สถานีตรวจสอบคุณภาพน้ำบนพื้นฐานไอโอทีและระบบพยากรณ์ด้วยการเรียนรู้ของเครื่อง

IOT-BASED WATER QUALITY MONITORING STATION AND FORECASTING SYSTEM WITH MACHINE LEARNING

ธนาตย์ จอมใจเอกชน

THANART JOMJAIEKACHORN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2567

KMITL-2025-EN-M-027-057

IOT-BASED WATER QUALITY MONITORING STATION AND FORECASTING SYSTEM WITH MACHINE LEARNING

THANART JOMJAIEKACHORN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN ELECTRICAL AND COMPUTER ENGINEERING
SCHOOL OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2024

KMITL-2025-EN-M-027-057

หัวข้อวิทยานิพนธ์	สถานีตรวจสอบคุณภาพน้ำบนพื้นฐานไอโอทีและระบบพยากรณ์ด้วยการเรียนรู้ของเครื่อง
นักศึกษา	นายธนาตย์ จอมใจเอกชน
รหัสประจำตัว	66016051
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
พ.ศ.	2567
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผู้ช่วยศาสตราจารย์ ดร.ธนวิชญ์ อนุวงศ์พินิจ

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้นำเสนอระบบตรวจสอบและพยากรณ์คุณภาพน้ำแบบเรียลไทม์โดยใช้เทคโนโลยีอินเตอร์เน็ตของสรรพสิ่ง ซึ่งออกแบบมาเพื่อประเมินคุณภาพแหล่งน้ำอย่างต่อเนื่อง ระบบนี้ผสานการทำงานของไอโอทีเกตเวย์สำหรับอุตสาหกรรม โดยทำการเก็บข้อมูลจากเซนเซอร์ที่ใช้วัดค่าการนำไฟฟ้า, ค่าความเป็นกรด-ด่าง, ค่าออกซิเจนละลายน้ำ และอุณหภูมิ โดยใช้การสื่อสารแบบ RS485 Modbus RTU กระบวนการประมวลผลข้อมูลเกิดขึ้นที่ Edge Computing โดยใช้ Node-RED และส่งข้อมูลไปยัง Amazon Web Services Cloud ผ่านโพรโทคอล Message Queuing Telemetry Transport เพื่อจัดเก็บและแสดงผลผ่านแดชบอร์ด และนำข้อมูลคุณภาพน้ำจากคุณภาพน้ำของแม่น้ำเจ้าพระยา ณ สถานีวัดมะขาม ย้อนหลัง 10 ปี จากหน่วยงานการประปานครหลวง เพื่อมาใช้เป็นข้อมูลฝึกสอนและทดสอบ เพื่อวิเคราะห์เชิงพยากรณ์โดยใช้โมเดลการเรียนรู้ของเครื่อง Extreme Gradient Boosting ที่ปรับแต่งพารามิเตอร์ด้วย Optuna และโครงข่ายประสาทเทียมด้วยโมเดล Long Short-Term Memory เพื่อพยากรณ์คุณภาพน้ำ โดยผลการทดลองแสดงให้เห็นว่าโมเดล Long Short-Term Memory มีประสิทธิภาพสูงกว่าสำหรับการพยากรณ์ค่าพารามิเตอร์ส่วนใหญ่ โดยมีค่าประสิทธิภาพการวัด R-Squared, Mean Absolute Error, Root Mean Square Error และ Mean Square Error เฉลี่ยแต่ละพารามิเตอร์ส่วนใหญ่ประมาณ 0.8744, 0.0271, 0.0435 และ 0.0020 ตามลำดับ ในขณะที่ Extreme Gradient Boosting ให้ผลลัพธ์ที่ดีกว่าในการพยากรณ์ค่าความเป็นกรดเป็นด่าง โดยมีค่าประสิทธิภาพการวัด R-Squared, Mean Absolute Error, Root Mean Square Error และ Mean Square Error ประมาณ 0.5111, 0.0110, 0.0399 และ 0.015 ตามลำดับ วิทยานิพนธ์ฉบับนี้แสดงให้เห็นถึงความสามารถในการขยายใช้งาน ความเชื่อถือได้ และศักยภาพในการปรับปรุงการบริหารจัดการคุณภาพน้ำในสภาพแวดล้อมที่หลากหลายโดยใช้เทคโนโลยีการเรียนรู้ของเครื่องร่วมกับระบบเทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง

คำสำคัญ: อินเทอร์เน็ตของสรรพสิ่ง, การตรวจสอบคุณภาพน้ำ, MQTT, การเรียนรู้ของเครื่องจักร, ระบบพยากรณ์

Title	IoT-based Water Quality Monitoring Station and Forecasting System with Machine Learning
Student	Mr.Thanart Jomjaiekachorn
Student ID.	66016051
Degree	Master of Engineering
Program	Electrical and Computer Engineering
Year	2024
Thesis Advisor	Assistant Professor Dr.Thanavit Anuwongpinit

ABSTRACT

This thesis presents an IoT-based water quality monitoring and forecasting system designed for real-time and continuous assessment of water resources. The system integrates an Industrial IoT Gateway, which collects data from sensors measuring conductivity, pH, dissolved oxygen, and temperature using RS485 Modbus RTU communication. Data processing occurs at the edge computing using Node-RED and is transmitted to Amazon Web Services Cloud via Message Queuing Telemetry Transport for storage and visualization on a dashboard. Historical water quality data of the Chao Phraya River at the Makham monitoring station, spanning the past 10 years and provided by the Metropolitan Waterworks Authority, is used for training and testing. For predictive analysis, machine learning models are employed, including Extreme Gradient Boosting (XGBoost) with parameter tuning via Optuna and Long Short-Term Memory (LSTM) networks. Experimental results demonstrate that the LSTM model outperforms in forecasting most water quality parameters, achieving average performance metrics of R-squared = 0.8744, Mean Absolute Error = 0.0271, Root Mean Square Error = 0.0435, and Mean Square Error = 0.0020. In contrast, the XGBoost model yields better performance in pH prediction, with corresponding values of R-squared = 0.5111, Mean Absolute Error = 0.0110, Root Mean Square Error = 0.0399, and Mean Square Error = 0.0150. This thesis demonstrates the scalability, reliability, and potential of integrating machine learning with IoT technologies to improve water quality management across diverse environments.

Keywords: Internet of Things, Water Quality Monitoring, MQTT, Machine Learning, Forecasting System

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี ด้วยความเมตตาและความอนุเคราะห์จาก ผู้ช่วยศาสตราจารย์ ดร.ธนวิชญ์ อนุวงศ์พินิจ และ รองศาสตราจารย์ ดร.บุณย์ชนะ ภู่ระหงษ์ ซึ่งเป็นอาจารย์ที่ปรึกษา ได้กรุณาให้คำแนะนำ ถ่ายทอดองค์ความรู้ และให้คำปรึกษาในการดำเนินงานวิจัยอย่างใกล้ชิด พร้อมทั้งชี้แนะแนวทางในการแก้ไขปัญหาต่าง ๆ และเป็นกำลังใจที่สำคัญตลอดระยะเวลาการจัดทำวิทยานิพนธ์

ขอขอบคุณ ดร.นัชนัยน์ รุ่งเหมือนฟ้า และคณาจารย์ประจำสาขาวิศวกรรมระบบไอโอทีและสารสนเทศทุกท่าน ที่ได้ให้คำปรึกษา ถ่ายทอดความรู้ และคำแนะนำอันเป็นประโยชน์ในทุก ๆ ด้าน รวมถึงเป็นแรงสนับสนุนและกำลังใจที่ดีเสมอมา

ขอขอบคุณน้อง ณัฐวินท์ แต้มประเสริฐ (อัส) และฐิติมา แนวพระยา (คิว) รุ่นน้องที่ให้ความช่วยเหลือ ทั้งในด้านคำแนะนำ การให้กำลังใจ และการร่วมทดสอบงานวิจัย จนทำให้งานวิจัยประสบความสำเร็จด้วยดี รวมถึงน้อง ๆ ในห้องวิจัย IoT and Electronics Research Laboratory (IERL) E12-1109 ทุกคน ที่คอยช่วยเหลือกัน และให้กำลังใจมาเสมอ

ขอแสดงความขอบพระคุณอย่างสูงต่อ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง สำหรับการสนับสนุนทุนอุดหนุนการวิจัย ซึ่งมีบทบาทสำคัญในการทำให้การดำเนินงานวิจัยครั้งนี้เป็นไปอย่างราบรื่น

ขอขอบพระคุณจากใจต่อ ครอบครัวของข้าพเจ้าทุกคน ที่ได้เป็นกำลังใจที่มั่นคง คอยให้คำปรึกษา และอยู่เคียงข้างด้วยความเข้าใจและความรักเสมอมา ทำให้ข้าพเจ้าสามารถก้าวผ่านอุปสรรคต่าง ๆ และดำเนินการจัดทำวิทยานิพนธ์ฉบับนี้จนแล้วเสร็จ

สำหรับคุณงามความดีใด ๆ ที่เกิดขึ้นจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ บิดา มารดา และครอบครัว ที่ข้าพเจ้ารักและเคารพยิ่ง ตลอดจน ครูบาอาจารย์ทุกท่าน ที่ได้กรุณาประสิทธิ์ประสาทวิชา ถ่ายทอดประสบการณ์ และเป็นแบบอย่างที่ดีให้แก่ข้าพเจ้า รวมทั้ง ผู้มีพระคุณทุกท่าน ที่มีส่วนสำคัญในการสนับสนุนการศึกษาของข้าพเจ้าอย่างหาที่สุดมิได้

ธนาตย์ จอมใจเอกชน

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูปภาพ

รูปที่ **หน้า**

สารบัญรูปภาพ (ต่อ)

รูปที่ **หน้า**

สารบัญรูปภาพ (ต่อ)

รูปที่ **หน้า**

สารบัญรูปภาพ (ต่อ)

รูปที่ **หน้า**

คำอธิบายสัญลักษณ์และคำย่อ

ABS	Acrylonitrile Butadiene Styrene
AC	Alternating Current
API	Application Programming Interface
AWS	Amazon Web Services
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DC	Direct Current
DNNs	Deep Neural Networks
DO	Dissolved Oxygen
DPU	Data Processing Unit, Distributed Processing Unit
EC	Electrical Conductivity
eMMC	embedded Multi-Media Card
FTP	File Transfer Protocol
GNU	GNU's Not Unix
GPU	Graphics Processing Unit
HTTP	HyperText Transfer Protocol
ID	Identifier, Identification
IIoT	Industrial Internet of Things
I/O	Input/Output
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MBAP	Modbus application Protocol
ML	Machine Learning
MPPT	Maximum Power Point Tracking
MQTT	Message Queuing Telemetry Transport
MSE	Mean Squared Error
NLP	Natural Language Processing
NoSQL	Not Only Structured Query Language
OS	Operating System
PDU	Protocol Data Unit
PH	Pondus Hydrogenii
PLC	Programmable Logic Controller
RMSE	Root Mean Squared Error

คำอธิบายสัญลักษณ์และคำย่อ (ต่อ)

RNN	Recurrent Neural Network
RS485	Recommended Standard no. 485
R-Squared	Coefficient of determination Score
RTU	Remote Terminal Unit
SCP	Secure Copy Protocol
SFTP	SSH File Transfer Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
UVC	Universal Serial Bus Video Class
VFD	Variable Frequency Drive
Wi-Fi	Wireless Fidelity
XGBoost	Extreme Gradient Boosting
f_t	เวกเตอร์ที่ timestep t กำหนดว่าค่าจากสถานะก่อนหน้าจะถูก Forget หรือเก็บไว้เท่าใด
σ	ตัวแปลทำให้ค่าผลลัพธ์อยู่ในช่วง [0,1]
W_f	น้ำหนักของ forget gate เรียนรู้ได้ผ่านการฝึก
$[h_{t-1}, x_t]$	การ concatenation ของ hidden state จาก timestep ก่อนหน้าที่ h_{t-1} กับ input ปัจจุบัน x_t
b_f	forget gate ช่วยปรับค่าความเอียงที่เกิดขึ้นจากค่าคงที่ถูกบวกเข้ากับผลรวมเชิงเส้นของการคำนวณ
i_t	input gate activation ที่ timestep t จะกำหนดว่าข้อมูลใหม่จะถูกเพิ่มเข้า cell state มากน้อยแค่ไหน
b_i	bias ของ input gate
$\tilde{c}_t$	candidate cell state เป็นค่าข้อมูลใหม่ที่นำไปอัปเดตใน cell state
$\tanh$	hyperbolic tangent function ให้ค่าอยู่ในช่วง [−1,1]
b_c	bias ของ candidate cell state
c_t	cell state ปัจจุบัน ที่ timestep t เก็บข้อมูลระยะยาวใน LSTM
C_{t-1}	cell state ก่อนหน้า
$\odot$	element-wise multiplication การคูณกันแบบจุดต่อจุดของเวกเตอร์
o_t	output gate activation กำหนดว่าจะส่งส่วนใดของ cell state ออกเป็น hidden state

คำอธิบายสัญลักษณ์และคำย่อ (ต่อ)

W_o	น้ำหนักของ output gate
b_o	bias ของ output gate
$\tanh(c_t)$	การปรับค่า cell state c_t ให้อยู่ในช่วง [−1,1]
$\hat{y}_i^{(t)}$	ค่าพยากรณ์ของตัวอย่าง i ที่รอบการเรียนรู้ t
$f_k(x_i)$	ฟังก์ชันอนุมูล (*base learner)* ลำดับที่ k ที่ใช้กับข้อมูล x_i
F	เซตของฟังก์ชันอนุมูลทั้งหมด
$\mathcal{L}$	*objective function*
n	จำนวนตัวอย่าง
$l(y_i, \hat{y}_i)$	ค่าความสูญเสีย (*loss)* ระหว่างค่าจริง y_i และค่าพยากรณ์ $\hat{y}_i$
$\Omega(f_k)$	ค่าความซับซ้อน (*regularization term)* ของฟังก์ชัน f_k
T	*leaves* หรือจำนวนใบไม้ในต้นไม้เพื่อใช้ในการตัดสินใจ f_k
γ	พารามิเตอร์ค่าคงที่ที่ควบคุมความซับซ้อน
λ	พารามิเตอร์กำกับความเรียบและความคงที่ของน้ำหนัก
w_j	ค่าการพยากรณ์ที่มีความสัมพันธ์กันกับใบไม้ใบที่ j
$\mathcal{L}^{(t)}$	ค่าความสูญเสียโดยประมาณ (approximate loss) ในรอบที่ t
g_i	ค่ากราดิเอนต์แรกของ $l(y_i, \hat{y}_i)$
h_i	ค่าฮีสเซียน (second derivative) ของ $l(y_i, \hat{y}_i)$
$f_i(x_i)$	ฟังก์ชันอนุมูลใหม่ที่กำลังเรียนรู้
I_L	เซตของตัวอย่างที่ตกลงบนใบไม้ทางฝั่งซ้าย
I_R	เซตของตัวอย่างที่ตกลงบนใบไม้ทางฝั่งซ้ายขวา
I	เซตของตัวอย่างทั้งหมดในโหนด
Gain	ค่า reduction in loss เมื่อแบ่งโหนด
η	learning rate (อัตราการเรียนรู้)
$\hat{y}_i^{(t-1)}$	ค่าพยากรณ์ก่อนรอบที่ t
$f_t(x_i)$	การอัปเดตจากอนุมูลรอบที่ t

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน ระดับน้ำทะเลที่เพิ่มสูงขึ้นส่งผลกระทบต่อคุณภาพน้ำและความสมดุลของระบบนิเวศในพื้นที่บางกระเจ้า จังหวัดสมุทรปราการ [1] ซึ่งเป็นพื้นที่อนุรักษ์สีเขียวที่ได้รับการประกาศโดยมติคณะรัฐมนตรีเมื่อปี พ.ศ. 2520 และได้รับการขนานนามว่า “ปอดของกรุงเทพฯ” เนื่องจากมีบทบาทสำคัญในการดูดซับมลพิษและรักษาความสมดุลของสิ่งแวดล้อมในเขตเมือง โดยการตรวจสอบคุณภาพน้ำจึงเป็นกระบวนการที่มีความสำคัญอย่างยิ่งต่อการจัดการทรัพยากรน้ำ การป้องกันปัญหาด้านสุขภาพ และการสนับสนุนการพัฒนาอย่างยั่งยืน เนื่องจากน้ำเป็นทรัพยากรพื้นฐานที่จำเป็นต่อมนุษย์และสิ่งแวดล้อม อย่างไรก็ตาม ในอดีตการตรวจสอบคุณภาพน้ำมักพึ่งพากำลังคนในการเก็บตัวอย่างและประเมินผล ซึ่งก่อให้เกิดข้อจำกัดทั้งในด้านค่าใช้จ่าย ความแม่นยำ และความรวดเร็วของการรายงานข้อมูล

การประยุกต์ใช้เทคโนโลยีเพื่อยกระดับกระบวนการตรวจสอบคุณภาพน้ำจึงเป็นแนวทางที่จำเป็น โดยเฉพาะเทคโนโลยี Internet of Things (IoT) [2-5] เป็นหนึ่งในเทคโนโลยีที่ได้รับความนิยมซึ่งตอบสนองความต้องการนี้ได้อย่างมีประสิทธิภาพ และสามารถนำไปใช้ในหลายพื้นที่ เช่น การทำฟาร์มอัจฉริยะ [6] เมืองอัจฉริยะ [7] สุขภาพอัจฉริยะ [8] และการเพาะเลี้ยงสัตว์น้ำอัจฉริยะ [9] เทคโนโลยีนี้ช่วยให้สามารถติดตั้งเซ็นเซอร์เพื่อรวบรวมข้อมูลจากแหล่งน้ำในพื้นที่ต่าง ๆ แบบต่อเนื่อง ลดความล่าช้า เพิ่มความแม่นยำ และสามารถส่งข้อมูลผ่านระบบไร้สายไปยังระบบคลาวด์เพื่อประมวลผลและแสดงผลแบบเรียลไทม์ผ่านแดชบอร์ดในรูปแบบที่เข้าใจง่าย จึงเป็นเครื่องมือสำคัญในการบริหารจัดการคุณภาพน้ำอย่างมีประสิทธิภาพ

ในงานวิจัยที่ผ่านมา [10] นำเสนอการพัฒนาอุปกรณ์พกพา IoT โดยใช้ไมโครคอนโทรลเลอร์สำหรับการตรวจสอบน้ำเค็มในพื้นที่บางกะเจ้าของประเทศไทย อย่างไรก็ตาม อุปกรณ์พกพานี้มีข้อจำกัดในการทำงานต่อเนื่องและการเชื่อมต่อเครือข่ายผ่าน Wi-Fi ส่งผลให้ข้อมูลพร้อมใช้งานได้เฉพาะในบางช่วงเวลา จากงานวิจัยที่ผ่านมา [11] นำเสนอการพัฒนาระบบแจ้งเตือนอัตโนมัติโดยจัดตั้งสถานีตรวจสอบคุณภาพน้ำโดยเฉพาะสำหรับระดับความเค็มในพื้นที่บางกะเจ้าสำหรับการตรวจสอบแบบเรียลไทม์ อย่างไรก็ตาม การพัฒนาสถานีนี้ถูกจำกัดด้วยความสามารถในการวัดและตัวเลือกแหล่งพลังงาน ส่งผลให้สูญเสียข้อมูลเป็นระยะๆ และการวัดคุณภาพน้ำที่จำกัดโดยเน้นที่ความเค็มเพียงอย่างเดียว และงานวิจัยที่ผ่านมา [12-15] เสนอระบบตรวจสอบคุณภาพน้ำโดยใช้บอร์ดไมโครคอนโทรลเลอร์เป็นตัวควบคุม อย่างไรก็ตาม การใช้ไมโครคอนโทรลเลอร์ยังมีข้อจำกัดในการทำงานต่อเนื่องและไม่เหมาะสำหรับระบบพลังงานแสงอาทิตย์หรือไฟฟ้ากระแสสลับ

วิทยานิพนธ์ฉบับนี้ จึงนำเสนอการออกแบบและพัฒนาสถานีตรวจสอบคุณภาพน้ำที่ใช้ IoT เพื่อประเมินความสามารถในการตรวจสอบคุณภาพน้ำในพื้นที่ต่างๆ อุปกรณ์นี้จะวัดค่าการนำไฟฟ้า ระดับ pH ปริมาณออกซิเจนที่ละลายน้ำ และอุณหภูมิของน้ำ และนำเสนอข้อมูลนี้ในรูปแบบกราฟิกผ่านแอปพลิเคชันบนเว็บ นอกจากนี้การทำงานของสถานีตรวจสอบคุณภาพน้ำที่ใช้ IoT ได้รับการปรับปรุงเพื่อรวบรวมข้อมูลสำหรับการวิเคราะห์เชิงทำนายค่าคุณภาพน้ำ ซึ่งเกี่ยวข้องกับการใช้เทคนิคการถดถอยเชิงเส้นหลายระดับควบคู่ไปกับแบบจำลอง Extreme Gradient Boosting

(XGBoost) [16-18] โดยมีการปรับพารามิเตอร์ผ่าน Optuna [19] เพื่อปรับความแม่นยำในการพยากรณ์ให้เหมาะสมที่สุด และยังรวมการเรียนรู้เชิงลึกโดยใช้แบบจำลอง Long Short-Term Memory (LSTM) [20-23] โดยมีจุดมุ่งหมายเพื่อเปรียบเทียบแบบจำลองทั้ง 2 เพื่อประเมินความเหมาะสมสำหรับการใช้งานจริง

1.2 วัตถุประสงค์การวิจัย

วิทยานิพนธ์ฉบับนี้มีวัตถุประสงค์เพื่อออกแบบและพัฒนาระบบตรวจสอบคุณภาพน้ำโดยใช้เทคโนโลยี Internet of Things (IoT) สำหรับการตรวจวัดและแสดงผลข้อมูลคุณภาพน้ำแบบเรียลไทม์ เพื่อให้สามารถติดตามและประเมินสภาพแหล่งน้ำได้อย่างต่อเนื่องและมีประสิทธิภาพ รวมถึงการพัฒนาแบบจำลองการพยากรณ์สำหรับคุณภาพน้ำ โดยใช้เทคนิคการถดถอยเชิงเส้นหลายระดับควบคู่กับแบบจำลอง Extreme Gradient Boosting (XGBoost) ซึ่งมีการปรับค่าพารามิเตอร์ด้วย Optuna เพื่อเพิ่มประสิทธิภาพในการพยากรณ์ และแบบจำลองการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM) ซึ่งเหมาะสำหรับข้อมูลที่มีลักษณะเป็นลำดับเวลา โดยจะทำการศึกษาวิจัยเปรียบเทียบประสิทธิภาพของแบบจำลอง XGBoost และ LSTM ในด้านความแม่นยำในการพยากรณ์และความเหมาะสมในการใช้งานจริง เพื่อเลือกแนวทางที่เหมาะสมที่สุดสำหรับการนำไปประยุกต์ใช้ในการจัดการคุณภาพน้ำอย่างยั่งยืน

1.3 พื้นที่ศึกษา

พื้นที่บางกระเจ้า ตั้งอยู่ในอำเภอพระประแดง จังหวัดสมุทรปราการ เป็นพื้นที่ที่เกิดจากการกัดเซาะและการไหลของแม่น้ำเจ้าพระยา ทำให้เกิดลักษณะโค้งบิดคล้ายกระเพาะหมู มีลักษณะเป็นแหลมยื่นออกมาจากแนวแม่น้ำ และถูกล้อมรอบด้วยแม่น้ำเจ้าพระยาจนมีลักษณะคล้ายเกาะขนาดเล็ก มีระยะทางจากทิศเหนือถึงทิศใต้ประมาณ 6 กิโลเมตร และจากทิศตะวันตกถึงทิศตะวันออกประมาณ 6 กิโลเมตร ครอบคลุมพื้นที่ประมาณ 18.9 ตารางกิโลเมตร ลักษณะภูมิประเทศส่วนใหญ่เป็นที่ราบ ไม่สูงจากระดับน้ำทะเล และอยู่ห่างจากอ่าวไทยไปทางทิศใต้ประมาณ 20 กิโลเมตร

ด้วยสภาพที่ตั้งทางภูมิศาสตร์ดังกล่าว พื้นที่บางกระเจ้าจึงได้รับอิทธิพลจากน้ำทะเลหนุนในช่วงเวลาต่าง ๆ ส่งผลให้พื้นที่มีลักษณะของแหล่งน้ำผสมผสานระหว่างน้ำจืด น้ำกร่อย และน้ำเค็ม พื้นที่นี้ยังมีความอุดมสมบูรณ์ทางทรัพยากรธรรมชาติ เหมาะสมสำหรับการเกษตร โดยเฉพาะสวนผลไม้ เช่น มะม่วงน้ำดอกไม้ ซึ่งเป็นพืชเศรษฐกิจที่มีความไวต่อความคุณภาพของน้ำ การรุกล้ำของน้ำทะเลจึงส่งผลกระทบโดยตรงต่อผลผลิตและสุขภาพของพืชสวน นอกจากนี้ พื้นที่บางกระเจ้ายังได้รับการกำหนดให้เป็นพื้นที่อนุรักษ์สีเขียวตามมติคณะรัฐมนตรีเมื่อปี พ.ศ. 2520 ซึ่งสะท้อนถึงความสำคัญของพื้นที่ในด้านสิ่งแวดล้อมและระบบนิเวศ โดยทั่วไปมักเรียกพื้นที่นี้ว่า “ปอดสีเขียวของกรุงเทพมหานคร” เนื่องจากมีบทบาทสำคัญในการดูดซับมลพิษ ฟอกอากาศ และรักษาสมดุลทางธรรมชาติให้กับพื้นที่เมืองหลวงที่มีความหนาแน่นของประชากรและกิจกรรมทางเศรษฐกิจสูง

การเลือกพื้นที่บางกระเจ้าเป็นพื้นที่ศึกษาในงานวิจัยนี้ มีเหตุผลสำคัญ 3 ประการ 1) พื้นที่บางกระเจ้าเป็นพื้นที่สีเขียวขนาดใหญ่ที่สุดของกรุงเทพมหานคร ซึ่งมีบทบาทสำคัญในการช่วยลดมลพิษและฟอกอากาศ 2) พื้นที่นี้ตั้งอยู่ในบริเวณปลายน้ำของแม่น้ำเจ้าพระยา และเชื่อมต่อกับอ่าวไทย จึงมีแนวโน้มได้รับผลกระทบของคุณภาพของน้ำจากการรุกของน้ำทะเลอย่างต่อเนื่อง 3)

พื้นที่บางกระเจ้ามีความหลากหลายทางระบบนิเวศสูง และเป็นพื้นที่ที่มีการเปลี่ยนแปลงของคุณภาพของน้ำตามฤดูกาลและอิทธิพลจากสิ่งแวดล้อม จึงเหมาะสมอย่างยิ่งสำหรับการศึกษา วิเคราะห์ และพยากรณ์คุณภาพน้ำ ทั้งในเชิงระบบนิเวศ วิศวกรรม และเศรษฐกิจ

