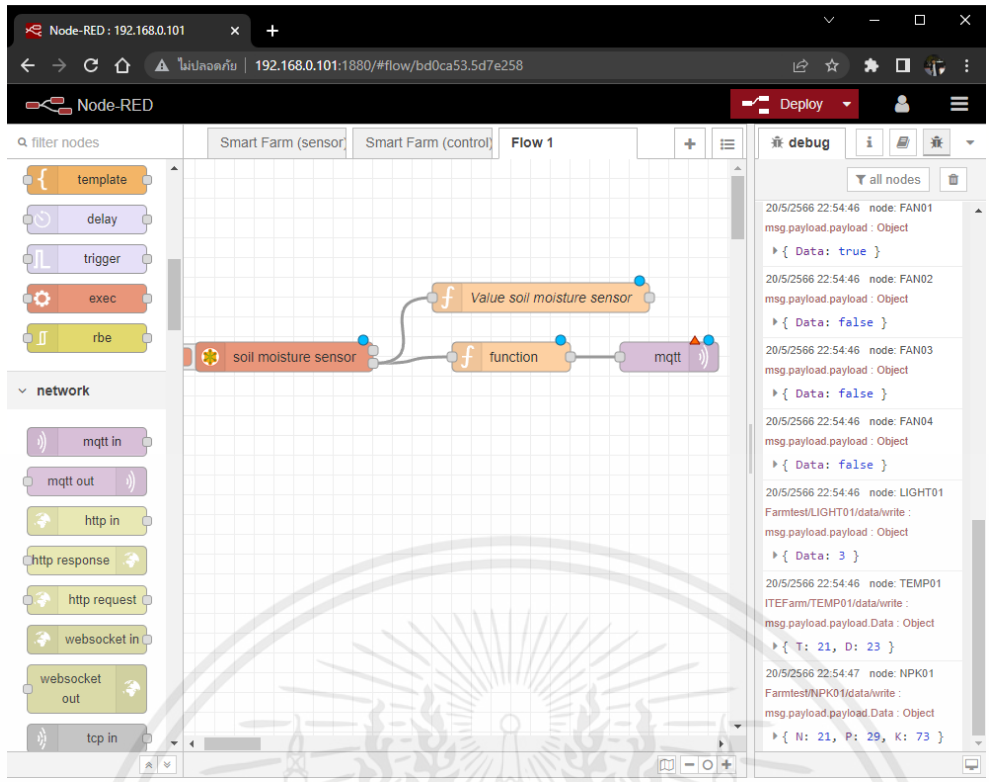


รูปที่ 3.32 การตั้งค่า Function node ในการส่งข้อมูลอุปกรณ์ sensor

หลังจะปรากฏการตั้งค่า Function node ให้เขียนโปรแกรมตามดังนี้

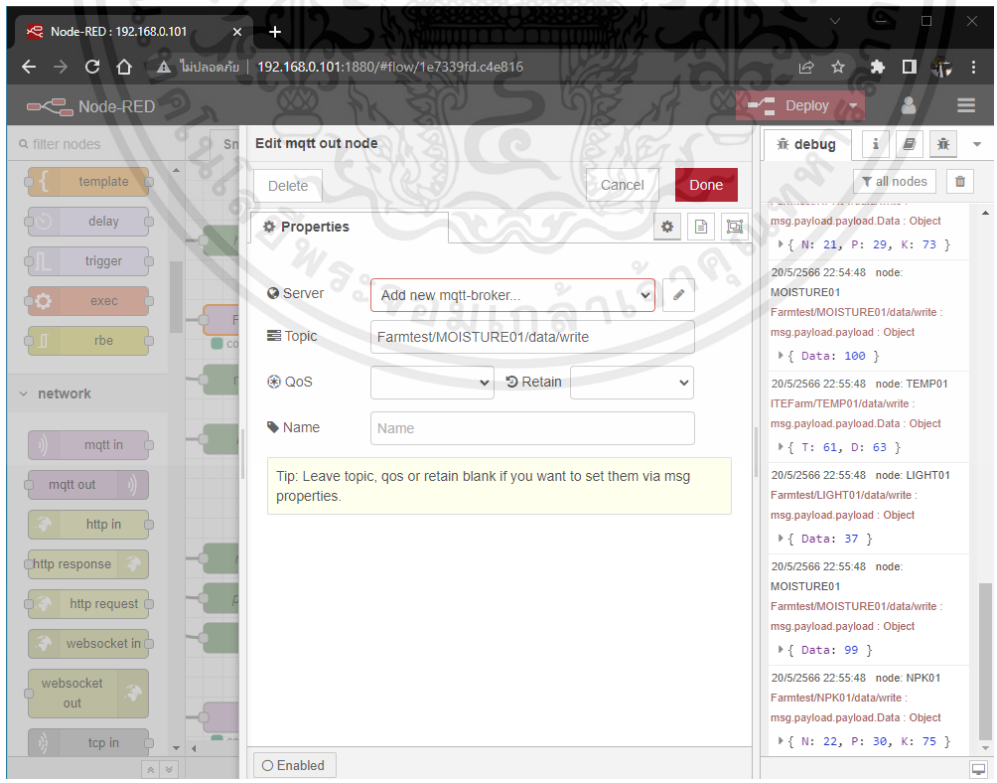
```
1)  if (global.get("MOISTURE01")!=undefined){
2)      msg.payload = {
3)          "DeviceLocation": "Farmtest",
4)          "DeviceState":true,
5)          "payload": {
6)              "Data" : global.get("MOISTURE01")
7)          }
8)      };
9)      return msg;
10) }
```

หลังจากกดปุ่ม Done



รูปที่ 3.33 การเชื่อมต่อระหว่าง mqtt out Node กับ Function Node ในการส่งข้อมูลอุปกรณ์ sensor

เลือกหา mqtt out Node ใน Network Node ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง mqtt out Node กับ Function Node หลังจากนั้นกด Double Click ที่ตัว Node ที่สร้างมาใหม่

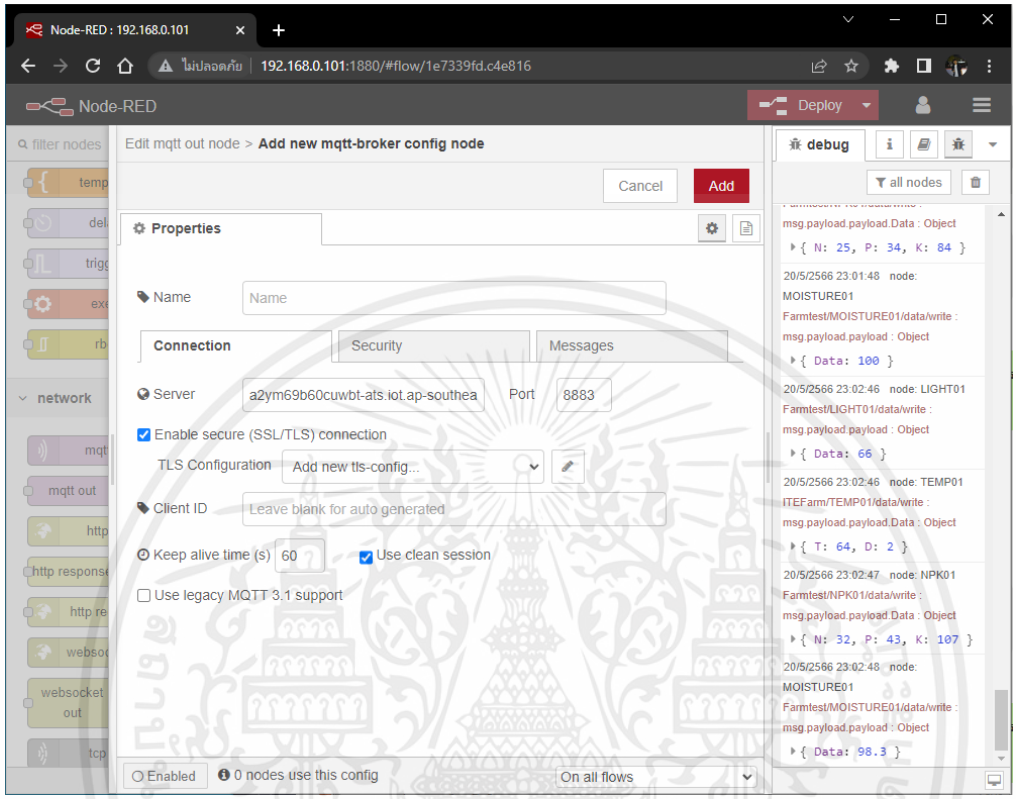


รูปที่ 3.34 การตั้งค่า mqtt out node ในการส่งข้อมูลอุปกรณ์ sensor

หลังจากนั้นจะปรากฏการตั้งค่า mqtt out node ให้ตั้งค่าตามดังนี้

1) Topic: Farmtest/MOISTURE01/data/write
(ชื่อฟาร์ม/Serial Number ของอุปกรณ์ Sensor/data/write)

หลังจากกดปุ่มแก้ไขตรง Server เพื่อสร้างมาตรฐานการสื่อสารด้วย MQTT Protocol

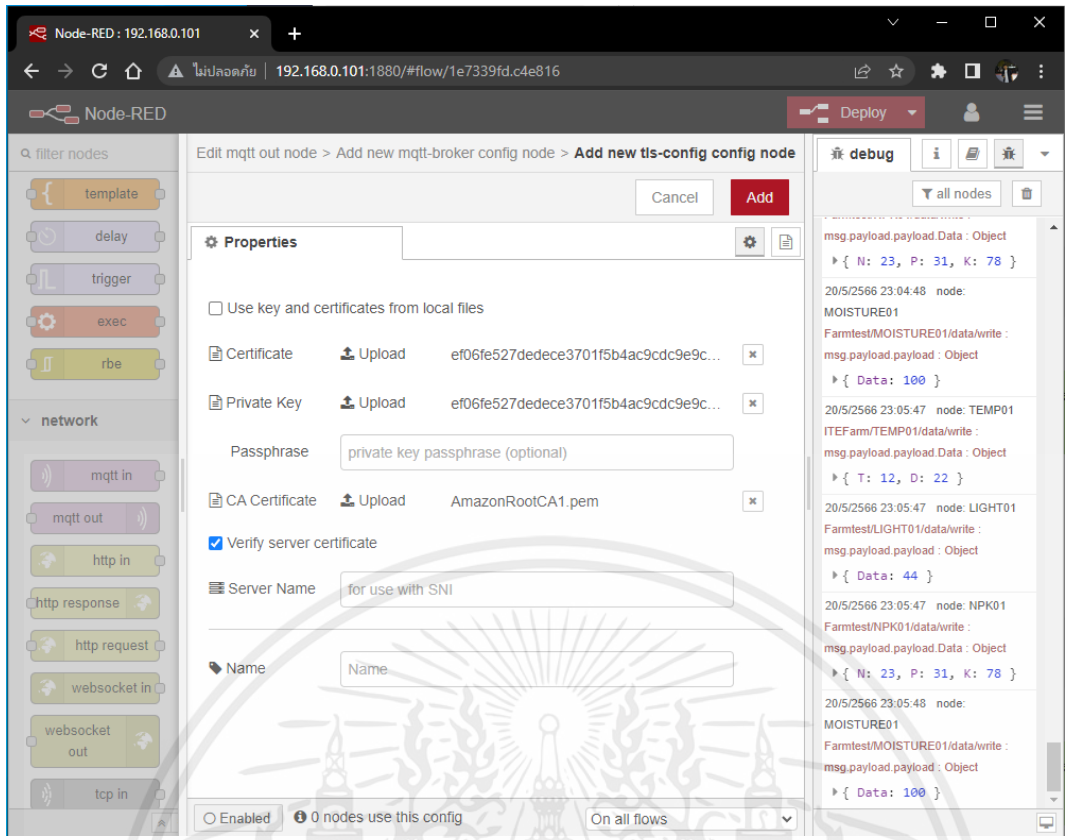


รูปที่ 3.35 การตั้งค่า mqtt-broker node ในการส่งข้อมูลอุปกรณ์ sensor

หลังจากนั้นจะปรากฏการตั้งค่า mqtt-broker node ให้ตั้งค่าตามดังนี้

1) Server: ชื่อ endpoint ของ AWS IoT Core
2) Port: 8883

หลังจากกดปุ่ม Enable secure (SSL/TLS) connection และปุ่มแก้ไขตรง TLS Configuration เพื่อยืนยันตัวตนในการเชื่อมต่อกับ AWS IoT Core

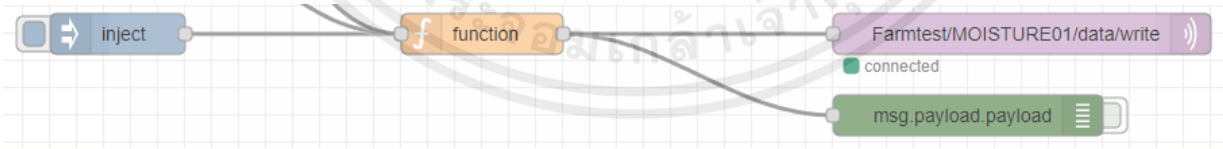


รูปที่ 3.36 การตั้งค่า tls-config node ในการส่งข้อมูลอุปกรณ์ sensor

หลังจากนั้นจะปรากฏการตั้งค่า tls-config node ให้อัพโหลดตามดังนี้

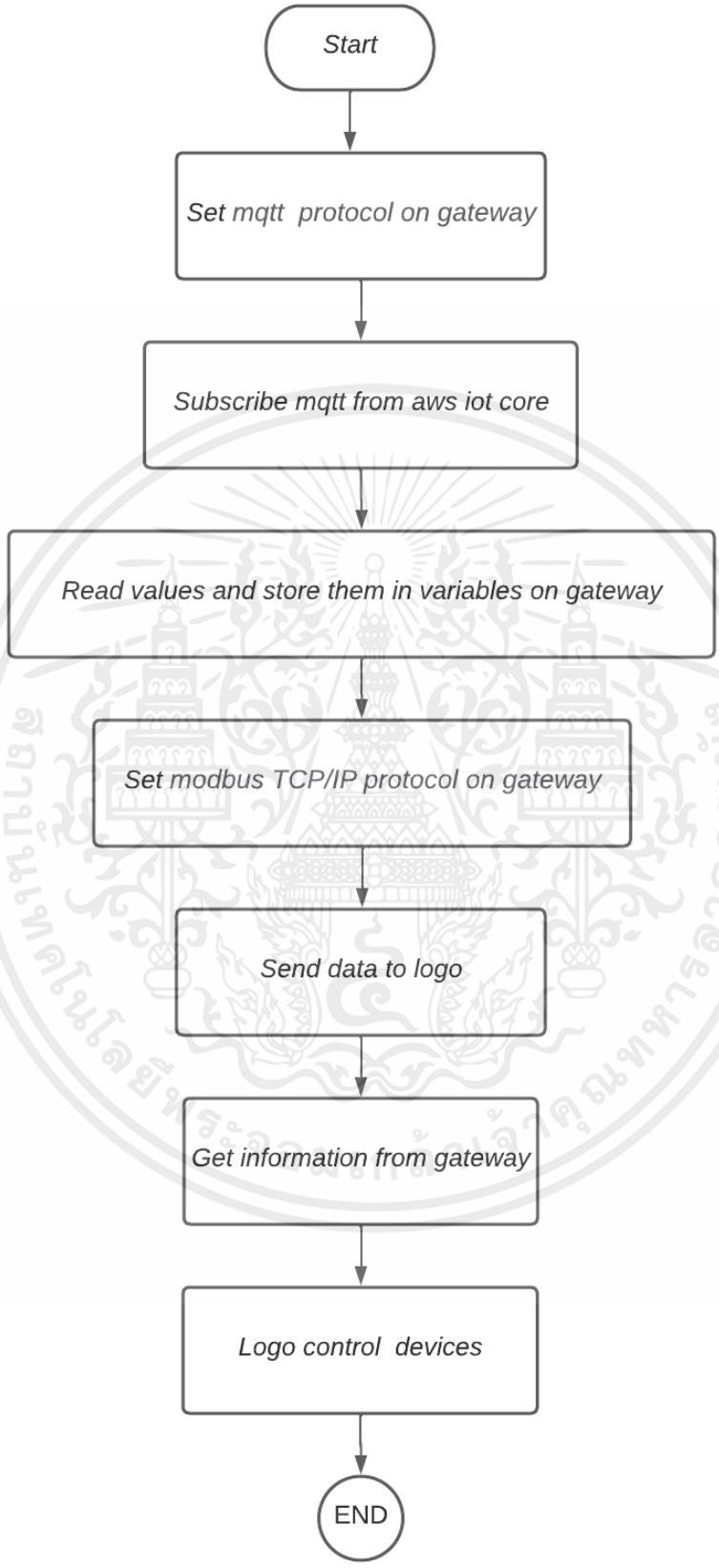
1) Certificate
2) Private Key
3) CA Certificate

หลังจากกดปุ่ม Add



รูปที่ 3.37 การส่งข้อมูลอุปกรณ์ sensor ไปยัง AWS IoT Core

3.5.2 การส่งข้อมูลเพื่อควบคุมอุปกรณ์สั่งการ

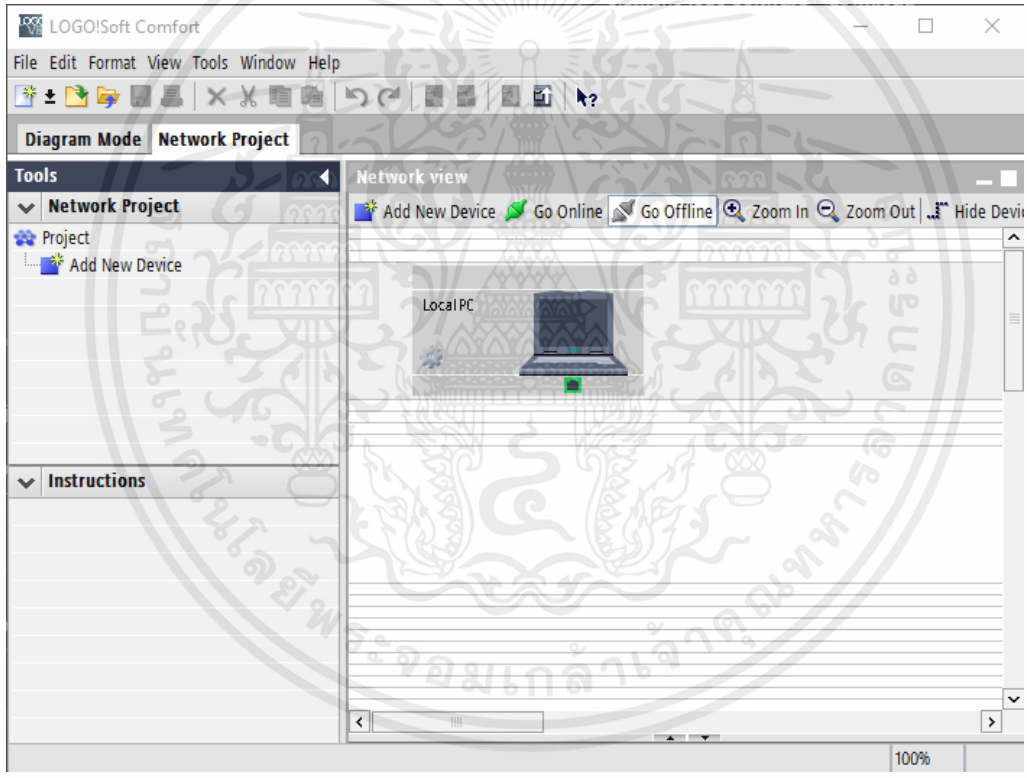


รูปที่ 3.38 ผังงานของการส่งข้อมูลเพื่อควบคุมอุปกรณ์สั่งการ

การส่งข้อมูลเพื่อควบคุมอุปกรณ์สั่งการ จะเริ่มจากการตั้งค่าการสื่อสารระหว่างอุปกรณ์ IoT Gateway กับ AWS IoT Core โดยใช้มาตรฐานการสื่อสาร MQTT หลังจากนั้น subscribe MQTT ไปยัง AWS IoT Core หลังจากนั้นอ่านค่าที่ได้จาก AWS IoT Core ใน IoT Gateway และเก็บค่าเพื่อรอส่งให้เครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ควบคุมอุปกรณ์สั่งการ โดยก่อนที่จะส่งให้เครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ จะต้องตั้งค่าการสื่อสารระหว่างเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้กับ IoT Gateway และรับค่ามาควบคุมอุปกรณ์สั่งการ โดยใช้มาตรฐานการสื่อสารอนุกรม Modbus TCP/IP

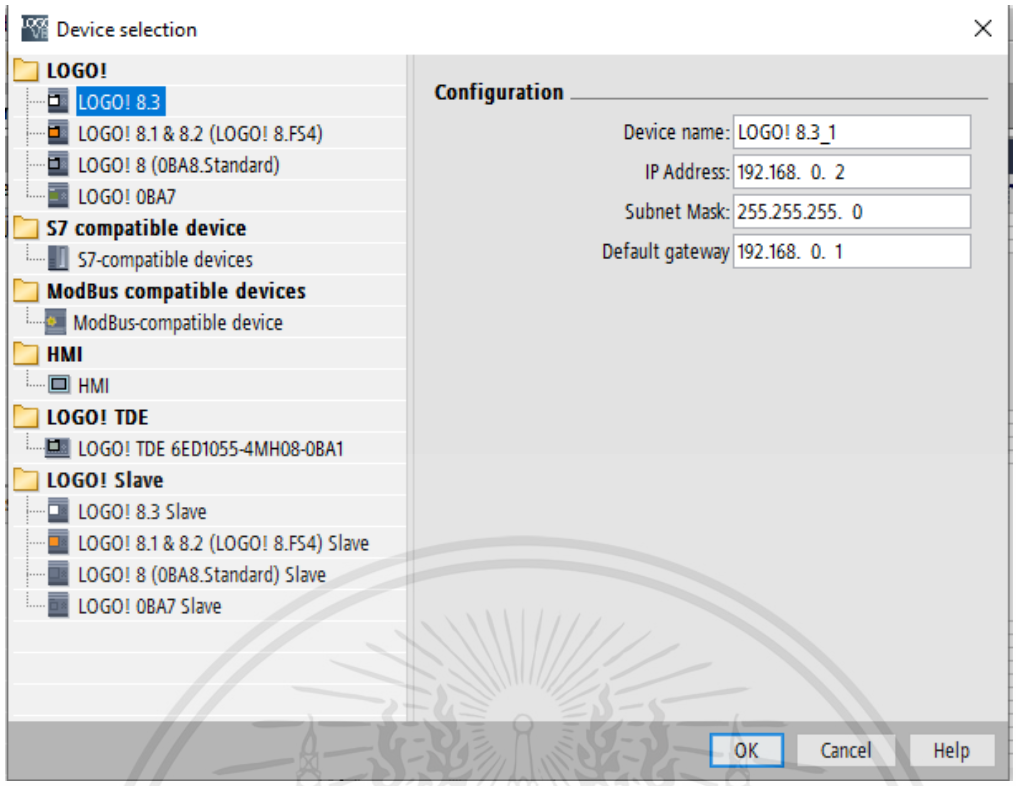
3.5.2.1 การตั้งค่าเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้

โดยเชื่อมต่ออุปกรณ์เครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ (PLC LOGO Controller) เข้ากับคอมพิวเตอร์โดยใช้สาย LAN เข้าด้วยกัน หลังจากนั้นการตั้งค่าเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ สามารถทำได้โดยการใช้โปรแกรม LOGO! Soft Comfort ดังนี้



รูปที่ 3.39 Add New Device ใน PLC LOGO Controller

กดปุ่ม Network Project อยู่ด้านบนของหน้าต่าง Tools และกดปุ่ม Add New Device ในหน้าต่างของ Network view



รูปที่ 3.40 การตั้งค่า Device selection ใน PLC LOGO Controller

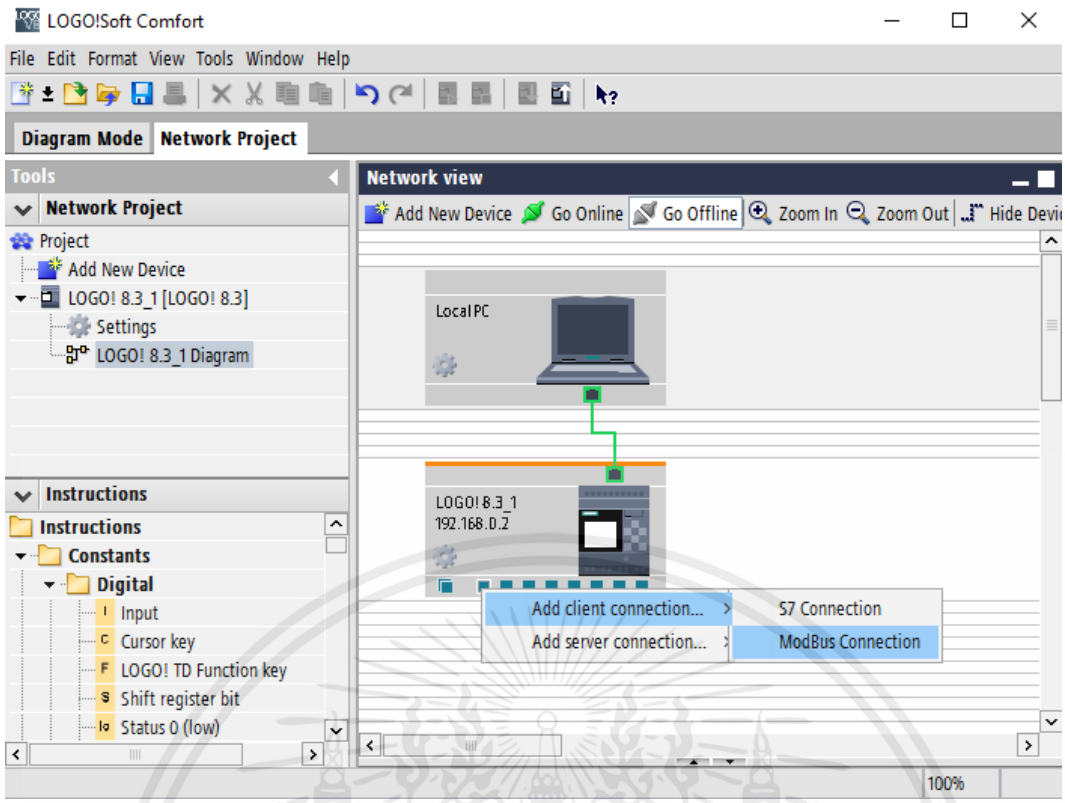
จะปรากฏหน้าต่างการตั้งค่า Device selection โดยให้ตั้งค่าตามดังนี้

1) IP Address 192.168.0.2

2) Subnet Mask 255.255.255.0

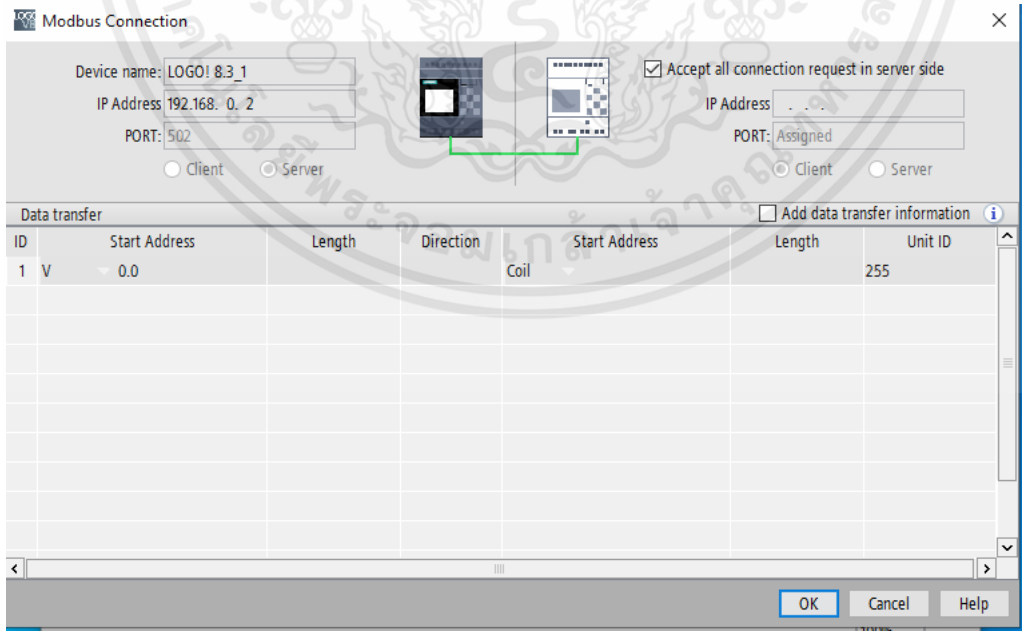
3) Default gateway 192.168.0.1

ตั้งค่าเรียบร้อยกด OK



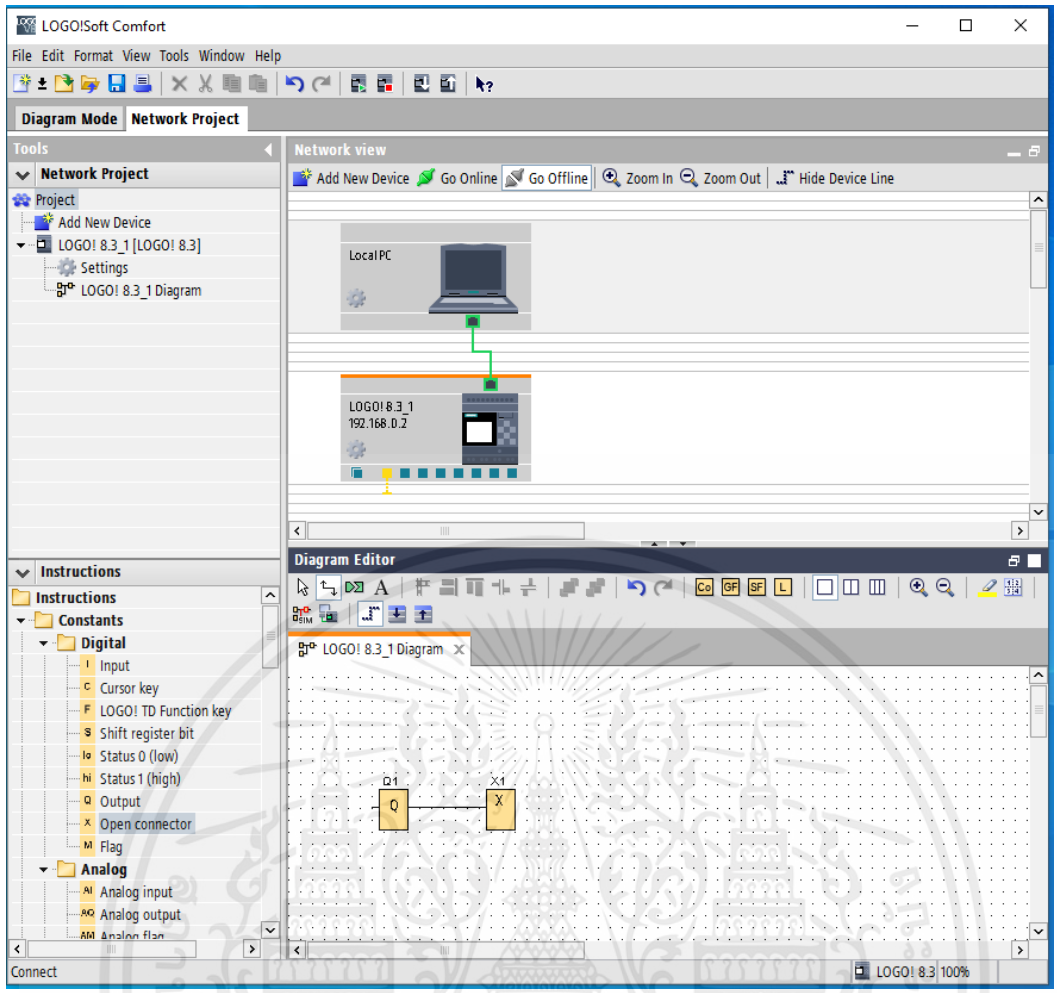
รูปที่ 3.41 Add client connection ใน PLC LOGO Controller

หลังจากนั้นจะปรากฏอุปกรณ์ LOGO ที่มีสายเชื่อมต่อไปยัง Local PC จากนั้นกดคลิกขวาปุ่มสีน้ำเงินด้านล่างบนอุปกรณ์ LOGO ที่เพิ่มขึ้นมา จะมีให้เลือกไปยัง Add client connection... และกดไปที่ Modbus Connection



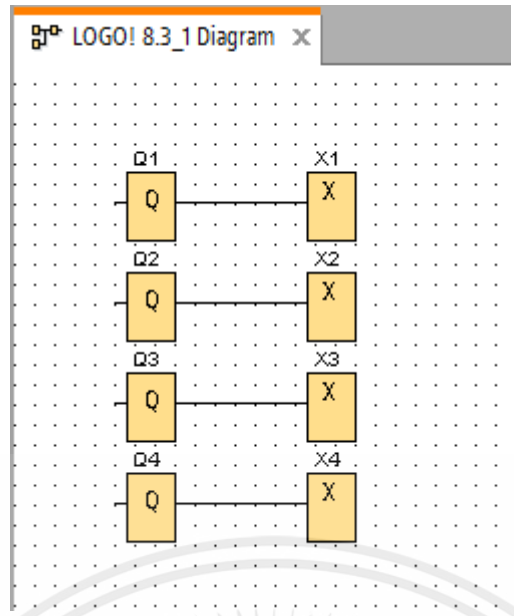
รูปที่ 3.42 การตั้งค่า Modbus TCP/IP ใน PLC LOGO Controller

หลังจากนั้นจะปรากฏการตั้งค่า Modbus TCP/IP จากนั้นกด OK



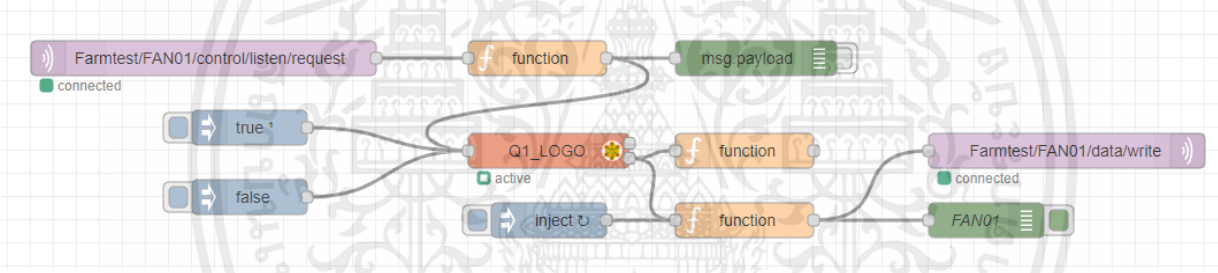
รูปที่ 3.43 Diagram ใน PLC LOGO Controller

หลังจากนั้นจะปรากฏปุ่มสีเหลืองและมีสายเชื่อมต่อเพิ่มขึ้นมา จากนั้นเลื่อนมาด้านล่างจะพบหน้าต่าง Diagram Editor ให้ไปดูที่หน้าต่างด้านซ้ายมือของ Diagram Editor และเลือก Output Node กับ Open connection ใน Instruction มาลากเชื่อมต่อกัน 4 ชุด (เนื่องจาก PLC Siemens LOGO Controller สามารถควบคุมได้สูงสุด 4 อุปกรณ์)



รูปที่ 3.44 ผลที่ได้จากการโปรแกรมเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้

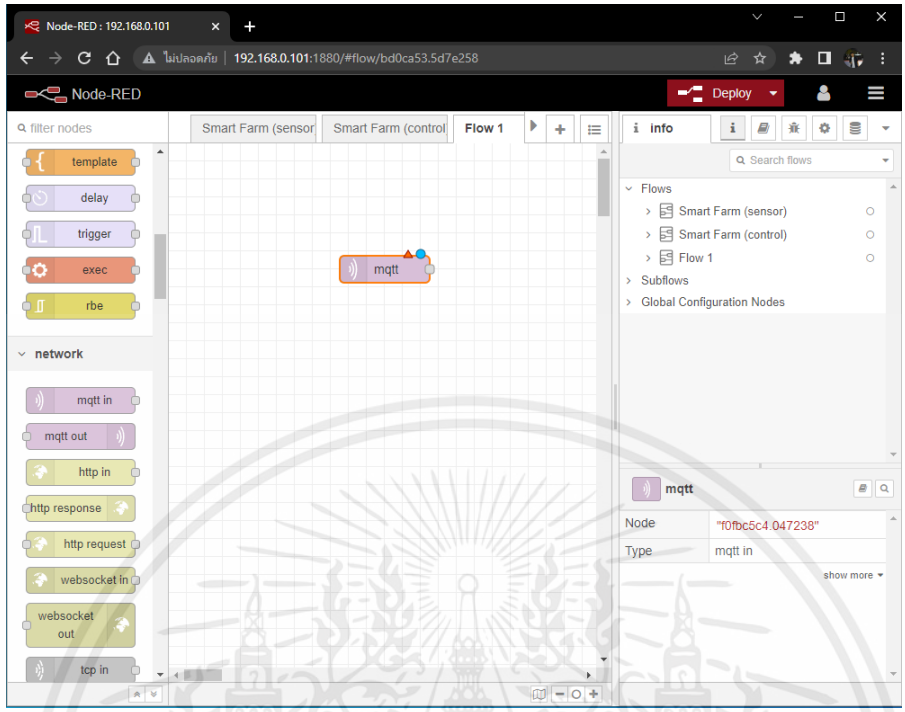
3.5.2.2 การตั้งค่า node-red ในเกตเวย์



รูปที่ 3.45 ภาพรวมการส่งข้อมูลอุปกรณ์สั่งการบนเกตเวย์

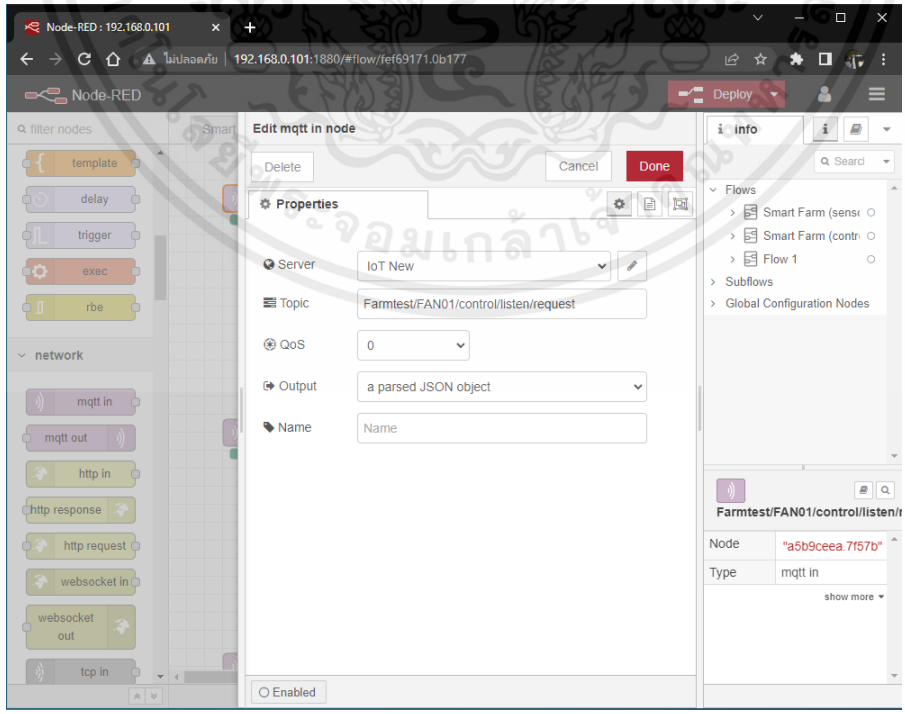
การตั้งค่า node-red ใน IoT Gateway เพื่อการส่งข้อมูลเพื่อควบคุมอุปกรณ์สั่งการ จะเริ่มจากการตั้งค่าการ MQTT in Node จากนั้นตั้งค่า Function Node เขียนฟังก์ชันเพื่ออ่าน เก็บค่า และการตั้งค่าตัวแปรเพื่อสื่อสารระหว่างอุปกรณ์ IoT Gateway กับเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรมได้ โดยตั้งค่า Modbus Write Node และใช้มาตรฐานการสื่อสาร Modbus TCP/IP จากนั้นตั้งค่า Function Node เขียนฟังก์ชันเพื่ออ่าน เก็บค่า และการตั้งค่าตัวแปรเพื่อสื่อสารระหว่างอุปกรณ์ IoT Gateway กับ AWS IoT Core โดยใช้มาตรฐานการสื่อสาร MQTT หลังจากนั้น publish MQTT ไปยัง AWS IoT Core โดย MQTT out Node

3.5.2.2.1 อ่านข้อมูลอุปกรณ์สั่งการจาก AWS IoT Core ด้วย Subscribe MQTT Protocol



รูปที่ 3.46 mqtt in Node ในการอ่านข้อมูลอุปกรณ์สั่งการบนเกตเวย์

เลือกหา mqtt in Node ใน netword Node ลากมาวางใน Flow แล้ว Double Click ที่ตัว Node