รูปที่ 1.1 พื้นที่บางกระเจ้า
(**ที่มา**: เจสซี่ อัลเลน (2557) https://earthobservatory.nasa.gov/images/85382)

1.4 ขอบเขตการวิจัย

1.4.1 ศึกษาและค้นคว้าระบบการตรวจสอบคุณภาพน้ำที่ใช้เทคโนโลยี IoT สำหรับการตรวจวัดและแสดงผลข้อมูลคุณภาพน้ำแบบเรียลไทม์

1.4.2 พัฒนาต้นแบบสถานีตรวจสอบคุณภาพน้ำที่ใช้เทคโนโลยี IoT สำหรับการตรวจวัดและแสดงผลข้อมูลคุณภาพน้ำแบบเรียลไทม์

1.4.3 ศึกษาและค้นคว้าแบบจำลองการพยากรณ์สำหรับคุณภาพน้ำ โดยใช้เทคนิคการถดถอยเชิงเส้นหลายระดับ ร่วมกับแบบจำลอง XGBoost ที่มีการปรับพารามิเตอร์ด้วย Optuna และแบบจำลองการเรียนรู้เชิงลึก LSTM

1.4.4 พัฒนาต้นแบบแบบจำลองการพยากรณ์สำหรับคุณภาพน้ำ โดยใช้เทคนิคการถดถอยเชิงเส้นหลายระดับร่วมกับแบบจำลอง XGBoost ที่มีการปรับพารามิเตอร์ด้วย Optuna และแบบจำลองการเรียนรู้เชิงลึกแบบ LSTM

1.4.5 เปรียบเทียบประสิทธิภาพของแบบจำลอง XGBoost และ LSTM ในด้านความแม่นยำและความเหมาะสมในการใช้งานจริงสำหรับการพยากรณ์คุณภาพน้ำ

1.5 วิธีการวิจัย

1.5.1 ศึกษา ค้นคว้า วิจัยเกี่ยวกับการตรวจสอบคุณภาพน้ำที่ใช้เทคโนโลยี IoT สำหรับการตรวจวัดและแสดงผลข้อมูลคุณภาพน้ำแบบเรียลไทม์

1.5.2 ศึกษา ค้นคว้า วิจัยเกี่ยวกับแบบจำลองการพยากรณ์สำหรับคุณภาพน้ำ โดยใช้เทคนิคการถดถอยเชิงเส้นหลายระดับ ร่วมกับแบบจำลอง XGBoost ที่มีการปรับพารามิเตอร์ด้วย Optuna และแบบจำลองการเรียนรู้เชิงลึกแบบ LSTM

1.5.3 ออกแบบภาพรวมของระบบและสถาปัตยกรรม

1.5.4 พัฒนาต้นแบบสถานีตรวจสอบคุณภาพน้ำที่ใช้เทคโนโลยี IoT สำหรับการตรวจวัดและแสดงผลข้อมูลคุณภาพน้ำแบบเรียลไทม์ที่ได้ทำการออกแบบไว้

1.5.5 พัฒนาต้นแบบแบบจำลองการพยากรณ์สำหรับคุณภาพน้ำ โดยใช้เทคนิคการถดถอยเชิงเส้นหลายระดับร่วมกับแบบจำลอง XGBoost ที่มีการปรับพารามิเตอร์ด้วย Optuna และแบบจำลองการเรียนรู้เชิงลึก LSTM ที่ได้ทำการออกแบบไว้

1.5.6 ประเมินและเปรียบเทียบประสิทธิภาพของแบบจำลองการพยากรณ์ XGBoost และ LSTM ด้วย R-Squared MAE RMSE และ MSE ในด้านความแม่นยำและความเหมาะสมในการใช้งานจริงสำหรับการพยากรณ์คุณภาพน้ำ

1.5.7 ทดสอบและปรับปรุงความสามารถและประสิทธิภาพของระบบที่ได้พัฒนาขึ้น

1.5.8 วิเคราะห์ผลการทดลอง และสรุปผลการทดลอง

1.6 อุปกรณ์ที่ต้องใช้

1.6.1 ฮาร์ดแวร์

- เครื่องคอมพิวเตอร์สำหรับการพัฒนาระบบ	จำนวน 1 เครื่อง
- เครื่องเกตเวย์ที่มีการต่อเชื่อมกับเน็ตเวิร์ค (IoT Gateway)	จำนวน 1 เครื่อง
- อุปกรณ์วัดค่าความเป็นกรดและด่างในน้ำ (pH Sensor)	จำนวน 1 ชุด
- อุปกรณ์วัดค่าความนำไฟฟ้าในน้ำ (EC Sensor)	จำนวน 1 ชุด
- อุปกรณ์วัดค่าออกซิเจนในน้ำ (DO Sensor)	จำนวน 1 ชุด
- อุปกรณ์แปลงแรงดันไฟฟ้ากระแสสลับ	จำนวน 1 ชุด
- เครื่องควบคุมและชาร์จพลังงานแสงพร้อมเซลล์แสงอาทิตย์	จำนวน 1 ชุด
- แบตเตอรี่ขนาด 20 แอมแปร์ชั่วโมง	จำนวน 1 ชุด
- ตู้ระบบคอนโทรล	จำนวน 1 ชุด

1.6.2 ซอฟต์แวร์

- โปรแกรม Modbus Poll ในการจัดการระบบการสื่อสารมาตรฐาน RS485 Modbus RTU ระหว่างเกตเวย์กับอุปกรณ์ต่างๆ
- โปรแกรม Node-RED ให้ทำงานเป็น Edge Computing รวมถึงการจัดการข้อมูลของอุปกรณ์เซ็นเซอร์ และข้อมูลจาก AWS Cloud Service โดยมาตรฐานการสื่อสาร MQTT Protocol
- โปรแกรม WinSCP สำหรับอัปโหลดและดาวน์โหลดไฟล์ผ่าน Protocol FTP จากคอมพิวเตอร์ของผู้ใช้ไปยัง Gateway

- โปรแกรม Putty สำหรับ Remote Server หรือ SSH (Secure Shell) จากคอมพิวเตอร์ของผู้ใช้ไปยัง Gateway
- โปรแกรม Python ในการพัฒนาระบบต้นแบบแบบจำลองการพยากรณ์สำหรับคุณภาพน้ำ
- Amazon Web Services DynamoDB เพื่อเป็นฐานข้อมูล
- Amazon Web Services IoT Core เพื่อช่วยให้สามารถเชื่อมต่ออุปกรณ์เกตเวย์กับบริการของ Amazon Web Services ผ่านโพรโทคอล MQTT

1.7 ประโยชน์ที่ได้จากการวิจัย

สามารถตรวจสอบคุณภาพน้ำได้อย่างแม่นยำและทันเวลาโดยสามารถวัดและแสดงผลข้อมูลแบบเรียลไทม์ ช่วยให้หน่วยงานหรือชุมชนสามารถติดตามคุณภาพแหล่งน้ำได้อย่างมีประสิทธิภาพมากขึ้น พร้อมทั้งสนับสนุนการบริหารจัดการทรัพยากรน้ำอย่างยั่งยืนผ่านแบบจำลองการพยากรณ์คุณภาพน้ำ ซึ่งช่วยคาดการณ์แนวโน้มล่วงหน้าและนำไปสู่การวางแผนการจัดการที่เหมาะสมในพื้นที่เสี่ยง เช่น การรุกล้ำของน้ำทะเล รวมถึงการเปรียบเทียบแบบจำลอง XGBoost และ LSTM ยังช่วยเลือกโมเดลที่เหมาะสมที่สุดสำหรับการพยากรณ์จริง เพิ่มความแม่นยำและความน่าเชื่อถือของข้อมูล ระบบที่พัฒนาขึ้นสามารถนำไปประยุกต์ใช้ในพื้นที่อื่นที่มีปัญหาด้านคุณภาพน้ำ และสามารถต่อยอดเป็นเครือข่ายตรวจสอบคุณภาพน้ำในระดับกว้าง นอกจากนี้ยังเป็นต้นแบบของการบูรณาการเทคโนโลยีดิจิทัลเพื่อสิ่งแวดล้อม ที่สามารถขยายผลสู่การพัฒนาเมืองอัจฉริยะและระบบนิเวศอัจฉริยะในอนาคตได้อย่างเป็นรูปธรรม

1.8 ส่วนประกอบของการวิจัย

เนื้อหาของวิทยานิพนธ์ฉบับนี้ ประกอบด้วย 5 บท ได้แก่

บทที่ 1 บทนำ

บทที่ 2 ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง

บทที่ 3 การออกแบบระบบและสถาปัตยกรรม

บทที่ 4 การดำเนินงานและผลการวิจัย

บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ

บทที่ 2
ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง

2.1 Internet of Things (IoT)

ระบบ Internet of Things (IoT) คือแนวคิดและเทคโนโลยีที่เชื่อมโยงอุปกรณ์อิเล็กทรอนิกส์ต่างๆ เข้าด้วยกันผ่านทางเครือข่ายอินเทอร์เน็ต โดยอุปกรณ์เหล่านี้สามารถรับ ส่งข้อมูล และสื่อสารกันได้โดยอัตโนมัติ โดยไม่จำเป็นต้องมีการควบคุมจากมนุษย์ตลอดเวลา จุดเด่นของ IoT คือความสามารถในการสั่งงานอุปกรณ์จากระยะไกลผ่านเครือข่ายอินเทอร์เน็ต เช่น การเปิด-ปิดเครื่องใช้ไฟฟ้า การตรวจสอบสถานะอุปกรณ์ และการรวบรวมข้อมูลจากอุปกรณ์ต่างๆ เพื่อใช้ในการวิเคราะห์และตัดสินใจ

ในอดีต อุปกรณ์อิเล็กทรอนิกส์มักทำหน้าที่เพียงเป็นตัวรับหรือแสดงผลข้อมูลเท่านั้น แต่ในยุคปัจจุบัน ด้วยเทคโนโลยี IoT อุปกรณ์ต่างๆ ได้กลายเป็น "อุปกรณ์อัจฉริยะ" ที่สามารถประมวลผล สื่อสาร และโต้ตอบกับสิ่งแวดล้อมหรือกับอุปกรณ์อื่นๆ ได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น อุปกรณ์ในบ้านอัจฉริยะ (Smart Home) อย่างหลอดไฟที่สามารถเปิด-ปิดอัตโนมัติตามเวลาหรือการตรวจจับความเคลื่อนไหว, ระบบรักษาความปลอดภัยที่แจ้งเตือนไปยังมือถือของผู้ใช้เมื่อมีสิ่งผิดปกติ, หรือแม้แต่ตู้เย็นที่สามารถตรวจสอบว่ามีอาหารอะไรเหลืออยู่บ้างและแจ้งเตือนเมื่อของใกล้หมดอายุ อีกตัวอย่างหนึ่งที่เห็นได้ชัดคืออุปกรณ์สวมใส่อัจฉริยะ เช่น นาฬิกาอัจฉริยะ หรือรองเท้าวิ่งที่สามารถบันทึกข้อมูลการเคลื่อนไหว ความเร็ว ระยะทาง สถานที่ และอัตราการเต้นของหัวใจ แล้วส่งข้อมูลเหล่านี้ไปเก็บบน Cloud หรือแหล่งจัดเก็บข้อมูลออนไลน์ เพื่อให้ผู้ใช้สามารถดูข้อมูลย้อนหลัง วิเคราะห์แนวโน้มสุขภาพของตนเอง หรือแม้แต่แชร์ให้แพทย์วินิจฉัยร่วมได้

หนึ่งในเทคโนโลยีที่มีความสำคัญมากในระบบ IoT คือ Cloud Storage หรือพื้นที่จัดเก็บข้อมูลออนไลน์ ซึ่งเป็นอีกหนึ่งรูปแบบของ IoT ที่ผู้คนใช้งานเป็นประจำโดยไม่รู้ตัว ระบบคลาวด์ช่วยให้สามารถจัดเก็บข้อมูลได้อย่างปลอดภัย ไม่ต้องกังวลเรื่องข้อมูลสูญหาย หรือการโจรกรรมข้อมูล อีกทั้งยังสามารถเข้าถึงได้ทุกที่ทุกเวลา ด้วยอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ที่เชื่อมต่ออินเทอร์เน็ต เช่น คอมพิวเตอร์ สมาร์ตโฟน หรือแท็บเล็ต นอกจากนี้ยังช่วยลดค่าใช้จ่าย เพราะไม่ต้องลงทุนซื้ออุปกรณ์จัดเก็บข้อมูลเอง เช่น ฮาร์ดไดรฟ์ หรือแฟลชไดรฟ์

ระบบ IoT ได้ถูกนำมาใช้ในหลายด้าน ทั้งในภาคอุตสาหกรรม การเกษตร การแพทย์ การจราจร และการศึกษา ตัวอย่างเช่น ระบบ Smart Grid ที่ช่วยควบคุมการใช้พลังงานไฟฟ้าให้มีประสิทธิภาพ, ระบบ Smart Transportation ที่ช่วยให้การบริหารจัดการจราจรเป็นไปอย่างแม่นยำและทันสมัย, หรือระบบ Smart Agriculture ที่ใช้เซนเซอร์วัดอุณหภูมิ ความชื้น และสภาพดินเพื่อปรับการให้น้ำและปุ๋ยได้อย่างเหมาะสม

ข้อดีของระบบ IoT มีมากมาย เช่น 1) ช่วยเพิ่มความสะดวกสบายและประหยัดเวลาในการดำเนินชีวิต2) เพิ่มความรวดเร็วและความแม่นยำในการทำงาน 3) ช่วยลดต้นทุนและเพิ่มโอกาสในการพัฒนาเทคโนโลยีใหม่ๆ 4) ทำให้สามารถวิเคราะห์ข้อมูลได้แบบ Real-time เป็นต้น

อย่างไรก็ตาม ระบบ IoT ก็มีข้อจำกัดและความท้าทายบางประการ เช่น ต้องพึ่งพาเครือข่ายอินเทอร์เน็ตในการทำงาน หากระบบอินเทอร์เน็ตล่มหรือขัดข้อง อุปกรณ์อาจไม่สามารถทำงานได้ตามปกติ อีกทั้งยังมีความเสี่ยงด้านความปลอดภัยของข้อมูล เพราะการส่งข้อมูลผ่านเครือข่ายอินเทอร์เน็ตอาจถูกโจมตีจากผู้ไม่หวังดีได้ นอกจากนี้ ระบบ IoT บางระบบยังมีความซับซ้อนในการออกแบบ การติดตั้ง และการบำรุงรักษา

2.1.1 MQTT Protocol

MQTT (Message Queue Telemetry Transport) [24-25] คือโปรโตคอลที่ใช้ในการส่งข้อความ ซึ่งได้รับความนิยมในอุปกรณ์ประเภท IoT (Internet of Things) โดยมีการออกแบบมาให้เหมาะสมกับการสื่อสารที่มีขนาดข้อมูลต่ำและการใช้งานในเครือข่ายที่มีแบนด์วิธต่ำ (low bandwidth) โปรโตคอลนี้ขับเคลื่อนโดยเหตุการณ์ (Event-driven) และใช้รูปแบบการเชื่อมต่อแบบ Publish-Subscribe ซึ่งช่วยให้การส่งและรับข้อมูลระหว่างอุปกรณ์ทำได้อย่างมีประสิทธิภาพ

ในระบบ MQTT จะมี 3 องค์ประกอบหลัก ได้แก่ Publisher, Subscriber, และ Broker:

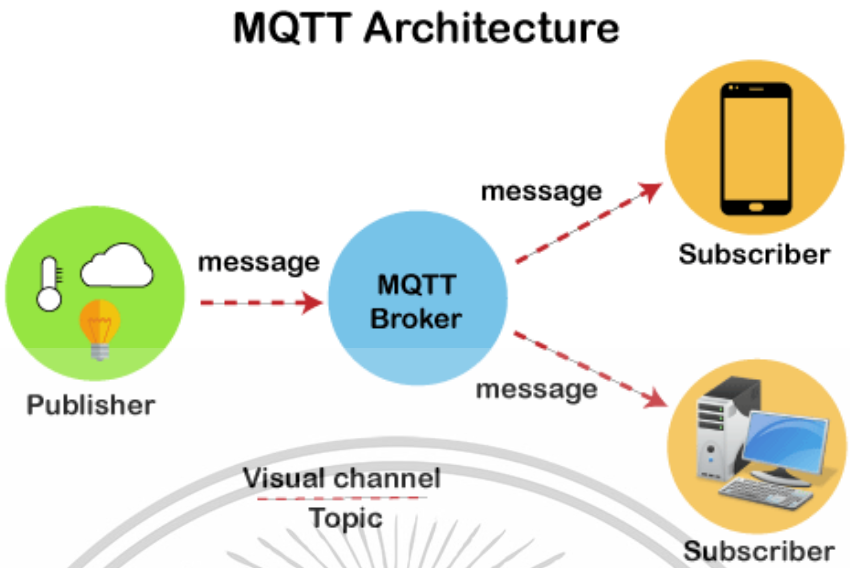
1) Publisher คือ อุปกรณ์หรือแอปพลิเคชันที่ส่งข้อมูล โดยต้องระบุ Topic (หัวข้อ) ที่ต้องการส่งข้อมูลไป
2) Subscriber คือ อุปกรณ์หรือแอปพลิเคชันที่รอรับข้อมูลจาก Topic ที่สมัครสมาชิก (Subscribe)
3) Broker คือ ตัวกลางที่ทำหน้าที่รับข้อมูลจาก Publisher และกระจายข้อมูลไปยัง Subscriber ที่สมัครรับข้อมูลจาก Topic นั้นๆ

จุดเด่นของ MQTT คือการใช้ Publish/Subscribe Model ที่ช่วยให้การส่งข้อมูลทำได้อย่างมีประสิทธิภาพ โดยที่ผู้สงข้อมูลไม่จำเป็นต้องรู้จักผู้รับข้อมูล และผู้รับข้อมูลก็ไม่จำเป็นต้องรู้จักผู้ส่งข้อมูล สิ่งที่สำคัญคือการ เชื่อมโยงผ่าน Topic ซึ่งเป็นการระบุหัวข้อที่ผู้ส่งข้อมูลต้องการเผยแพร่ และผู้รับข้อมูลต้องการรับข้อมูลในหัวข้อนั้น

สำหรับการใช้งานในระบบ IoT, MQTT ได้รับความนิยมเป็นอย่างมากเนื่องจากเหมาะสมกับการส่งข้อมูลในรูปแบบ real-time โดยไม่ใช้พลังงานมากเกินไป ซึ่งมีความสำคัญสำหรับอุปกรณ์ IoT ที่มักจะมีข้อจำกัดด้านพลังงานและแบนด์วิธ เช่น เซ็นเซอร์, อุปกรณ์สมาร์ตโฮม, หรืออุปกรณ์ในโรงงานอุตสาหกรรมที่ต้องการการสื่อสารแบบประหยัดพลังงานและมีประสิทธิภาพสูง

การส่งข้อมูลผ่าน MQTT จะเริ่มต้นจาก Publish ซึ่งเป็นการส่งข้อมูลไปยัง Topic ที่กำหนดไว้ และ Subscribe ซึ่งเป็นการรับข้อมูลที่ส่งผ่านมาใน Topic ที่ผู้ใช้งานหรืออุปกรณ์ได้สมัครรับข้อมูลไว้ ตัวอย่างเช่น หากต้องการรับข้อมูลอุณหภูมิจากเซ็นเซอร์ในบ้าน สามารถสมัครรับข้อมูลจาก home/office/temperature ในขณะที่เซ็นเซอร์จะ Publish ข้อมูลไปยัง Topic นี้

นอกจากนี้ Broker ยังสามารถเป็นบริการที่มีอยู่ในคลาวด์หรือสามารถตั้งค่าในเครือข่ายภายในองค์กรได้เอง โดยใช้โปรแกรม เช่น Mosquitto ซึ่งเป็น MQTT Broker ที่ได้รับความนิยมและสามารถใช้งานได้ฟรี ช่วยให้ผู้ใช้งานสามารถจัดการการเชื่อมต่อและการส่งผ่านข้อมูลในระบบ IoT ได้ง่ายและมีประสิทธิภาพ



รูปที่ 2.1 ภาพรวมการทำงานของ MQTT
(Learning Inventions Laboratory 2565: https://medium.com/@lil.eng.cmu/การสื่อสารกับสมองกลฝังตัวผ่าน-mqtt-47633bb76e58)

2.1.2 Smart water

Smart Water หรือ น้ำอัจฉริยะ [26] คือการใช้เทคโนโลยีและระบบดิจิทัลในการตรวจสอบ, บริหารจัดการ, และเพิ่มประสิทธิภาพในการใช้น้ำผ่านการเชื่อมโยงและการวิเคราะห์ข้อมูลจากอุปกรณ์ต่างๆ ที่ติดตั้งในระบบน้ำ เช่น เซ็นเซอร์, ตัวตรวจวัด, และระบบ IoT (Internet of Things) ที่สามารถส่งข้อมูลจากการวัดค่าต่างๆ เช่น ระดับน้ำ, ความขุ่น, อุณหภูมิ, หรือความเค็ม ไปยังระบบคลาวด์เพื่อการวิเคราะห์และการจัดการในระดับที่มีประสิทธิภาพ โดยไม่จำเป็นต้องมีการตรวจสอบด้วยตนเอง การใช้เทคโนโลยีเหล่านี้ช่วยให้สามารถจัดการและใช้ทรัพยากรน้ำได้อย่างมีประสิทธิภาพ, ลดการสูญเสียของน้ำ, และส่งเสริมการอนุรักษ์น้ำอย่างยั่งยืน

ตัวอย่างการใช้งานของ Smart Water ได้แก่ การตรวจสอบคุณภาพน้ำในแหล่งน้ำต่างๆ เช่น น้ำประปา, น้ำใช้ในเกษตรกรรม, หรือแหล่งน้ำในอุตสาหกรรม ผ่านการติดตั้งเซ็นเซอร์วัดค่าและการใช้ Big Data และ Machine Learning เพื่อทำนายและจัดการการใช้น้ำได้อย่างแม่นยำและมีประสิทธิภาพสูงสุด นอกจากนี้ยังสามารถใช้งานในการตรวจสอบสภาพแวดล้อมและช่วยในการตัดสินใจสำหรับการจัดสรรน้ำในพื้นที่ต่างๆ ได้อย่างเหมาะสม

2.2 Amazon Web Services

คลาวด์ (Cloud) คือการนำเครื่องเซิร์ฟเวอร์หลายๆ ตัวมาเชื่อมต่อและทำงานร่วมกันในลักษณะของ Cluster ซึ่งทำหน้าที่ในการจัดเก็บข้อมูล, ประมวลผล หรือฟังก์ชันอื่นๆ ที่เกี่ยวข้อง โดยที่ผู้พัฒนาสามารถใช้งานได้โดยไม่จำเป็นต้องมีเครื่องเซิร์ฟเวอร์ของตัวเอง การใช้งานนี้คือการจำลองเซิร์ฟเวอร์บนเครื่องเซิร์ฟเวอร์จริง ๆ ซึ่งช่วยให้สะดวกในการเข้าถึงทรัพยากรคอมพิวเตอร์และการจัดการข้อมูลในระบบคลาวด์ ในช่วงการทดสอบเริ่มแรก โปรเจคนี้ได้เลือกใช้ Firebase และในขั้นตอนต่อมาจึงได้มีการปรับเปลี่ยนมาใช้ Amazon Web Services (AWS) [27]

Amazon Web Services (AWS) คือแพลตฟอร์มคลาวด์ที่ให้บริการโดย Amazon ซึ่งมีบริการมากกว่า 200 บริการจากศูนย์ข้อมูล (Data Centers) ทั่วโลก AWS เน้นการให้บริการออนไลน์ที่สามารถขยายขนาดได้ตามความต้องการ และมีประสิทธิภาพในด้านต้นทุน การให้บริการของ AWS ครอบคลุมหลายประเภท เช่น บริการการประมวลผล, การจัดเก็บข้อมูล, ฐานข้อมูล, การทำเครือข่ายและการขนส่งข้อมูล, เครื่องมือความปลอดภัย, เครื่องมือพัฒนา และเครื่องมือในการจัดการระบบฯลฯ สำหรับในโปรเจคนี้ ผู้จัดทำได้เลือกใช้เครื่องมือจาก AWS ที่เหมาะสมกับการดำเนินงานในระบบโดยเลือกใช้บริการต่างๆ ดังต่อไปนี้

2.2.1 AWS IoT Core

AWS IoT Core เป็นบริการที่มีอยู่ใน AWS Cloud ซึ่งทำหน้าที่ในการเชื่อมต่ออุปกรณ์ IoT เข้ากับระบบคลาวด์และบริการอื่นๆ โดยรองรับการตั้งค่าการส่งข้อมูลต่างๆ ด้วยโปรโตคอลต่างๆ เช่น MQTT ทำให้การส่งข้อมูลจากอุปกรณ์สามารถเกิดขึ้นได้อย่างมีประสิทธิภาพผ่านหัวข้อที่กำหนดไว้ล่วงหน้า โดยก่อนที่จะทำการเชื่อมต่อ การยืนยันความปลอดภัยของอุปกรณ์จะทำผ่าน Certificate ของอุปกรณ์นั้น ๆ ซึ่งจะช่วยรับประกันได้ว่า การรับส่งข้อมูลจะเกิดขึ้นระหว่างอุปกรณ์ที่ได้รับการยืนยันตัวตนแล้วเท่านั้น ดังนั้น การเชื่อมต่อและส่งข้อมูลจึงมีความปลอดภัยและสามารถควบคุมได้อย่างแม่นยำ

2.2.2 DynamoDB

DynamoDB เป็นฐานข้อมูลแบบ NoSQL ที่จัดเก็บข้อมูลในรูปแบบ key-value หรือกุญแจ-ค่า ซึ่งมีข้อได้เปรียบในเรื่องของความยืดหยุ่นในการจัดเก็บข้อมูลที่สามารถรองรับรายละเอียดที่ไม่จำกัดจำนวน และสามารถเก็บข้อมูลที่มีหลายค่าในแต่ละระเบียนได้ อีกทั้งยังสามารถปรับขนาดการทำงานได้อัตโนมัติตามปริมาณข้อมูลที่เพิ่มขึ้น โดยผู้ใช้ไม่จำเป็นต้องจัดการกับการติดตั้งและการดูแลรักษาฐานข้อมูลเอง ทำให้ DynamoDB เป็นตัวเลือกที่ดีในการจัดเก็บข้อมูลที่มีการเติบโตสูงและต้องการความเร็วในการเข้าถึงข้อมูลที่สูง

2.3 สถานีตรวจสอบคุณภาพน้ำ

2.3.1 เกตเวย์

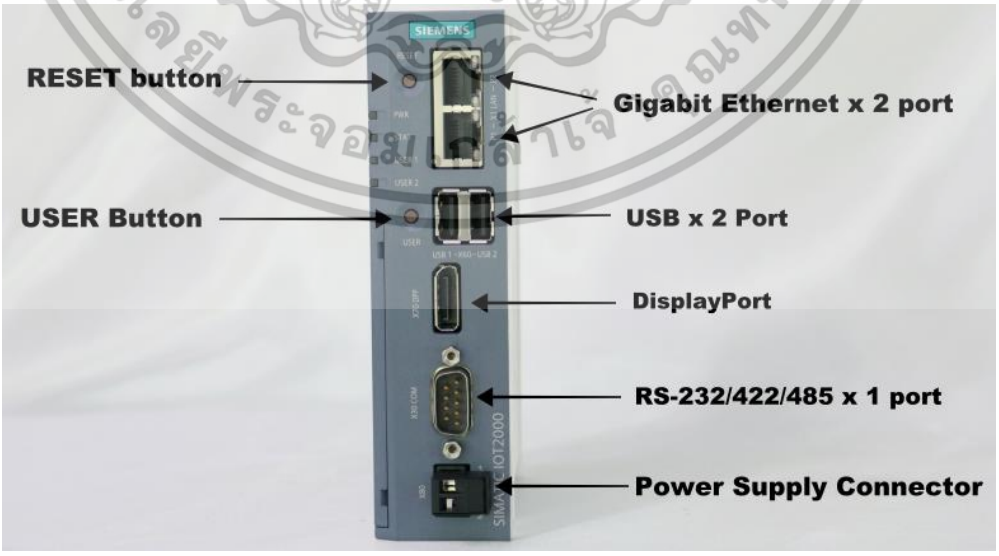
เกตเวย์ (Gateway) คือ อุปกรณ์ฮาร์ดแวร์ที่ทำหน้าที่เชื่อมต่อเครือข่ายต่าง ๆ เข้าด้วยกัน โดยสามารถทำงานกับเครือข่ายที่ใช้โปรโตคอลหรือเทคโนโลยีการสื่อสารที่แตกต่างกันได้ เช่น การเชื่อมต่อเครือข่ายที่ใช้โปรโตคอล TCP/IP กับเครือข่ายที่ใช้โปรโตคอลอื่น ๆ หรือการเชื่อมต่อเครือข่ายที่ใช้สื่อส่งข้อมูลประเภทต่าง ๆ เข้าด้วยกันอย่างไม่มีข้อจำกัด เช่น การใช้เกตเวย์ในการเชื่อมต่อระหว่างคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการ Windows กับระบบ MacOS หรือการเชื่อมต่อเครือข่าย Ethernet LAN ที่ใช้สาย UTP เข้ากับเครือข่าย Token Ring LAN

เกตเวย์ช่วยให้การส่งข้อมูลระหว่างเครือข่ายที่มีเทคโนโลยีหรือโครงสร้างต่างกันสามารถทำงานร่วมกันได้อย่างราบรื่น โดยทำหน้าที่แปลงข้อมูลหรือโปรโตคอลที่ใช้ในเครือข่ายหนึ่งให้อยู่ในรูปแบบที่สามารถรับรู้และประมวลผลในเครือข่ายอื่นได้

2.3.1.1 SIMATIC IoT2050 gateways

SIMATIC IoT2050 [28] เป็น Open industrial IoT gateway ที่เปิดให้ใช้งานในลักษณะโอเพ่นซอร์ส ซึ่งออกแบบมาเพื่อสนับสนุนการพัฒนาของนักพัฒนารุ่นใหม่ที่ต้องการศึกษาและใช้งาน Internet of Things (IoT) ในอุตสาหกรรม โดยมันถูกสร้างขึ้นเพื่อขยายจากโปรโตไทป์ที่พัฒนาด้วยบอร์ดเปิดอย่าง Arduino, Intel Galileo หรือ Raspberry Pi ให้สามารถนำไปใช้งานจริงในสภาพแวดล้อมอุตสาหกรรม

แม้ว่า IoT2050 จะไม่ใช่ PLC (Programmable Logic Controller) แต่ก็ถูกออกแบบให้ทำหน้าที่เป็นศูนย์กลางของทุกแอปพลิเคชันใน IoT โดยสามารถเก็บข้อมูลจาก PLC หรือเซ็นเซอร์ต่าง ๆ ได้จากทุกจุดในโลก มันยังรองรับการต่ออุปกรณ์เสริม เช่น Arduino shields และสามารถสื่อสารกับอุปกรณ์อื่น ๆ ในระบบอุตสาหกรรมผ่านโปรโตคอลอุตสาหกรรม เช่น MODBUS และ PROFINET รวมถึงสามารถใช้งานโปรโตคอลอินเทอร์เน็ตอย่าง MQTT ได้ เพื่อส่งข้อมูลและรับคำสั่งจากระบบต่าง ๆ อย่างมีประสิทธิภาพ



รูปที่ 2.2 พอร์ตการเชื่อมต่อของ SIMATIC IoT2050 gateways
(IBCONBLOG 2563 : https://blog.ibcon.com/?p=739)

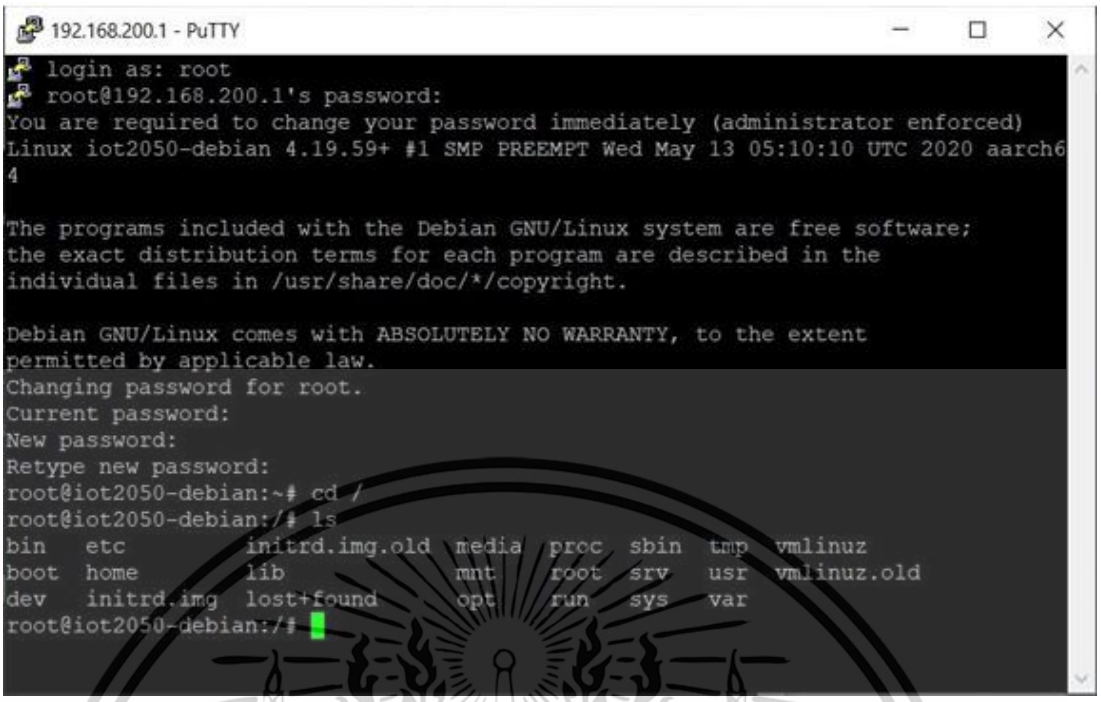
ตารางที่ 2.1 เปรียบเทียบความสามารถในการทำงาน SIMATIC IoT gateways ในแต่ละเวอร์ชัน

Topic	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 16GB
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	2x Gbit Ethernet RJ45	2 x Gbit Ethernet RJ45
serial interface	2x 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
Mounting	Din rail, Wall mount (Optional)	Din rail, Wallmount	Din rail, Wall mount

2.3.1.2 ระบบปฏิบัติการลีนุกซ์บนเกตเวย์

ระบบปฏิบัติการ (Operating System) คือ ชุดโปรแกรมที่ทำหน้าที่ควบคุมการทำงานของระบบคอมพิวเตอร์ รวมถึงการจัดการอุปกรณ์ต่างๆ และเชื่อมโยงการทำงานระหว่างผู้ใช้และเครื่องคอมพิวเตอร์ ระบบปฏิบัติการจะรับผิดชอบในการจัดสรรทรัพยากรในระบบให้มีประสิทธิภาพสูงสุด เช่น หน่วยความจำ, การประมวลผล, การจัดการไฟล์ และการเชื่อมต่อกับอุปกรณ์ต่างๆ

ระบบปฏิบัติการลีนุกซ์ (Linux) เป็นระบบปฏิบัติการที่คล้ายคลึงกับ ดอส, ไมโครซอฟต์วินโดวส์ หรือ ยูนิกซ์ ซึ่งลีนุกซ์จัดเป็นหนึ่งในระบบปฏิบัติการที่พัฒนาบนพื้นฐานของยูนิกซ์ โดยลีนุกซ์มีคุณสมบัติที่โดดเด่นคือ ฟรีแวร์ (Freeware) หรือเป็นระบบที่ไม่ต้องเสียค่าใช้จ่ายในการซื้อโปรแกรม ซึ่งแตกต่างจากหลายๆ ระบบปฏิบัติการที่ต้องเสียค่าลิขสิทธิ์ ดังตัวอย่างรูปที่ 2.2



รูปที่ 2.3 ตัวอย่างระบบปฏิบัติการลีนุกซ์บนเกตเวย์

ระบบลีนุกซ์สามารถทำงานร่วมกับโปรแกรมต่างๆ โดยเฉพาะโปรแกรมในตระกูล GNU (GNU's Not UNIX) ซึ่งเป็นโปรแกรมที่เปิดเผยโค้ดให้ผู้ใช้สามารถพัฒนาและปรับปรุงได้ ในขณะที่ลีนุกซ์ยังไม่สามารถทดแทน ไมโครซอฟต์ วินโดวส์ หรือ แมคโอเอส (Mac OS) ในการใช้งานบนพีซีได้ทั้งหมด แต่ก็มีผู้ใช้จำนวนไม่น้อยที่เลือกใช้และช่วยกันพัฒนาโปรแกรมบนลีนุกซ์ ด้วยเครื่องมือและฟีเจอร์ต่างๆ ที่ช่วยให้การดูแลระบบลีนุกซ์ทำได้สะดวกและมีประสิทธิภาพมากยิ่งขึ้น

2.3.2 อุปกรณ์วัดข้อมูลต่างๆภายในสถานีตรวจสอบคุณภาพน้ำ

2.3.2.1 อุปกรณ์วัดค่าความเป็นกรดและด่างในน้ำ (pH Sensor)

รายละเอียดของอุปกรณ์วัดค่าความเป็นกรดและด่างในน้ำ (PH and Temperature Transmitter)

- ช่วงการวัดค่า pH: 0.0~14.0
- ช่วงการวัดค่าอุณหภูมิ: -20℃ ~+80℃
- ความแม่นยำในการวัดค่า pH: ±0.1PH
- ความแม่นยำในการวัดค่าอุณหภูมิ: ± 0.5℃
- แรงดันไฟฟ้า DC ที่ใช้งานได้: 12V ~ 24V (ริปเปิล < 50mV)
- เอาต์พุตสัญญาณ: Modbus RTU-485
- บอดการสื่อสาร: ค่าเริ่มต้น 4800 (สามารถตั้งค่า 2400, 4800, 9600 ได้)
- พลังงานที่สิ้นเปลือง: <1W
- อุณหภูมิสภาพแวดล้อมในการจัดเก็บ: 10℃ - 50℃ (-20℃ ~ +80℃ พีค)
- ความชื้นในสภาพแวดล้อมการจัดเก็บ: 20 - 60%RH
- อุณหภูมิสภาพแวดล้อมในการทำงาน: -20℃ ~+80℃
- ความชื้นในสภาพแวดล้อมการทำงาน: 5 - 95%RH (ไม่ควบแน่น)
- ขนาดรูปร่าง: 65 มม. * 46 มม.* 28.5 มม.

รูปที่ 2.4 อุปกรณ์วัดค่าความเป็นกรดและด่างในน้ำ (pH Sensor)

2.3.2.2 อุปกรณ์วัดค่าความนำไฟฟ้าในน้ำ (EC Sensor)

รายละเอียดของอุปกรณ์วัดค่าความนำไฟฟ้าในน้ำ (EC transmitter RS485)

- แรงดันไฟฟ้าในการทำงาน: DC12-24V
- การใช้พลังงาน: <1W
- ช่วง EC: 0-4400us/cm
- ความแม่นยำในการตรวจจับ: ≤±2%F*S (เต็มสเกล)
- สัญญาณเอาต์พุต: Modbus RTU-485
- บอดการสื่อสาร: ค่าเริ่มต้น 4800 (สามารถตั้งค่า 2400, 4800, 9600 ได้)
- ความต้านทานโหลด: เอาต์พุตกระแส: โหลด R ≤ (Uvcc-3) / 0.02Ω
- สภาพแวดล้อมการทำงาน: อุณหภูมิ: 0-100 ° C, ความชื้น: 0-100% RH
- สภาพแวดล้อมในการจัดเก็บ: อุณหภูมิ: 10-50 ° C (-20 ~ 80 ° C สูงสุด) ความชื้น: 20-60% RH
- ขนาด: 65 มม. * 46 มม. * 28.5 มม.
- วัสดุอิเล็กโทรด: อิเล็กโทรด ABS แพลตตินัมสีดำ 5 ม.วัสดุโพรบ: สแตนเลส 316
- วัสดุปิดผนึก: อีพอกซีเรซินทนไฟสีดำ

รูปที่ 2.5 อุปกรณ์วัดค่าความนำไฟฟ้าในน้ำ (EC Sensor)

2.3.2.3 อุปกรณ์วัดค่าออกซิเจนในน้ำ (DO Sensor)

รายละเอียดของอุปกรณ์วัดค่าออกซิเจนในน้ำ (Dissolved Oxygen transmitter RS485)

- แรงดันไฟใช้งาน: 10 - 30 VDC
- กำลังไฟใช้งาน: 0.2W
- สัญญาณเอาต์พุต: Modbus RTU-RS485
- บอดการสื่อสาร: ค่าเริ่มต้น 4800 (สามารถตั้งค่า 2400, 4800, 9600 ได้)
- ช่วงการวัดค่า: 0 - 20 mg/L (0-200% Saturation)
- เทคนิคการวัดค่า: Fluorescent
- ความแม่นยำ: ±0.2%FS
- ความละเอียด: 0.01mg/L,0.1%,0.1C
- ความถี่ในการตอบสนอง: ≤60 วินาที
- อุณหภูมิใช้งาน: 0 - 40 ° C
- อุณหภูมิในการจัดเก็บ: -10 - 60 ° C
- มาตรฐานการกันน้ำ: IP68

รูปที่ 2.6 อุปกรณ์วัดค่าออกซิเจนในน้ำ (DO Sensor)

2.3.3 การสื่อสารระหว่างอุปกรณ์วัดข้อมูลต่างๆกับเกตเวย์

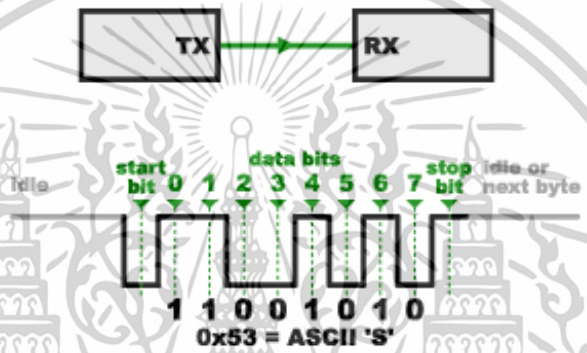
การสื่อสารระหว่างอุปกรณ์ต่างๆ กับเกตเวย์มักใช้การสื่อสารแบบอนุกรม (Serial Communication) [29-31] ซึ่งทำให้ข้อมูลถูกส่งผ่านทีละบิต โดยต่างจากการสื่อสารแบบขนาน (Parallel Communication) ที่ส่งข้อมูลพร้อมกันทุกบิต การสื่อสารแบบอนุกรมมีข้อดีคือสามารถใช้สายเชื่อมต่อน้อยกว่าการสื่อสารแบบขนาน แต่มีข้อเสียคือต้องใช้เวลามากกว่าในการส่งข้อมูล

การสื่อสารแบบอนุกรมแบ่งออกเป็น 3 รูปแบบหลัก ได้แก่:

1) Simplex – การสื่อสารทางเดียว ซึ่งเลือกได้แค่การส่งข้อมูลไปหรือกลับ แต่ไม่สามารถทำได้ทั้งสองทิศทางพร้อมกัน
2) Half-Duplex – การสื่อสารสองทาง แต่ไม่สามารถส่งข้อมูลทั้งสองทิศทางพร้อมกันได้ การส่งข้อมูลจะสลับไปมาระหว่างการส่งไปและกลับ
3) Full-Duplex – การสื่อสารสองทางที่สามารถส่งข้อมูลไปและกลับพร้อมกันได้

การสื่อสารแบบอนุกรมสามารถแบ่งออกเป็น 2 ลักษณะหลัก คือ

1) Synchronous เป็นการรับ-ส่งข้อมูลเป็นบล็อกในคราวเดียวหลายๆ ไบต์ โดยใช้สัญญาณนาฬิกาช่วยในการประสานการทำงานระหว่างตัวรับและตัวส่งข้อมูลให้ทำงานในลำดับเดียวกัน
2) Asynchronous เป็นการรับ-ส่งข้อมูลทีละ 1 ไบต์ (8 บิต) โดยใช้รูปแบบของ Stream bit ซึ่งประกอบด้วย
   - Start bit: ใช้ระบุจุดเริ่มต้นของข้อมูล ขนาด 1 บิต
   - Data bit: แทนข้อมูล ขนาด 5 ถึง 8 บิต
   - Parity bit: ใช้ตรวจสอบความถูกต้องของข้อมูล ขนาด 0 หรือ 1 บิต
   - Stop bit: ใช้ระบุจุดสิ้นสุดของข้อมูล ขนาด 1, 1.5 หรือ 2 บิต



รูปที่ 2.7 ความหมายของ Stream bit

ระยะเวลาการส่งข้อมูลและอัตราบิต (bit rate) ขึ้นอยู่กับการใช้งานและลักษณะของระบบ ซึ่งจะถูกวัดเป็นจำนวนบิตต่อวินาที (bit/second) โดยมีค่ามาตรฐานหลายช่วง เช่น 2400, 4800, 9600 บิตต่อวินาที เป็นต้น ค่าของอัตราบิตนี้จะกำหนดความเร็วในการส่งข้อมูลระหว่างอุปกรณ์ในระบบการสื่อสาร

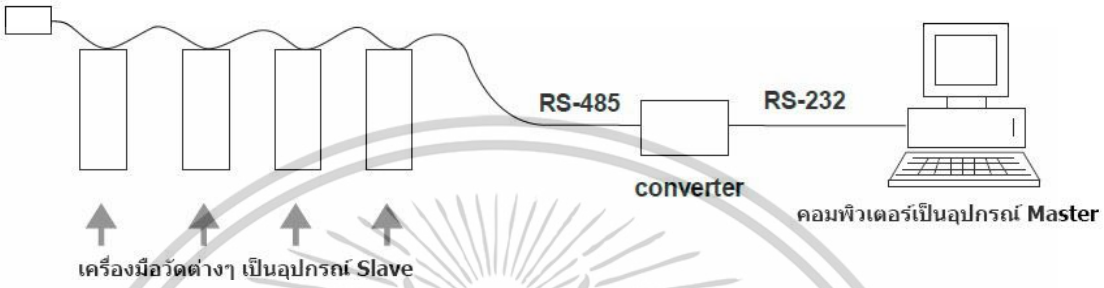
2.3.3.1 การสื่อสารด้วยมาตรฐาน RS485

Recommended Standard No. 485 (RS-485) [32-33] เป็นมาตรฐานการรับ-ส่งข้อมูลแบบ Half-duplex ซึ่งใช้สำหรับการสื่อสารระหว่างอุปกรณ์ต่างๆ โดยใช้เพียงแค่ 2 สาย คือ สาย A และ B ที่ทำหน้าที่เป็นตัวส่งสัญญาณดิจิตอล การส่งข้อมูลจะใช้ค่าความต่างของแรงดันระหว่างสาย A และ B เพื่อบ่งบอกข้อมูลที่ถูกส่งไปยังปลายทาง

รูปที่ 2.8 การเชื่อมต่อ RS485 ระหว่างเครื่องมือวัดกับตัวแปลงสัญญาณ

- เมื่อแรงดันระหว่าง Va – Vb มีค่าต่ำกว่า -200 mV จะเป็นสัญญาณดิจิตอล 1
- เมื่อแรงดันระหว่าง Va – Vb มีค่ามากกว่า +200 mV จะเป็นสัญญาณดิจิตอล 0

การทำงานในระบบเครือข่าย (Network) อุปกรณ์สามารถเชื่อมต่อกันได้สูงสุดถึง 32 ตัว โดยมี 1 ตัวที่ทำหน้าที่เป็นตัวจัดการลำดับการสื่อสาร (Master) ส่วนอุปกรณ์ที่เหลือจะทำหน้าที่เป็น Slave ทุกตัวอุปกรณ์จะมี ID ที่เป็นของตัวเอง โดย Master จะส่งคำสั่งพร้อมกับ ID ไปยังทุก Slave และหากคำสั่งนั้นตรงกับ ID ของอุปกรณ์ใด อุปกรณ์นั้นจะทำการปฏิบัติตามคำสั่งที่ได้รับ



รูปที่ 2.9 การทำงาน RS485 แบบ Network

ข้อดีของ RS485

- สามารถส่งสัญญาณได้ไกลสูงสุดถึง 1200 เมตร ซึ่งเหมาะสมสำหรับการใช้งานในระบบอุตสาหกรรม
- รองรับการเชื่อมต่อในรูปแบบเครือข่าย (Network) แบบ Multipoint ได้สูงสุดถึง 32 อุปกรณ์
- เป็นมาตรฐานที่ใช้เพียงสายไฟ 2 เส้นในการรับส่งข้อมูล

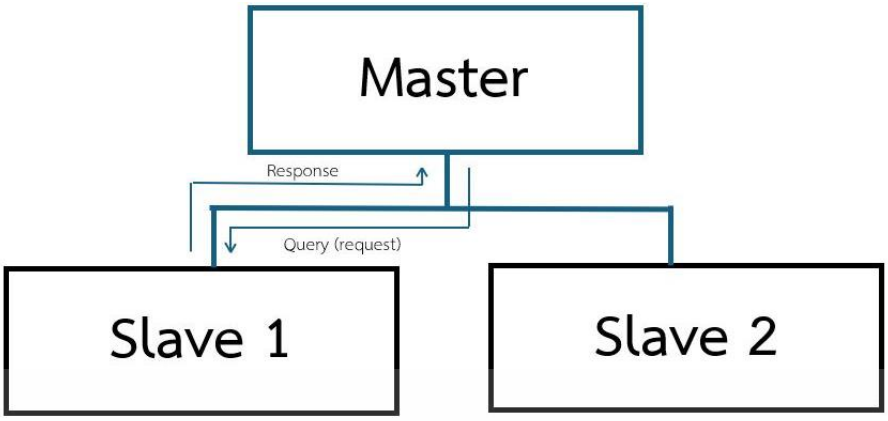
ข้อเสียของ RS485

- จำเป็นต้องใช้ตัวแปลงสัญญาณในการเชื่อมต่อกับคอมพิวเตอร์หรือเกตเวย์
- ความเร็วในการรับส่งข้อมูลค่อนข้างต่ำ

2.3.3.2 การสื่อสารด้วยมาตรฐาน Modbus

Modbus [34] คือ โปรโตคอลการสื่อสารแบบอนุกรมที่ใช้รูปแบบการสื่อสารแบบ Master-Slave (หรือ Client-Server) โดยที่อุปกรณ์ตัวเดียวในระบบจะทำหน้าที่เป็น Master (หรือ Client) ซึ่งเป็นผู้เริ่มต้นการสื่อสาร (queries) ส่วนอุปกรณ์อื่นๆ ในระบบจะทำหน้าที่เป็น Slave (หรือ Server) ที่ตอบสนองต่อคำขอจาก Master โดยการส่งข้อมูลที่ร้องขอกลับไป หรือดำเนินการตามคำขอที่ Master ระบุไว้

Slave สามารถเป็นอุปกรณ์ต่อพ่วงหลากหลายชนิด เช่น I/O transducer, วาล์ว, Inverter (VFD) หรือเครื่องมือวัดต่างๆ ที่ประมวลผลข้อมูลและส่งข้อมูลเหล่านั้นไปยัง Master ซึ่งจะทำหน้าที่ควบคุมการสื่อสารทั้งหมดในระบบ



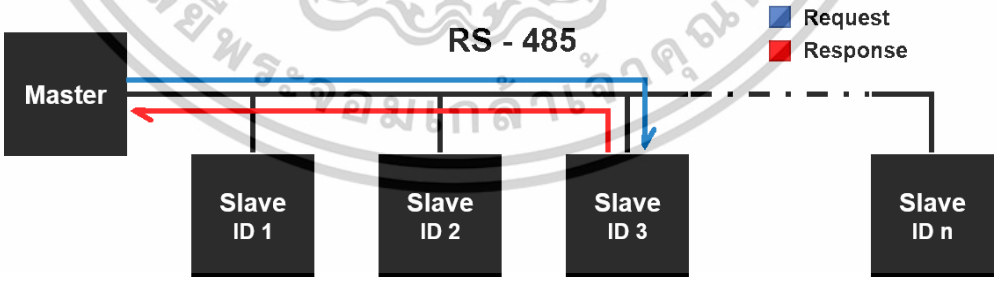
รูปที่ 2.10 การทำงานและการเชื่อมต่อของ Modbus

Master สามารถติดต่อกับ Slave แต่ละตัวได้โดยตรง หรือสามารถส่งข้อความในลักษณะของการ Broadcast ถึง Slave ทุกตัวได้ แต่ Slave จะตอบสนองเฉพาะสิ่งที่ Master ต้องการเท่านั้น ข้อมูลที่ Master ส่งไปจะประกอบด้วย Slave address (ที่อยู่ของ Slave), function code (รหัสคำสั่งหรือคำขอที่ต้องการให้ทำ), Data (ข้อมูลที่ต้องการให้ดำเนินการ) และ Checksum (ค่าตรวจสอบความถูกต้องของข้อมูล)

ในขณะเดียวกัน ข้อมูลที่ Slave ส่งกลับมาจะประกอบด้วยคำสั่งที่ได้รับการดำเนินการตามคำขอจาก Master พร้อมข้อมูลที่ต้องการ และ Checksum เพื่อยืนยันความถูกต้องของข้อมูลที่ส่งกลับ