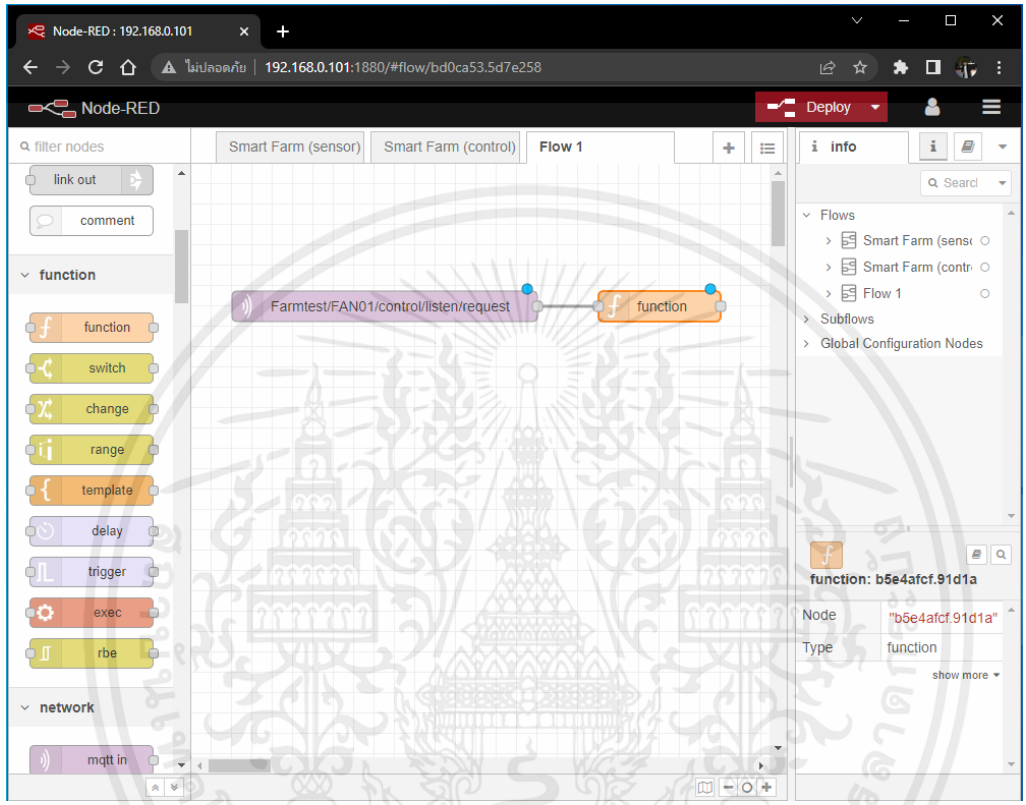


รูปที่ 3.47 การอ่านข้อมูลอุปกรณ์สั่งการบนเกตเวย์

จะปรากฏหน้าต่างการตั้งค่า mqtt in Node โดยให้ตั้งค่าตามดังนี้

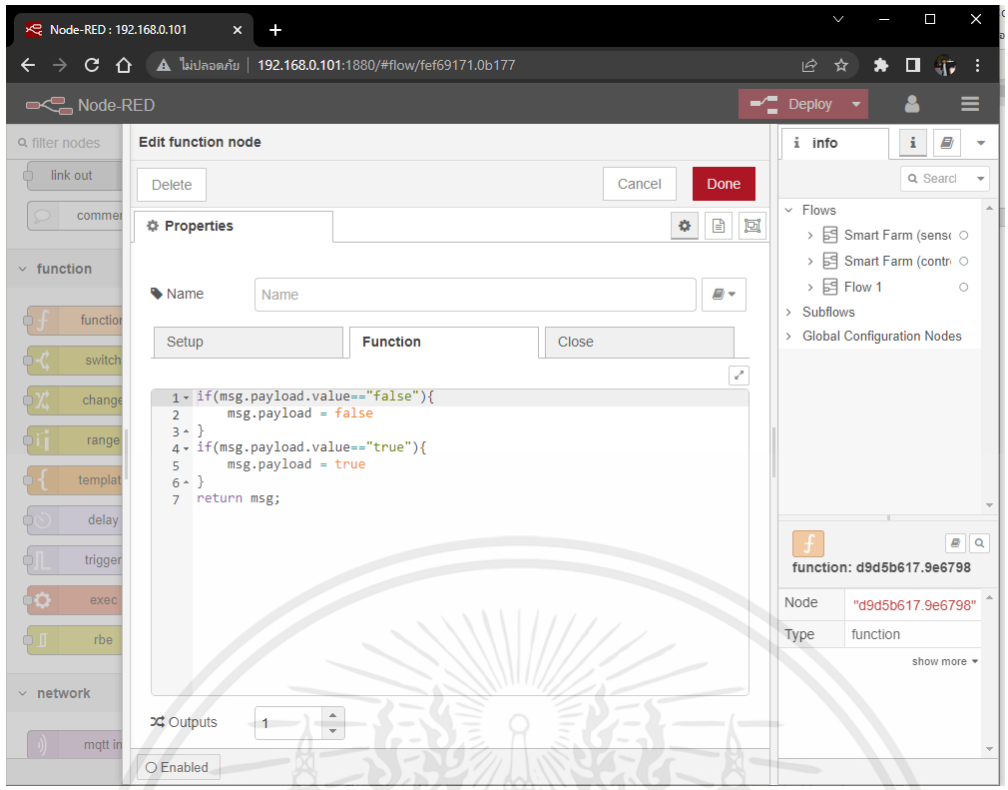
1) เลือก Server ที่สร้างขึ้นมาแล้วในการอ่านข้อมูลอุปกรณ์ sensor
2) Topic: Farmtest/FAN01/control/listen/request
   (ชื่อฟาร์ม/Serial Number ของอุปกรณ์/control/listen/request)

หลังจากกดปุ่ม Done



รูปที่ 3.48 การเชื่อมต่อระหว่าง Function Node กับ mqtt in Node
ในการอ่านข้อมูลอุปกรณ์สั่งการบนเกตเวย์

เลือกหา Function Node ใน Function ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง Function Node กับ mqtt in Node หลังจากนั้นกด Double Click ที่ตัว Node

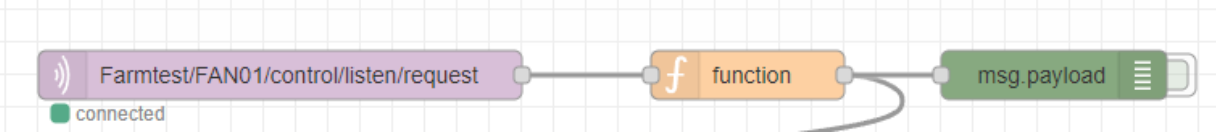


รูปที่ 3.49 การตั้งค่า Function node ในการอ่านข้อมูลอุปกรณ์สั่งการบนเกตเวย์

หลังจะปรากฏการตั้งค่า Function node ให้เขียนโปรแกรมตามดังนี้

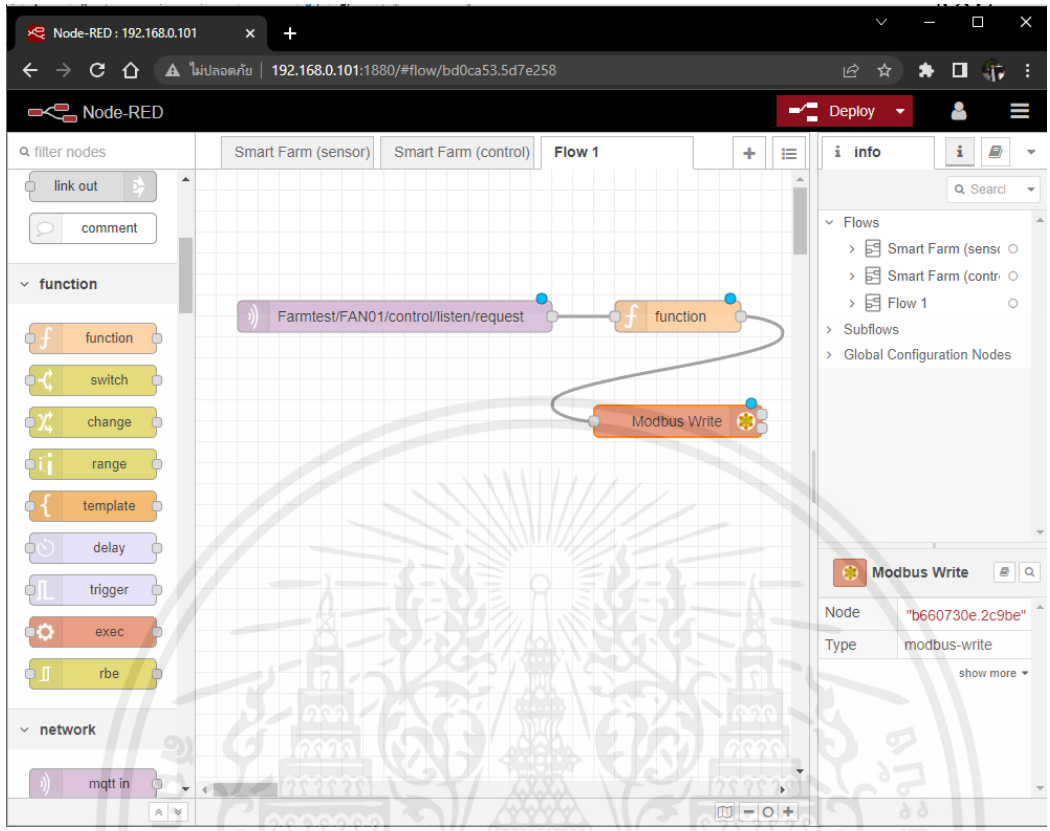
```
1) if(msg.payload.value=="false"){
2)         msg.payload = false
3) }
4) if(msg.payload.value=="true"){
5)          msg.payload = true
6) }
7) return msg;
```

หลังจากกดปุ่ม Done



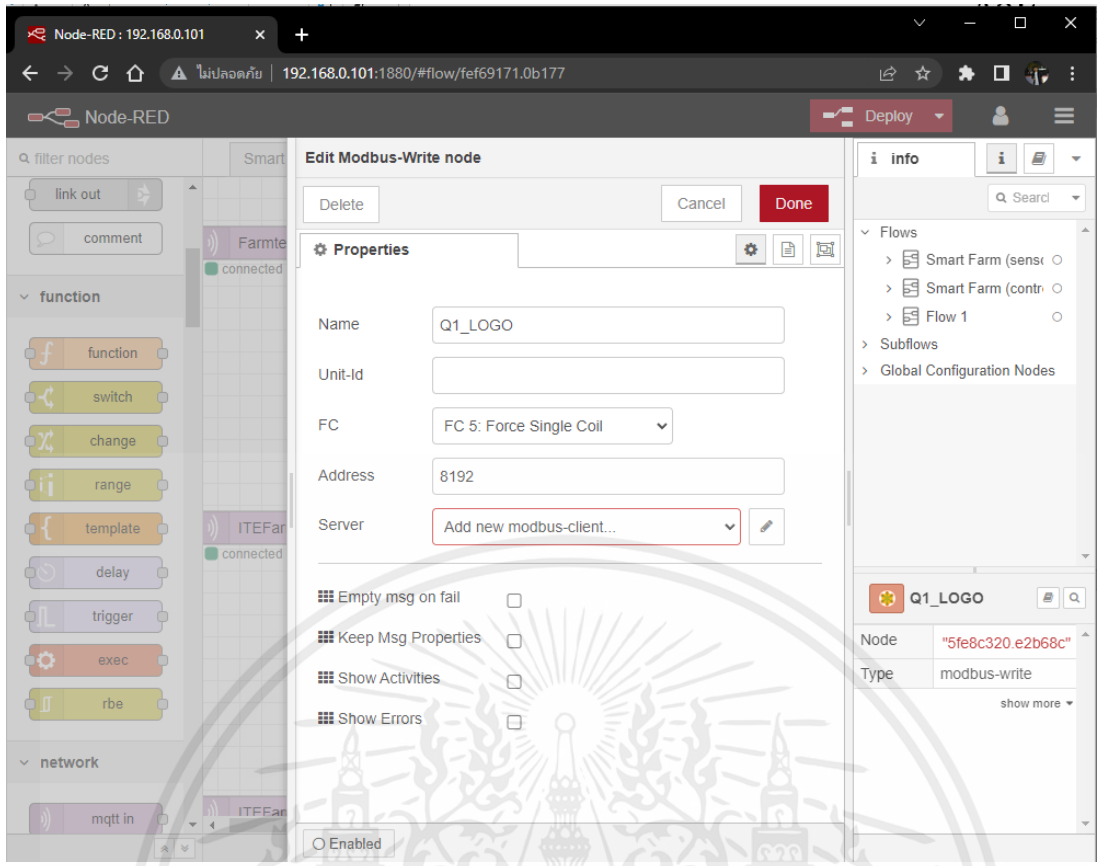
รูปที่ 3.50 การอ่านข้อมูลอุปกรณ์สั่งการบนเกตเวย์

3.5.2.2.2 ส่งข้อมูลอุปกรณ์สั่งการไปยัง Controller ด้วย Modbus TCP/IP Protocol



รูปที่ 3.51 การเชื่อมต่อระหว่าง Modbus Write Node กับ Function Node ในส่งข้อมูลอุปกรณ์สั่งการไปยัง Controller

เลือกหา Modbus Write Node ใน Modbus Node ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง Modbus Write Node กับ Function Node หลังจากนั้นกด Double Click ที่ตัว Node ที่สร้างมาใหม่

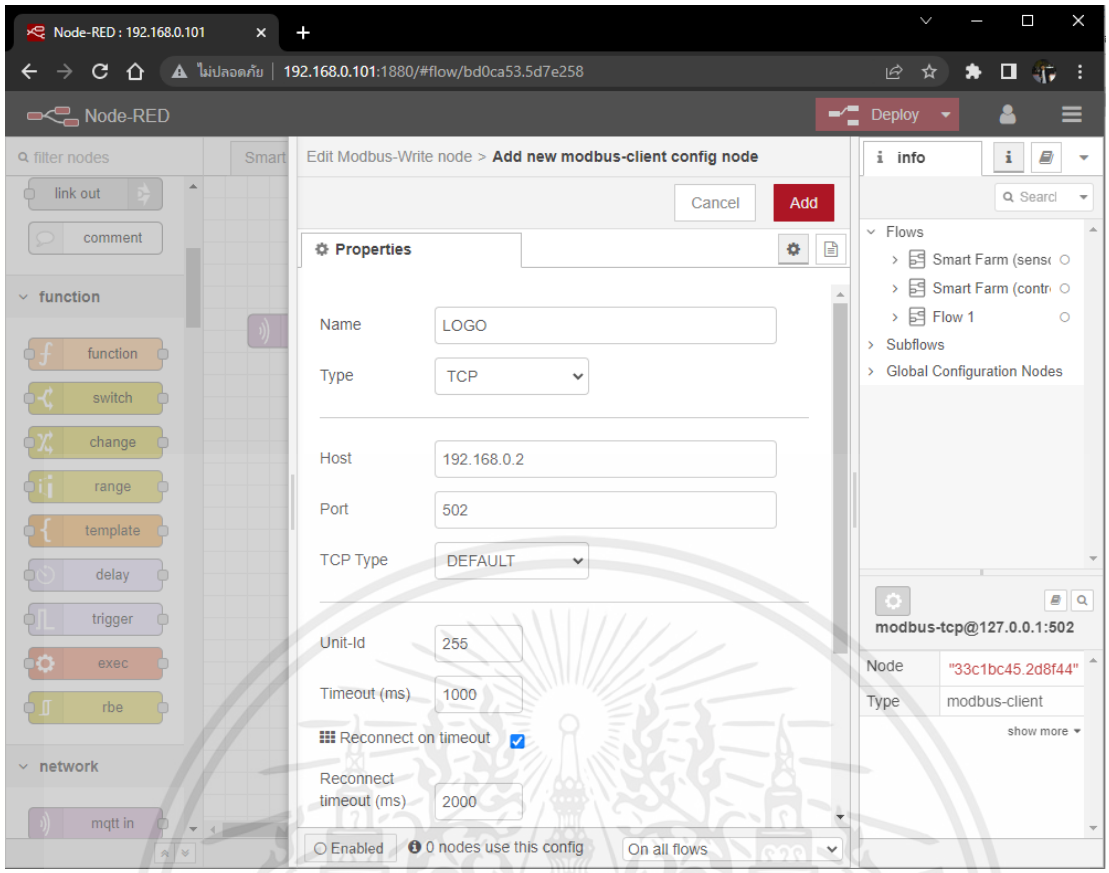


รูปที่ 3.52 การตั้งค่า Modbus-Write Node ในส่งข้อมูลอุปกรณ์สั่งการไปยัง Controller

หลังจากนั้นจะปรากฏการตั้งค่า Modbus-Write Node ให้ตั้งค่าตามข้อมูลเฉพาะของ PLC Siemens LOGO Controller ดังนี้

1) ตั้งชื่อ: Q1_LOGO
2) FC 5: Force Single Coil
3) Address: 8192

หลังจาก Server ให้เลือกเป็น Add new modbus-client และกดปุ่มแก้ไขเพื่อสร้างมาตรฐานการสื่อสารด้วย Modbus TCP/IP Protocol

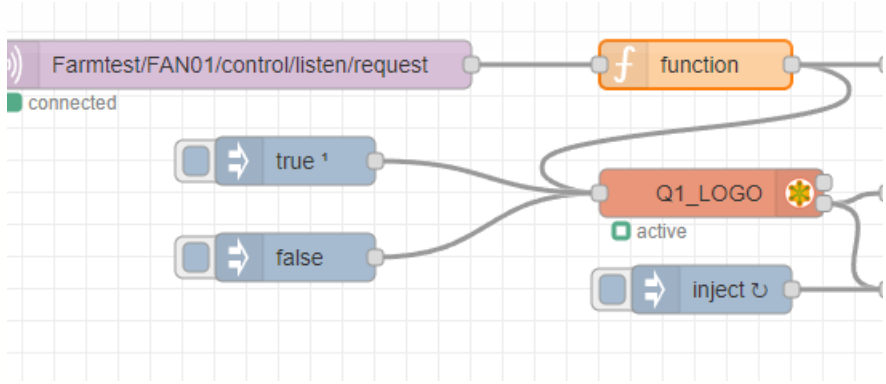


รูปที่ 3.53 การตั้งค่า Modbus-client config node ในส่งข้อมูลอุปกรณ์สั่งการไปยัง Controller

หลังจากนั้นจะปรากฏการตั้งค่า Modbus-client config node ให้ตั้งค่าตามดังนี้

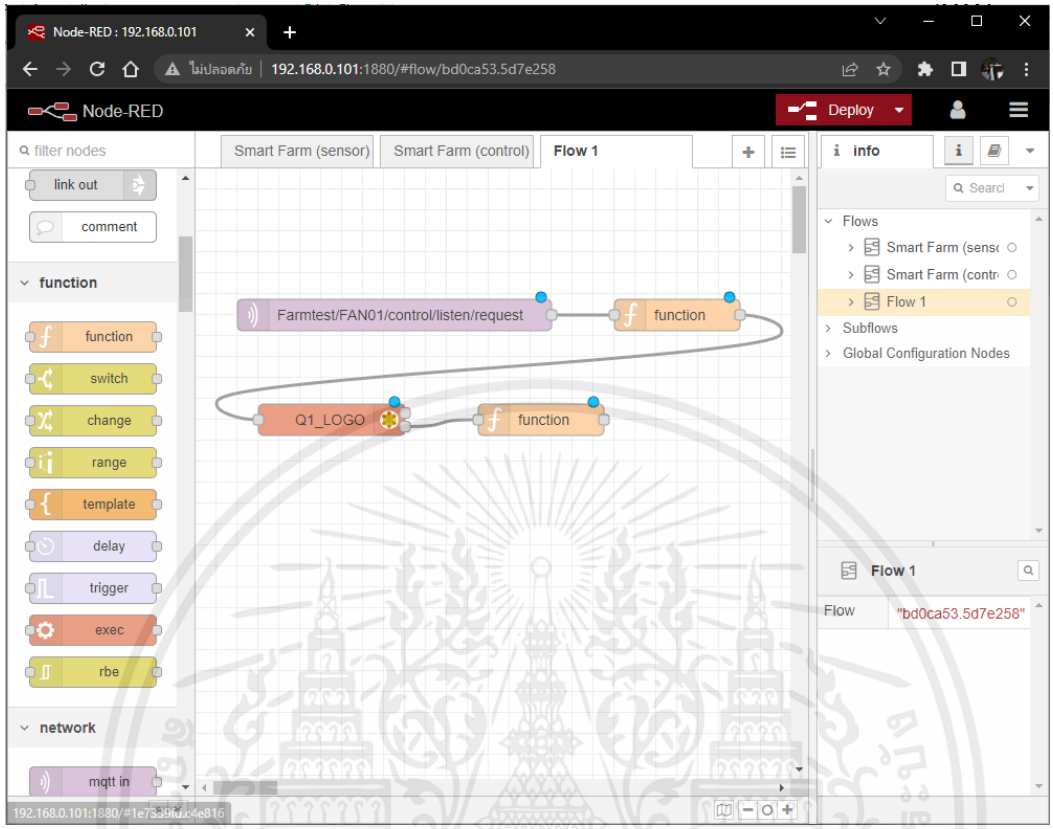
1) ตั้งชื่อ: LOGO
2) Type: TCP
3) Host: 192.168.0.2
4) Port 502
5) Unit-Id: 255

หลังจากกดปุ่ม Add



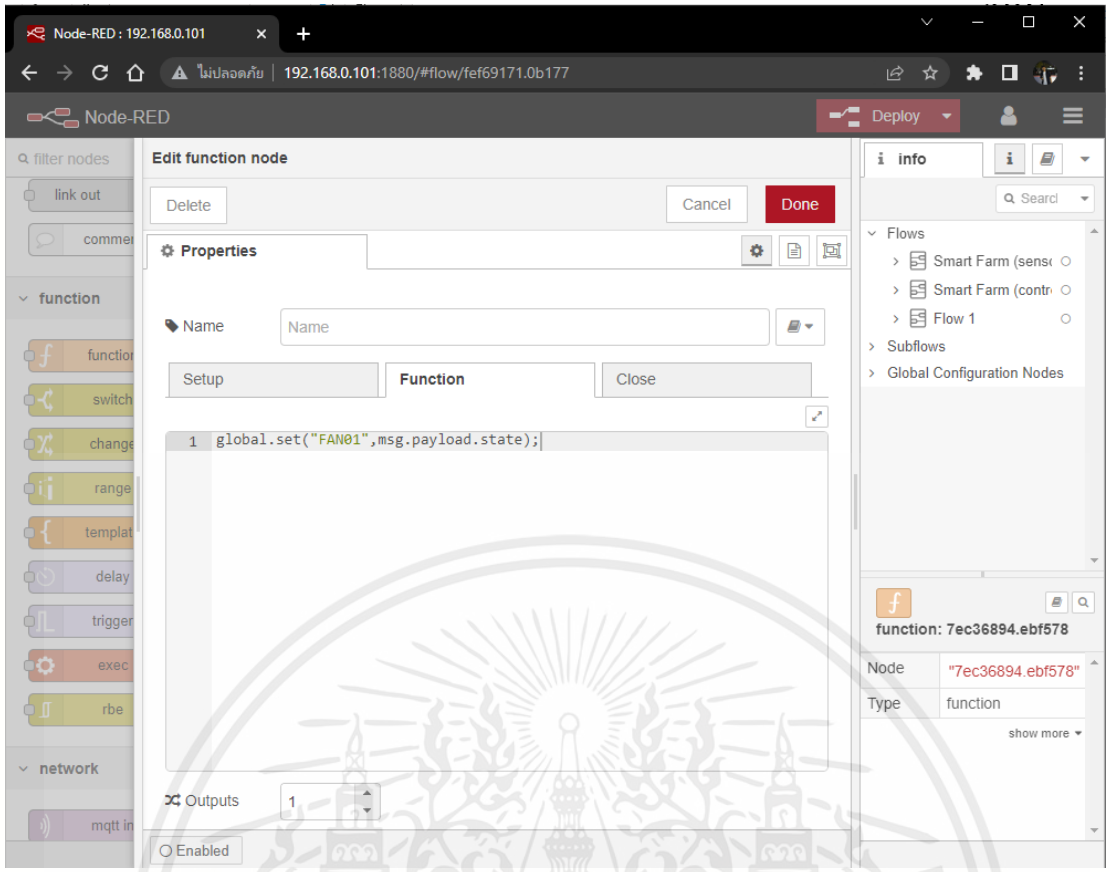
รูปที่ 3.54 การส่งข้อมูลอุปกรณ์สั่งการไปยัง Controller

3.5.2.2.3 ส่งข้อมูลอุปกรณ์สั่งการไปยัง AWS IoT Core ด้วย Publish MQTT Protocol



รูปที่ 3.55 การเชื่อมต่อระหว่าง Function Node กับ Modbus-Write Node ในการส่งข้อมูลอุปกรณ์สั่งการไปยัง AWS IoT Core

เลือกหา Function Node ใน Function ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง Function Node กับ Modbus-Write Node หลังจากนั้นกด Double Click ที่ตัว Node ที่สร้างมาใหม่

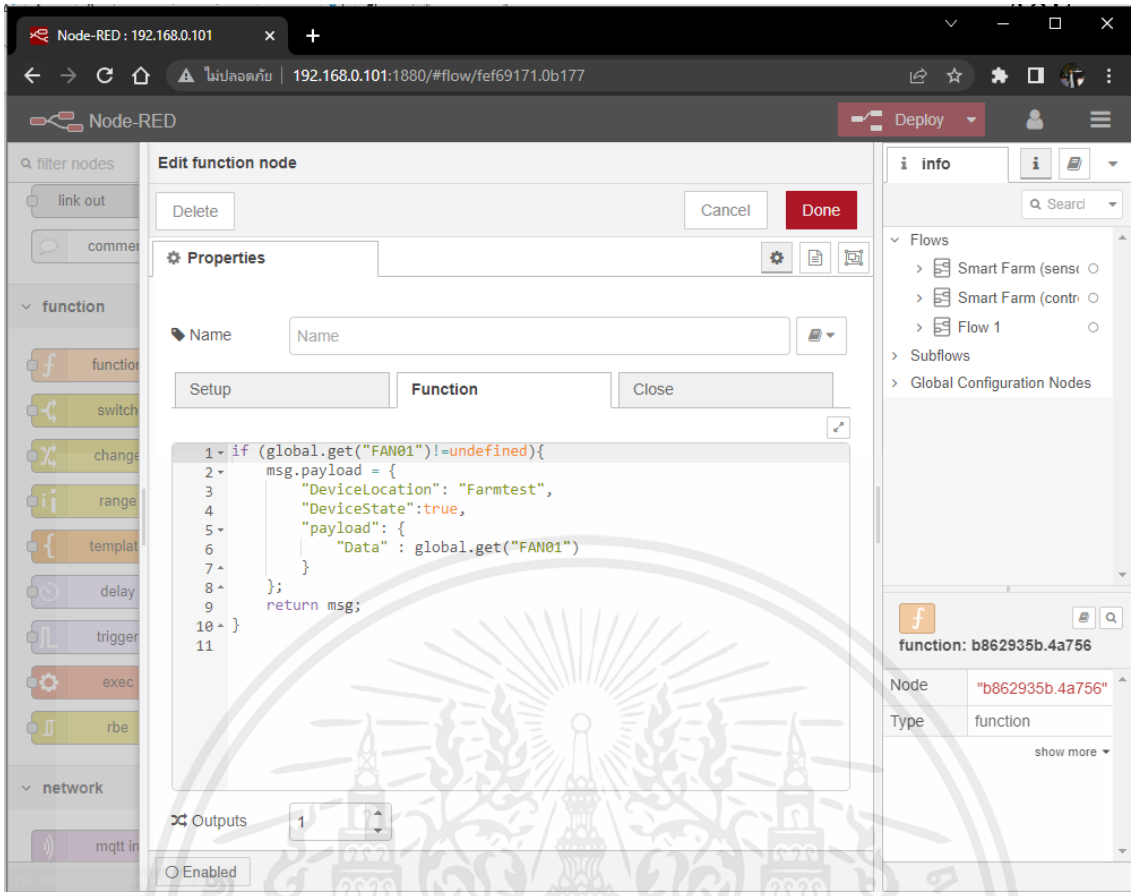


รูปที่ 3.56 การตั้งค่า Function node ในการส่งข้อมูลอุปกรณ์สั่งการไปยัง AWS IoT Core

หลังจะปรากฏการตั้งค่า Function node ให้เขียนโปรแกรมตามดังนี้

1) global.set("FAN01",msg.payload.state);

หลังจากกดปุ่ม Done และเลือกหา Function Node ใน Function ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง Function Node กับ Modbus-Write Node หลังจากนั้นกด Double Click ที่ตัว Node ที่สร้างมาใหม่

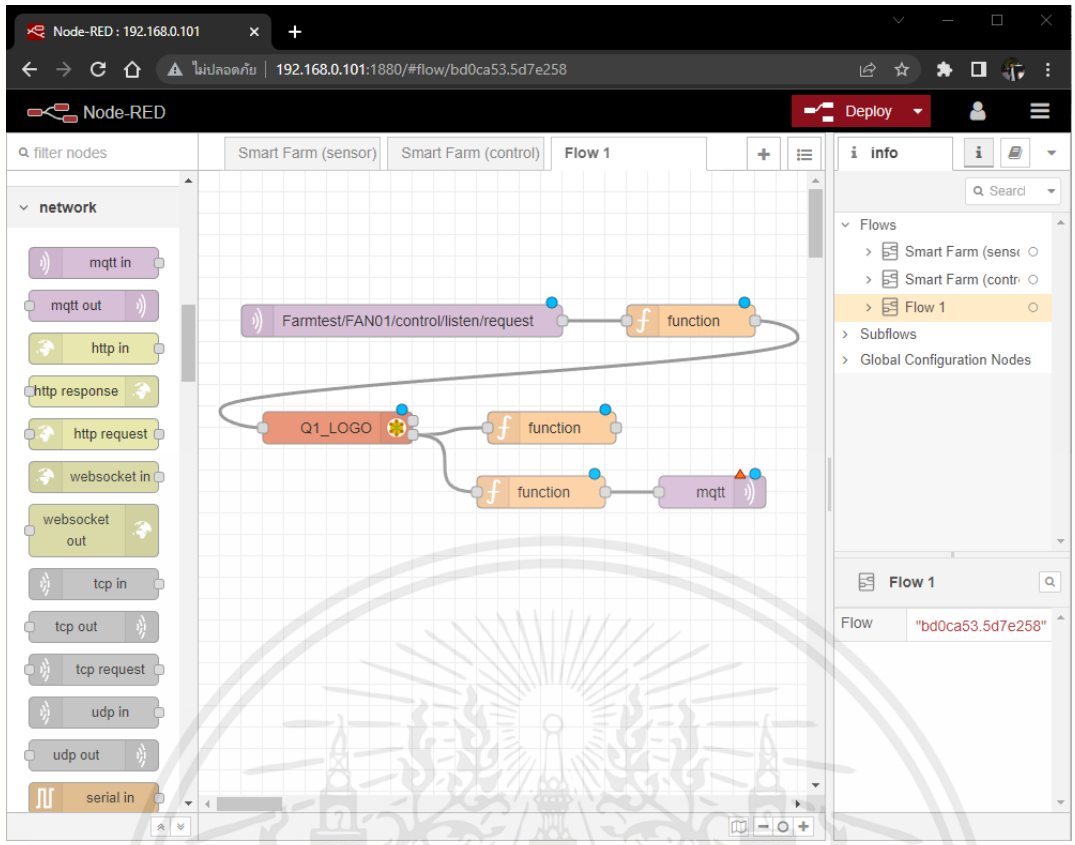


รูปที่ 3.57 การตั้งค่า Function node 2 ในการส่งข้อมูลอุปกรณ์สั่งการไปยัง AWS IoT Core

หลังจะปรากฏการตั้งค่า Function node ให้เขียนโปรแกรมตามดังนี้

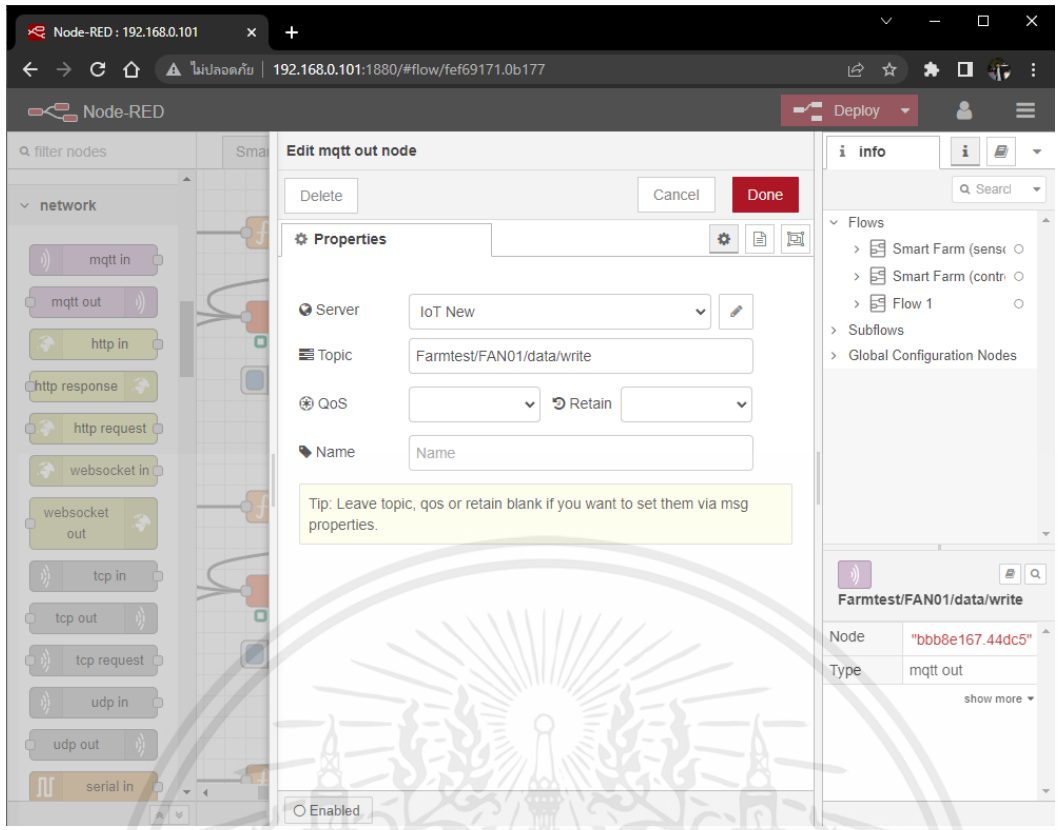
```
1)  if (global.get("FAN01")!=undefined){
2)          msg.payload = {
3)         "DeviceLocation": "Farmtest",
4)         "DeviceState":true,
5)              "payload": {
a.                  "Data" : global.get("FAN01")
6)              }
7)         };
8)         return msg;
9)  }
```

หลังจากกดปุ่ม Done



รูปที่ 3.58 การเชื่อมต่อระหว่าง mqtt out Node กับ Function Node ในการส่งข้อมูลอุปกรณ์สั่งการไปยัง AWS IoT Core

เลือกหา mqtt out Node ใน Network Node ลากมาวางใน Flow และลากสายเชื่อมต่อระหว่าง mqtt out Node กับ Function Node หลังจากนั้นกด Double Click ที่ตัว Node ที่สร้างมาใหม่

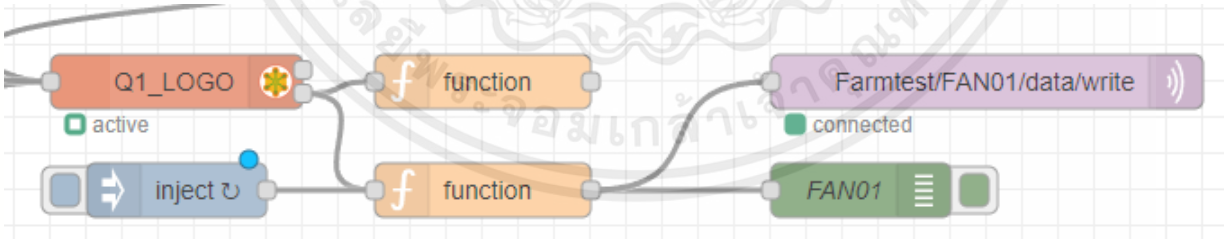


รูปที่ 3.59 การตั้งค่า mqtt out node ในการส่งข้อมูลอุปกรณ์สั่งการไปยัง AWS IoT Core

หลังจากนั้นจะปรากฏการตั้งค่า mqtt out node ให้ตั้งค่าตามดังนี้

1) เลือก Server ที่สร้างขึ้นมาแล้วในการอ่านข้อมูลอุปกรณ์ sensor

2) Topic: Farmtest/FAN01/data/write (ชื่อฟาร์ม/Serial Number ของอุปกรณ์ Sensor/data/write) หลังจากกดปุ่ม Done



รูปที่ 3.60 การส่งข้อมูลอุปกรณ์สั่งการไปยัง AWS IoT Core

บทที่ 4
ผลการดำเนินงาน

4.1 การทดลองในส่วนอุปกรณ์เพื่อจัดการฟาร์ม

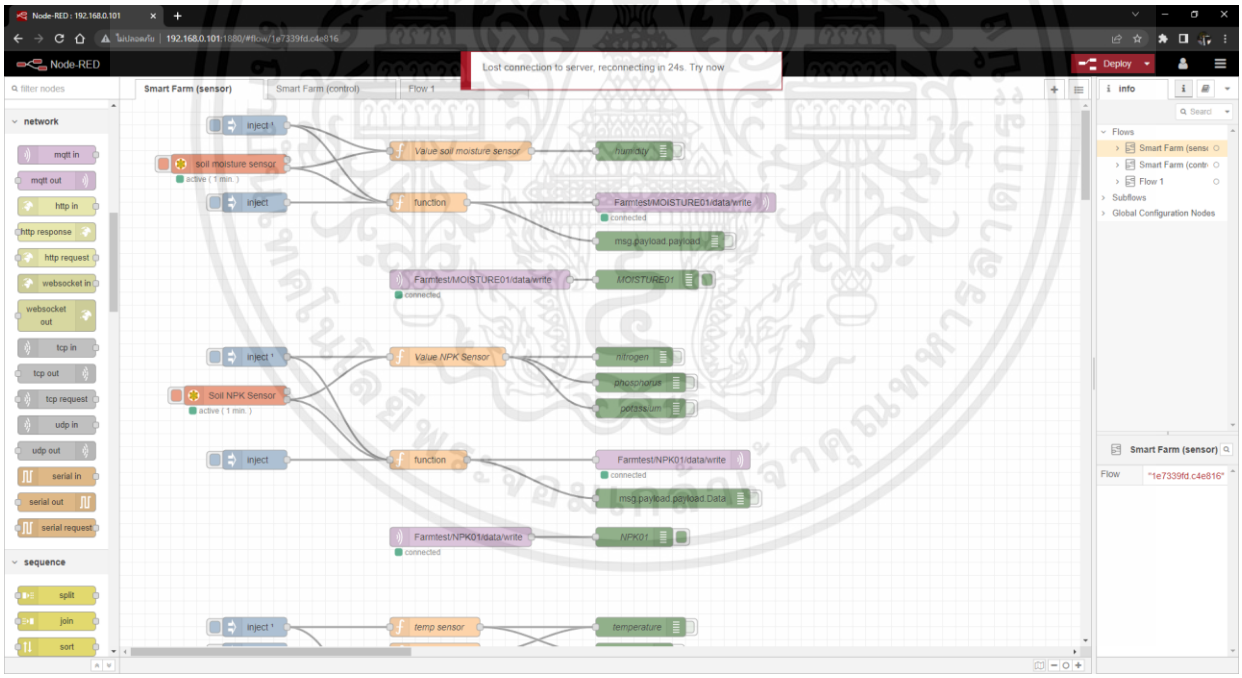
เชื่อมต่ออุปกรณ์ต่างๆ ภายในฟาร์ม โดยติดตั้งอุปกรณ์เซ็นเซอร์วัดความชื้นในดินและอุปกรณ์วัดเซ็นเซอร์ดธาตุสารอาหารในดินไว้ในโรงเรือนปลูกผักจำลอง และติดตั้งพัดลมเป็นอุปกรณ์สั่งการในตู้ควบคุม นอกจากนั้นยังมีอุปกรณ์ที่เอาไว้ควบคุม PLC Siemens LOGO Controller และ IoT2050 gateways พร้อมเสารับ-ส่งสัญญาณอินเตอร์เน็ต โดยทำงานเป็น Edge Computing รวมถึงการจัดการข้อมูลอ่านค่าและควบคุมอุปกรณ์ โดยใช้มาตรฐานการสื่อสารในอุปกรณ์ต่างๆ คือ MQTT Protocol เชื่อมต่อกับ AWS IoT Core เพื่อบันทึกข้อมูลในฐานข้อมูล AWS DynamoDB, RS485 Modbus RTU Protocol เชื่อมต่อกับอุปกรณ์เซนเซอร์, Modbus TCP/IP Protocol เชื่อมต่อกับคอนโทรลเลอร์ (PLC Siemens LOGO) เพื่อควบคุมการเปิด-ปิดของพัดลม นอกจากนี้ยังมีตัวแปลงไฟ AC เป็น 12 VDC จ่ายพลังงานให้กับอุปกรณ์ทั้งระบบอีกด้วย

รูปที่ 4.1 ภาพรวมของการเชื่อมต่ออุปกรณ์ภายในฟาร์ม

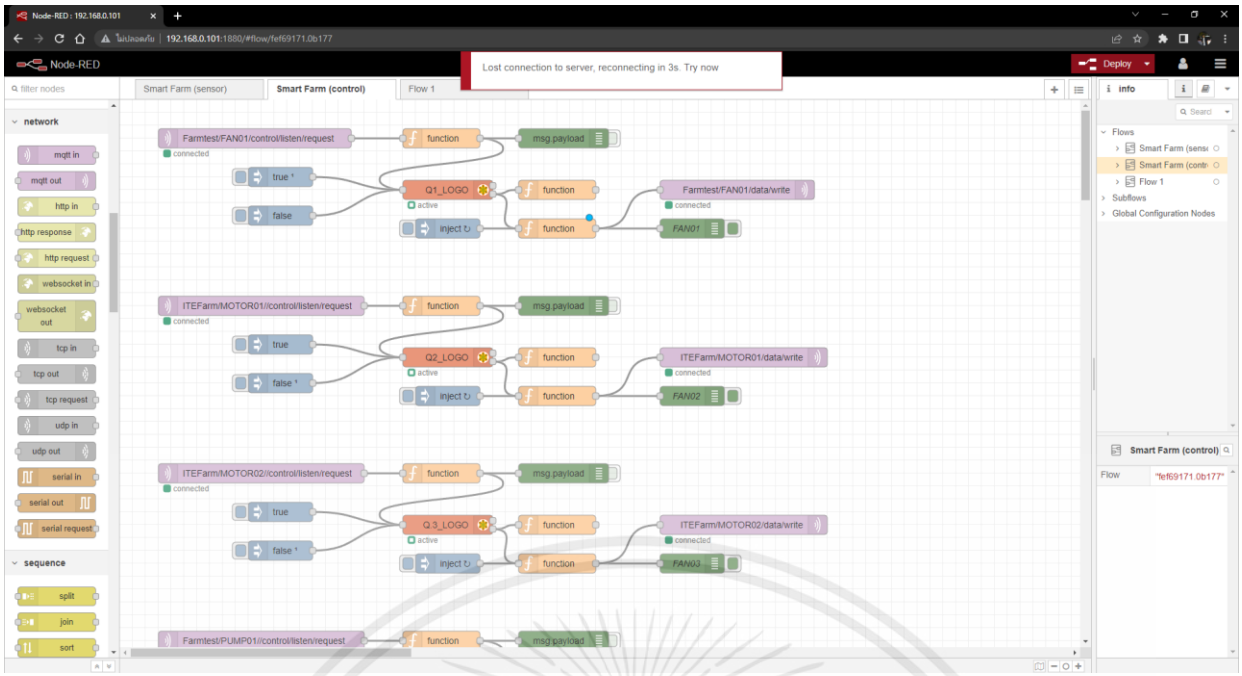
โดยทางซ้ายมือจะเป็นการจำลองโรงเรือนปลูกผัก และทางขวามือจะเป็นตู้ควบคุมภายในฟาร์ม

รูปที่ 4.2 ตู้ควบคุมภายในฟาร์ม

โดยมีในตู้จะประกอบไปด้วย PLC Siemens LOGO Controller, Siemens IoT2050 gateways พร้อมเสารับ-ส่งสัญญาณอินเตอร์เน็ต, พัดลม และตัวแปลงไฟ AC เป็น 12 VDC จ่ายพลังงานให้กับอุปกรณ์ทั้งระบบ

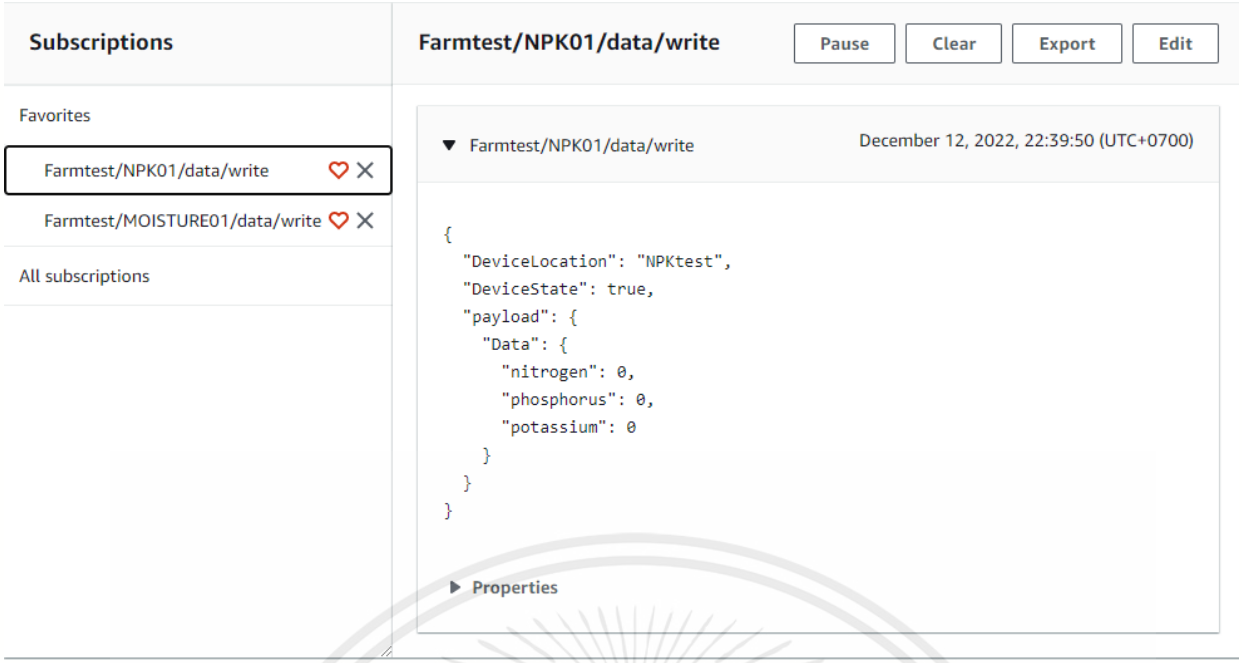


รูปที่ 4.3 ภาพรวมของการเชื่อมต่อในการอ่านค่าอุปกรณ์เซนเซอร์บน node-red ของเกตเวย์