2.3.3.3 การสื่อสารด้วยมาตรฐาน Modbus RTU

Modbus RTU คือ โปรโตคอลที่ใช้การสื่อสารแบบอนุกรม (Serial-based Protocol) โดยมีการสื่อสารในรูปแบบ Master/Slave ซึ่งอุปกรณ์ที่ทำหน้าที่เป็น Master จะเป็นผู้เริ่มต้นการร้องขอข้อมูลจาก Slave และอุปกรณ์ Slave จะตอบกลับข้อมูล (Response) เฉพาะเมื่อมีการร้องขอจาก Master เท่านั้น



รูปที่ 2.11 การทำงานและการเชื่อมต่อของ Modbus RTU
(ธีรเชษฐ์ 2563 : https://www.nectec.or.th/news-public-document/modbus-protocol.html)

Modbus RTU โดยทั่วไปจะใช้ในการสื่อสารผ่าน RS-232 หรือ RS-485 และข้อมูลในโปรโตคอล Modbus จะถูกจัดเก็บใน 4 รูปแบบหลัก ได้แก่

1) Output coils: ตำแหน่ง address เริ่มต้นที่ 000001 ใช้เก็บข้อมูลที่มีค่าเป็นบิตเดียว เช่น การเปิดหรือปิดของอุปกรณ์รีเลย์หรือสวิตช์

2) Input contacts: ตำแหน่ง address เริ่มต้นที่ 100001 ใช้เก็บข้อมูลการรับสัญญาณจากอุปกรณ์ที่ตรวจสอบสถานะ เช่น สวิตช์
3) Input registers: ตำแหน่ง address เริ่มต้นที่ 300001 ใช้เก็บข้อมูลที่มาจากอุปกรณ์ตรวจวัด เช่น เซ็นเซอร์ หรืออุปกรณ์ที่ส่งข้อมูลแบบอนาล็อก โดยเก็บเป็นตัวเลข 16 บิต
4) Holding registers: ตำแหน่ง address เริ่มต้นที่ 400001 ใช้เก็บข้อมูลที่สามารถอ่านและเขียนได้ เช่น ค่าที่อ่านจากอุปกรณ์ตรวจวัดหรือเก็บค่าตัวแปรในระบบอัตโนมัติ

โดย Output coils และ Input contacts จะเก็บค่าเพียง 1 บิตเท่านั้น (ค่า "0" หรือ "1") ขณะที่ Input registers และ Holding registers สามารถเก็บค่าเป็นตัวเลข 16 บิต ซึ่งเป็นค่าที่มาจากอุปกรณ์ตรวจวัดที่ส่งข้อมูลในรูปแบบอนาล็อก (Analog) ตามตารางที่ 2.2

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 จะรับส่งข้อมูลเป็นชุดข้อมูล ซึ่งแต่ละชุดจะประกอบด้วย 6 ส่วนหลักๆ ตามตารางที่ 2.3

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 × 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

ชุดข้อมูลนี้จะมีการจัดส่งในรูปแบบที่เป็นระเบียบ เพื่อให้การสื่อสารระหว่าง Master และ Slave เป็นไปได้อย่างถูกต้องและมีประสิทธิภาพ

Function code ในโปรโตคอล Modbus RTU สามารถแบ่งหน้าที่ได้ตามรหัส ซึ่งจะมีฟังก์ชันการทำงานหลักๆ 2 แบบ คือ การอ่าน (Read) และการเขียน (Write) ไปยัง Coils, Contacts หรือ Registers ตามตารางที่ 2.4

ตารางที่ 2.4 Function code ของการรับ-ส่ง RS485 Modbus RTU

Function Code (DEC)	Action	Data Type	Object Type
1	Read	Single bit	Output Coils
5	Write Single	Single bit	Output Coils
15	Write Multiple	Single bit	Output Coils
2	Read	Single bit	Input Contacts
4	Read	Word (16bit)	Input Registers
3	Read	Word (16bit)	Holding Registers
6	Write Single	Word (16bit)	Holding Registers
16	Write Multiple	Word (16bit)	Holding Registers

ในแต่ละฟังก์ชันจะมีรหัสที่แตกต่างกันเพื่อกำหนดการทำงานและประเภทของข้อมูลที่จะถูกส่งหรือรับจากอุปกรณ์ Slave

ในส่วนชุดข้อมูล Data Field ของโปรโตคอล Modbus RTU จะถูกแบ่งออกเป็น 2 ชุดหลัก ได้แก่ ชุดคำสั่งสำหรับการอ่านและชุดคำสั่งสำหรับการเขียน ตามตารางที่ 2.5-2.6

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)

ชุดคำสั่งนี้จะใช้เมื่ออุปกรณ์ Master ต้องการดึงข้อมูลจากอุปกรณ์ Slave โดยจะมีการส่งคำสั่งไปยัง Slave เพื่อนำข้อมูลจาก Coils, Registers หรือ Contacts มาให้ Master เช่น การอ่านค่า Input Registers หรือการตรวจสอบสถานะของ Output Coils

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)

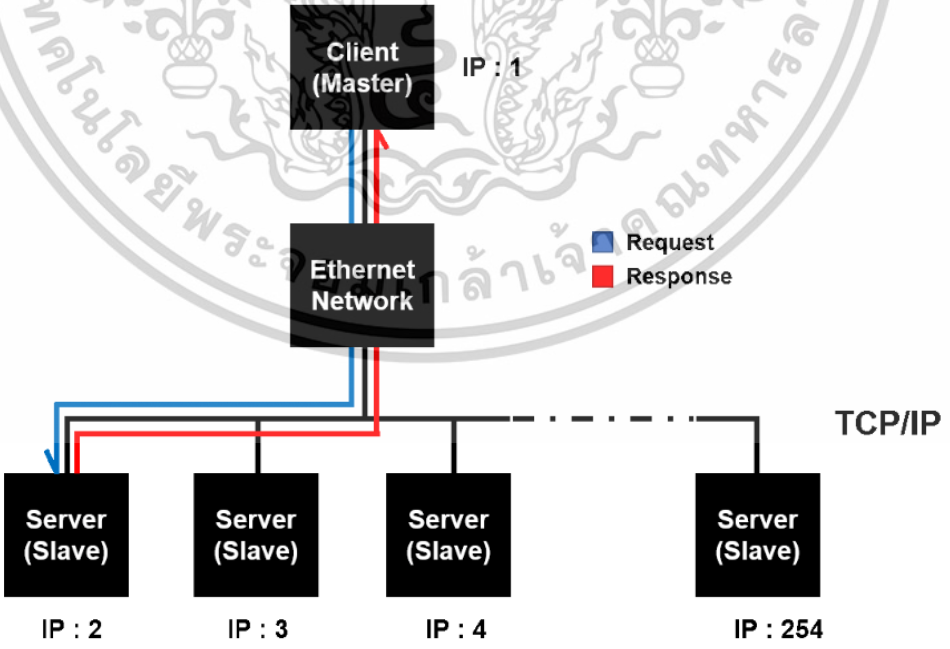
ชุดคำสั่งนี้ใช้เมื่ออุปกรณ์ Master ต้องการส่งคำสั่งไปยังอุปกรณ์ Slave เพื่อให้ Slave ทำการเปลี่ยนแปลงข้อมูล เช่น การเขียนค่าลงใน Holding Registers หรือการควบคุมการเปิด/ปิด Output Coils

ชุดคำสั่งทั้ง 2 ประเภทนี้จะถูกส่งจากอุปกรณ์ที่ทำหน้าที่เป็น Master เท่านั้น เพื่อสั่งการให้กับอุปกรณ์ Slave ที่ต้องการสื่อสาร ซึ่งจะทำหน้าที่ตามคำสั่งที่ได้รับ

2.3.3.4 การสื่อสารด้วยมาตรฐาน Modbus TCP/IP

Modbus TCP คือโปรโตคอลที่พัฒนาต่อยอดจาก Modbus RTU โดยมีการนำมาใช้งานร่วมกับระบบเครือข่าย Ethernet ผ่านโปรโตคอล TCP/IP ซึ่งใช้พอร์ตสื่อสารมาตรฐานคือ Port 502 แทนการสื่อสารแบบอนุกรมที่ใช้ใน Modbus RTU เดิม

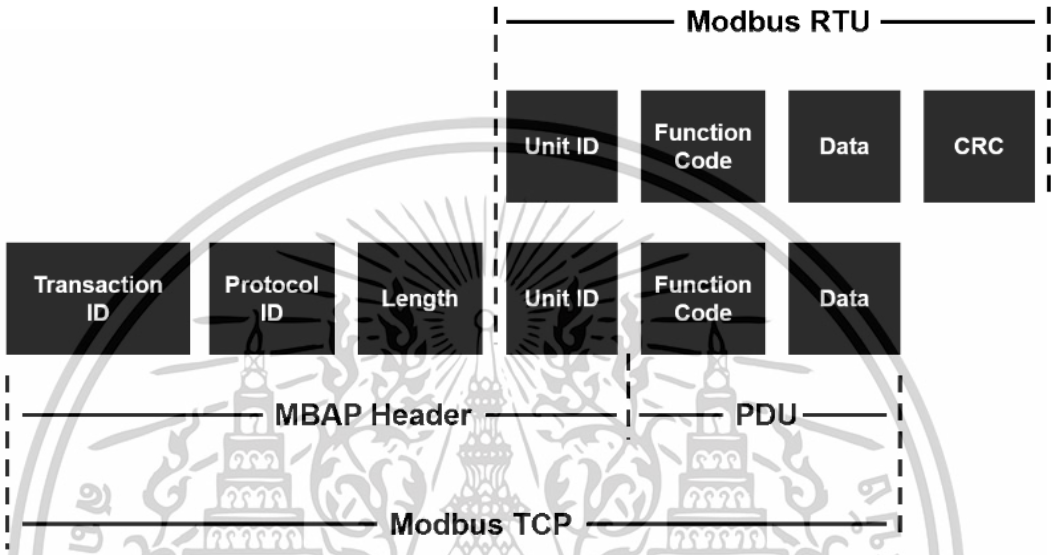
ด้วยการทำงานบนเครือข่าย Local Area Network (LAN) หรือ Internet ทำให้ Modbus TCP สามารถเชื่อมต่ออุปกรณ์ต่างๆ ได้อย่างสะดวกมากยิ่งขึ้น ทั้งในรูปแบบ มีสาย (Ethernet) และ ไร้สาย (Wireless) โดยใช้อุปกรณ์เชื่อมต่อ เช่น Router หรือ Access Point เป็นตัวกลางในการส่งข้อมูลระหว่าง Master และ Slave แสดงไว้ดังรูปที่ 2.12



รูปที่ 2.12 การทำงานและการเชื่อมต่อของ Modbus TCP/IP
(ธีรเชษฐ์ 2563 : https://www.nectec.or.th/news-public-document/modbus-protocol.html)

การนำ Modbus TCP มาใช้ในระบบอัตโนมัติ ช่วยให้การสื่อสารรวดเร็ว รองรับการขยายระบบได้ง่าย และสามารถเข้าถึงข้อมูลจากระยะไกลได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

จุดที่แตกต่างสำคัญคือ Modbus TCP จะไม่มีการใช้ CRC (Cyclic Redundancy Check) สำหรับตรวจสอบความผิดพลาดในชุดข้อมูล เพราะมีการใช้ระบบตรวจสอบความถูกต้องของข้อมูลที่มีอยู่แล้วใน Data Link Layer ของ Ethernet แทน ดังรูปที่ 2.11



รูปที่ 2.13 ส่วนประกอบชุดข้อมูลของ Modbus TCP เทียบกับ Modbus RTU
(ธีรเชษฐ์ 2563 : https://www.nectec.or.th/news-public-document/modbus-protocol.html)

ในชุดข้อความของ Modbus TCP จะเริ่มต้นด้วยส่วนที่เรียกว่า MBAP Header (Modbus Application Protocol Header) ซึ่งเป็นส่วนที่เพิ่มเติมเข้ามาจาก Modbus RTU เพื่อรองรับการทำงานบนเครือข่าย TCP/IP โดยประกอบด้วยข้อมูลสำคัญ 4 ส่วน ได้แก่:

1) Transaction ID ใช้จับคู่ว่าข้อมูลที่ส่งออกไปกับข้อมูลที่ได้รับกลับมาจาก Slave เป็นชุดเดียวกัน
2) Protocol ID ใช้ระบุโปรโตคอลที่ใช้ (สำหรับ Modbus จะมีค่าเป็น 0)
3) Length ระบุความยาวของข้อมูลที่ตามหลัง MBAP Header
4) Unit ID ใช้แทนหมายเลข Slave ID เพื่อระบุอุปกรณ์ปลายทาง

หลังจาก MBAP Header แล้ว จะตามด้วย Function Code และ Data Field ซึ่งยังคงมีโครงสร้างเหมือนกับ Modbus RTU ตามตารางที่ 2.7

ตารางที่ 2.7 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.

2.3.4 พลังงานแสงอาทิตย์

พลังงานแสงอาทิตย์ (Solar Energy) [35-36] คือ พลังงานที่ได้จากการแผ่รังสีของดวงอาทิตย์มายังพื้นโลก โดยพลังงานนี้อยู่ในรูปของรังสีคลื่นแม่เหล็กไฟฟ้าทั้งในช่วงแสงที่ตามองเห็น รังสีอินฟราเรด และรังสีอัลตราไวโอเลต พลังงานแสงอาทิตย์ถือเป็นพลังงานสะอาด (Clean Energy) และเป็นพลังงานหมุนเวียน (Renewable Energy) ที่ไม่มีวันหมดอีกด้วย

2.3.4.1 การประยุกต์ใช้พลังงานแสงอาทิตย์

การประยุกต์ใช้พลังงานแสงอาทิตย์ โดยพลังงานแสงอาทิตย์สามารถนำไปประยุกต์ใช้ได้หลากหลาย ทั้งนำมาใช้ในการผลิตพลังงานไฟฟ้าโดยตรงผ่านเซลล์แสงอาทิตย์ (Solar Cells) หรือใช้ในการอบแห้งผลิตผลทางการเกษตรในรูปแบบต่างๆ ใช้ในการถนอมอาหารให้อยู่ได้นานขึ้น เพื่อนำเข้าและส่งออกผลผลิต โดยคนส่วนมากนิยมแปลงพลังงานแสงอาทิตย์เป็นพลังงานไฟฟ้า เพื่อนำมาใช้งาน มากกว่าการนำความร้อนของแสงอาทิตย์ไปใช้โดยตรง โดยใช้เซลล์แสงอาทิตย์ (Solar Cells) หรือแผงโซลาร์เซลล์ในการแปลงแสงจากดวงอาทิตย์เป็นพลังงานไฟฟ้า ทั้งใช้กับอุปกรณ์พกพา เช่น พาวเวอร์แบงก์แสงอาทิตย์และโคมไฟโซลาร์เซลล์ เป็นต้น หรือหากยังไม่ต้องการใช้งานไฟฟ้าที่ได้มาในทันที สามารถนำมาเก็บไว้ในแบตเตอรี่เพื่อสำรองพลังงานไว้ ก่อนนำออกมาใช้ได้

2.3.4.2 แผงโซลาร์เซลล์

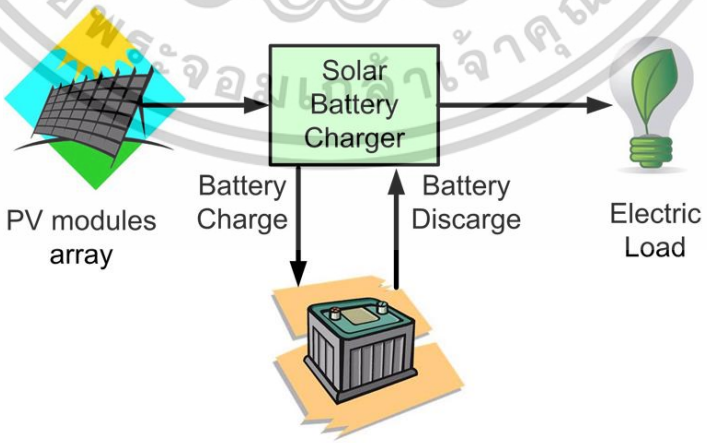
แผงโซลาร์เซลล์ (Solar Panel) ทำหน้าที่แปลงพลังงานแสงอาทิตย์ให้เป็นพลังงานไฟฟ้าโดยตรง ผ่านกระบวนการที่เรียกว่า Photovoltaic Effect ซึ่งภายในแผงจะประกอบด้วยเซลล์แสงอาทิตย์ที่ทำจากสารกึ่งตัวนำ เช่น ซิลิกอน (Silicon) เมื่อแสงแดดตกกระทบกับผิวของเซลล์แสงอาทิตย์ โฟตอน (Photons) จากแสงจะกระตุ้นให้เกิดการเคลื่อนที่ของอิเล็กตรอนในวัสดุกึ่งตัวนำ ทำให้เกิดกระแสไฟฟ้ากระแสตรง (DC) ออกมาจากแผง จากนั้นกระแสนี้จะถูกส่งไปยังตัวควบคุมการชาร์จเพื่อจัดการก่อนเก็บเข้าที่แบตเตอรี่

2.3.4.3 ตัวควบคุมการชาร์จ

ตัวควบคุมการชาร์จ (Solar Charge Controller) หน้าที่ของตัวควบคุมการชาร์จ หรือ Solar Charger คือ ควบคุมการไหลของกระแสไฟฟ้าจากแผงโซลาร์เซลล์เข้าสู่แบตเตอรี่ เพื่อป้องกันไม่ให้แบตเตอรี่เกิดการชาร์จเกิน (Overcharging) หรือจ่ายไฟมากเกินจนแรงดันไฟตก (Over-discharge) โดย Solar Charger มีอยู่ 2 ประเภทหลัก คือ PWM (Pulse Width Modulation) ควบคุมแรงดันไฟฟ้าอย่างง่าย มีต้นทุนต่ำ เหมาะกับระบบขนาดเล็ก และ MPPT (Maximum Power Point Tracking) มีความสามารถในการดึงพลังงานสูงสุดจากแผงได้อย่างมีประสิทธิภาพ เหมาะกับระบบขนาดกลางถึงใหญ่ที่ต้องการใช้ไฟฟ้าในระบบค่อนข้างมาก นอกจากนี้ Solar Charger ยังมีบทบาทสำคัญในการแปลงแรงดันให้เหมาะสมกับแบตเตอรี่ เช่น ลดแรงดันจาก 18 โวลล์ ของแผงให้เหลือ 12 โวลล์ เพื่อให้แบตเตอรี่ 12 โวลล์ เก็บพลังงานได้อย่างปลอดภัย

2.3.4.4 แบตเตอรี่

แบตเตอรี่ (Battery) ทำหน้าที่ในการเก็บพลังงานไฟฟ้าที่ได้จากแผงโซลาร์เซลล์ผ่านตัวชาร์จ เพื่อให้สามารถนำมาใช้งานได้ภายหลัง โดยเฉพาะในช่วงเวลากลางคืนหรือเมื่อไม่มีแสงแดด ชนิดของแบตเตอรี่ที่นิยมใช้ในระบบโซลาร์เซลล์ ได้แก่ แบตเตอรี่ตะกั่ว-กรด (Lead-acid battery) มีราคาถูก ทนทาน แต่มีน้ำหนักมาก หรือ แบตเตอรี่ลิเธียมไอออน (Lithium-ion battery) มีน้ำหนักเบา ประสิทธิภาพสูง อายุการใช้งานยาว แต่ราคาสูงกว่า และ LiFePO4 (ลิเธียมเฟอรัสฟอสเฟต) มีความปลอดภัยสูง ไม่ระเบิดง่าย เหมาะกับงานพลังงานหมุนเวียน



รูปที่ 2.14 ภาพรวมการทำงานของการประยุกต์ใช้พลังงานแสงอาทิตย์
(Michele 2555 : https://www.powersystemsdesign.com/articles/solar-battery-charger/28/5809)

การทำงานของระบบทั้งสามส่วนร่วมกันนั้นแผงโซลาร์เซลล์จะรับพลังงานแสงเข้ามาและแปลงเป็นพลังงานไฟฟ้า หลังจากนั้น Solar Charge Controller รับไฟฟ้าจากแผง โดยมีการควบคุมแรงดันไฟฟ้าและกระแสไฟฟ้า ให้อยู่ในระดับคงที่ และ สามารถนำไปเก็บไว้ในแบตเตอรี่ได้ โดยแบตเตอรี่จะทำการเก็บพลังงานไว้สำหรับใช้งานภายหลัง เช่น จ่ายไฟให้หลอดไฟหรือจ่ายไฟให้กับระบบที่ต้องการใช้ไฟฟ้า บางระบบอาจต่อ Inverter เพื่อแปลงไฟเป็นไฟฟ้ากระแสสลับ (AC) สำหรับใช้กับเครื่องใช้ไฟฟ้าทั่วไป โดยประโยชน์ของระบบการทำงานร่วมกันของโซลาร์เซลล์ ตัวควบคุมการชาร์จและแบตเตอรี่นั้น สามารถลดค่าใช้จ่ายทางไฟฟ้าให้กับงานได้ สามารถใช้งานและตั้งตั้งได้ง่ายในทุกพื้นที่ ทั้งที่ห่างไกลหรือพื้นที่ที่ไฟฟ้าเข้าถึงได้ยาก ที่สำคัญสามารถพกพาได้ง่าย ใช้กับระบบโซลาร์แบบเคลื่อนที่ได้ มีหลากหลายขนาดให้เลือกใช้ เพื่อความเหมาะสมกับระบบการทำงานและสามารถลดปัญหาพลังงานไม่เสถียรในพื้นที่ต่างๆได้อีกด้วย

2.3.5 เครื่องมือที่ใช้ในการพัฒนา

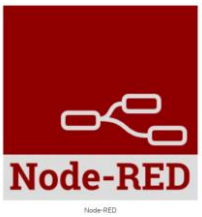
2.3.5.1 node-red

Node-RED [37-39] เป็นเครื่องมือที่ช่วยในการจัดการและควบคุมเหตุการณ์ต่างๆ โดยใช้ Node.js ซึ่งช่วยให้การพัฒนาแอปพลิเคชัน IoT ง่ายขึ้น โดยทำงานเป็นเว็บเซิร์ฟเวอร์ที่ผู้ใช้สามารถเข้าถึงและปรับแต่งได้จากเว็บเบราว์เซอร์ของคอมพิวเตอร์เครื่องใดก็ได้ ทำให้ผู้ใช้สามารถเชื่อมต่อและจัดการกับฮาร์ดแวร์ต่างๆ รวมถึงสร้างกระบวนการทำงาน (workflow) ได้อย่างสะดวกและรวดเร็ว โดยไม่จำเป็นต้องเขียนโค้ดมากมาย

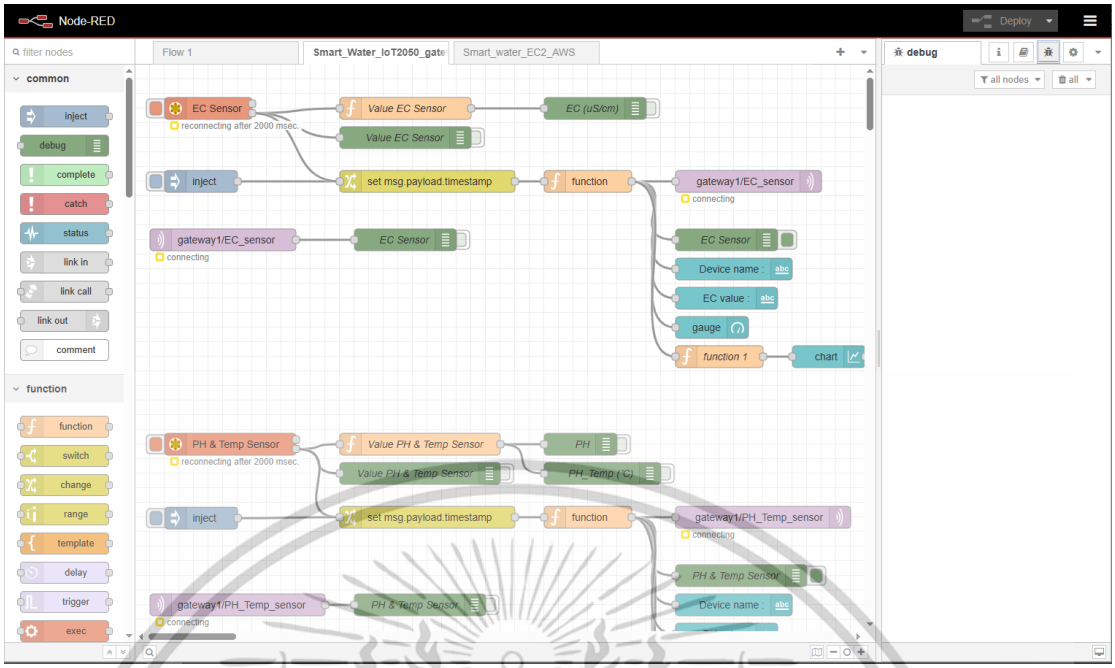
Node-RED ใช้ Flow-Based Programming ซึ่งช่วยให้การพัฒนาโครงการ IoT เป็นไปอย่างง่ายดายและเข้าใจได้ง่าย โดยการลากและเชื่อมต่อ Node ต่างๆ ที่มีอยู่ในระบบ เมื่อใช้ Node-RED ผู้ใช้สามารถเลือกจากชุด Node ที่มีบริการต่างๆ ให้เลือกใช้งาน เช่น MQTT, HTTP, TCP/UDP, และอื่นๆ โดยไม่ต้องเขียนโค้ดจากศูนย์ สิ่งนี้ช่วยลดความยุ่งยากในการพัฒนาแอปพลิเคชัน IoT

ในส่วนของ API Development Node-RED ช่วยให้ผู้ใช้สามารถออกแบบ API ที่เกี่ยวข้องกับ IoT ได้อย่างง่ายดายผ่านเว็บเบราว์เซอร์ โดยการใช้ Node ต่างๆ ที่ช่วยในการรับข้อมูล คำนวณหรือแปลงข้อมูล และเชื่อมต่อกับบริการต่างๆ ผ่านการลากและเชื่อมต่อ Node ตามต้องการ นอกจากนี้ Node-RED ยังรองรับการใช้งานฟอร์แมตข้อมูล JSON (JavaScript Object Notation) ที่สามารถนำเข้าและส่งออกได้ง่าย ทั้งในรูปแบบของคลิปบอร์ดหรือแชร์ออนไลน์ ซึ่งช่วยให้การพัฒนาโครงการ IoT เป็นเรื่องที่สามารถทำได้อย่างรวดเร็วและมีประสิทธิภาพ

Node-RED ยังเป็นโอเพนซอร์สที่พัฒนาและแบ่งปันใน GitHub ซึ่งสามารถดาวน์โหลดหรือแชร์โค้ดในรูปแบบ JSON ได้ ทำให้สามารถนำโค้ดที่พัฒนาขึ้นไปใช้ต่อในโครงการอื่นๆ ได้อย่างสะดวกและง่ายดาย