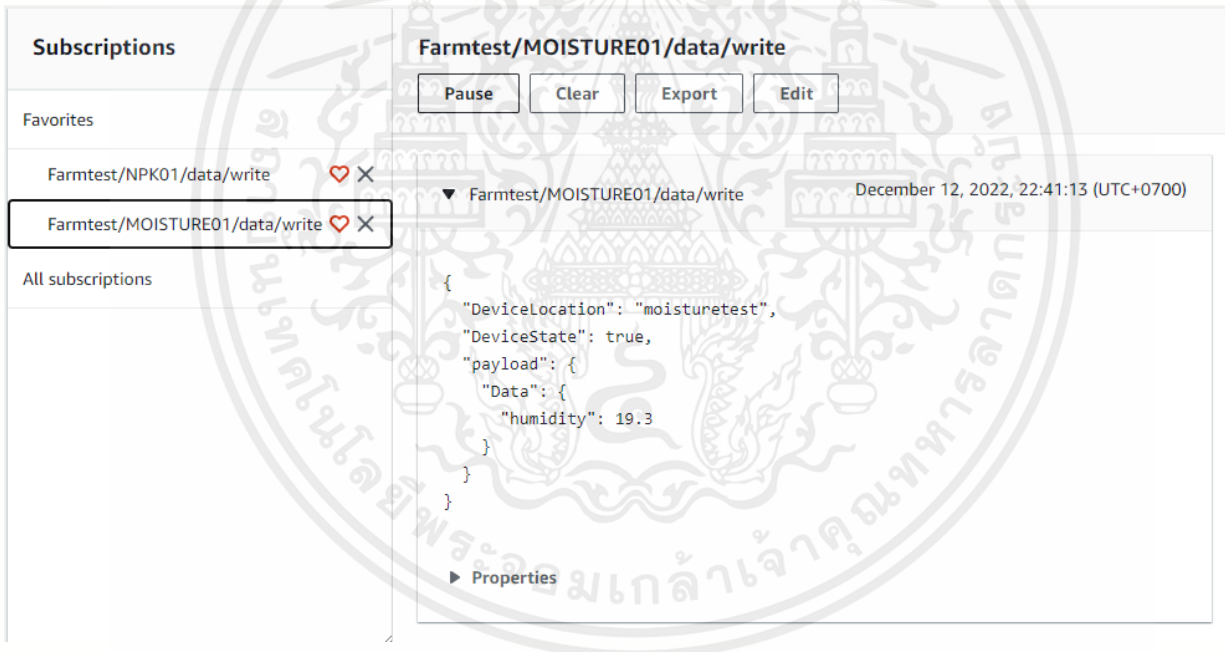


รูปที่ 4.4 ภาพรวมของการเชื่อมต่อในการควบคุมอุปกรณ์สั่งการบน node-red ของเกตเวย์

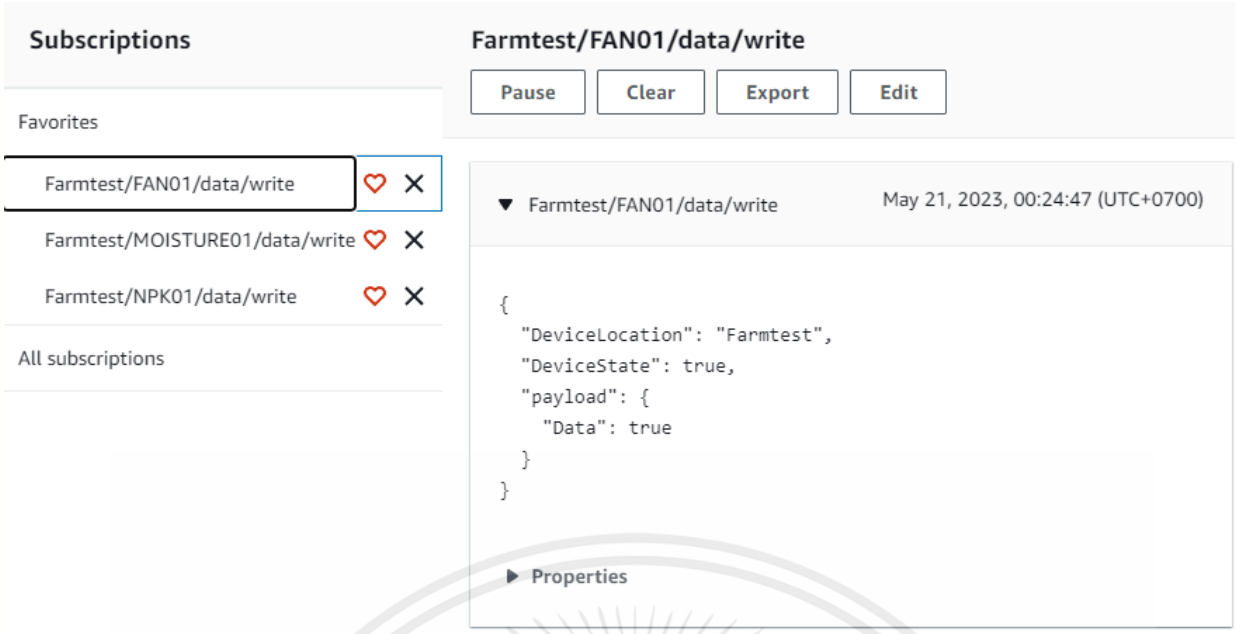
รูปที่ 4.5 แสดงค่าข้อมูลของอุปกรณ์ที่อ่านได้จาก node-red ของเกตเวย์



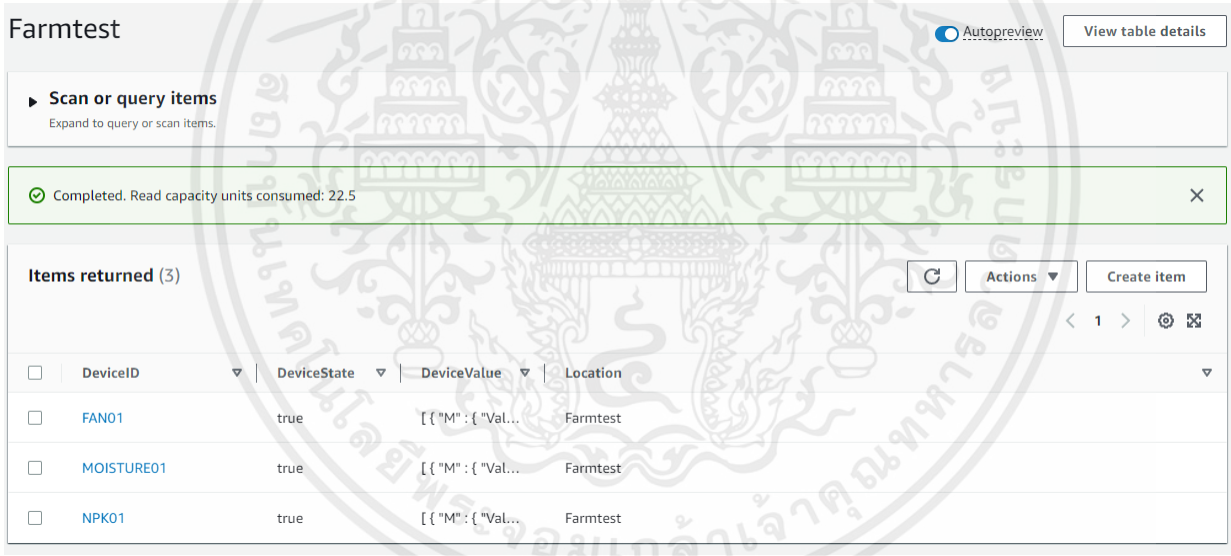
รูปที่ 4.6 แสดงค่าข้อมูลของอุปกรณ์วัดธาตุสารอาหารในดินที่อ่านได้จาก AWS IoT Core



รูปที่ 4.7 แสดงค่าข้อมูลของอุปกรณ์วัดความชื้นในดินที่อ่านได้จาก AWS IoT Core



รูปที่ 4.8 แสดงค่าข้อมูลของอุปกรณ์สั่งการที่อ่านได้จาก AWS IoT Core



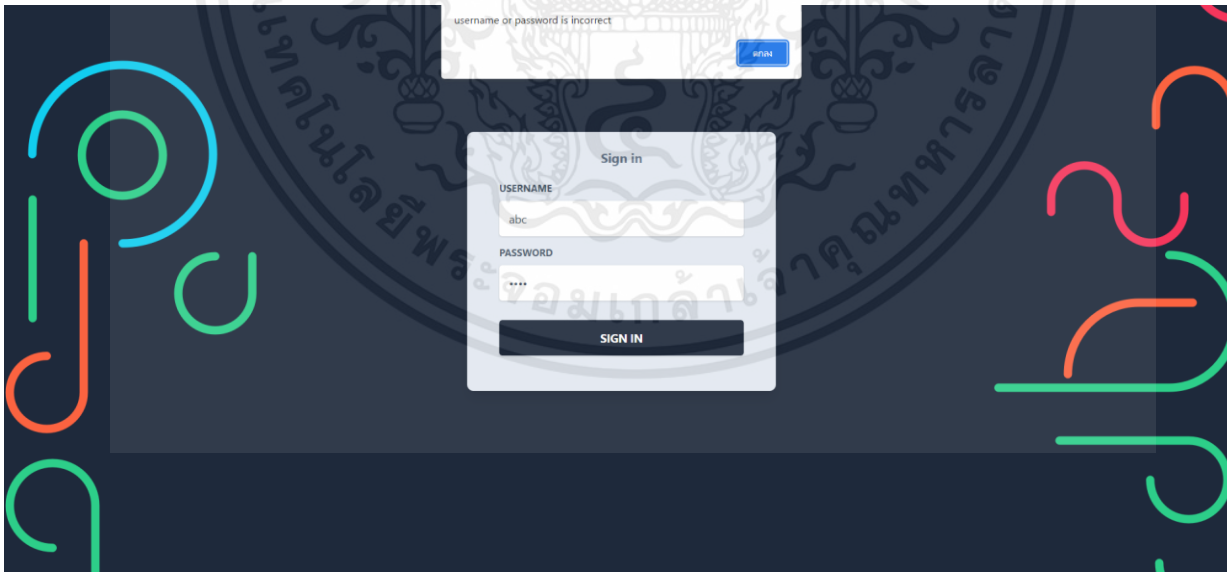
รูปที่ 4.9 แสดงข้อมูลอุปกรณ์ของฐานข้อมูลในฟาร์มบน AWS DynamoDB

4.2 การทำงานของเว็บแอปพลิเคชัน

4.2.1 หน้าการเข้าสู่ระบบ

ในการเริ่มใช้งานเว็บแอปพลิเคชัน ผู้ใช้จะถูกนำมาที่หน้าเข้าสู่ระบบและไม่สามารถไปที่หน้าอื่นๆได้ จนกว่าผู้ใช้เข้าระบบสำเร็จ โดยผู้ใช้ต้องระบุ username และpassword ที่ลงทะเบียนไว้เพื่อเข้าระบบ เมื่อใส่ username และ password แล้วให้กด SIGH IN ถ้าข้อมูลถูกต้องจะไปสู่หน้าแดชบอร์ดแต่ถ้าข้อมูลไม่ถูกต้องจะมีแจ้งเตือนขึ้นมาว่า “username or password is incorrect”

รูปที่ 4.10 หน้าเข้าสู่ระบบ

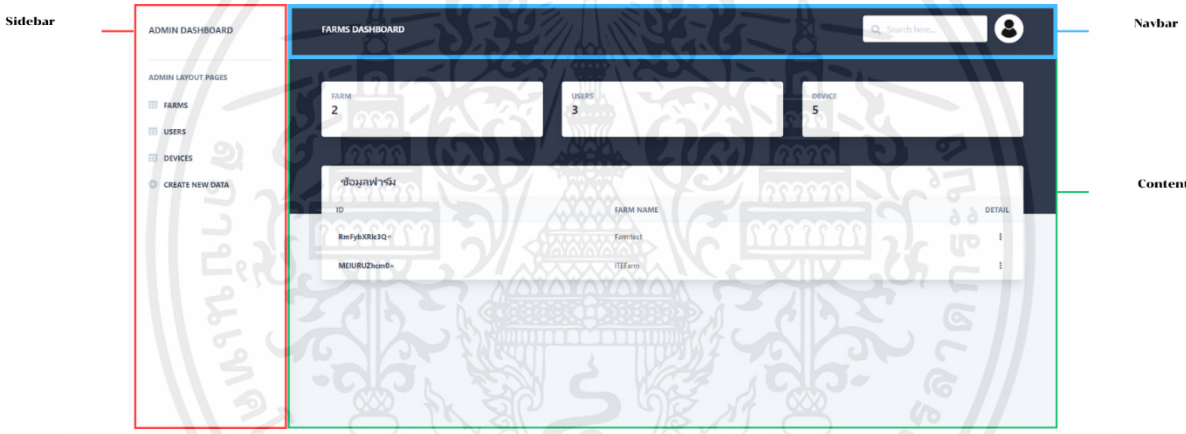


รูปที่ 4.11 แจ้งเตือนข้อมูลไม่ถูกต้อง

4.2.2 หน้าแดชบอร์ด

ในหน้าแดชบอร์ดจะมีอยู่ 3 ส่วนสำคัญ ดังนี้

1) Content คือส่วนที่แสดงข้อมูลต่างๆโดยเริ่มต้นจะแสดงของข้อมูลต่างๆของฟาร์มในรูปแบบของตารางและจะมีปุ่มป็อปอัพเพื่อให้ผู้ใช้สามารถดูข้อมูลเพิ่มเติม, แก้ไขและลบข้อมูลได้ โดยการคลิกที่ Detail จะไปที่หน้าแสดงรายละเอียด การคลิกที่ Edit จะไปที่หน้าแก้ไขข้อมูล การคลิกที่ Delete จะเป็นการลบข้อมูลตัวนั้นไป ซึ่งผู้ใช้สามารถเปลี่ยนข้อมูลที่แสดงใน content ได้ในส่วนของ sidebar
2) Sidebar คือส่วนที่ผู้ใช้คลิกเพื่อเปลี่ยนไปหน้าอื่นๆซึ่งจะมีส่วนของ content ที่จะไม่เหมือนกันโดยจะมีข้อมูลฟาร์ม, ข้อมูลอุปกรณ์,ข้อมูลผู้ใช้และสร้างข้อมูลใหม่
3) Navbar คือส่วนที่ผู้ใช้สามารถค้นหาข้อมูลที่ต้องการได้โดยพิมพ์ข้อมูลที่ต้องการค้นหาไปในช่อง Search แล้วกด Enter ข้อมูลที่ค้นหาก็จะมาแสดงในส่วนของ content และผู้ใช้สามารถออกจากระบบโดยคลิกที่ปุ่มรูปคนแล้วเลือก Logout เพื่อออกจากระบบแล้วไปที่หน้าเข้าสู่ระบบ



รูปที่ 4.12 หน้าแดชบอร์ด

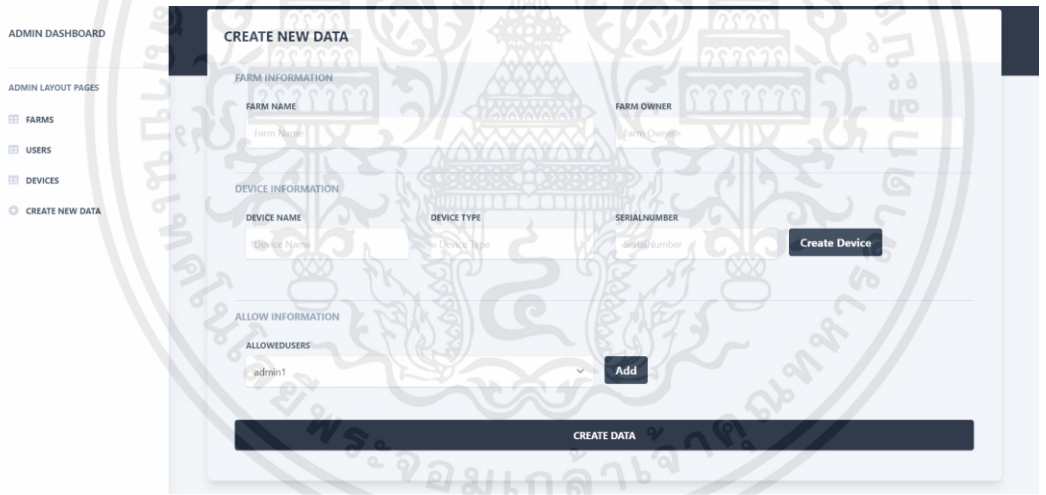
รูปที่ 4.13 ปุ่มป็อปอัพตัวเลือกและป็อปอัพ Logout

4.2.3 หน้าสร้างข้อมูลใหม่

เมื่อคลิกที่ CREATE NEW DATA ที่ส่วนของ Sidebar จะทำให้ส่วนของ Content มาที่หน้าสร้างข้อมูลใหม่โดยในหน้าสร้างข้อมูลใหม่จะให้ผู้ใช้ระบุข้อมูลต่างๆที่จะต้องการดังนี้

- Farm Name ชื่อฟาร์ม
- Farm Owner ชื่อเจ้าของฟาร์ม
- Device Name ชื่ออุปกรณ์
- Device Type ประเภทของอุปกรณ์
- Serial Number หมายเลขผลิตภัณฑ์ของอุปกรณ์
- Allowed Users ผู้ใช้ที่ต้องการให้สามารถเข้าถึงฟาร์มนี้ได้

โดยอุปกรณ์และผู้ใช้สามารถเพิ่มได้มากกว่า 1 ตัว ทำได้โดยการคลิก Create Device และ Add ซึ่งจะทำให้เพิ่มข้อมูลเข้าไปได้เรื่อย ๆ หลังจากเพิ่มข้อมูลครบถ้วนแล้วให้คลิกที่ Create Data ก็จะเป็นการสร้างข้อมูลใหม่สำเร็จและกลับไปที่หน้าแดชบอร์ดในตอนแรก



รูปที่ 4.14 หน้าสร้างข้อมูลใหม่

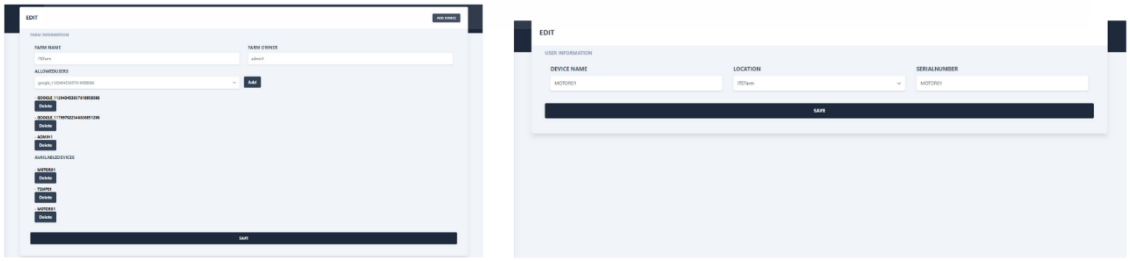
4.2.4 หน้าแสดงรายละเอียดและหน้าแก้ไขข้อมูล

โดยในหน้าแสดงรายละเอียดจะเป็นการแสดงข้อมูลจากที่เราเลือกในหน้าแดชบอร์ดซึ่งก็จะมีข้อมูลแตกต่างกันตามที่เลือกดูเช่นถ้าดูข้อมูลอุปกรณ์ก็จะแสดงข้อมูลเช่น ชื่อ, ประเภท, หมายเลขผลิตภัณฑ์ เป็นต้น แต่ถ้าเลือกดูข้อมูลฟาร์มก็จะแสดงข้อมูลเช่น ผู้ใช้ภายในฟาร์ม ,อุปกรณ์ภายในฟาร์ม เป็นต้น และเมื่อเราคลิกไปที่ Detail ของอุปกรณ์ต่างๆภายในฟาร์มจะทำให้ไปที่หน้าแสดงค่าของอุปกรณ์