รูปที่ 2.15 สัญลักษณ์ของ Node-Red



รูปที่ 2.16 โปรแกรม Node-Red

2.3.5.2 Modbus Poll

Modbus Poll เป็นซอฟต์แวร์ที่ใช้สำหรับติดต่อสื่อสารระหว่างคอมพิวเตอร์ (PC) กับอุปกรณ์ที่ใช้การสื่อสารผ่าน Modbus RTU โดยเชื่อมต่อผ่านพอร์ต RS485 ซึ่งเป็นโปรแกรมจำลองการทำงานของ Modbus master (Modbus master simulator) ที่ออกแบบมาเพื่อช่วยนักพัฒนาในการทดสอบและจำลองโปรโตคอล Modbus โดยไม่จำเป็นต้องเขียนโปรแกรมเอง

โปรแกรมนี้มาพร้อมกับอินเทอร์เฟซที่ใช้งานง่ายและหลากหลายฟังก์ชัน สามารถตรวจสอบสถานะของ Modbus slaves และพื้นที่ข้อมูลหลายๆ ตัวได้พร้อมกัน โดยผู้ใช้สามารถกำหนดค่าได้อย่างง่ายดาย เช่น Modbus slave ID, function code, address, size และ poll rate สำหรับแต่ละหน้าต่าง เมื่อระบุค่าต่างๆ ได้แล้ว สามารถอ่านและเขียนรีจิสเตอร์ (register) และคอยส์ (coil) จากหน้าต่างใดก็ได้

หากต้องการเปลี่ยนค่าในรีจิสเตอร์เพียงแค่ดับเบิลคลิกที่ค่าตัวเลข หรือสามารถเปลี่ยนค่าในรีจิสเตอร์หรือคอยส์หลายๆ ตัวได้ในครั้งเดียว อีกทั้งโปรแกรมยังรองรับการแสดงข้อมูลในหลายรูปแบบ เช่น float, double และ long รวมถึงการสลับลำดับคำ (byte order) ซึ่งทำให้สะดวกในการทดสอบและทำงานกับโปรโตคอล Modbus โดยไม่ต้องเขียนโปรแกรมจากศูนย์

ด้วยฟังก์ชันที่หลากหลาย Modbus Poll จึงเป็นเครื่องมือที่สะดวกและมีประโยชน์สำหรับการทดสอบและพัฒนาอุปกรณ์ที่ใช้ Modbus protocol ได้อย่างมีประสิทธิภาพ

2.3.5.3 WinSCP

WinSCP คือโปรแกรมที่ใช้สำหรับการถ่ายโอนไฟล์ระหว่างคอมพิวเตอร์ของผู้ใช้และเซิร์ฟเวอร์ โดยรองรับหลายโปรโตคอลการสื่อสาร เช่น FTP, SFTP, SCP และ WebDAV ช่วยให้การอัพโหลดและดาวน์โหลดไฟล์จากคอมพิวเตอร์ไปยังเซิร์ฟเวอร์หรือจากเซิร์ฟเวอร์มายังคอมพิวเตอร์ทำได้อย่างสะดวกและรวดเร็ว

โปรแกรมนี้มีอินเทอร์เฟซที่ใช้งานง่าย ซึ่งสามารถใช้งานทั้งในโหมดกราฟิกและโหมดบรรทัดคำสั่ง (Command Line) ผู้ใช้สามารถลากและวางไฟล์ได้โดยตรงหรือใช้คำสั่งในการจัดการไฟล์บนเซิร์ฟเวอร์ได้อย่างมีประสิทธิภาพ WinSCP เป็นเครื่องมือที่ได้รับความนิยมในหมู่นักพัฒนาและผู้ดูแลระบบสำหรับการจัดการไฟล์ระยะไกล โดยไม่ต้องใช้เครื่องมือหรือวิธีการที่ซับซ้อน

2.3.5.4 PuTTY

Putty เป็นโปรแกรมนี้มักถูกใช้ในการเชื่อมต่อกับเซิร์ฟเวอร์ที่ใช้ระบบปฏิบัติการ Linux หรือ Unix โดยสามารถส่งคำสั่งเพื่อควบคุมหรือจัดการเซิร์ฟเวอร์ผ่านทางอินเทอร์เฟซที่เรียบง่าย เช่น การเชื่อมต่อผ่าน SSH เพื่อรักษาความปลอดภัยในระหว่างการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ของผู้ใช้และเซิร์ฟเวอร์ อีกทั้ง PuTTY ยังรองรับการเชื่อมต่อที่หลากหลายรูปแบบ เช่น

- Raw: การเชื่อมต่อโดยตรงกับพอร์ตที่กำหนด
- Telnet: การเชื่อมต่อแบบเก่าใช้ในเครือข่าย
- Rlogin: การเชื่อมต่อแบบที่ใช้ในเซิร์ฟเวอร์ Unix
- SSH: การเชื่อมต่อที่ปลอดภัยที่สุดในการใช้งานกับ Linux/Unix
- Serial: เชื่อมต่อผ่านพอร์ตอนุกรม

PuTTY เป็นเครื่องมือที่ได้รับความนิยมในวงการ IT และเป็นเครื่องมือสำคัญสำหรับการดูแลและควบคุมเซิร์ฟเวอร์ระยะไกลจากเครื่องคอมพิวเตอร์ของผู้ใช้

2.4 การเรียนรู้ของเครื่อง (Machine Learning)

Machine Learning (ML) [40-43] คือการทำให้คอมพิวเตอร์สามารถเรียนรู้และพัฒนาการทำงานได้ด้วยตัวเองจากข้อมูลและสภาพแวดล้อมที่ได้รับ โดยไม่จำเป็นต้องมีมนุษย์คอยกำกับหรือเขียนโปรแกรมใหม่ทุกครั้งเมื่อมีข้อมูลใหม่เกิดขึ้น ซึ่งหมายความว่าเมื่อระบบได้เรียนรู้จากข้อมูลในอดีตแล้ว ก็สามารถทำงานได้โดยอัตโนมัติในอนาคต แม้ว่าจะมีข้อมูลรูปแบบใหม่เข้ามา ระบบยังสามารถตีความและตอบสนองได้โดยไม่ต้องเขียนโปรแกรมเพิ่มเติม นี่คือเหตุผลที่ทำให้ Machine Learning กลายเป็นเครื่องมือที่สำคัญในการปรับปรุงประสิทธิภาพของการทำงานในหลายๆ อุตสาหกรรม เช่น การคาดการณ์พฤติกรรมลูกค้า การตรวจจับการฉ้อโกง หรือแม้แต่การพัฒนาผลิตภัณฑ์ใหม่ ๆ ด้วยการใช้ Machine Learning ธุรกิจสามารถลดต้นทุนการดำเนินงานได้มาก เนื่องจากไม่ต้องพึ่งพาแรงงานมนุษย์ในการวิเคราะห์ข้อมูลจำนวนมหาศาลหรือการทำงานที่ซ้ำซาก อีกทั้งยังสามารถลดเวลาในการตัดสินใจด้วยข้อมูลที่แม่นยำและทันเวลา ซึ่งจะช่วยเพิ่มขีดความสามารถในการแข่งขันในตลาดได้เป็นอย่างด

การเรียนรู้ใน Machine Learning สามารถเปรียบเทียบกับการเรียนรู้ของมนุษย์ที่จำเป็นต้องเรียนรู้จากประสบการณ์ ในลักษณะเช่นเดียวกับการสอนเด็กคนหนึ่งให้แยกแยะความแตกต่างระหว่างดินสอ และ ปากกา โดยเริ่มต้นจากการแนะนำคุณลักษณะของทั้งสองสิ่งเพื่อให้เด็กสามารถแยกแยะได้อย่างถูกต้อง ในทำนองเดียวกัน Machine Learning จะใช้การป้อนข้อมูลพื้นฐานและคำสั่งต่าง ๆ ให้กับคอมพิวเตอร์เพื่อให้มันสามารถเรียนรู้และพัฒนาความสามารถในการแยกแยะหรือวิเคราะห์ข้อมูลที่ได้รับมา

ใน Machine Learning จะมีการแบ่งประเภทการเรียนรู้เป็น 3 รูปแบบหลัก ๆ ตามลักษณะและวิธีการเรียนรู้ของระบบ ได้แก่ Supervised Learning, Unsupervised Learning และ Reinforcement Learning ซึ่งแต่ละประเภทจะใช้เทคนิคและวิธีการที่แตกต่างกันในการประมวลผลข้อมูลและการเรียนรู้

2.4.1 รูปแบบการเรียนรู้ของเครื่อง

2.4.1.1 Supervised Learning (การเรียนรู้แบบมีผู้สอน)

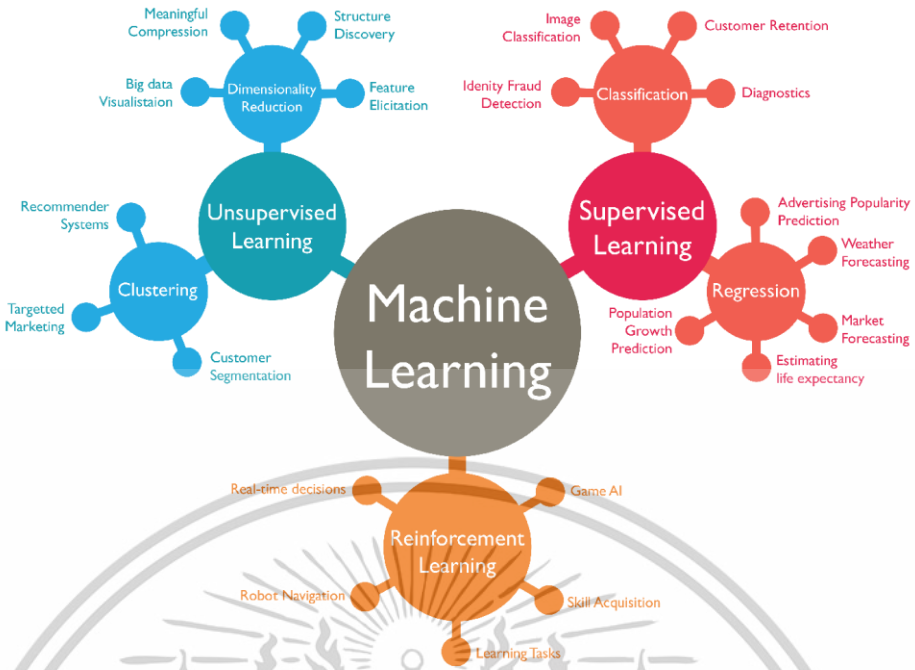
เป็นรูปแบบที่คอมพิวเตอร์เรียนรู้จากชุดข้อมูลตัวอย่างที่มนุษย์ได้กำหนดคำตอบหรือค่าที่ต้องการไว้ล่วงหน้า ตัวอย่างเช่น การป้อนข้อมูลรูปภาพของ ปากกา ให้คอมพิวเตอร์ เมื่อเราให้คอมพิวเตอร์เห็นข้อมูลที่มีคำตอบ (label) แล้ว คอมพิวเตอร์จะสามารถเรียนรู้ที่จะวิเคราะห์ข้อมูลนั้นและหาคำตอบที่ถูกต้องได้ในอนาคต เช่น หลังจากคอมพิวเตอร์ได้รับข้อมูลว่า ปากกา คืออะไร ก็สามารถแยกรูปปากกาออกจากวัตถุอื่น ๆ ได้ เช่น ดินสอ หรือ ดินสอสี โดยการใช้กระบวนการ Classification ในการจัดหมวดหมู่ข้อมูล เมื่อคอมพิวเตอร์ได้รับข้อมูลใหม่ในอนาคตมันก็สามารถตัดสินใจได้ทันทีว่าเป็นปากกา หรือไม่ใช่ปากกา

2.4.1.2 Unsupervised Learning (การเรียนรู้แบบไม่มีผู้สอน)

ในรูปแบบนี้ คอมพิวเตอร์จะไม่ได้รับคำตอบที่ต้องการจากมนุษย์ แต่จะเรียนรู้จากข้อมูลที่ได้รับด้วยตัวเอง โดยไม่มีกำหนดผลลัพธ์ล่วงหน้า เช่น ถ้าเราป้อนข้อมูลภาพของ ปากกา ให้คอมพิวเตอร์ในลักษณะที่ไม่บอกว่ามันคืออะไร คอมพิวเตอร์จะทำการวิเคราะห์ข้อมูลเหล่านั้นโดยใช้กระบวนการ Clustering เพื่อหาความสัมพันธ์ระหว่างข้อมูลต่าง ๆ และแบ่งกลุ่มข้อมูลที่มีคุณลักษณะคล้ายคลึงกัน เช่น คอมพิวเตอร์อาจจัดกลุ่มรูปภาพที่มีปลายด้ามและใช้หมึกในการเขียนเข้าด้วยกัน เช่น ปากกา และ ปากกาไฮไลท์ ที่มีคุณสมบัติคล้ายกัน โดยไม่มีการบอกล่วงหน้าว่าภาพที่ใส่เข้าไปคือ ปากกา หรือไม่

2.4.1.3 Reinforcement Learning (การเรียนรู้แบบเสริมกำลัง)

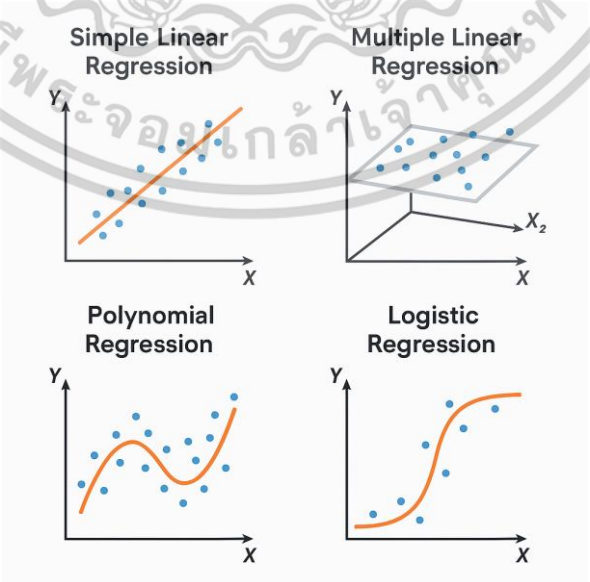
การเรียนรู้ในรูปแบบนี้จะเกิดขึ้นจากการปฏิสัมพันธ์ระหว่าง Agent (ตัวเรียนรู้) กับ Environment (สภาพแวดล้อม) โดยที่ Agent จะทำการเลือกกระทำสิ่งต่าง ๆ ในสภาพแวดล้อมและได้รับ Feedback (ผลตอบกลับ) จากการกระทำนั้น ๆ ซึ่งจะช่วยให้ Agent เรียนรู้วิธีการที่จะทำให้ได้ผลลัพธ์ที่ดีที่สุด โดยผ่านการลองผิดลองถูก เช่น ในกรณีของ AlphaGo ที่จะเรียนรู้การเล่นหมากล้อม โดยการทดลองเลือกหมากในการเล่นในแต่ละรอบ เพื่อควบคุมพื้นที่บนกระดานให้ได้มากที่สุด ในที่สุด AlphaGo ก็สามารถเรียนรู้กลยุทธ์ต่าง ๆ จากการปฏิสัมพันธ์กับเกมและคู่ต่อสู้ จนทำให้มันกลายเป็นผู้เล่นหมากล้อมที่เก่งที่สุดในโลก



รูปที่ 2.17 ภาพรวมของ Machine Learning
(Vithan 2561 : https://medium.com/investic/machine-learning-คืออะไร-fa8bf6663c07)

2.4.2 การถดถอย (Regression)

การถดถอย [44-45] เป็นวิธีการทางสถิติที่ใช้ในการศึกษาความสัมพันธ์ระหว่างตัวแปรตั้งแต่สองตัวขึ้นไป โดยแยกออกเป็น ตัวแปรอิสระ (Independent Variable) และ ตัวแปรตาม (Dependent Variable) ซึ่งตัวแปรอิสระมักเรียกว่า “ตัวแปรพยากรณ์ (Predictor Variable)” ในขณะที่ตัวแปรตามมักเรียกว่า “ตัวแปรตอบสนอง (Response Variable)” โดยมีจุดประสงค์เพื่อการตรวจสอบความสัมพันธ์ระหว่างตัวแปร และสามารถใช้เพื่อทำนายค่าของตัวแปรตามจากค่าของตัวแปรอิสระได้



รูปที่ 2.18 การวิเคราะห์การถดถอยแบบต่างๆ

ประเภทของการถดถอยมีหลายรูปแบบ ขึ้นอยู่กับลักษณะของข้อมูลและวัตถุประสงค์ในการวิเคราะห์ โดยสามารถแบ่งประเภทหลัก ๆ ได้ดังนี้

2.4.2.1 การถดถอยเชิงเส้นอย่างง่าย (Simple Linear Regression)

การถดถอยเชิงเส้นอย่างง่ายเป็นรูปแบบพื้นฐานที่สุดของการถดถอย ซึ่งใช้ในการศึกษาความสัมพันธ์ระหว่าง ตัวแปรอิสระเพียงหนึ่งตัว กับ ตัวแปรตามหนึ่งตัว โดยสมมุติว่าความสัมพันธ์ระหว่างตัวแปรทั้งสองเป็นเส้นตรง โมเดลที่ใช้ในการถดถอยเชิงเส้นอย่างง่ายสามารถเขียนได้ในรูปของสมการ

$$Y = \beta_0 + \beta_1 X + \varepsilon \quad (1)$$

โดยที่ Y คือตัวแปรตาม, X คือตัวแปรอิสระ, β_0 คือค่าคงที่ (intercept), β_1 คือสัมประสิทธิ์ (slope) ซึ่งบ่งชี้ถึงอัตราการเปลี่ยนแปลงของ Y เมื่อ X เพิ่มขึ้น และ ε คือค่าความคลาดเคลื่อน ในเบื้องต้นเราสามารถตรวจสอบความสัมพันธ์ระหว่าง Y กับ X ว่ามีความสัมพันธ์เชิงเส้นตรงหรือไม่ โดยการนำค่าของ Y และ X มาพล็อตเป็นจุดในแผนภาพ ซึ่งเรียกว่าแผนภาพการกระจาย (Scatter Diagram) หากพบว่าความสัมพันธ์ระหว่าง Y และ X มีลักษณะเป็นเส้นตรง ก็สามารถประมาณตัวแบบโดยการใช้ สมการการถดถอย เพื่อหาความสัมพันธ์ที่แน่นอนระหว่างตัวแปรทั้งสองได้

การถดถอยเชิงเส้นอย่างง่ายมักใช้ในกรณีที่ต้องการทำนายค่าของตัวแปรตามจากตัวแปรอิสระ เช่น การศึกษาความสัมพันธ์ระหว่างชั่วโมงการเรียนกับคะแนนสอบ

2.4.2.2 การถดถอยเชิงเส้นพหุคูณ (Multiple Linear Regression)

การถดถอยเชิงเส้นพหุคูณเป็นการขยายจากการถดถอยเชิงเส้นอย่างง่าย ที่ใช้ตัวแปรอิสระมากกว่าหนึ่งตัว การใช้หลายตัวแปรอิสระช่วยให้สามารถอธิบายความสัมพันธ์ระหว่างตัวแปรได้ดียิ่งขึ้น

การถดถอยเชิงเส้นพหุคูณช่วยให้สามารถศึกษาอิทธิพลของหลายปัจจัยที่มีผลต่อผลลัพธ์เดียวกัน ตัวอย่างเช่น การพยากรณ์ราคาบ้านที่ขึ้นอยู่กับหลายปัจจัย เช่น ขนาดบ้าน, จำนวนห้องนอน, และทำเลที่ตั้ง

2.4.2.3 การถดถอยพหุนาม (Polynomial Regression)

การถดถอยพหุนามเหมาะสำหรับกรณีที่ความสัมพันธ์ระหว่างตัวแปรอิสระและตัวแปรตาม ไม่เป็นเส้นตรง โดยการใช้สมการพหุนามเพื่อจับลักษณะความโค้งของข้อมูล

การถดถอยพหุนามสามารถใช้เพื่อศึกษาข้อมูลที่มีลักษณะโค้ง เช่น การศึกษาการเติบโตของพืชตามระยะเวลา หรือการศึกษาพฤติกรรมที่ไม่เป็นเส้นตรงในระบบต่าง ๆ

2.4.2.4 การถดถอยโลจิสติก (Logistic Regression)

การถดถอยโลจิสติกใช้สำหรับกรณีที่ตัวแปรตามเป็นข้อมูลประเภท จำแนก (Categorical Data) หรือ ไบนารี (Binary) เช่น การทำนายว่าเหตุการณ์จะเกิดขึ้นหรือไม่ โดยค่าผลลัพธ์จะอยู่ในช่วง 0 ถึง 1 ซึ่งแสดงถึงความน่าจะเป็นที่จะเกิดเหตุการณ์นั้น ๆ

การถดถอยโลจิสติกมักใช้ในงานที่ต้องการทำนายประเภทหรือสถานะ เช่น การทำนายว่าลูกค้าจะซื้อสินค้าหรือไม่

2.4.2.5 การถดถอยแบบไม่เชิงเส้น (Nonlinear Regression)

เมื่อความสัมพันธ์ระหว่างตัวแปรอิสระและตัวแปรตามไม่สามารถอธิบายได้ด้วยสมการเชิงเส้นหรือพหุนาม การถดถอยแบบไม่เชิงเส้นจะถูกนำมาใช้ในการพยากรณ์ที่ซับซ้อนกว่า โดยมักใช้ในกรณีที่ข้อมูลแสดงพฤติกรรมที่ซับซ้อนเช่น การเติบโตแบบ exponential หรือ logistic growth

การถดถอยแบบไม่เชิงเส้นสามารถใช้ได้ในหลากหลายกรณี เช่น การศึกษาการเติบโตของประชากรในลักษณะของการเติบโตที่ไม่เป็นเชิงเส้น

2.4.2.6 การถดถอยแบบ Ridge, Lasso, และ Elastic Net

การถดถอยแบบ Ridge, Lasso และ Elastic Net เป็นเทคนิคที่ใช้เมื่อมีปัญหา multicollinearity หรือเมื่อมีตัวแปรจำนวนมากที่อาจทำให้เกิดปัญหากับการคำนวณค่าพารามิเตอร์ การใช้เทคนิคเหล่านี้จะช่วยในการลดความซับซ้อนของโมเดล และเพิ่มความเสถียรในการประมาณค่าของตัวแปรโดยการเพิ่ม ค่าลงโทษ (penalty) ต่อค่าพารามิเตอร์ที่มีขนาดใหญ่

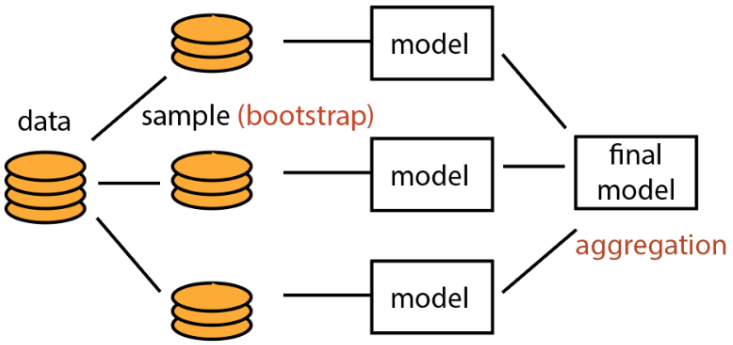
- Ridge Regression: ใช้การเพิ่ม penalty term ที่ขึ้นอยู่กับค่ากำลังสองของพารามิเตอร์
- Lasso Regression: ใช้การเพิ่ม penalty term ที่ขึ้นอยู่กับค่าค่าสัมบูรณ์ของพารามิเตอร์
- Elastic Net: เป็นการผสมผสานระหว่าง Ridge และ Lasso

2.4.3 Ensemble Learning

Ensemble Learning [46-47] เป็นการนำแบบจำลองมาเรียนรู้หลาย ๆ ครั้ง เพื่อเพิ่มประสิทธิภาพของแบบจำลอง ซึ่งเทคนิคที่ใช้ กันบ่อยได้แก่ Bagging และ Boosting มีรายละเอียดดังนี้

2.4.3.1 Bagging (Bootstrap Aggregating)

Bagging (Bootstrap Aggregation) เป็นเทคนิคใน Ensemble Learning ที่มีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพของโมเดลโดยการสร้างโมเดลหลายตัวจากชุดข้อมูลเดียวกัน ด้วยการใช้การสุ่มตัวอย่างข้อมูลจากชุดข้อมูลฝึกสอน (training data) เพื่อสร้างชุดข้อมูลย่อยหลายชุด โดยวิธีการสุ่มที่ใช้ใน Bagging คือการสุ่ม แบบแทนที่ (Random with Replacement) ซึ่งหมายความว่าในระหว่างการสุ่มแต่ละรอบ ข้อมูลอาจถูกเลือกซ้ำได้ และชุดข้อมูลที่ได้จะมีขนาดเท่ากับชุดข้อมูลเดิม แต่ไม่จำเป็นต้องมีข้อมูลที่เหมือนกันในทุกๆ ชุด แสดงดังรูปที่ 2.18



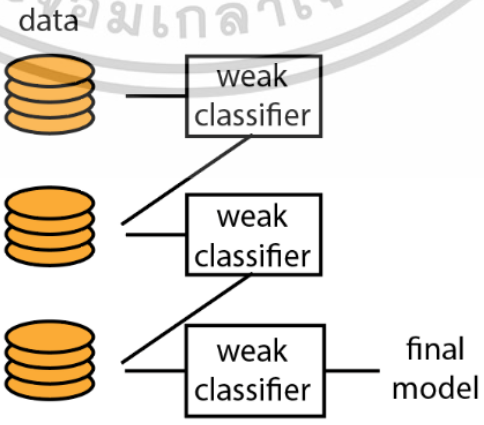
รูปที่ 2.19 การทำงานแบบ Bagging
(Titipata 2561 : https://tupleblog.github.io/bagging-boosting)

การใช้ Bagging มีข้อดีหลักคือช่วยลดความแปรปรวน (variance) ของโมเดล เนื่องจากการสุ่มเลือกข้อมูลและการฝึกหลายโมเดลทำให้โมเดลไม่พึ่งพาข้อมูลชุดใดชุดหนึ่งมากเกินไป ซึ่งช่วยลดความเสี่ยงของการ overfitting (การเรียนรู้ข้อมูลฝึกมากเกินไปจนไม่สามารถทำนายข้อมูลใหม่ได้ดี) โดยเฉพาะในกรณีที่ข้อมูลมีความผันผวนหรือมี noise

การทำนายโดยการใช้หลายโมเดลในกระบวนการนี้ทำให้ผลลัพธ์ที่ได้มีความเสถียรและแม่นยำยิ่งขึ้น เนื่องจากการรวมผลการทำนายจากหลายๆ โมเดลทำให้ข้อผิดพลาดที่เกิดจากโมเดลเดี่ยวถูกลดลง