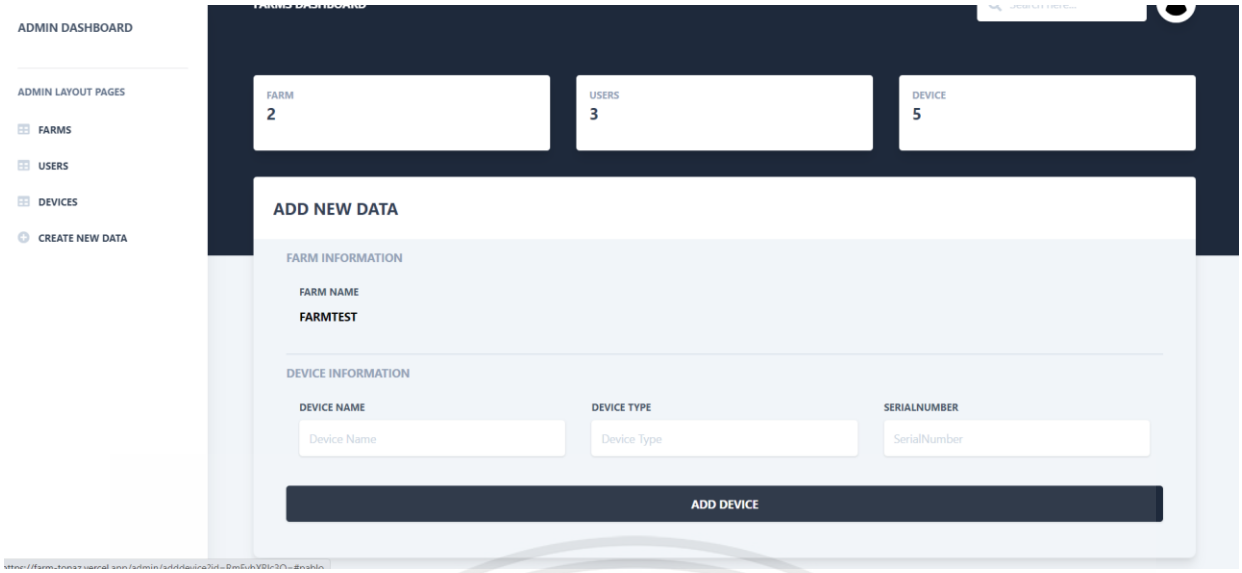
ส่วนหน้าแก้ไขข้อมูลจะแสดงข้อมูลเหมือนกับหน้าแสดงรายละเอียดแต่จะสามารถแก้ไขข้อมูลต่างๆได้ซึ่งแบ่งออกเป็น 2 แบบ

1) แก้ไขข้อมูลเดิมโดยไม่มีอุปกรณ์เพิ่มเข้ามาใหม่ โดยหลังจากจากที่ป้อนข้อมูลใหม่เข้าไปแล้วคลิกที่ปุ่ม Save ก็จะแก้ไขข้อมูลเสร็จสิ้นแล้วกลับไปหน้าแดชบอร์ด
2) แก้ไขข้อมูลเดิมโดยเพิ่มอุปกรณ์เข้ามาใหม่ โดยการคลิกที่ปุ่ม ADD DEVICE หลังจากนั้นจะมีหน้าให้กรอกข้อมูลของอุปกรณ์ตัวใหม่จากนั้นเมื่อคลิกไปที ADD DEVICE ก็จะเป็นการเพิ่มอุปกรณ์ตัวนั้นเข้าไปใหม่แล้วกลับไปที่หน้าแก้ไขข้อมูล

รูปที่ 4.15 หน้าแสดงรายละเอียดของฟาร์มและอุปกรณ์



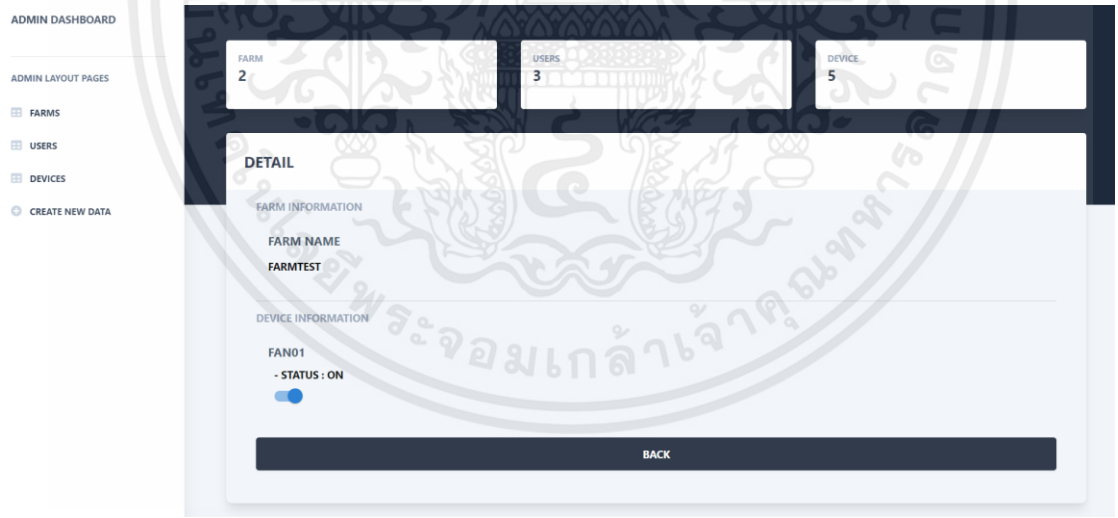
รูปที่ 4.16 หน้าแก้ไขข้อมูลของฟาร์มและอุปกรณ์



รูปที่ 4.17 หน้าเพิ่มอุปกรณ์ตัวใหม่เข้าไปในฟาร์ม

4.2.5 หน้าแสดงค่าของอุปกรณ์

โดยในหน้าแสดงค่าของอุปกรณ์จะเป็นการแสดงข้อมูลค่าต่างๆของอุปกรณ์ที่เราเลือกและในบางอุปกรณ์ที่เราสามารถสั่งเปิด/ปิดได้ก็สามารถสั่งได้จากหน้านี้เช่นกัน โดยการคลิกที่ปุ่มสวิตช์เพื่อควบคุมการเปิด/ปิดของอุปกรณ์ตัวนั้นๆ



รูปที่ 4.18 หน้าอุปกรณ์ที่สามารถควบคุมได้

บทที่ 5
สรุปและอภิปรายผลการดำเนินงาน

การพัฒนาระบบจัดการฟาร์มอัจฉริยะ มีวัตถุประสงค์เพื่อศึกษาการพัฒนาระบบจัดการฟาร์มอัจฉริยะที่สามารถจัดการอุปกรณ์ต่างๆ ภายในฟาร์ม และการพัฒนา Web Application ในการใช้ในเพื่อจัดการข้อมูลในฐานข้อมูล โดยการใช้ Amazon Web Services ในการเก็บข้อมูล แสดงผล และฟังก์ชันบริการบนเซิร์ฟเวอร์ โดยทำให้บริการต่างๆ เชื่อมต่อกัน ได้แก่ การยืนยันตัวตน ฐานข้อมูล REST-API และ MQTT แก้ปัญหาการจัดการระบบภายในฟาร์มในแบบเดิม เพิ่มความสะดวกสบายมากขึ้นและลดความซับซ้อน ซึ่งผู้จัดทำได้สรุปการดำเนินงาน อภิปรายผล และข้อเสนอแนะ ดังต่อไปนี้

5.1 ผลการดำเนินงาน

1) การพัฒนาเว็บแอปพลิเคชัน สำหรับการจัดการระบบฟาร์มอัจฉริยะ โดยใช้ Next.js เป็น React Web Framework ในการพัฒนามีการเชื่อมต่อไปยังระบบอื่น ได้แก่ เซิร์ฟเวอร์ และฮาร์ดแวร์ โดยในแอปพลิเคชันมีความสามารถในการทำงาน ดังนี้
   - การสร้าง ลบ แก้ไข อ่าน ข้อมูลในฐานข้อมูล
   - การแสดงข้อมูลทั้งหมดที่มีในฐานข้อมูล
   - การแสดงผลค่าอุปกรณ์แบบตามเวลาจริงในรูปแบบต่างๆ
   - การควบคุมเปิด/ปิดอุปกรณ์
   - การยืนยันตัวตน
   - การค้นหาข้อมูลในหน้าแดชบอร์ด
2) การพัฒนาฟังก์ชันบนเซิร์ฟเวอร์ AWS ด้วย AWS Lambda ที่ใช้โครงสร้างการทำงานแบบ Serverless เพื่อใช้ในการทำระบบอัตโนมัติที่รองรับการทำงานจากฮาร์ดแวร์และไปเชื่อมต่อกับบริการอื่นๆ โดยมีฟังก์ชันการทำงานหลักดังต่อไปนี้
   - การบันทึกค่าจากอุปกรณ์ลงฐานข้อมูลที่ AWS DynamoDB
   - การรับค่าและส่งต่อไปที่ฮาร์ดแวร์สำหรับการควบคุมอุปกรณ์
   - การจัดการข้อมูลในฐานข้อมูล เช่น การดึงข้อมูลฟาร์ม, การสร้างตารางฟาร์มสำหรับการเก็บค่าตามเวลา เป็นต้น
   - การจัดการข้อมูลในขณะที่กำลังมีการส่งค่าจากฮาร์ดแวร์เพื่อใช้ในการแสดงผลบนแอปพลิเคชัน

3) การพัฒนาอุปกรณ์ในระบบจัดการฟาร์มอัจฉริยะ โดยใช้อุปกรณ์ IoT Gateway ที่มีการเชื่อมต่ออินเตอร์เน็ต โดยใช้ Sim 4G และเครื่องควบคุมขนาดเล็กที่สามารถโปรแกรม PLC ได้ในการพัฒนา และมีการเชื่อมต่อไปยังระบบอื่น ได้แก่ เซิร์ฟเวอร์ โดยโปรแกรมอุปกรณ์ให้มีความสามารถในการทำงานดังนี้

- ตั้งค่า IoT Gateway ด้วย Node-Red ให้ทำงานเป็น Edge Computing รวมถึงการจัดการข้อมูลของอุปกรณ์เซ็นเซอร์ อุปกรณ์สั่งการ คอนโทรลเลอร์ (PLC Siemens LOGO) และข้อมูลจาก AWS Cloud Service
- เชื่อมต่ออุปกรณ์เซ็นเซอร์และอุปกรณ์สั่งการกับคอนโทรลเลอร์ (PLC Siemens LOGO) และ IoT Gateway (IoT2050) โดยสามารถอ่านค่าและควบคุมอุปกรณ์
- ส่งค่าไป AWS Cloud Service ผ่าน 4G ไปยังกับ AWS IoT Core ด้วย MQTT Protocol และเก็บข้อมูลไปยัง DynamoDB

5.2 อภิปรายผล

จากการศึกษาเว็บแอปพลิเคชันและอุปกรณ์ที่ใช้ในระบบฟาร์มอัจฉริยะ พบว่า มีหลากหลายปัจจัยที่ส่งผลทำให้เกิดข้อจำกัดภายในเว็บแอปพลิเคชันและอุปกรณ์ เช่น การเพิ่มจำนวนอุปกรณ์ที่ทำได้จำกัดและทำได้แค่เพียง 1 ฟาร์มอันเนื่องจากมาจากพื้นที่บนเซิร์ฟเวอร์หรือคลาวด์มีจำนวนจำกัดส่งผลให้การเก็บข้อมูลทำได้น้อยลงและจะต้องใช้จ่ายเพื่อเพิ่มพื้นที่ให้มากขึ้นด้วย หรืออาจติดปัญหาในเรื่องของการได้รับค่าของอุปกรณ์ล่าช้า ดังนั้นจึงได้ออกแบบและสร้างเว็บแอปพลิเคชันและระบบบนคลาวด์ขึ้นมาเพื่อรองรับการปรับขนาดของข้อมูลต่างๆ ด้วยการใช้ชุดเครื่องมือ NEXT.JS ในการพัฒนาเว็บแอปพลิเคชันและใช้ Serverless ในการพัฒนาฟังก์ชัน หลังการสร้างระบบดังกล่าวพบว่า อุปกรณ์สามารถส่งค่าและแสดงผลในเว็บแอปพลิเคชันได้ตามเวลาจริง และในกรณีที่มีการเพิ่มอุปกรณ์ของผู้ใช้ลงในฟาร์มหรือฟาร์มใหม่ก็จะปรากฏในเว็บแอปพลิเคชันทันที โดยที่ผู้ใช้ไม่ต้องไปเพิ่มอุปกรณ์ด้วยตนเอง และเมื่อดูค่าใช้จ่ายของคลาวด์ว่า การทำงานด้วยการใช้บริการต่างๆนี้จะต้องมีค่าใช้จ่ายเท่าไหร่ พบว่าเมื่อเปรียบเทียบกับราคาของบริการจากแอปพลิเคชันอื่นๆแล้วนั้นมีราคาถูกกว่ามาก เพราะบริการใน AWS มีลำดับขั้นของการใช้งานที่ถูกแบ่งออกมา ถ้ายังไม่เกินจำนวนที่กำหนดก็จะยังไม่คิดราคา เพราะยังถือว่าอยู่ในเกณฑ์ของระดับฟรีเทียร์ สมมติในกรณีที่คิดราคาแล้วสำหรับ 1 อุปกรณ์ที่ส่งข้อมูลทุกๆ 5 นาที จะมีค่าใช้จ่ายอยู่ที่ประมาณ 120 บาทต่อเดือน ทั้งนี้ราคานี้เป็นเพียงการคาดการณ์ของ 1 อุปกรณ์เท่านั้น และไม่ได้รวมการขยายระดับขนาดของข้อมูลซึ่งคาดว่าอาจจะขึ้นไปถึงประมาณ 3,000 บาทหรือมากกว่าได้

5.3 ข้อเสนอแนะ

1) ทำเว็บแอปพลิเคชันให้สามารถรองรับประเภทของอุปกรณ์ให้หลากหลายมากขึ้น เพื่อรองรับการแสดงผลในหน้าแดชบอร์ดให้หลากลาย
2) รองรับการแสดงผลเป็นภาษาไทยในเว็บแอปพลิเคชัน และเพิ่มอินเตอร์เฟสเพื่อให้ผู้ใช้เข้าใจง่ายมากขึ้น
3) ทำเว็บแอปพลิเคชันให้รองรับกับระบบปฏิบัติการอื่น
4) ทำการตั้งค่าให้เป็นระบบอัตโนมัติเมื่อถึงเกณฑ์ที่ผู้ใช้กำหนด ทำให้สามารถควบคุมอุปกรณ์เองได้โดยไม่ต้องผ่านผู้ใช้
5) พัฒนาการวิเคราะห์ให้เป็นการทำ machine learning เพื่อทำให้วิเคราะห์และคาดเดาในเรื่องต่างๆ เช่น ผลผลิต หรือ ค่าเกณฑ์ที่ผู้ใช้ควรจะตั้ง
6) พัฒนาอุปกรณ์เกตเวย์และเครื่องควบคุมให้สามารถการเข้าถึงจากระยะไกลในการตั้งค่าหรือแก้ไขข้อมูลเฉพาะ
7) พัฒนาอุปกรณ์เกตเวย์และเครื่องควบคุมให้สามารถรองรับอุปกรณ์เซนเซอร์และอุปกรณ์สั่งการโดยอัตโนมัติ เมื่อเชื่อมต่ออุปกรณ์

บรรณานุกรม

[1] Aykut Bal, Native Mobile App Architecture – Everything You Need to Know, 2022, Online: https://www.storyly.io/post/native-mobile-app-architecture-everything-you-need-to-know

[2] chaiyo, Next.js, Online: https://www.chaiyohosting.com/nextjs-hosting.html

[3] MQTT, Online: https://mqtt.org/

[4] HTTP, Online: https://httpwg.org/specs/rfc9110.htm

[5] REST, Online: https://restfulapi.net/http-methods/

[6] Amazon, AWS, Online: https://aws.amazon.com/th/what-is-aws/?nc1=f_cc

[7] Amazon, AWS Lambda, Online: https://aws.amazon.com/th/lambda/faqs/

[8] Amazon, AWS IoT Core, Online: https://aws.amazon.com/th/iot-core/faqs/

[9] Amazon, API Gateway, Online: https://aws.amazon.com/th/api-gateway/faqs/

[10] Amazon, DynamoDB, Online: https://aws.amazon.com/th/dynamodb/faqs/

[11] Amazon, IAM, Online: https://aws.amazon.com/th/iam/faqs/?nc=sn&loc=5

[12] Amazon, Cognito, Online: https://aws.amazon.com/th/cognito/?nc2=h_ql_prod_se_cog

[13] Amazon, CloudWatch, Online: https://aws.amazon.com/th/cloudwatch/faqs/

[14] Amazon, CloudFormation, Online: https://aws.amazon.com/th/cloudformation/faqs/

[15] Application, Online: https://sites.google.com/site/psupattar475/khwam-hmay-laea-prapheth-khxng-xaeph-phli-khechan

[16] BorntoDev, React, Online: https://www.borntodev.com/2020/07/15/react-101/

[17] 1stCraft Team, Web Application, Online: https://1stcraft.com/website-application-vs-general-website/

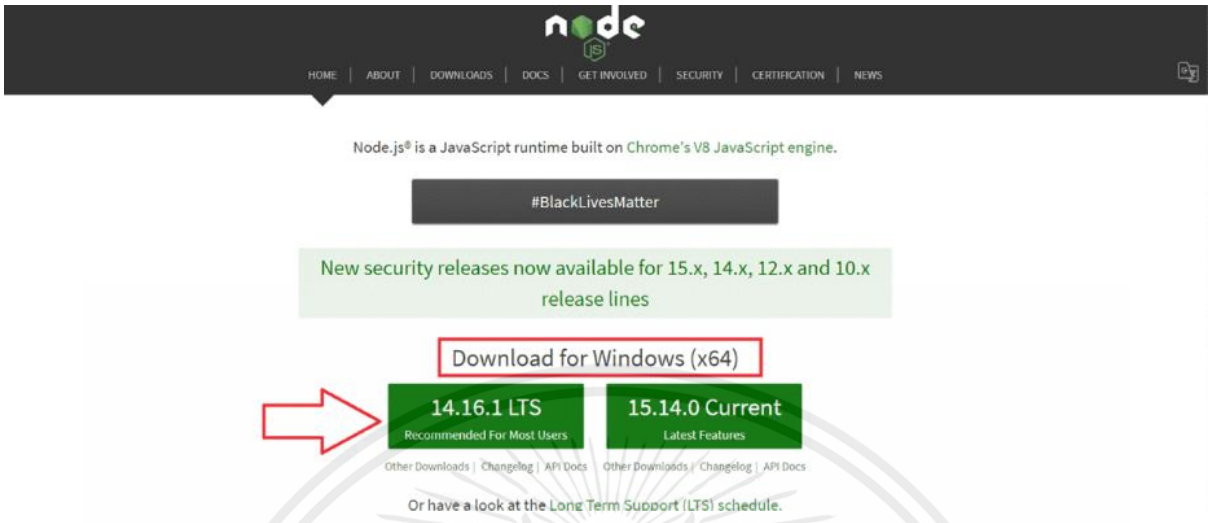
ภาคผนวก

ภาคผนวก ก.

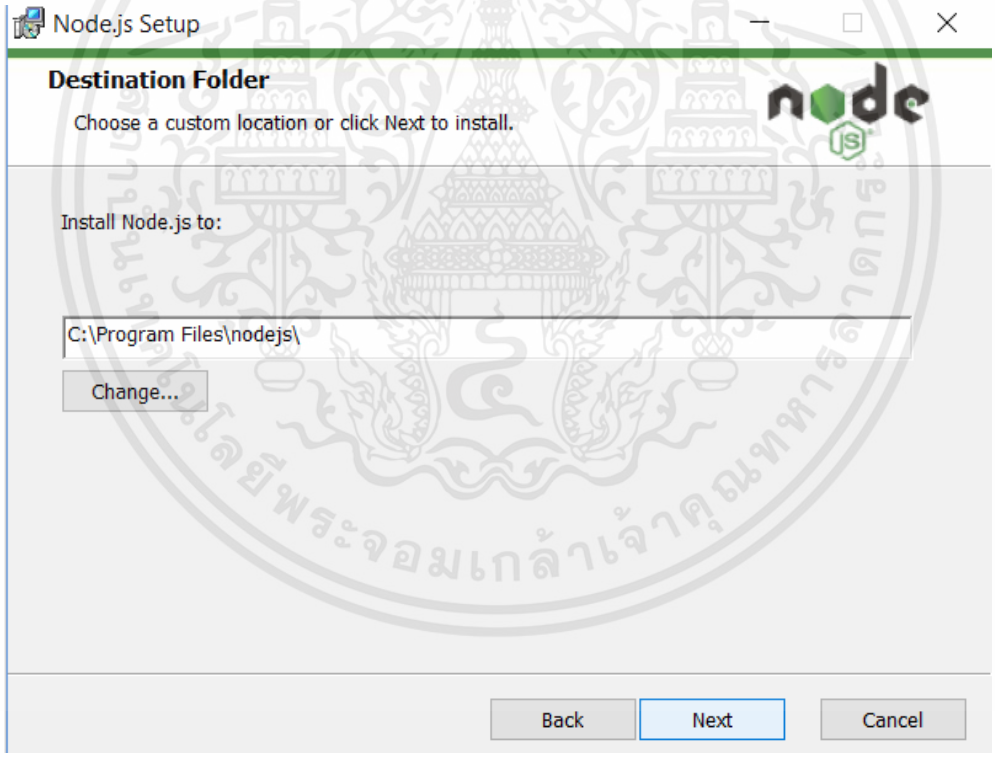
การติดตั้งโปรแกรมที่จำเป็นเว็บแอปพลิเคชัน

การติดตั้ง Node.js เพื่อใช้งาน Next.js

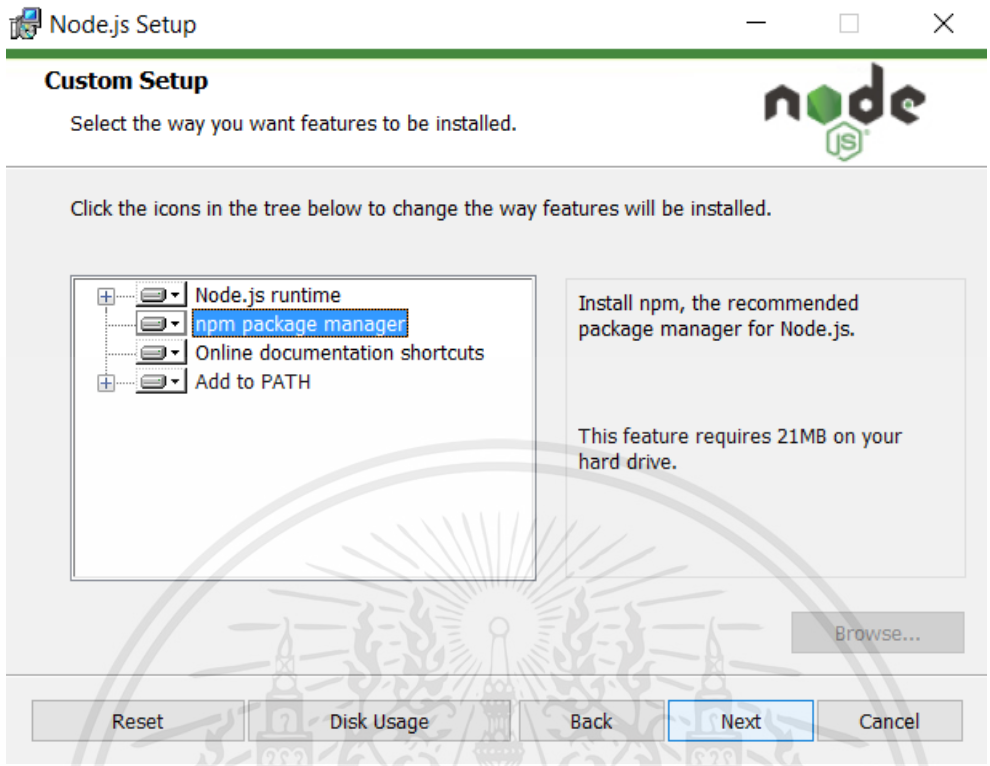
1. โหลดไฟล์ติดตั้ง node.js จาก https://nodejs.org/en/download



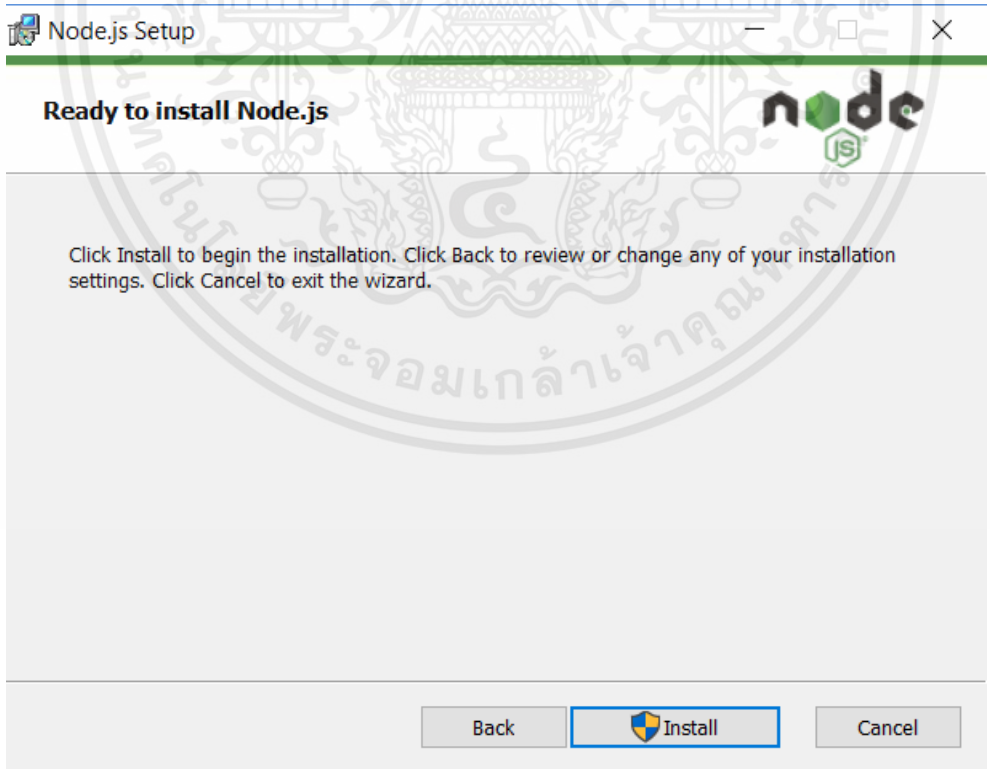
2. รันไฟล์ที่ดาวน์โหลดและกำหนดตำแหน่งที่ต้องการจะติดตั้งแล้วคลิก Next



3.กำหนดคุณสมบัติที่จะติดตั้งแล้วคลิก Next



4.คลิก Install เพื่อติดตั้ง Node.js



การสร้างโปรเจ็คด้วย Next.js

1. เปิด Terminal แล้วรันคำสั่งดังนี้ npx create-next-app@latest

2. กำหนดค่าต่างๆของโปรเจ็ค

```
What is your project named?  my-app
Would you like to add TypeScript with this project?  Y/N
Would you like to use ESLint with this project?  Y/N
Would you like to use Tailwind CSS with this project? Y/N
Would you like to use the `src/ directory` with this project? Y/N
What import alias would you like configured? `@/*`
```

3. ลงส่วนเสริมต่างที่จำเป็นด้วยการรันคำสั่งต่อไปนี้

- npm install @mui/material @emotion/react @emotion/styled
- npm install -D tailwindcss postcss autoprefixer
- npx tailwindcss init -p
- npm install aws-sdk
- npm install aws-amplify

การ Configure Tailwindcss

1. ไปที่ไฟล์ tailwind.config.js แล้วเพิ่มข้อมูลในไฟล์ดังนี้

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./app/**/*.{js,ts,jsx,tsx,mdx}",
    "./pages/**/*.{js,ts,jsx,tsx,mdx}",
    "./components/**/*.{js,ts,jsx,tsx,mdx}",

    // Or if using `src` directory:
    "./src/**/*.{js,ts,jsx,tsx,mdx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