2.4.3.2 Boosting

Boosting เป็นเทคนิคใน Ensemble Learning ที่มีวัตถุประสงค์ในการพัฒนาโมเดลที่มีประสิทธิภาพสูงขึ้น โดยการนำ แบบจำลองที่มีความแม่นยำต่ำ (weak learners) มาทำนายข้อมูลในขั้นตอนแรก จากนั้นจะใช้ผลลัพธ์ที่ได้จากการทำนายเหล่านั้นเพื่อแก้ไขข้อผิดพลาด (errors) ที่เกิดขึ้นในแต่ละรอบของการฝึก การทำเช่นนี้จะช่วยให้โมเดลสามารถปรับปรุงตัวเองได้อย่างต่อเนื่องจนกว่าจะได้ แบบจำลองที่มีความแม่นยำสูงสุด โดยกระบวนการนี้จะดำเนินไปจนกว่าผลลัพธ์จากแบบจำลองที่สร้างขึ้นจะไม่มีข้อผิดพลาดหรือค่าคลาดเคลื่อน (error) ที่สามารถแก้ไขได้อีก แสดงดังรูปที่ 2.19



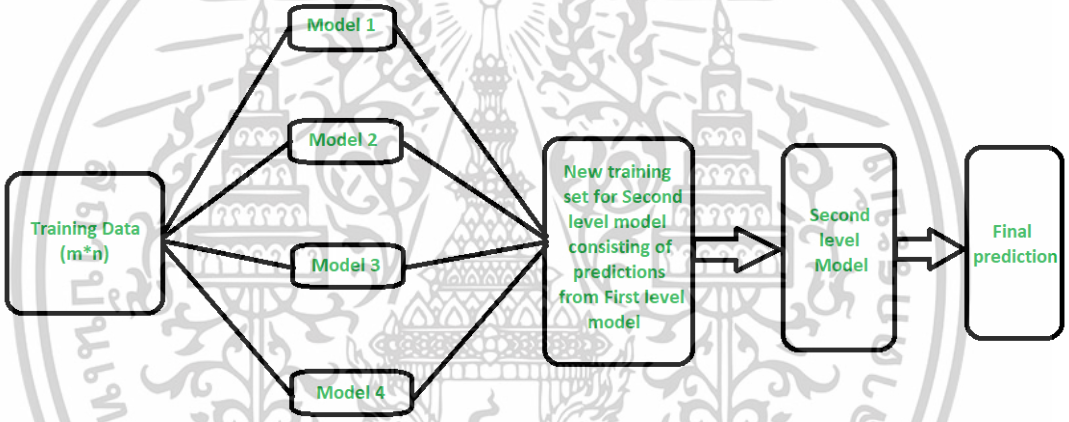
รูปที่ 2.20 การทำงานแบบ Boosting
(Titipata 2561 : https://tupleblog.github.io/bagging-boosting)

การใช้ Boosting มีข้อดีหลักคือ การเพิ่มความแม่นยำของโมเดลโดยการรวมผลจากหลายโมเดลที่มีความแม่นยำต่ำให้กลายเป็นโมเดลที่มีความแม่นยำสูงขึ้น กระบวนการฝึกซ้ำ ๆ ช่วยลดค่า bias และปรับปรุงโมเดลอย่างต่อเนื่องโดยเรียนรู้จากข้อผิดพลาดในแต่ละรอบ

นอกจากนี้ Boosting ยังสามารถจัดการกับข้อมูลที่มี noise หรือซับซ้อนได้ดี และสามารถใช้ weak learners ที่มีความแม่นยำต่ำในการเริ่มต้นเพื่อสร้างโมเดลที่มีความแม่นยำสูงได้อย่างมีประสิทธิภาพ นอกจากนี้ยังยืดหยุ่นในการใช้งานทั้งในปัญหาการจำแนกประเภทและการทำนายค่าต่อเนื่อง

2.4.3.3 Stacking (Stacked Generalization)

Stacking หรือ Stacked Generalization เป็นเทคนิคใน Ensemble Learning ที่ใช้การรวมหลายๆ โมเดลที่ต่างกันเพื่อเพิ่มความแม่นยำ โดยการใช้โมเดลต่างๆ หลายตัว (ที่เรียกว่า Base Models) มาทำนายข้อมูล แล้วเอาผลลัพธ์จากโมเดลเหล่านั้นมาฝึกอีกโมเดลหนึ่ง (เรียกว่า Meta-Model) เพื่อทำการทำนายผลสุดท้าย



รูปที่ 2.21 การทำงานแบบ Stacking
(Geeksforgeeks 2562 : https://www.geeksforgeeks.org/stacking-in-machine-learning)

กระบวนการของ Stacking ช่วยให้โมเดลสุดท้ายมีความแม่นยำสูงกว่าโมเดลเดี่ยว เนื่องจากมันสามารถรวมความแข็งแกร่งจากหลาย ๆ โมเดลที่อาจมีจุดเด่นหรือข้อดีที่แตกต่างกันออกไปในแต่ละแง่มุมของข้อมูล

ในการวิจัยฉบับนี้ การเรียนรู้ของเครื่องจะถูกนำไปใช้ในรูปแบบ การเรียนรู้แบบมีผู้สอน (Supervised Learning) โดยมีจุดประสงค์หลักเพื่อประยุกต์ใช้กับงานประเภท Regression เพื่อทำนายคุณภาพของน้ำ โดยการเรียนรู้จากข้อมูลที่มีการระบุคำตอบหรือค่าที่ต้องการล่วงหน้า ซึ่งจะช่วยให้ระบบสามารถทำนายค่าความเค็มของน้ำได้จากคุณสมบัติต่างๆ ที่ได้รับการป้อนเข้าไปในระบบ เช่น ค่าของอุณหภูมิ, ความขุ่น, หรือพารามิเตอร์อื่นๆ ที่เกี่ยวข้อง การใช้ Supervised Learning ในลักษณะนี้จะทำให้ระบบสามารถเรียนรู้จากข้อมูลตัวอย่างเพื่อสร้างแบบจำลองที่สามารถทำนายผลลัพธ์ที่ถูกต้องในอนาคต ในกระบวนการนี้ โมเดล Extreme Gradient Boosting (XGBoost) จะถูกนำมาใช้เป็นเครื่องมือหลักในการพัฒนาแบบจำลอง โดยโมเดล XGBoost เป็นเทคนิคการเรียนรู้ที่มีความสามารถสูงในการจัดการกับข้อมูลที่มีความซับซ้อนและหลากหลาย โดยการทำงานของ

XGBoost จะเน้นการเพิ่มประสิทธิภาพของแบบจำลองผ่านกระบวนการ "boosting" ที่ช่วยลดข้อผิดพลาดในแต่ละรอบของการฝึกฝน ทำให้สามารถคาดการณ์ผลลัพธ์ได้อย่างมีประสิทธิภาพและแม่นยำมากยิ่งขึ้น

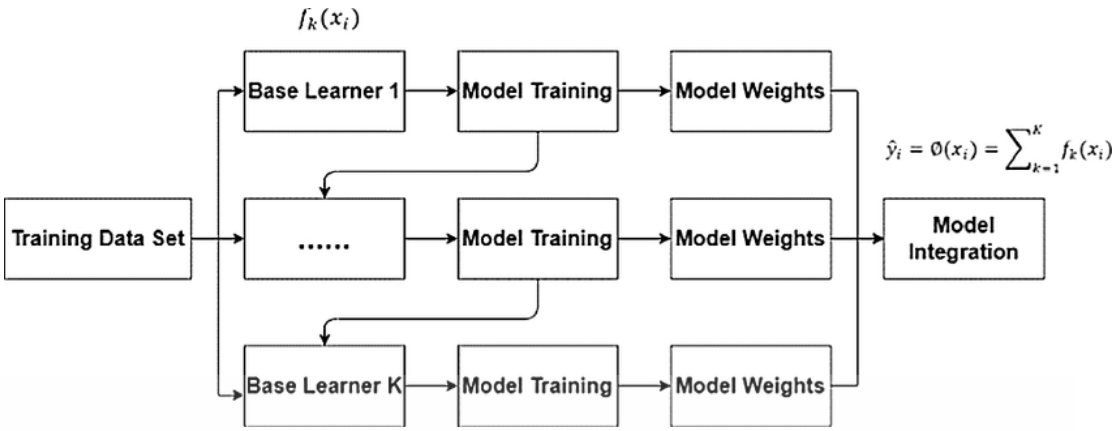
2.4.4 Extreme Gradient Boosting (XGBoost)

XGBoost หรือ Extreme Gradient Boosting เป็นหนึ่งในอัลกอริธึมที่ได้รับความนิยมสูงสุดในการเรียนรู้ของเครื่อง (Machine Learning) โดยเฉพาะอย่างยิ่งในหมู่นักวิเคราะห์ข้อมูล นักวิจัย และผู้เข้าแข่งขันแพลตฟอร์มวิเคราะห์ข้อมูลเช่น Kaggle หรือ Zindi อัลกอริธึมนี้ถูกพัฒนาโดย Tianqi Chen โดยมีวัตถุประสงค์หลักเพื่อพัฒนาประสิทธิภาพของเทคนิค Gradient Boosting ให้รวดเร็วขึ้น ใช้หน่วยความจำน้อยลง และสามารถจัดการกับข้อมูลที่มีความซับซ้อนได้ดียิ่งขึ้น

Gradient Boosting ซึ่งเป็นพื้นฐานของ XGBoost นั้นเป็นเทคนิคการรวมแบบจำลอง (Ensemble Learning) ที่ใช้การสร้างโมเดลอย่างต่อเนื่อง โดยแต่ละโมเดลใหม่จะพยายามลดข้อผิดพลาด (Residual) ที่เกิดจากโมเดลก่อนหน้า กลไกสำคัญของมันคือการใช้ความชันของฟังก์ชันค่าความผิดพลาด (Gradient of Loss Function) ในการชี้นำการเรียนรู้ของแต่ละต้นไม้ การตัดสินใจ (Decision Tree) ที่ถูกสร้างขึ้นอย่างต่อเนื่องเพื่อให้ผลลัพธ์สุดท้ายแม่นยำมากที่สุด

XGBoost เป็นการพัฒนาและปรับปรุงจากเทคนิค Gradient Boosting โดยมีประสิทธิภาพสูงและสามารถปรับขนาดการทำงานได้ดี ทั้งในด้านความเร็วและความแม่นยำในการทำนาย และสร้างโมเดลโดยการรวมชุดของต้นไม้ตัดสินใจ (decision trees) หลายต้นเข้าด้วยกันในลักษณะเป็นขั้นตอน (iterative) โดยในแต่ละขั้นจะพยายามแก้ไขข้อผิดพลาดที่เกิดจากโมเดลก่อนหน้า การ Boosting เป็นกระบวนการเรียนรู้แบบบวกสะสม (additive learning process) ซึ่งในแต่ละรอบจะมีการเพิ่มตัวเรียนรู้แบบ weak learner เข้ามา เพื่อช่วยปรับปรุงผลการพยากรณ์ของโมเดลให้ดีขึ้นเรื่อยๆ

นอกจากนี้ XGBoost ยังมีคุณสมบัติเด่นคือสามารถประมวลผลข้อมูลจำนวนมากได้อย่างรวดเร็วด้วยเทคนิคการประมวลผลแบบ Parallel Processing และการจัดการหน่วยความจำอย่างมีประสิทธิภาพ นอกจากนี้ยังสามารถจัดการกับค่าที่ขาดหายไป (Missing Values) ได้โดยอัตโนมัติ และมีเทคนิค Regularization ที่ช่วยลดความเสี่ยงของการ overfitting อีกทั้งสามารถทำงานร่วมกับเครื่องมือวิเคราะห์ข้อมูลต่าง ๆ ได้อย่างมีประสิทธิภาพ เช่น Pandas, NumPy, Scikit-learn และ SHAP ซึ่งช่วยในการจัดการข้อมูลและอธิบายผลลัพธ์ของโมเดลได้อย่างชัดเจน เหมาะสำหรับอุตสาหกรรมที่ต้องการความโปร่งใส เช่น การแพทย์และการเงิน รองรับหลายภาษาการเขียนโปรแกรม และสามารถใช้งานบน GPU ได้ จึงเป็นเครื่องมือที่ใช้งานง่าย ยืดหยุ่น และสามารถปรับแต่งได้ลึกซึ้ง เหมาะกับการใช้งานจริงมากกว่าโมเดลเชิงลึกที่ซับซ้อนและตีความยาก



รูปที่ 2.22 หลักการทำงานของ XGBoost

สมการของ XGBoost นั้นเกี่ยวข้องกับการสร้างโมเดลโดยการใช้ชุดของต้นไม้ตัดสินใจ (Decision Trees) ซึ่งจะพยายามลดข้อผิดพลาดจากต้นไม้ที่สร้างมาก่อนหน้าในแต่ละรอบผ่านการปรับปรุงที่มีลักษณะเป็นขั้นตอน (iterative process) โดยมีการเพิ่ม "weak learners" หรือ ตัวเรียนรู้ที่มีประสิทธิภาพต่ำเข้ามาช่วยในการปรับปรุงการทำนายของโมเดลในแต่ละรอบ

สมมติว่าเรามีชุดข้อมูล $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ โดยที่ x_i คือข้อมูลคุณลักษณะ (input features) และ y_i คือค่าที่ต้องการทำนาย (target values) โดยผลการทำนายของโมเดลในรอบที่ t จะถูกแสดงในสมการที่ 2 ซึ่งจะกล่าวถึงวิธีการรวมผลลัพธ์ของต้นไม้ก่อนหน้าและต้นไม้ใหม่เข้าด้วยกัน โดยสามารถนิยามได้ดังนี้

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i), \quad f_k \in F \tag{2}$$

โดยที่ $f_k(x_i)$ คือต้นไม้การตัดสินใจ ต้นที่ k, F คือเซตของฟังก์ชันที่แทนต้นไม้การตัดสินใจ, $\hat{y}_i^{(t)}$ คือ ค่าพยากรณ์ของตัวอย่าง i ที่รอบการเรียนรู้ t , $f_k(x_i)$ คือ ฟังก์ชันอนุมูล (*base learner)* ลำดับที่ k ที่ใช้กับข้อมูล x_i และ F คือ เซตของฟังก์ชันอนุมูลทั้งหมด

Objective Function (ฟังก์ชันวัตถุประสงค์) คือหัวใจหลักของกระบวนการเรียนรู้ ซึ่งมีหน้าที่ ควบคุมทั้งความแม่นยำของโมเดล (ผ่าน Loss Function) และ ควบคุมความซับซ้อนของโมเดล (ผ่าน Regularization) เพื่อป้องกันไม่ให้เกิดการฟิตมากเกินไป (overfitting) XGBoost กำหนดฟังก์ชันวัตถุประสงค์ (Objective Function) ดังสมการที่ 3 โดยผสมผสานระหว่างค่าความผิดพลาด (Loss Function) กับ Regularization (การทำให้โมเดลมีความเรียบง่าย) เพื่อควบคุมความซับซ้อนของโมเดล โดยมีนิยามดังนี้

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{t} \Omega(f_k) \tag{3}$$

โดยที่ $\mathcal{L}$ คือ objective function, n คือจำนวนตัวอย่าง, $l(y_i, \hat{y}_i)$ คือค่าความสูญเสีย (loss) ระหว่างค่าจริง y_i และค่าพยากรณ์ $\hat{y}_i$ และ $\Omega(f_k)$ คือค่าความซับซ้อน (regularization term) ของฟังก์ชัน f_k

ℓ(yi,y^i) คือ ฟังก์ชันการสูญเสีย (Loss Function) ซึ่งใช้วัดความคลาดเคลื่อนระหว่างค่าจริง y_i และค่าทำนาย $\hat{y}_i$ โดย Loss Function จะทำให้โมเดลเรียนรู้ได้อย่างแม่นยำมากขึ้น เช่น ถ้าเป็นปัญหา regression (การพยากรณ์ค่า) มักใช้ Mean Squared Error (MSE) และ ถ้าเป็นปัญหา classification (จำแนกประเภท) มักใช้ Log Loss หรือ Cross-Entropy โดยที่ Regularization คือ กระบวนการที่ช่วยทำให้โมเดลไม่ซับซ้อนเกินไป จัดการกับต้นไม้ที่มีความซับซ้อน เพื่อไม่ให้โมเดลจำเฉพาะข้อมูลชุดฝึก (training data) จนไม่สามารถทำนายข้อมูลใหม่ได้ดี โดยทั้งสองส่วนจะทำงานรวมกันใน Objective Function ของ XGBoost เพื่อให้โมเดลเรียนรู้ได้อย่างแม่นยำและช่วยทำให้โมเดลไม่ซับซ้อนมากจนเกินไป โดยการจัดการแบบ Regularization จะถูกใช้เพื่อควบคุมไม่ให้โมเดลมีความซับซ้อนเกินไป ซึ่งอาจนำไปสู่การเกิด Overfitting ดังสมการที่ 4 โดยมีนิยามดังนี้

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \quad (4)$$

โดยที่ T คือ leaves หรือจำนวนใบไม้ในต้นไม้เพื่อใช้ในการตัดสินใจ f_k, γ คือพารามิเตอร์ค่าคงที่ที่ควบคุมความซับซ้อน, λ คือพารามิเตอร์กำกับความเรียบและความคงที่ของน้ำหนัก, w_j คือค่าการพยากรณ์ที่มีความสัมพันธ์กันกับใบไม้ใบที่ j

การควบคุมความซับซ้อนของโมเดล (Regularization) จะอิงกับจำนวนใบของต้นไม้ T และค่าน้ำหนักของแต่ละใบ w_j เพราะต้นไม้ที่มีจำนวนใบเยอะหรือมีน้ำหนักบางใบสูงมาก มักจะจำข้อมูลมากเกินไป (Overfitting) จึงต้องมีการจัดการความซับซ้อนนี้ผ่านการ Regularization โดย γ จะกำหนดว่า ถ้าการเพิ่มใบใหม่ไม่ช่วยลดค่าความผิดพลาด (loss) มากพอ ก็จะไม่เพิ่มใบนั้น เพื่อป้องกันการสร้างต้นไม้โดยไม่จำเป็นส่วน λ จะลงโทษน้ำหนักที่สูงเกินไป (ผ่าน L2 regularization) เพื่อให้ต้นไม้ไม่เน้นเฉพาะบางใบมากเกินไป

สำหรับการปรับปรุงโมเดล XGBoost จะใช้ Taylor expansion ลำดับที่สองของฟังก์ชัน loss สมการที่ 4 เพื่อคำนวณการเปลี่ยนแปลงค่าพารามิเตอร์อย่างแม่นยำ โดยใช้ทั้งอนุพันธ์อันดับที่ 1 (ความชัน) และอันดับที่ 2 (ความโค้ง) แทนที่จะใช้แค่อนุพันธ์อันดับเดียวแบบวิธีทั่วไป ทำให้การเรียนรู้มีประสิทธิภาพและรวดเร็วมากขึ้น แสดงดังสมการที่ 5

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i) \right] + \Omega(f_t) \quad (5)$$

โดยที่ $\mathcal{L}^{(t)}$ คือค่าความสูญเสียโดยประมาณ (approximate loss) ในรอบที่ t, g_i คือค่ากราดิเอนต์แรกของ $l(y_i, \hat{y}_i)$, h_i คือค่าฮีสเซียน (second derivative) ของ $l(y_i, \hat{y}_i)$ และ $f_i(x_i)$ คือ ฟังก์ชันอนุมูลใหม่ที่กำลังเรียนรู้

$g_i = \frac{\partial l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial \hat{y}_i^{(t-1)}}$ เป็นความชันอันดับหนึ่ง (First-order Gradient) และ

$h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)2}}$ เป็นความชันอันดับสอง (Second-order Gradient)

เมื่อสร้างต้นไม้ XGBoost จะเลือกจุดแบ่ง (Split) โดยใช้เกณฑ์การเพิ่มค่าผลประโยชน์ (Gain) จากการลดค่าฟังก์ชันวัตถุ จะมีการเลือกการแบ่งของต้นไม้ (Tree Splitting) โดยหาค่าสูงสุด

ของสมการ โดย I_L และ I_R คือเซตข้อมูลที่อยู่ฝั่งซ้ายและขวาหลังจากการแบ่ง และ I คือเซตข้อมูลก่อนการแบ่ง

$$Gain = \frac{1}{2}\left[\frac{\left(\sum_{i\epsilon I_L} g_i\right)^2}{\sum_{i\epsilon I_L} h_i+\lambda} + \frac{\left(\sum_{i\epsilon I_R} g_i\right)^2}{\sum_{i\epsilon I_R} h_i+\lambda} + \frac{(\sum_{i\epsilon I} g_i)^2}{\sum_{i\epsilon I} h_i+\lambda}\right] - \gamma \quad (6)$$

โดยที่ I_L, I_R คือเซตของตัวอย่างที่ตกลงบนใบไม้ทางฝั่งซ้ายและใบไม้ทางฝั่งขวา, I คือเซตของตัวอย่างทั้งหมดในโหนด และ $Gain$ คือค่า reduction in loss เมื่อแบ่งโหนด

การปรับค่าทำนายด้วยอัตราการเรียนรู้ (Learning Rate) หลังจากที่แต่ละต้นไม้ถูกสร้างขึ้น การทำนายจะได้รับการแก้ไขเพื่อลดความซับซ้อนข้องข้อมูล โดยใช้อัตราการเรียนรู้ η เพื่อลดความเสี่ยงในการ Overfitting อ้างอิงจาก

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i) \quad (7)$$

โดยที่ η คือlearning rate (อัตราการเรียนรู้), $\hat{y}_i^{(t-1)}$ คือค่าพยากรณ์ก่อนรอบที่ t, $f_t(x_i)$ คือการอัปเดตจากอนุมูลรอบที่ t และ $0 < \eta \leq 1$

2.4.4.1 Optuna Tuning

Optuna Tuning เป็นเฟรมเวิร์กซอฟต์แวร์ที่ออกแบบมาเพื่อการปรับจูนไฮเปอร์พารามิเตอร์ของโมเดลแบบอัตโนมัติ โดยมีเป้าหมายเพื่อค้นหาค่าของไฮเปอร์พารามิเตอร์ที่เหมาะสมที่สุดสำหรับโมเดลที่เลือกใช้งาน การดำเนินการของ Optuna แบ่งออกเป็น การศึกษา (study) ซึ่งประกอบด้วยหลายๆ การทดลอง (trial) แต่ละ trial จะประเมินชุดค่าพารามิเตอร์ผ่าน ฟังก์ชันวัตถุประสงค์ (objective function) ที่ผู้ใช้กำหนด และจุดเด่นของ Optuna คือการสนับสนุน พื้นที่การค้นหาแบบไดนามิก (dynamic search space) ซึ่งช่วยให้ผู้ใช้สามารถกำหนดขอบเขตของไฮเปอร์พารามิเตอร์แบบปรับเปลี่ยนได้ตามเงื่อนไขระหว่างการทำงาน โดยอาศัยการโต้ตอบกับอ็อบเจกต์ trial ภายในฟังก์ชันวัตถุประสงค์ ส่งผล ห้สามารถปรับแต่งกระบวนการค้นหาให้เหมาะสมกับสถานการณ์ได้มากขึ้น

ค่าผลลัพธ์ของแต่ละ trial จะถูกนำมาใช้ในกระบวนการ minimizing หรือ maximizing ขึ้นอยู่กับเป้าหมายของฟังก์ชันวัตถุประสงค์ โดยมุ่งหวังให้ได้ค่าผลลัพธ์ที่ดีที่สุดจากการป้อนค่าพารามิเตอร์ต่างๆ เข้าไป กลยุทธ์การค้นหาแบบไดนามิกของ Optuna มีความสำคัญอย่างยิ่งต่อการเลือกและประมาณค่าชุดไฮเปอร์พารามิเตอร์ที่มีประสิทธิภาพ โดยอาจใช้เทคนิคเช่นการ ตัด learning curves ที่ไม่มีแนวโน้มดี (early stopping/discarding) เพื่อเพิ่มความเร็วในการปรับจูนและลดทรัพยากรที่ใช้ อีกทั้ง Optuna ใช้เทคนิคที่เรียกว่า Tree-structured Parzen Estimator (TPE) ซึ่งเป็นหนึ่งในวิธีการของ Bayesian Optimization โดยจะเลือกจุดทดลองใหม่ตามข้อมูลจากผลลัพธ์ก่อนหน้านี้ ทำให้การค้นหาค่าที่ดีที่สุดมีประสิทธิภาพสูงกว่าการสุ่มทดลองแบบไม่มีทิศทาง

TPESampler (Tree-Structured Parzen Estimator) เป็นอัลกอริทึมการปรับแต่งไฮเปอร์พารามิเตอร์ในรูปแบบของ Bayesian Optimization ซึ่ง Optuna ใช้ในการเลือกค่าพารามิเตอร์อย่างมีประสิทธิภาพ โดยมีลำดับขั้นตอนการทำงานดังนี้:

1) เริ่มต้นโดยการสุ่มเลือกค่าพารามิเตอร์จากชุดของพารามิเตอร์ที่เป็นไปได้ และจัดเรียงค่าที่ได้ตามลำดับของประสิทธิภาพจากฟังก์ชันวัตถุประสงค์

2) แบ่งค่าพารามิเตอร์ทั้งหมดออกเป็น 2 กลุ่ม โดยอ้างอิงจากค่า quantile ที่กำหนดไว้ล่วงหน้า ได้แก่ กลุ่มที่มีค่าผลลัพธ์ดี (ค่าฟังก์ชันต่ำหรือสูงขึ้นอยู่กับเป้าหมาย) และ กลุ่มที่เหลือ

3) ใช้เทคนิค Parzen Estimators (Kernel Density Estimation) เพื่อประมาณฟังก์ชันความหนาแน่นความน่าจะเป็นของแต่ละกลุ่ม ได้แก่ l(x) สำหรับกลุ่มค่าที่ดี และ g(x) สำหรับกลุ่มค่าที่เหลือ

4) ประเมินความเป็นไปได้ของค่าพารามิเตอร์ใหม่โดยพิจารณาจากอัตราส่วนของฟังก์ชันความหนาแน่นทั้งสอง (l(x)/g(x)) และเลือกค่าที่มีอัตราส่วนสูงที่สุด (หมายถึงค่าที่มีแนวโน้มจะให้ผลลัพธ์ที่ดี)

5) ค่าพารามิเตอร์ที่ได้จะถูกนำกลับมาใช้ในการทดลองใหม่ และกระบวนการนี้จะทำซ้ำไปเรื่อย ๆ จนกว่าจะได้ชุดไฮเปอร์พารามิเตอร์ที่ดีที่สุด

2.5 การเรียนรู้เชิงลึก (Deep Learning)

การเรียนรู้เชิงลึก (Deep Learning) [48-50] คือเทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ที่มุ่งเน้นการสร้างแบบจำลองคอมพิวเตอร์ที่สามารถเรียนรู้การแทนค่าข้อมูลในระดับนามธรรม (abstract representations) ได้โดยอัตโนมัติ ผ่านโครงสร้างที่เลียนแบบการทำงานของสมองมนุษย์ ซึ่งเรียกว่า โครงข่ายประสาทเทียมหลายชั้น (Deep Neural Networks - DNNs) โดยทั่วไป โครงข่ายนี้จะประกอบด้วยชั้นของนิวรอนจำนวนมากที่เชื่อมโยงกัน โดยแต่ละชั้นจะมีหน้าที่ประมวลผลข้อมูลในรูปแบบต่าง ๆ และส่งผ่านผลลัพธ์ไปยังชั้นถัดไป

คำว่า “ลึก” (deep) ในที่นี้ หมายถึงจำนวนชั้นที่ซ้อนกันหลายชั้นของโครงข่ายประสาท ซึ่งช่วยให้ระบบสามารถเรียนรู้รูปแบบที่ซับซ้อน เช่น การตรวจจับวัตถุในภาพ การแยกเสียงพูด หรือการวิเคราะห์ข้อความในภาษาได้ดีกว่าโมเดลที่มีชั้นเดียว (shallow networks)

กระบวนการเรียนรู้ของ Deep Learning ประกอบด้วย 3 ขั้นตอนหลัก ได้แก่

1) การป้อนข้อมูล (Input): ระบบรับข้อมูลดิบ เช่น รูปภาพ เสียง หรือข้อความ
2) การแปลงข้อมูลผ่านชั้นซ่อน (Hidden Layers): ในแต่ละชั้น ระบบจะเรียนรู้คุณลักษณะ (features) ที่สำคัญ เช่น ขอบของวัตถุในภาพ หรือโครงสร้างของประโยคในข้อความ
3) การแสดงผลลัพธ์ (Output): ระบบจะสรุปผล เช่น ระบุว่าสิ่งในภาพคือแมวหรือสุนัข แปลภาษาจากอังกฤษเป็นไทย หรือทำนายค่าตัวเลขในอนาคต

เพื่อให้โมเดลเรียนรู้ได้อย่างแม่นยำ จะใช้กระบวนการ “ย้อนกลับค่าความผิดพลาด” (Backpropagation) ร่วมกับอัลกอริทึมการปรับค่าพารามิเตอร์ เช่น Gradient Descent โดยจะปรับน้ำหนักของการเชื่อมต่อระหว่างนิวรอนอย่างต่อเนื่องตามค่าความผิดพลาดที่เกิดขึ้นจากการทำนายเมื่อเทียบกับคำตอบจริง

หนึ่งในจุดแข็งของ Deep Learning คือ ความสามารถในการเรียนรู้โดยไม่ต้องพึ่งพาการกำหนดฟีเจอร์ล่วงหน้า (Feature Engineering) ซึ่งในอดีตเป็นสิ่งที่นักวิจัยต้องใช้เวลาและความรู้

เฉพาะทางในการออกแบบ แต่ Deep Learning สามารถเรียนรู้ฟีเจอร์เหล่านี้ได้เองจากข้อมูลขนาดใหญ่ โดยเฉพาะอย่างยิ่งเมื่อใช้ร่วมกับหน่วยประมวลผลที่ทรงพลัง เช่น GPU และ TPU

อย่างไรก็ตาม Deep Learning ก็มีข้อจำกัด เช่น ต้องการข้อมูลจำนวนมากในการฝึกสอน ใช้ทรัพยากรคอมพิวเตอร์สูง และอธิบายการทำงานได้ยากในบางกรณี (ลักษณะเป็น “กล่องดำ”) ซึ่งอาจเป็นอุปสรรคในงานบางประเภท เช่น การแพทย์หรือกฎหมายที่ต้องการความโปร่งใสในการตัดสินใจ

ปัจจุบัน Deep Learning ได้กลายเป็นเทคโนโลยีพื้นฐานในหลายสาขา ตั้งแต่ระบบอัจฉริยะในอุตสาหกรรม ไปจนถึงแอปพลิเคชันในชีวิตประจำวัน เช่น ระบบแนะนำของ YouTube, การรู้จำใบหน้าในโทรศัพท์มือถือ และระบบควบคุมรถยนต์ไร้คนขับ

ตัวอย่างสถาปัตยกรรมของ Deep Learning ได้แก่:

- Convolutional Neural Networks (CNNs) ใช้กับข้อมูลภาพ เช่น การจำแนกรูปภาพหรือวิเคราะห์วิดีโอ
- Recurrent Neural Networks (RNNs) และ Long Short-Term Memory (LSTM) ใช้กับข้อมูลลำดับ เช่น ข้อความหรือเสียงพูด
- Transformer เป็นโครงข่ายสมัยใหม่ที่โดดเด่นในงานด้านภาษาธรรมชาติ (NLP) เช่น โมเดล ChatGPT, BERT และ Google Translate

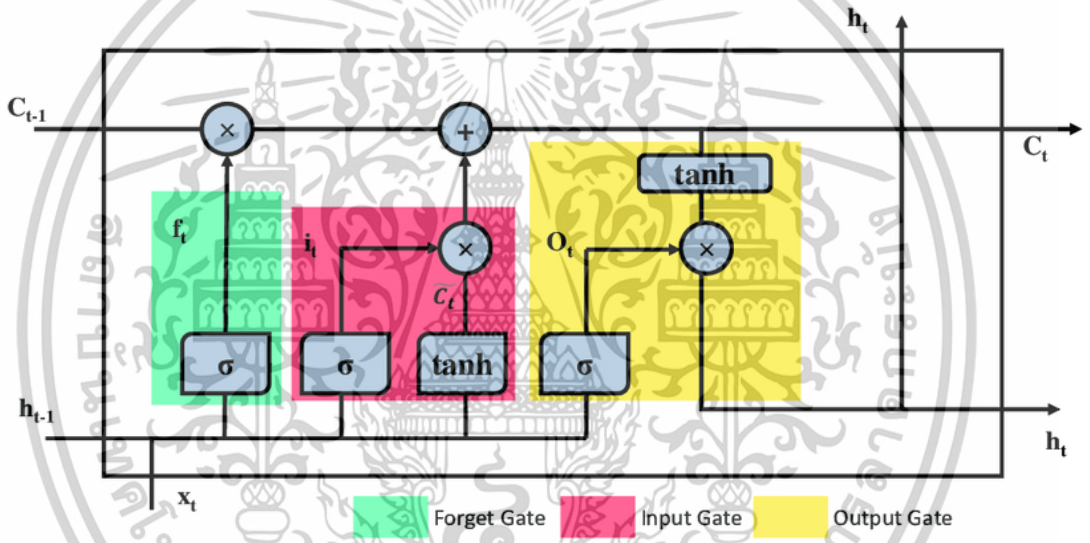
ในการวิจัยฉบับนี้ ได้มีการประยุกต์ใช้เทคนิคการเรียนรู้ของเครื่องในรูปแบบการเรียนรู้แบบมีผู้สอน (Supervised Learning) โดยเลือกใช้สถาปัตยกรรมโครงข่ายประสาทเทียมแบบ Long Short-Term Memory (LSTM) ซึ่งเป็นรูปแบบของ Recurrent Neural Network (RNN) ที่เหมาะสมอย่างยิ่งสำหรับการวิเคราะห์ข้อมูลลำดับเวลา (Time Series) จุดประสงค์หลักคือเพื่อพัฒนาแบบจำลองที่สามารถทำนายคุณภาพของน้ำในอนาคต โดยใช้ข้อมูลในอดีตเป็นปัจจัยในการคาดการณ์ เช่น ค่าความเป็นกรดเป็นด่าง อุณหภูมิ หรือค่าความการนำไฟฟ้าในช่วงเวลาที่ต่อเนื่องกัน และเนื่องจากข้อมูลคุณภาพน้ำมีลักษณะเป็นลำดับเวลา LSTM จึงสามารถเรียนรู้รูปแบบของความสัมพันธ์และแนวโน้มของข้อมูลในอดีตได้อย่างมีประสิทธิภาพ โดยแบบจำลองที่สร้างขึ้นจะสามารถนำไปใช้ทำนายค่าพารามิเตอร์ต่างๆ ของคุณภาพน้ำในช่วงเวลาถัดไป ซึ่งเป็นประโยชน์อย่างยิ่งต่อการบริหารจัดการทรัพยากรน้ำ และการเฝ้าระวังคุณภาพน้ำในระบบนิเวศหรือพื้นที่ชลประทาน

2.5.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) คือสถาปัตยกรรมของโครงข่ายประสาทเทียมแบบวนซ้ำ (Recurrent Neural Network: RNN) ที่ได้รับการพัฒนาเพื่อช่วยในการเรียนรู้ลำดับข้อมูลที่มีระยะเวลายาว โดยเฉพาะเมื่อต้องการจำข้อมูลที่ห่างไกลจากกันในลำดับ ซึ่ง RNN ปกติจะมีปัญหาในการรักษาข้อมูลในระยะยาวเนื่องจากปรากฏการณ์ Vanishing Gradient ที่ทำให้การฝึกฝนไม่สามารถทำได้ดีเมื่อข้อมูลมีระยะห่างจากกันมากๆ ในลำดับ เช่น ข้อความยาวๆ หรือชุดข้อมูลที่มีลำดับเวลาก็จะสูญเสียข้อมูลสำคัญจากช่วงต้น

LSTM แก้ปัญหานี้โดยการใช้ Cell State ซึ่งเป็นหน่วยความจำที่สามารถเก็บข้อมูลในระยะยาว และมีการควบคุมการไหลของข้อมูลที่ดีขึ้นด้วย Gating Mechanisms ที่ประกอบด้วยประตูหลัก 3 ประตู ได้แก่ Forget Gate, Input Gate, และ Output Gate ซึ่งแต่ละตัวมีหน้าที่ควบคุมการไหลของข้อมูลเข้าออกจากหน่วยความจำอย่างเป็นระบบ กระบวนการทำงานของ LSTM ในกระบวนการทำงานในแต่ละช่วงเวลา t ของ LSTM จะมีการอัปเดตข้อมูลในแต่ละ cell state ตามลำดับ โดยการควบคุมที่มีประสิทธิภาพนี้ช่วยให้ LSTM สามารถจดจำลำดับข้อมูลที่ห่างกันหลายขั้นตอนและเรียนรู้ข้อมูลในระยะยาวได้อย่างมีประสิทธิภาพมากขึ้น

LSTM มักถูกนำไปใช้ในหลายๆ งาน เช่น การประมวลผลภาษาธรรมชาติ (Natural Language Processing), การแปลภาษา, การทำนายชุดข้อมูลทางเวลาหรือ time series, การจดจำลำดับเสียง, และอื่นๆ ที่ต้องการการจัดการข้อมูลลำดับที่ยาวหรือมีการเชื่อมโยงระยะยาว เนื่องจากมันสามารถเก็บรักษาและลืมข้อมูลได้อย่างมีประสิทธิภาพ ทำให้ LSTM เหมาะสมกับการประมวลผลข้อมูลที่มีลำดับยาว



รูปที่ 2.23 หลักการทำงานของเซลล์ LSTM

สมการของ Long Short-Term Memory (LSTM) ประกอบด้วยหลายสมการที่ใช้ในการคำนวณการทำงานของแต่ละประตูใน LSTM (Forget Gate, Input Gate, Output Gate) และการอัปเดต Cell State ที่สำคัญ ซึ่งสามารถอธิบายได้ตามขั้นตอนต่างๆ ดังนี้:

1) Forget Gate

Forget Gate ทำหน้าที่ตัดข้อมูลที่ไม่จำเป็นออกจากสถานะความจำก่อนหน้า C_{t-1} โดยใช้สมการ

$$f_t = \sigma\left(W_f\left[h_{t-1,}\, x_t\right] + b_f\right) \tag{8}$$

โดยที่ f_t คือเวกเตอร์ที่ timestep t กำหนดว่าค่าจากสถานะก่อนหน้าจะถูก Forget หรือเก็บไว้เท่าใด, σ คือตัวแปลทำให้ค่าผลลัพธ์อยู่ในช่วง [0,1], W_f คือน้ำหนักของ forget gate เรียนรู้ได้ผ่านการฝึก, $\left[h_{t-1,}\, x_t\right]$ คือการ concatenation ของ hidden state จาก timestep

ก่อนหน้าที่, h_{t-1} กับ input ปัจจุบัน x_t และ b_f ของ forget gate ช่วยปรับค่าความเอียงที่เกิดขึ้นจากค่าคงที่ถูกบวกเข้ากับผลรวมเชิงเส้นของการคำนวณ

ฟังก์ชันซิกมอยด์ (Sigmoid) σ จะให้ค่าอยู่ในช่วง 0 ถึง 1 เพื่อบอกว่าเราควรลืมหรือเก็บข้อมูลจากสถานะก่อนหน้าเท่าใด โดยพิจารณาจากอินพุตปัจจุบัน x_t และ hidden state ก่อนหน้า h_{t-1}

2) Input Gate

Input Gate ควบคุมว่าเราควรเพิ่มข้อมูลใหม่เข้าไปใน cell state ปัจจุบันมากน้อยเพียงใดจาก candidate cell state หรือไม่ โดยจะคำนวณจาก hidden state ที่ผ่านมาและ input ในปัจจุบันผ่านฟังก์ชันซิกมอยด์ (Sigmoid) และ Tanh. Sigmoid ใช้ในการตัดสินใจว่าจะเก็บข้อมูลใหม่ลงใน cell state หรือไม่ และ Tanh ใช้ในการคำนวณ candidate cell state ซึ่งเป็นข้อมูลใหม่ที่ควรเก็บ

$$i_t = \sigma\left(W_i\left[h_{t-1,}\, x_t\right] + b_i\right) \tag{9}$$

โดยที่ i_t คือ input gate activation ที่ timestep t จะกำหนดว่าข้อมูลใหม่จะถูกเพิ่มเข้า cell state มากน้อยแค่ไหน, W_i คือน้ำหนักของ Input gate และ b_i คือ bias ของ input gate

3) Candidate Cell State

Candidate Cell State คือข้อมูลใหม่ที่เราต้องการเก็บลงใน cell state ซึ่งจะคำนวณจาก input และ hidden state ผ่านฟังก์ชัน Tanh เพื่อให้ข้อมูลมีค่าระหว่าง -1 และ 1

$$\tilde{c}_t = \tanh\left(W_c\left[h_{t-1,}\, x_t\right] + b_c\right) \tag{10}$$

โดยที่ $\tilde{c}_t$ คือ candidate cell state เป็นค่าข้อมูลใหม่ที่จะนำไปอัปเดตใน cell state, $\tanh$ คือ hyperbolic tangent function ให้ค่าอยู่ในช่วง [–1,1], W_c คือน้ำหนักของ candidate cell state และ b_c คือ bias ของ candidate cell state

4) Cell State Update

Cell State จะถูกอัปเดตโดยการผสมผสานข้อมูลที่ไม่จำเป็นจาก Forget Gate และข้อมูลใหม่จาก Input Gate และ Candidate Cell State การอัปเดตนี้จะช่วยให้ Cell State สามารถเก็บข้อมูลระยะยาวได้อย่างมีประสิทธิภาพในเวลาปัจจุบัน C_t

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{11}$$

โดยที่ c_t คือ cell state ปัจจุบัน ที่ timestep t เก็บข้อมูลระยะยาวใน LSTM, c_{t-1} คือ cell state ก่อนหน้า และ $\odot$ คือ element-wise multiplication การคูณกันแบบจุดต่อจุดของเวกเตอร์

สมการนี้แสดงให้เห็นว่า cell state ปัจจุบันเกิดจากการลบข้อมูลที่ไม่จำเป็น (ผ่าน $f_t \odot C_{t-1}$) แล้วเพิ่มข้อมูลใหม่ที่สำคัญ (ผ่าน $i_t \odot \tilde{c}_t$)

5) Output Gate

Output Gate มีหน้าที่ตัดสินใจว่าเราจะใช้ข้อมูลจาก Cell State เพื่อสร้าง Hidden State ฟังก์ชันซิกมอยด์ Sigmoid ใช้ในการตัดสินใจนี้ และ Tanh ใช้ในการแปลง Cell State ก่อนที่จะส่งไปเป็น Hidden State

$$o_t = \sigma\left(W_o\left[h_{t-1,}\, x_t\right] + b_o\right) \tag{12}$$

โดยที่ o_t คือ output gate activation กำหนดว่าจะส่งส่วนใดของ cell state ออกเป็น hidden state, W_o คือ น้ำหนักของ output gate และ b_o คือ bias ของ output gate

6) Hidden State

Hidden state คือผลลัพธ์สุดท้ายจาก LSTM หรือจะถูกส่งไปยังขั้นตอนถัดไปในโมเดล ซึ่งจะถูกคำนวณจาก Cell State ที่อัปเดตและ Output Gate

$$h_t = o_t \odot \tanh(c_t) \tag{13}$$

โดยที่ h_t คือ hidden state ของ LSTM ที่จะใช้ในเวลาต่อไป และ $\tanh(c_t)$ คือ การปรับค่า cell state c_t ให้อยู่ในช่วง [−1,1]

2.6 การวัดประสิทธิภาพของแบบจำลอง

การวัดค่าประสิทธิภาพของแบบจำลอง [51] ที่ใช้ในการศึกษานี้ใช้ มีดังนี้

Coefficient of determination Score (R-Squared) คือ ค่าความผันแปรของตัวแปรตอบสนองที่สามารถอธิบายได้มีอยู่ในตัวแบบเชิงเส้นนี้ กี่เปอร์เซ็นต์ หรือ ความผันแปรที่สามารถอธิบายได้ / ความผันแปรทั้งหมด ซึ่งหากค่า R-Squared มีค่าเข้าใกล้ 1 ยิ่งดี โดยแสดงไว้ในสมการที่ 14

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{i=N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{i=N}(y_i - \bar{y})^2}\right) \tag{14}$$

โดยที่ N คือจำนวนตัวอย่างทั้งหมด, y_i คือค่าจริง (observed value) ของตัวอย่างที่ i, $\hat{y}_i$ คือค่าพยากรณ์ (predicted value) ของตัวอย่างที่ i และ$\hat{y}_i$ คือค่าเฉลี่ยของค่าจริงทั้งหมด

Mean Absolute Error (MAE) คือ วิธีการวัดความแตกต่างระหว่างค่าที่คาดการณ์และค่าจริง โดยจะคำนวณได้จากการนำเอาผลต่างระหว่างค่าที่ทำนายกับคำตอบจริงของแต่ละตัวอย่างมายกกำลังสอง แล้วนำมาบวกกันทั้งหมด (machine learning) ซึ่งหากค่า MAE มีค่าเข้าใกล้ 0 ยิ่งดี โดยแสดงไว้ในสมการที่ 15

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{15}$$

Mean Squared Error (MSE) คือ วิธีการวัดความแตกต่างระหว่างค่าที่คาดการณ์และค่าจริง โดยจะคำนวณค่าผิดพลาดกำลังสอง (squared errors) และหาค่าเฉลี่ย ซึ่งหากค่า MSE มีค่าเข้าใกล้ 0 ยิ่งดี โดยแสดงไว้ในสมการที่ 16

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (16)$$

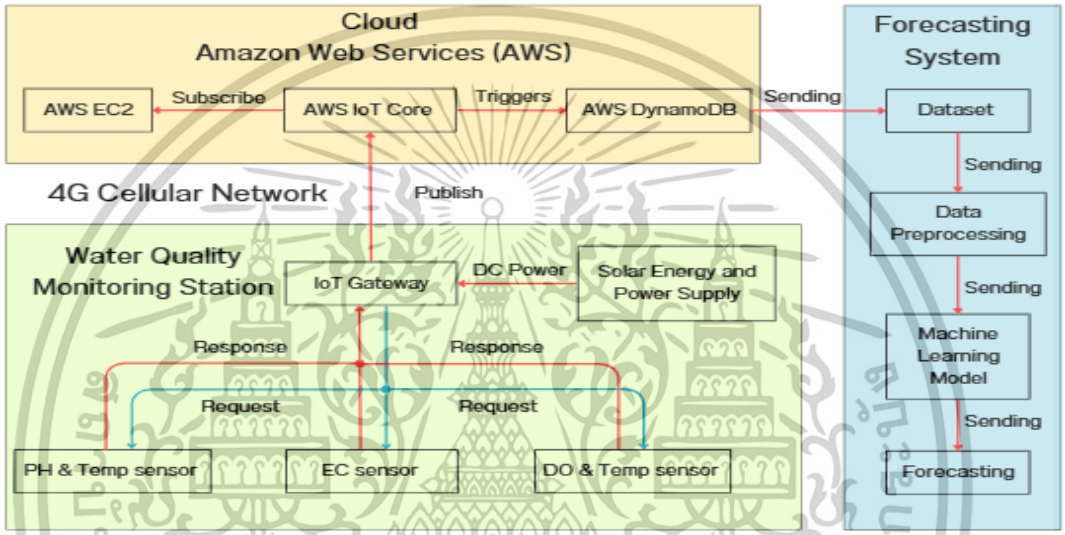
Root Mean Squared Error (RMSE) คือ วิธีการวัดความแตกต่างระหว่างค่าที่คาดการณ์และค่าจริง โดยการหาค่ารากที่สองของค่าเฉลี่ยของความผิดพลาดกำลังสอง (squared errors) ซึ่งหากค่า RMSE มีค่าเข้าใกล้ 0 ยิ่ง โดยแสดงไว้ในสมการที่ 17

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|} \quad (17)$$

บทที่ 3
การออกแบบระบบและสถาปัตยกรรม

3.1 ภาพรวมของระบบ

การออกแบบภาพรวมของการพัฒนาสถานีตรวจวัดคุณภาพน้ำด้วย Internet of Things (IoT) ประกอบไปด้วย 3 ส่วน คือ การออกแบบฮาร์ดแวร์สำหรับสถานีตรวจวัดคุณภาพน้ำ, การออกแบบซอฟแวร์สำหรับ IIoT Gateway และ Amazon Web Services (AWS) Cloud และการออกแบบระบบการพยากรณ์ โดยภาพรวมของงานวิจัยที่นำเสนอนี้แสดงไว้ในรูปที่ 3.1



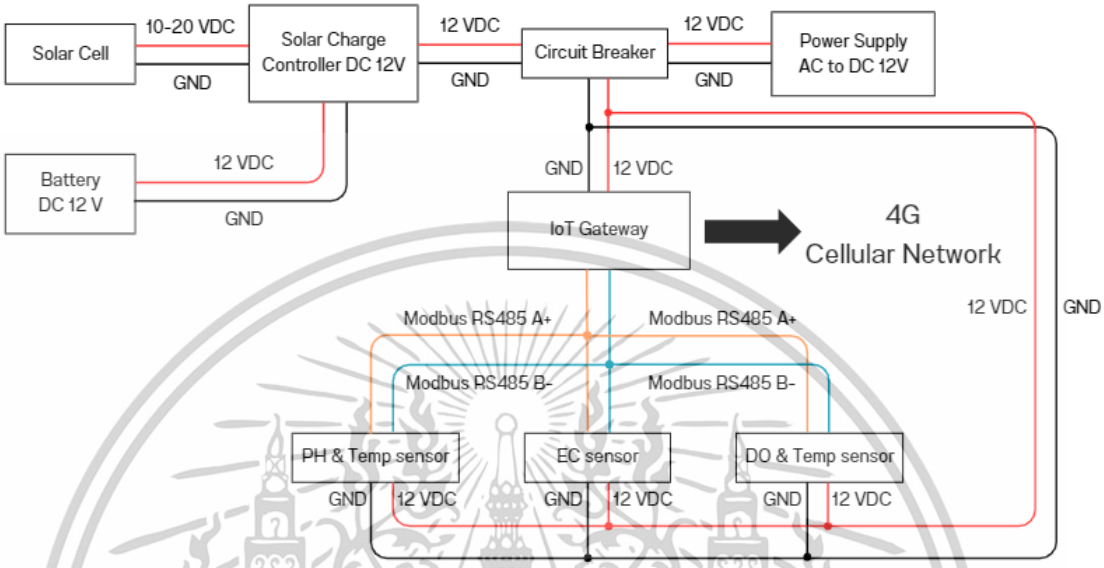
รูปที่ 3.1 ภาพรวมการออกแบบระบบที่นำเสนอ

ใน AWS Cloud จะเชื่อมต่อข้อมูลระหว่าง AWS Cloud และสถานีตรวจวัดคุณภาพน้ำผ่านทาง Sevice ของ AWS Cloud IoT Core ซึ่งจะใช้ Message Queuing Telemetry Transport Protocol (MQTT Protocol) และมี Broker ทำหน้าที่เป็นตัวกลางในการรับข้อมูลระหว่างกัน โดยเรียกใช้งานฟังก์ชั่น (Triggers) เขียนข้อมูลที่ได้รับจาก Sensor ในสถานีตรวจวัดคุณภาพน้ำไปเก็บใน Database ของ AWS Cloud DynamoDB และมี AWS Cloud EC2 ทำหน้าที่เป็น Server ของ Dashboard เพื่อแสดงข้อมูลต่างๆ ที่ได้รับจาก Sensor ในสถานีตรวจวัดคุณภาพน้ำ ซึ่งไปอ่านข้อมูล (Subscribe) จาก Server Broker ของ AWS Cloud IoT Core ผ่าน MQTT Protocol

ในสถานีตรวจวัดคุณภาพน้ำ "จะทำงานจาก Sensor ส่งข้อมูลค่าต่างๆไปยัง Siemens SIMATIC IOT2050 ซึ่งเป็น Industrial Internet of Things Gateway (IIoT Gateway) ผ่าน Recommended Standard no. 485 (RS485) Modbus RTU (Serial Communications Protocol) หลังจากนั้น IIoT Gateway ทำหน้าที่เป็น Edge Computing ประมวลผลข้อมูลต่างๆ และส่งข้อมูล (Publish) ไปยัง AWS Cloud ผ่านระบบอินเทอร์เน็ตบนเครือข่ายเซลลูลาร์ 4G โดยใช้ MQTT Protocol และนอกจากนี้สถานีตรวจวัดคุณภาพน้ำจะใช้พลังงานอยู่ 2 แหล่งพลังงาน ได้แก่ Solar Energy และ Power Supply