3. ไปที่ไฟล์ globals.css แล้วเพิ่มข้อมูลในไฟล์ดังนี้

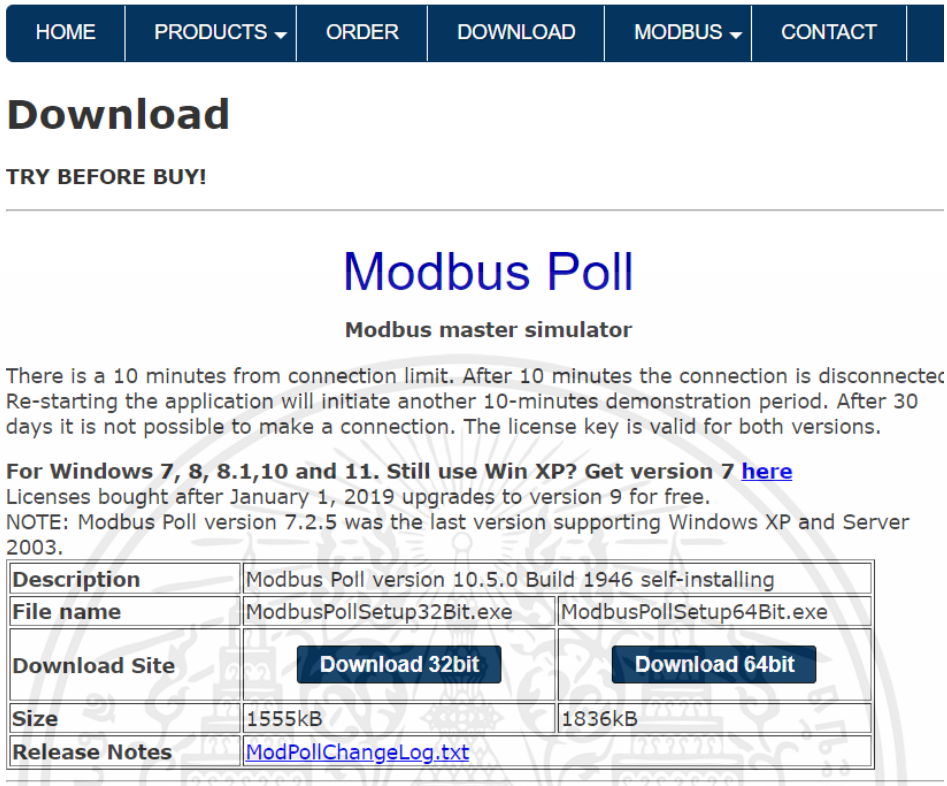
```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

ภาคผนวก ข.

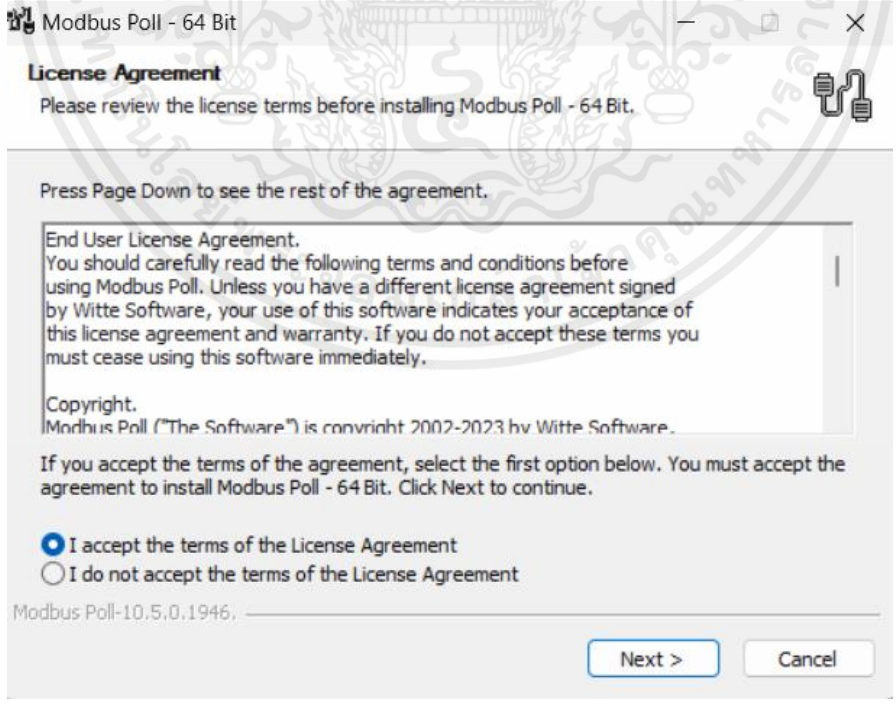
การติดตั้งโปรแกรมที่จำเป็นการตั้งค่าอุปกรณ์

การติดตั้งโปรแกรม Modbus Pull

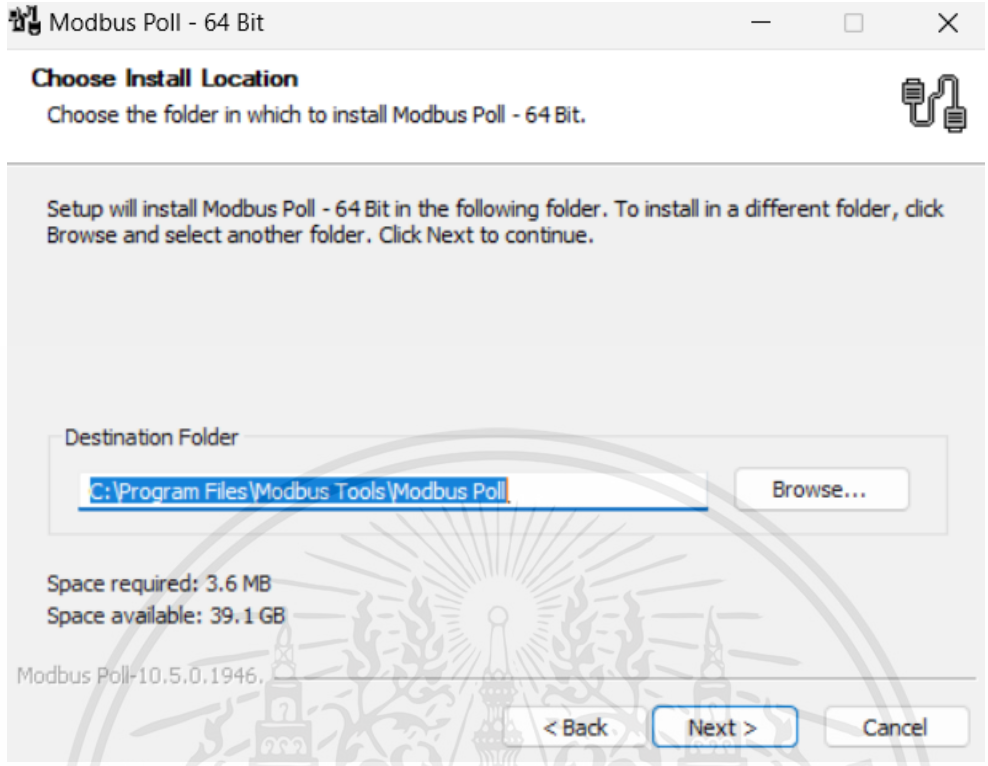
1. โหลดไฟล์ติดตั้ง Modbus Pull จาก https://www.modbustools.com/download.html



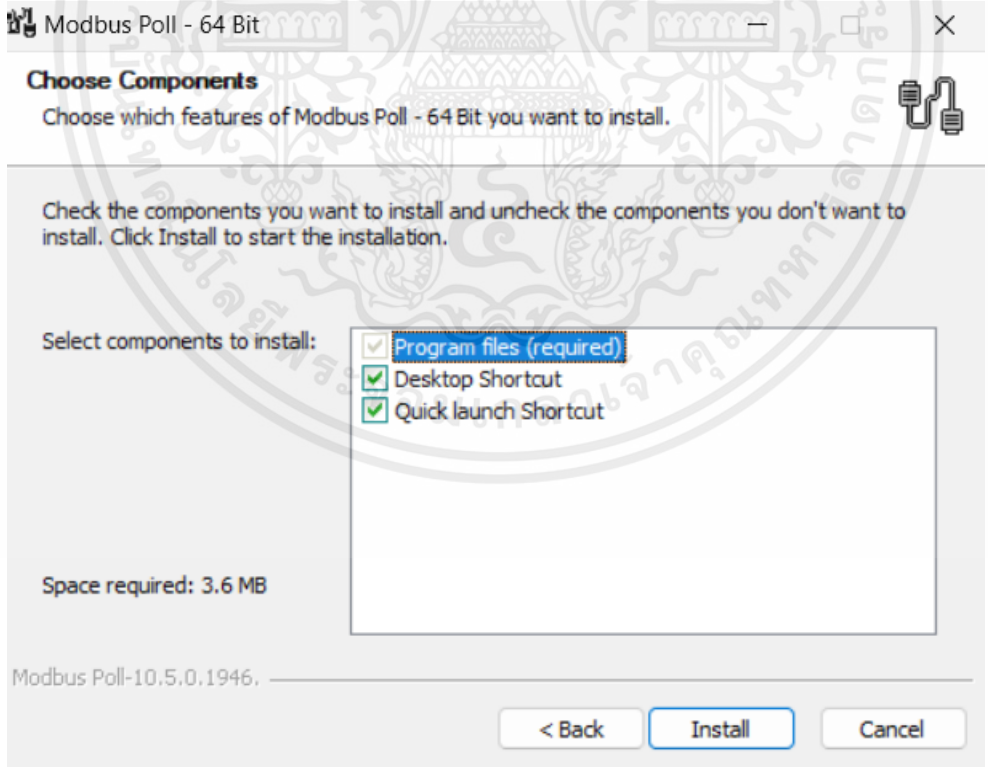
2. รันไฟที่ดาวโหลดและกดยอมรับข้อตกลงแล้วคลิก Next



3. กำหนดตำแหน่งที่ต้องการจะติดตั้งแล้วคลิก Next

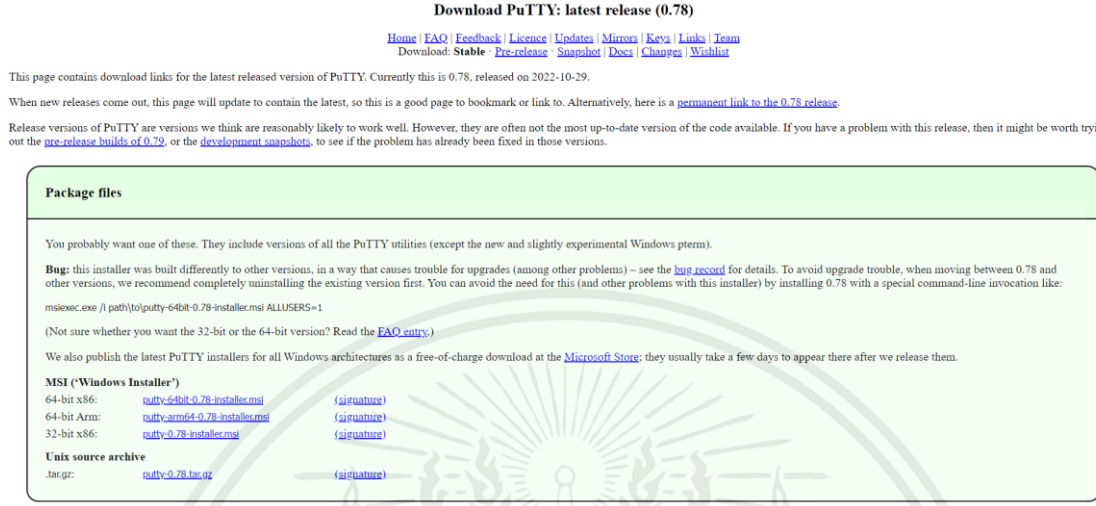


4. เลือก component ที่ต้องการจะติดตั้งแล้วคลิก Install



การติดตั้งโปรแกรม PuTTy

1. โหลดไฟล์ติดตั้ง PuTTy จาก https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html



Download PuTTY: latest release (0.78)

Home | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team
Download: **Stable** · Pre-release · Snapshot | Docs | Changes | Wishlist

This page contains download links for the latest released version of PuTTY. Currently this is 0.78, released on 2022-10-29.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a permanent link to the 0.78 release.

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the pre-release builds of 0.79, or the development snapshots, to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include versions of all the PuTTY utilities (except the new and slightly experimental Windows pterm).

Bug: this installer was built differently to other versions, in a way that causes trouble for upgrades (among other problems) – see the bug record for details. To avoid upgrade trouble, when moving between 0.78 and other versions, we recommend completely uninstalling the existing version first. You can avoid the need for this (and other problems with this installer) by installing 0.78 with a special command-line invocation like:

msiexec.exe /i path\to\putty-64bit-0.78-installer.msi ALLUSERS=1

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

We also publish the latest PuTTY installers for all Windows architectures as a free-of-charge download at the Microsoft Store; they usually take a few days to appear there after we release them.

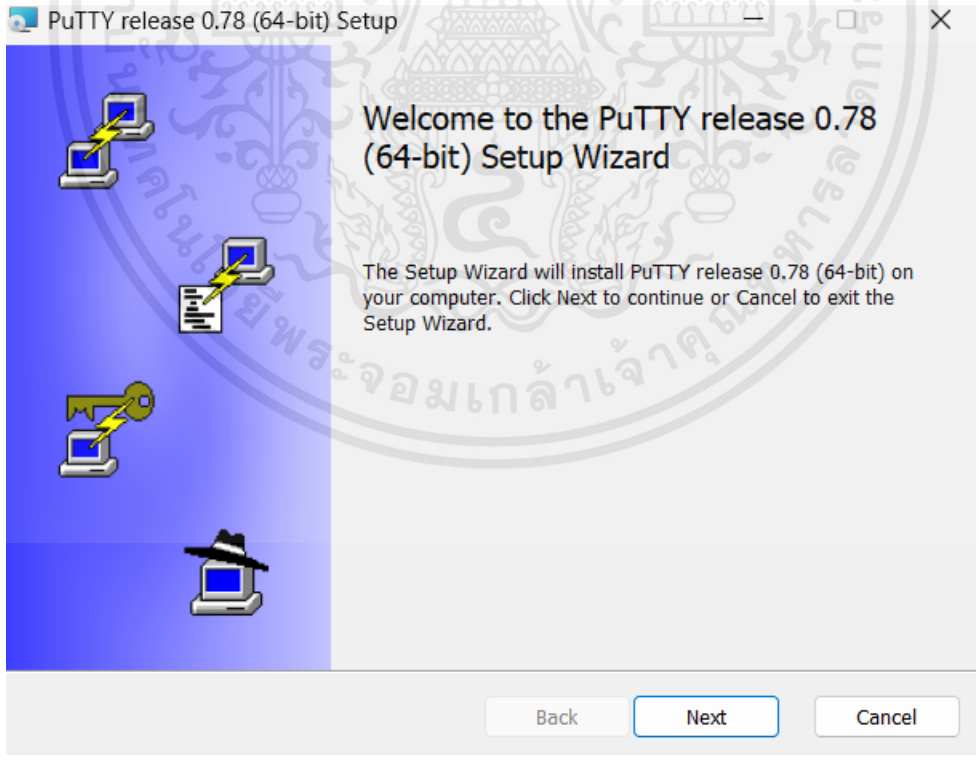
MSI ('Windows Installer')

64-bit x86:	putty-64bit-0.78-installer.msi	(signature)
64-bit Arm:	putty-arm64-0.78-installer.msi	(signature)
32-bit x86:	putty-0.78-installer.msi	(signature)

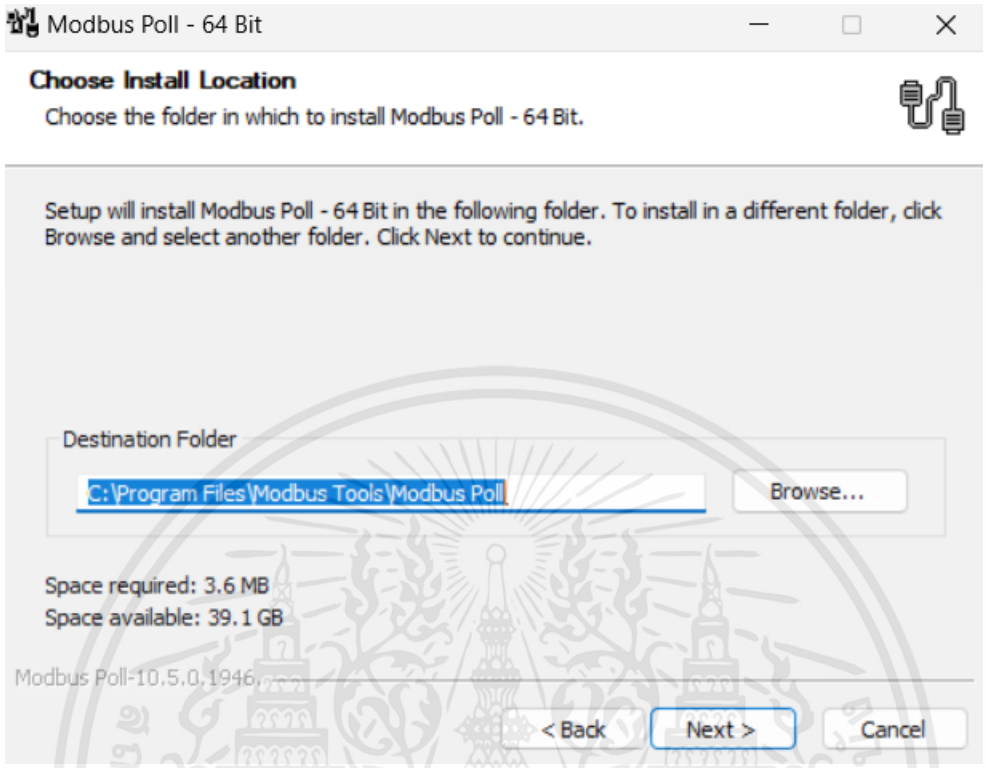
Unix source archive

.tar.gz:	putty-0.78.tar.gz	(signature)

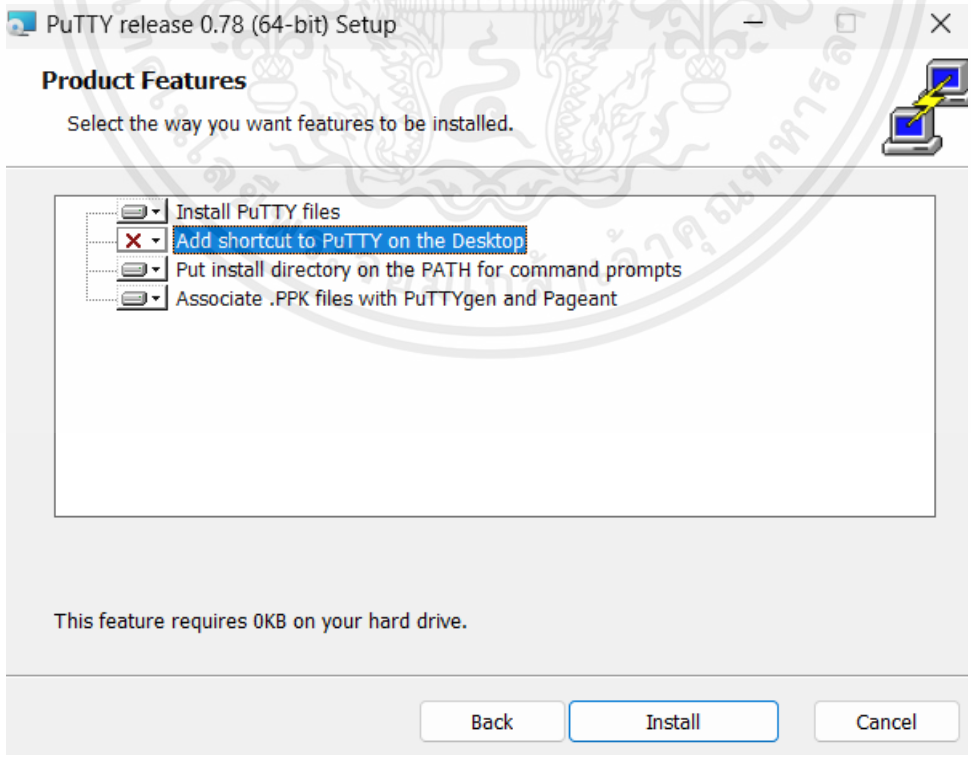
2. รันไฟที่ดาวโหลดและกดยอมรับข้อตกลงแล้วคลิก Next



3. กำหนดตำแหน่งที่ต้องการจะติดตั้งแล้วคลิก Next



4. เลือก feature ที่ต้องการจะติดตั้งแล้วคลิก Install



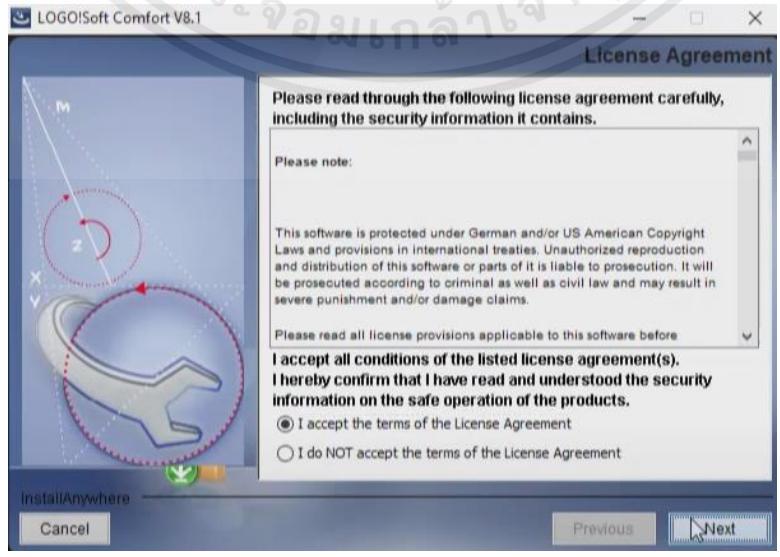
การติดตั้งโปรแกรม LOGO! Soft Comfort

1. โหลดไฟล์ติดตั้ง LOGO! Soft Comfort จาก https://support.industry.siemens.com/cs/document/109783154/download-for-logo!-8-3-software-upgrade?dti=0&lc=en-TH

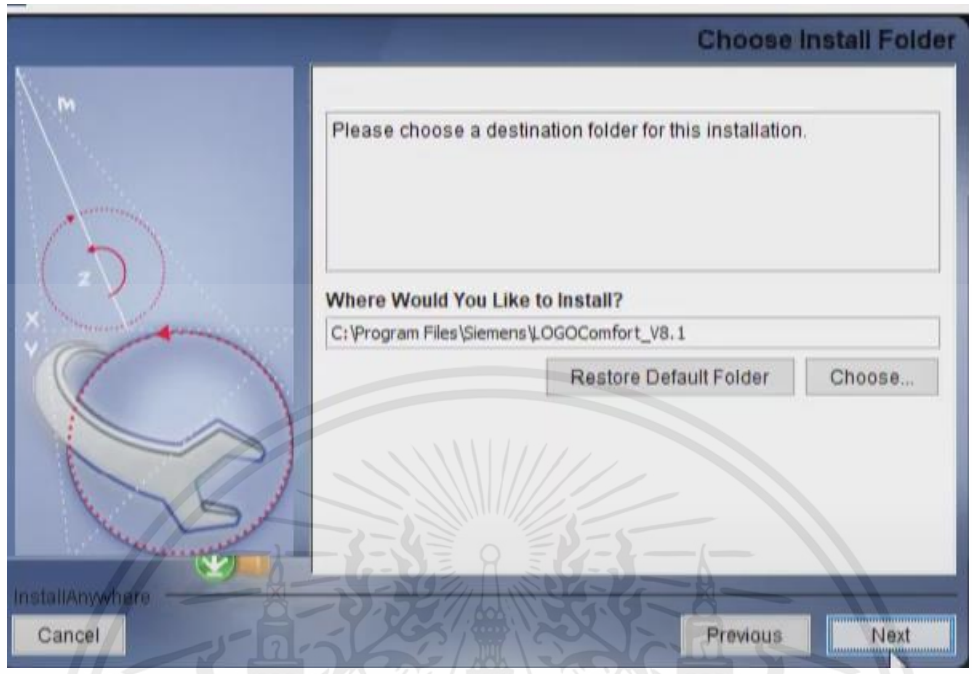
2. ไปที่ตำแหน่งที่เก็บตัวติดตั้งไว้ C:\Users\boom\Downloads\LOGO8\LOGO Soft Comfort V8.3\CD_logo8\CDROM_Installers\Disk1\InstData\Win64\VM จากนั้นรันตัวติดตั้ง

3. เลือกภาษาที่ต้องการจะติดตั้งแล้วคลิก OK

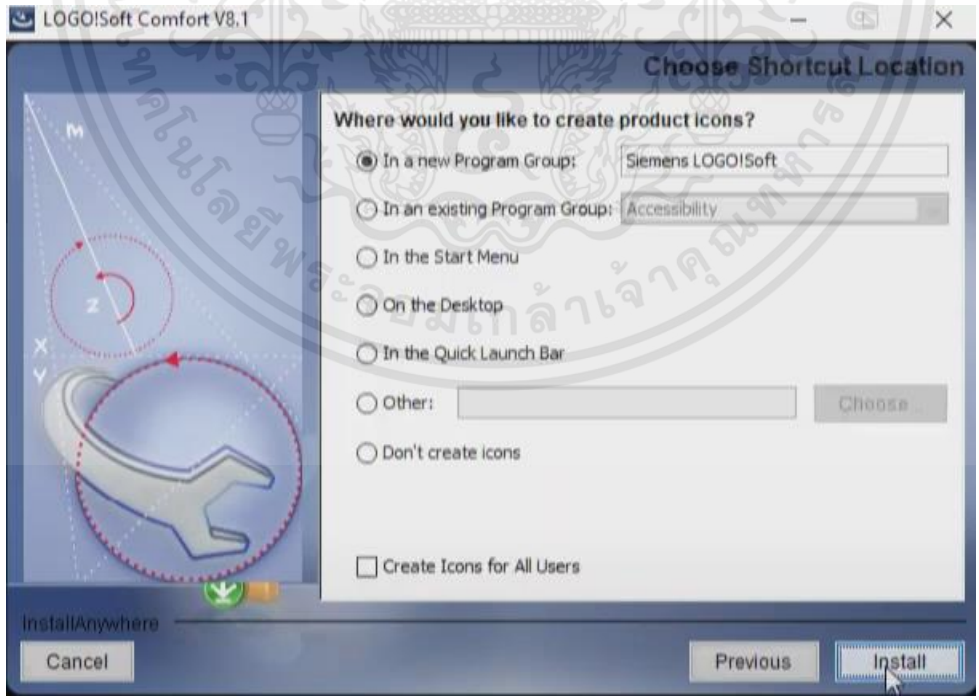
4. กดยอมรับข้อตกลงแล้วคลิก Next



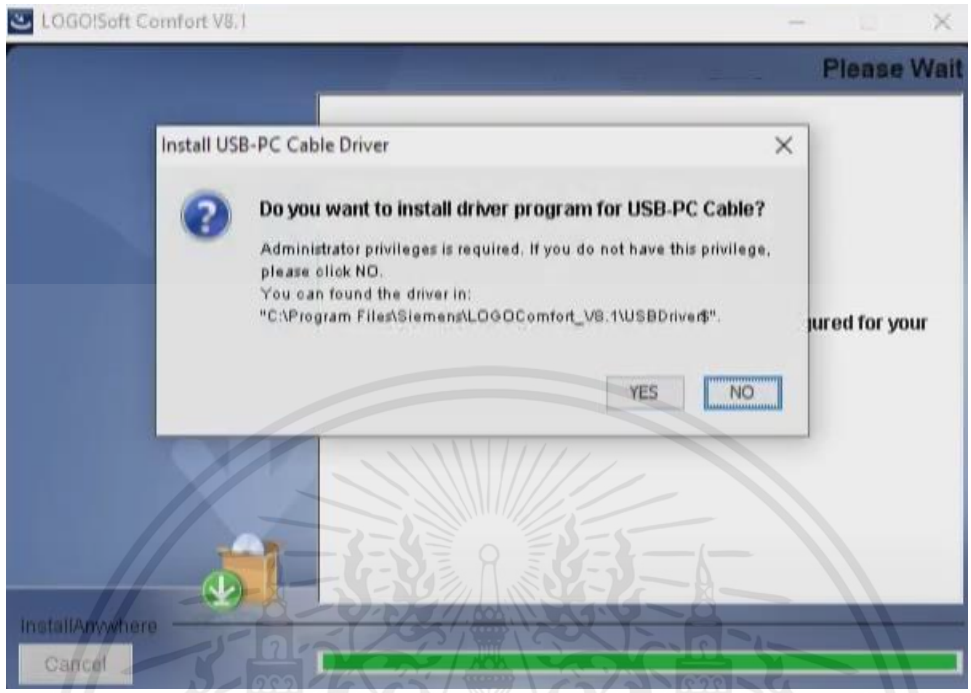
5. เลือกตำแหน่งที่ต้องการจะติดตั้งแล้วคลิก Next



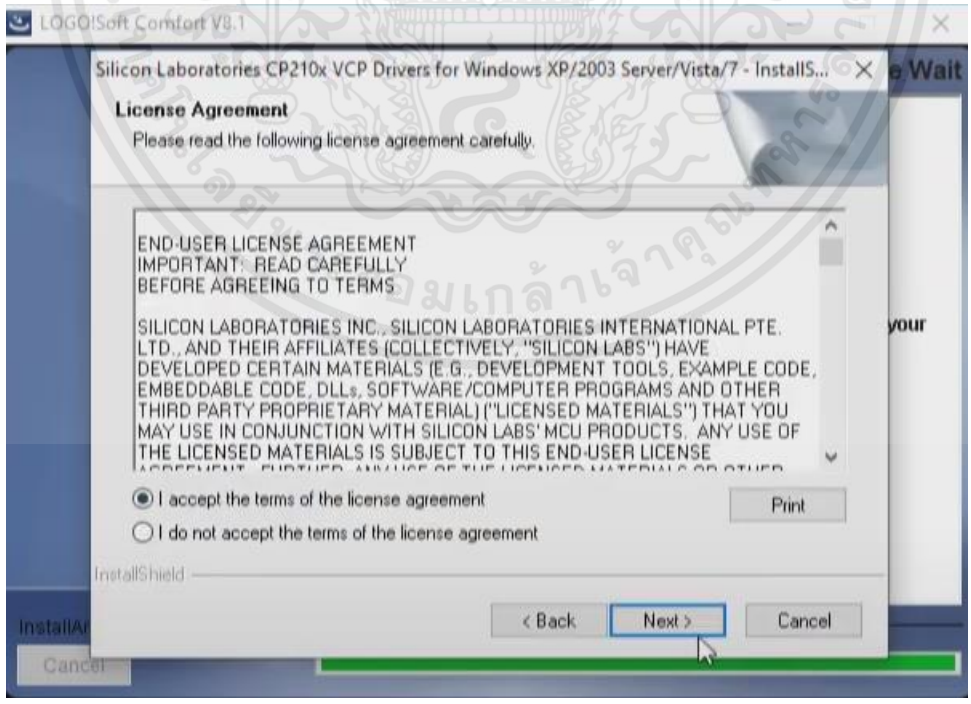
6. กำหนดตำแหน่งที่ต้องการสร้าง icon แล้วคลิก Install



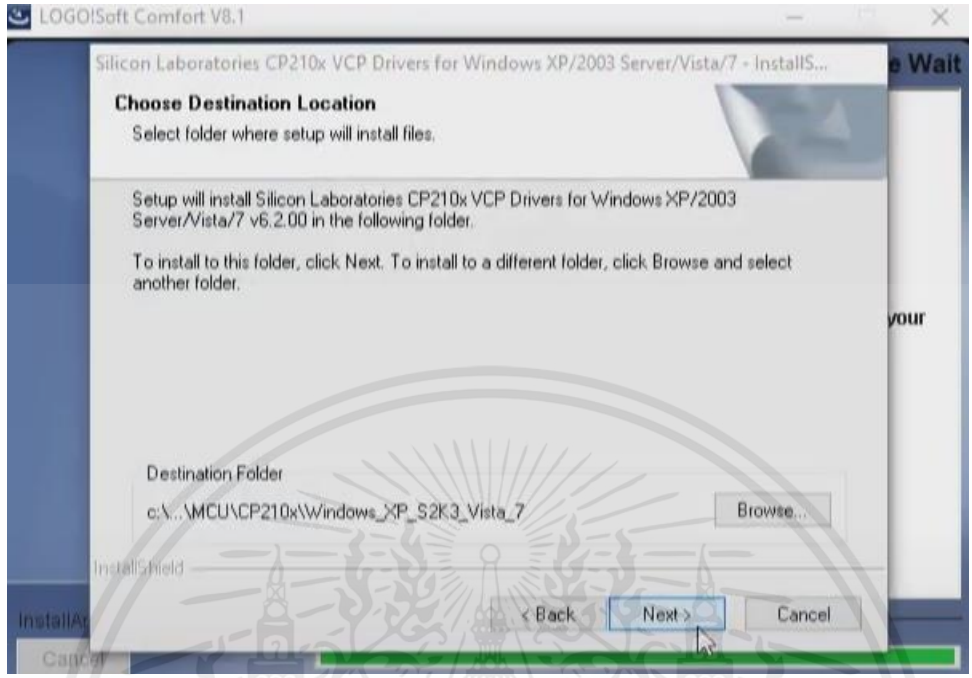
7. คลิก YES เพื่อติดตั้ง driver



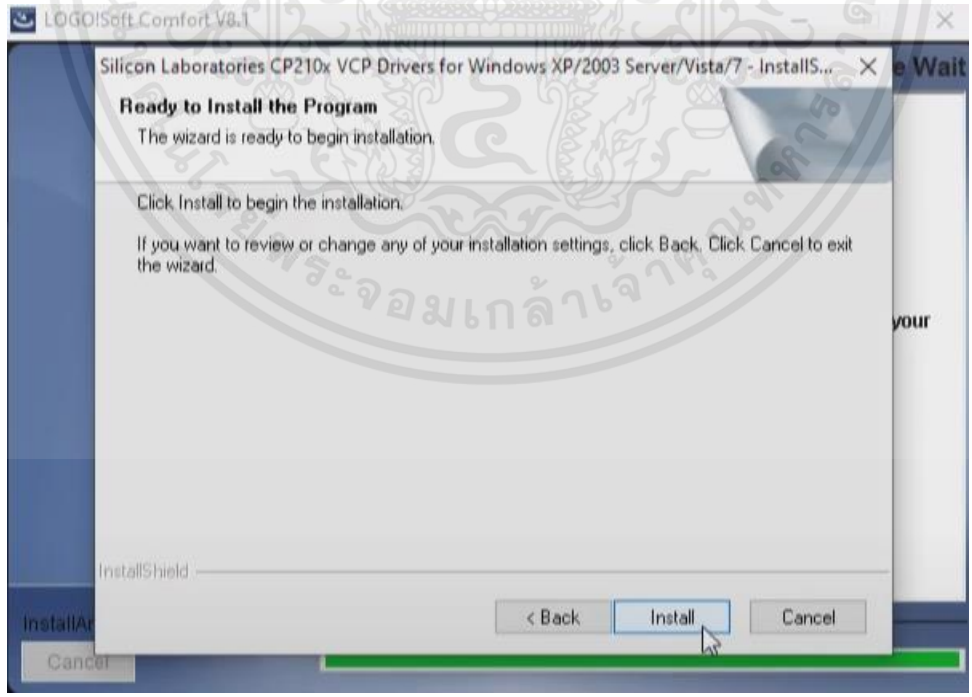
8. กดยอมรับข้อตกลงแล้วคลิก Next



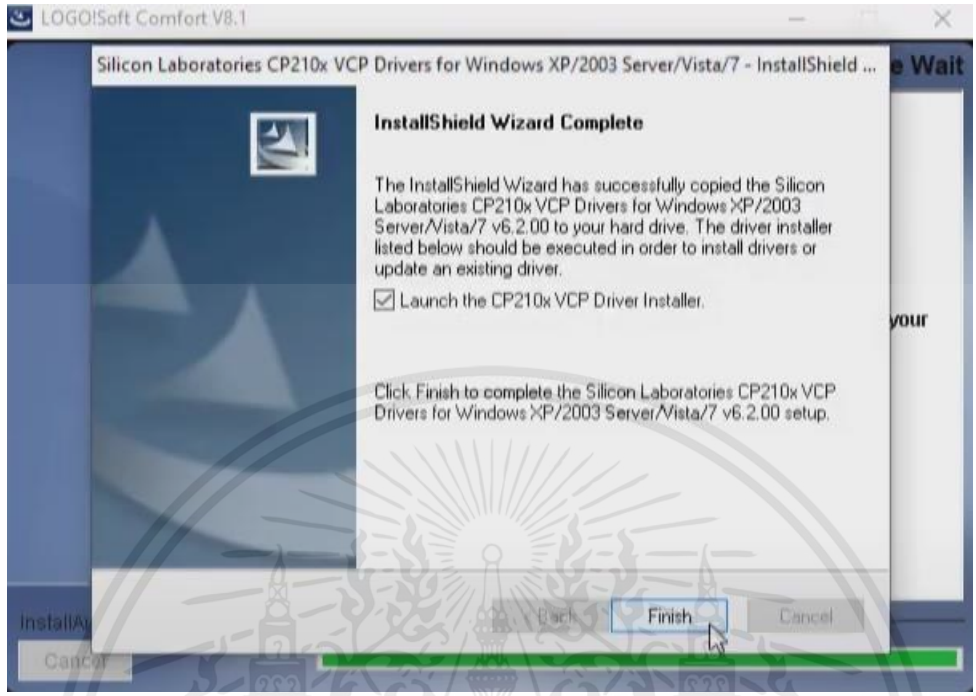
9. เลือกตำแหน่งที่ต้องการจะติดตั้งแล้วคลิก Next



10. คลิก Install เพื่อติดตั้ง



11. คลิก Finish เพื่อติดตั้ง CP210x VCP Driver



12. เลือกตำแหน่งที่ต้องการจะติดตั้งแล้วคลิก Install