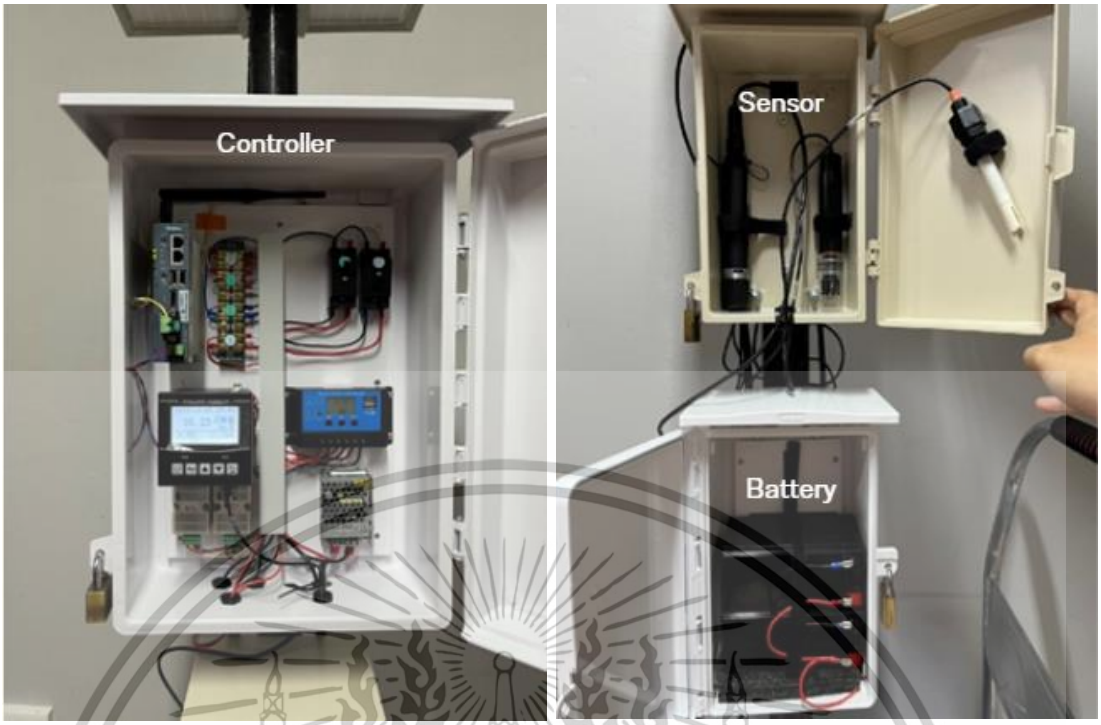
3.2 การออกแบบฮาร์ดแวร์

การออกแบบระบบฮาร์ดแวร์ของสถานีตรวจวัดคุณภาพน้ำประกอบไปด้วย 3 ส่วน คือ แหล่งพลังงาน, IIoT Gateway และ Sensor โดยภาพรวมของระบบฮาร์ดแวร์ของสถานีแสดงไว้ในรูปที่ 3.2-3.4



รูปที่ 3.2 ภาพรวมของระบบฮาร์ดแวร์ของสถานี

รูปที่ 3.3 ตัวอย่างภายนอกสถานีที่ออกแบบ



รูปที่ 3.4 ตัวอย่างภายในสถานีที่ออกแบบ

จากรูปเป็นระบบที่แสดงเป็นการสื่อสารข้อมูลระหว่างอุปกรณ์ IIoT Gateway กับเซนเซอร์ โดยใช้มาตรฐาน RS485 Modbus RTU ซึ่งใช้สายส่งข้อมูลเพียง 2 เส้น คือสาย Modbus RS485 A+ และ Modbus RS485 B- เชื่อมต่อกันระหว่างเซนเซอร์แต่ละตัว ได้แก่ เซนเซอร์วัดค่าความเป็นกรดเป็นด่าง (pH), เซนเซอร์วัดค่าการนำไฟฟ้า (EC sensor) และเซนเซอร์วัดค่าออกซิเจนละลายน้ำ (DO sensor) รวมถึงอุณหภูมิ โดยการเชื่อมต่อในลักษณะ Bus Topology ทำให้สามารถส่งข้อมูลผ่านสายคู่บิดเกลียวร่วมกันไปยัง IIoT Gateway ได้ IIoT Gateway จะเป็นอุปกรณ์หลักในการรวบรวมข้อมูลจากเซนเซอร์ทั้งหมด จากนั้นทำหน้าที่ ประมวลผลเบื้องต้นและส่งข้อมูลขึ้นสู่ระบบ Cloud ของ AWS ผ่านทางระบบอินเทอร์เน็ต โดยใช้เครือข่ายเซลลูลาร์ 4G ที่ติดตั้งโมดูลไว้ภายในตัว Gateway เอง ข้อมูลจะถูกส่งออกไป (Publish) ไปยัง AWS IoT Core โดยใช้ MQTT Protocol ซึ่งเป็นโปรโตคอลที่เหมาะสมสำหรับงานด้าน IoT เนื่องจากมีความเบาและประหยัดแบนด์วิดธ์

ในด้านพลังงาน ระบบได้รับการออกแบบให้ใช้แหล่งจ่ายไฟฟ้ากระแสตรง (DC) ที่มีแรงดันไฟฟ้าคงที่ที่ 12 โวลต์ (12 VDC) ซึ่งมีอยู่ 2 ระบบหลัก ได้แก่:

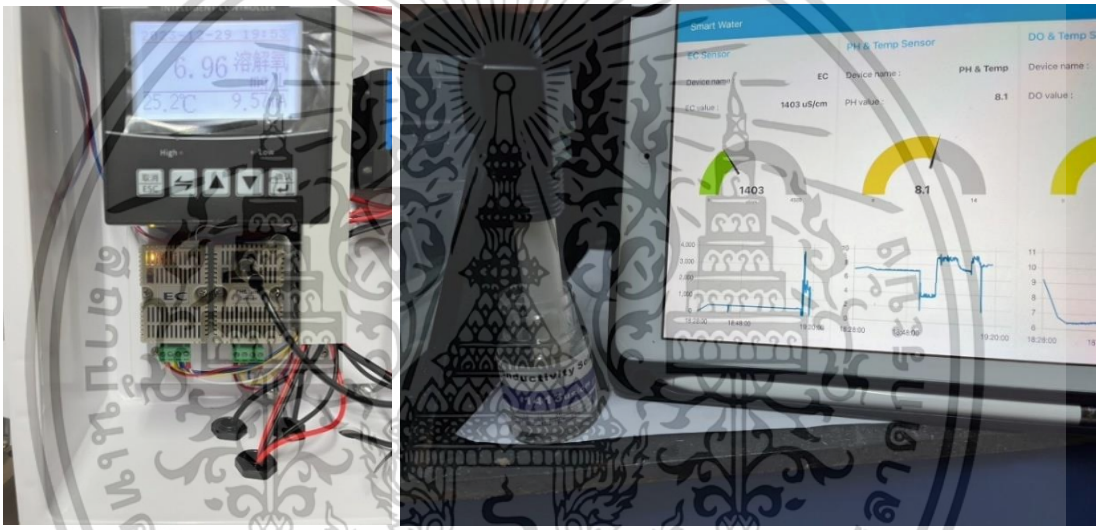
1) ระบบจ่ายไฟฟ้าจาก Power Supply: เป็นการแปลงพลังงานจากไฟฟ้ากระแสสลับ (AC) มาเป็นไฟฟ้ากระแสตรง (DC) โดยจะจ่ายไฟเข้าไปยังวงจรต่าง ๆ ผ่านตัว Circuit Breaker ซึ่งทำหน้าที่เป็นสวิตช์ควบคุมและป้องกันระบบไฟฟ้า
2) ระบบจ่ายไฟฟ้าจาก Solar Energy: ประกอบด้วย แผง Solar Cell ที่ผลิตกระแสไฟฟ้าจากพลังงานแสงอาทิตย์ โดยจ่ายไฟฟ้าให้กับ Solar Charge Controller ซึ่งทำหน้าที่ควบคุมแรงดันให้คงที่ที่ 12VDC และควบคุมการชาร์จพลังงานเข้าสู่

Battery ซึ่งใช้เป็นแหล่งพลังงานสำรองในกรณีที่ไม่มีแสงอาทิตย์ ทั้งนี้ยังสามารถจ่ายพลังงานให้กับอุปกรณ์ต่าง ๆ ผ่านทาง Circuit Breaker ได้เช่นเดียวกับระบบ Power Supply

การออกแบบระบบในลักษณะนี้ทำให้สามารถใช้งานได้อย่างต่อเนื่อง แม้ในกรณีที่แหล่งพลังงานหลักไม่สามารถจ่ายไฟได้ โดยมี Battery เป็นตัวรองรับการทำงานอย่างมีประสิทธิภาพ อีกทั้งยังรองรับการทำงานในพื้นที่ห่างไกลที่ไม่มีแหล่งจ่ายไฟฟ้าปกติได้อย่างเหมาะสม

3.3 การทดสอบเทียบเซนเซอร์วัด (Calibration)

การทดสอบเทียบเครื่องมือวัดของ Sensor (Calibration) เพื่อเพิ่มความแม่นยำของข้อมูล มีอยู่ 2 Sensor คือ EC Sensor และ PH and Temperature Sensor แสดงไว้ในรูปที่ 3.5-3.6



รูปที่ 3.5 การสอบเทียบเซนเซอร์วัดการนำไฟฟ้า

การทดสอบเทียบเครื่องมือวัดของ EC Sensor (Calibration) จากรูปที่ 3.5 เริ่มต้นด้วยการทำความสะอาด Probe แล้วเช็ดให้แห้งสนิท เตรียมสารละลายมาตรฐานค่าความนำไฟฟ้า 1413 uS/cm หลังจากนั้นกดปุ่มสอบเทียบค้างไว้จนกว่าจะแสดงไฟสีเหลือง หลังจากนั้นปล่อยปุ่ม และกดปุ่มสอบเทียบ 2 ครั้ง จนกระทั่งไฟเปลี่ยนเป็นสีแดง จะเห็นได้ว่าไฟสีแดงทำการกระพริบ ให้รอจนกว่าไฟสีแดงจะหยุดกระพริบ นำ Probe มาจุ่มสารละลายมาตรฐานค่าความนำไฟฟ้า 1413 uS/cm หลังจากนั้นทำการกดปุ่มสอบเทียบเพียงครั้งเดียว เมื่อทำการ Calibration สารละลายมาตรฐานค่าความนำไฟฟ้า 1413 uS/cm เรียบร้อย จะเห็นได้ว่าไฟดับ

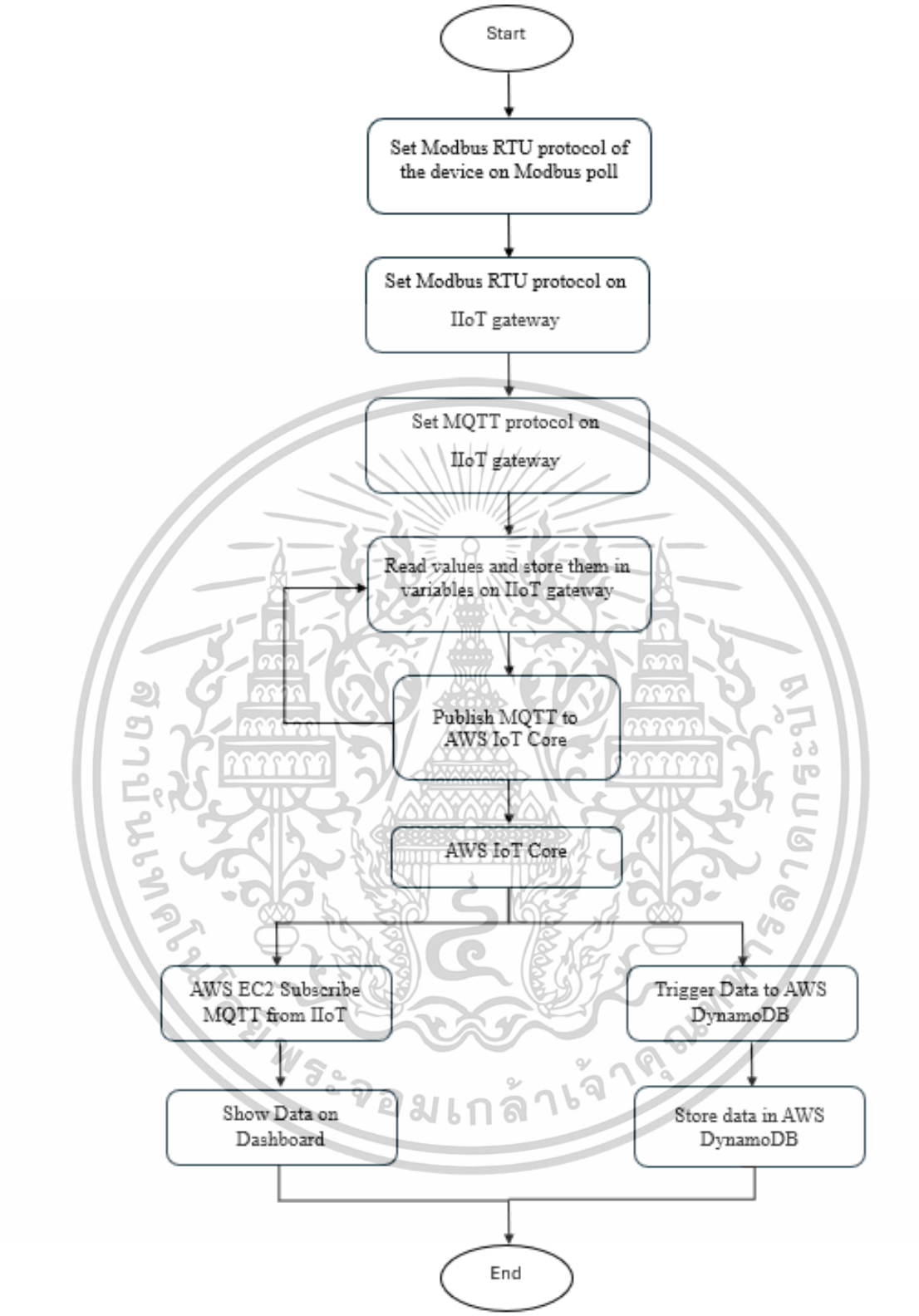
รูปที่ 3.6 การสอบเทียบเซนเซอร์วัดความเป็นกรดเป็นเบส

การทดสอบเทียบเครื่องมือวัดของ PH and Temperature Sensor (Calibration) จากรูปที่ 3.6 เริ่มต้นด้วยการทำความสะอาด Probe และเซ็นเซอร์วัดอุณหภูมิด้วยน้ำสะอาดแล้วเช็ดให้แห้งสนิท เตรียมสารละลายบัฟเฟอร์ PH 4.0 และ PH 9.18 นำ Probe และเซ็นเซอร์วัดอุณหภูมิ มาจุ่มสารละลายบัฟเฟอร์ PH 4.00 ก่อน หลังจากนั้นกดปุ่มสอบเทียบค้างไว้จนกว่าจะแสดงไฟสีเหลือง หลังจากนั้นปล่อยปุ่ม และกดปุ่มสอบเทียบ 2 ครั้ง จนกระทั่งไฟเปลี่ยนเป็นสีแดง จะเห็นได้ว่าไฟสีแดงทำการกระพริบ ให้รอ 30 วินาที เมื่อทำการ Calibration สารละลายบัฟเฟอร์ PH 4.00 เรียบร้อยจะเห็นได้ว่าไฟสีแดงไม่ ต่อมาให้ทำการสะอาด Probe การทำความสะอาด Probe และเซ็นเซอร์วัดอุณหภูมิด้วยน้ำสะอาดแล้วเช็ดให้แห้งสนิท และเตรียมสารละลายบัฟเฟอร์ PH 9.18 นำ Probe จุ่มลงไปในสารละลายบัฟเฟอร์ PH 9.18 หลังจากนั้นทำการกดปุ่มสอบเทียบเพียงครั้งเดียว จะเห็นได้ว่าไฟแสดงเป็นสีเขียว กระพริบ รอ 30 วินาที หลังจากนั้นเมื่อไฟสีเขียวหยุดกระพริบ ให้รอ 20 วินาที จนกระทั่งไฟเปลี่ยนเป็นสีเหลืองกระพริบ

หมายเหตุ : ถ้าไฟแสดง สีเขียว สีแดง กระพริบสลับกัน 20 วินาทีหมายถึงบันทึกค่าไม่สำเร็จ ให้ทดสอบอีกครั้ง

3.4 การออกแบบซอฟต์แวร์

การออกแบบระบบซอฟต์แวร์ของสถานีตรวจวัดคุณภาพน้ำประกอบไปด้วย 2 ส่วน คือ ภายใน IIoT Gateway และ AWS Cloud EC2 โดยผังงานภาพรวมของระบบซอฟต์แวร์ของสถานีแสดงไว้ในรูปที่ 3.7



รูปที่ 3.7 ผังงานภาพรวมของระบบซอฟต์แวร์ของสถานี

จากรูปที่ 3.7 การสร้างระบบซอฟต์แวร์ของสถานีเริ่มต้นจากการกำหนดค่าการสื่อสารระหว่างอุปกรณ์เซนเซอร์กับ IIoT Gateway โดยเลือกใช้โปรโตคอล Modbus RTU ซึ่งเป็นมาตรฐาน

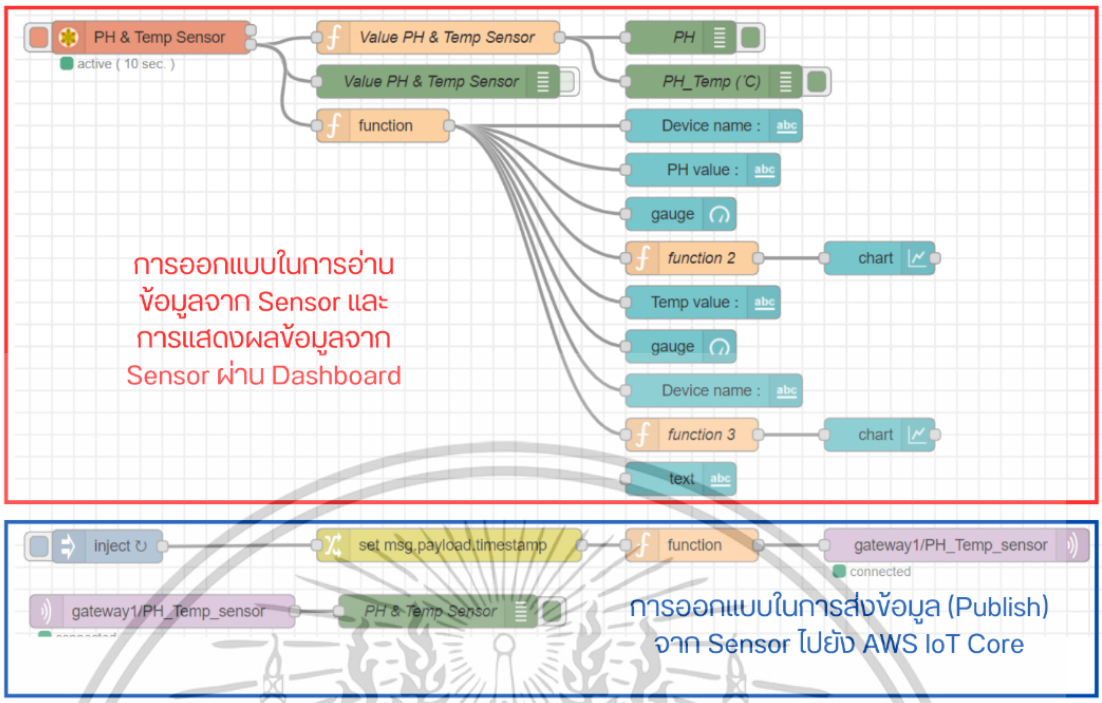
การสื่อสารแบบอนุกรมที่นิยมใช้ในงานอุตสาหกรรม เพื่อให้สามารถอ่านค่าข้อมูลจากเซนเซอร์ได้อย่างถูกต้องและต่อเนื่อง IIoT Gateway จะทำหน้าที่อ่านค่าและจัดเก็บข้อมูลภายในระบบเพื่อรอการส่งต่อ จากนั้นจะมีการตั้งค่าการเชื่อมต่อระหว่าง Gateway กับบริการ AWS IoT Core โดยใช้โปรโตคอล MQTT ซึ่งเหมาะสำหรับการส่งข้อมูลแบบเบาและรวดเร็ว เมื่อการเชื่อมต่อสมบูรณ์ Gateway จะทำการ publish ข้อมูลที่ได้จากเซนเซอร์ผ่าน MQTT ไปยัง Topic บน AWS IoT Core ข้อมูลที่ได้รับจะถูกประมวลผลตามกฎที่ตั้งไว้ในระบบ เช่น การ Trigger ให้จัดเก็บข้อมูลลงในฐานข้อมูล AWS DynamoDB อย่างเป็นระบบ เพื่อรองรับการเรียกใช้งานย้อนหลังหรือการวิเคราะห์ข้อมูล นอกจากนี้ ข้อมูลยังสามารถถูก subscribe โดยระบบบน AWS EC2 เพื่อนำไปแสดงผลแบบเรียลไทม์บน Dashboard ช่วยให้สามารถติดตามและตรวจสอบสถานะจากระยะไกลได้อย่างมีประสิทธิภาพ

การออกแบบ Software ภายใน IIoT Gateway ที่ออกแบบจะใช้เครื่องมือสำหรับนักพัฒนาโปรแกรมในการเชื่อมต่ออุปกรณ์ คือ Node-red ทำงานบน Node.js ภายในระบบปฏิบัติการ Debian (Linux) โดยโปรแกรมจะเริ่มการทำงานอ่านข้อมูลจาก Sensor แต่ละ Sensor ผ่านมาตรฐาน RS485 Modbus RTU ซึ่งมีการกำหนดค่าแต่ละ Sensor ตามตารางที่ 3.1 ดังนี้

ตารางที่ 3.1 การกำหนดค่าของเซนเซอร์แต่ละชนิดผ่านมาตรฐาน RS485 Modbus RTU

Topic	Electrical Conductivity (EC) sensor	Positive Potential of the Hydrogen Ions (PH) and Temperature	Dissolved Oxygen (DO) and Temperature Sensor
Baud rate	9600	9600	9600
Slave ID	3	4	5
Function Code	3 (Read Holding Registers)	3 (Read Holding Registers)	3 (Read Holding Registers)
Data address	Address 1 : EC Value (uS/cm)	Address 0 : PH Value Address 1 : Temperature Value (°C)	Address 0 : DO Value (mg/L) Address 1 : Temperature Value (°C)
Voltage	12VDC	12VDC	12VDC

การกำหนดค่าพารามิเตอร์แต่ละชนิดของเซนเซอร์ตามตารางที่ 3.1 เพื่อให้สามารถใช้งานได้โดยข้อมูลไม่ซ้อนทับกันในมาตรฐาน RS485 Modbus RTU และเป็นการบอกถึงรายละเอียดต่างๆในการรับ-ส่งค่าข้อมูล เช่น อัตราในการส่งข้อมูล ที่อยู่และฟังชันก์ในการเรียกใช้งาน รวมไปถึงแรงดันไฟฟ้าของแต่ละเซนเซอร์

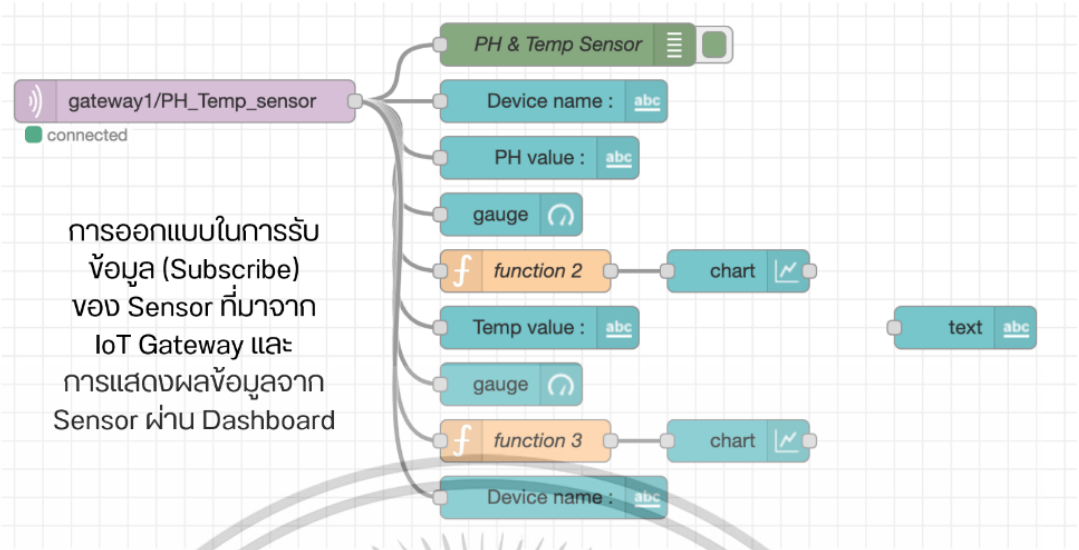


รูปที่ 3.8 ภาพรวมของระบบด้วย Node-RED ภายใน IIoT Gateway

จากรูปที่ 3.8 แสดงให้เห็นภาพรวมของการออกแบบซอฟต์แวร์ภายใน IIoT Gateway โดยใช้เครื่องมือ Node-RED ซึ่งเป็นแพลตฟอร์มที่เหมาะสำหรับงาน IoT โดยเฉพาะแบ่งการทำงานออกเป็น 2 ส่วนหลัก ได้แก่ การอ่านข้อมูลจาก Sensor และการส่งข้อมูลไปยังระบบคลาวด์

ในกรอบสีแดง เป็นการออกแบบระบบสำหรับการอ่านข้อมูลจาก Sensor ซึ่งในที่นี้ประกอบด้วย Sensor วัดค่า pH และอุณหภูมิ โดยใช้ Node ที่ชื่อว่า PH & Temp Sensor ทำหน้าที่ดึงข้อมูลผ่านโปรโตคอล Modbus RTU ซึ่งเชื่อมต่อผ่าน serial port หรือ RS-485 จากนั้นข้อมูลจะถูกส่งต่อมายัง Function Node เพื่อเขียนฟังก์ชันแยกค่า และจัดโครงสร้างข้อมูลให้อยู่ในรูปแบบที่เหมาะสม จากนั้นจึงเชื่อมต่อกับ Node แสดงผลต่างๆ เช่น text, gauge และ chart เพื่อให้ผู้ใช้สามารถตรวจสอบค่าต่าง ๆ ได้แบบเรียลไทม์ผ่าน Dashboard บนเว็บอินเทอร์เฟซของ Gateway

ในกรอบสีน้ำเงินเป็นการออกแบบกระบวนการส่งข้อมูล (Publishing) ไปยัง AWS IoT Core ซึ่งเริ่มต้นจาก inject node เพื่อจำลองการส่งข้อมูลแบบเป็นช่วงเวลา โดยเชื่อมต่อผ่าน Function Node เพื่อจัดรูปแบบข้อมูลให้เหมาะสม จากนั้นข้อมูลจะถูกส่งผ่าน MQTT-Out Node ไปยัง Topic ที่กำหนดไว้ใน AWS IoT Core MQTT Broker โดยผ่านเครือข่ายเซลลูลาร์ 4G ซึ่งรองรับการสื่อสารระยะไกลและมีความเสถียรในระดับที่เหมาะสม



รูปที่ 3.9 ภาพรวมของระบบด้วย Node-RED ภายใน AWS Cloud EC2

จากรูปที่ 3.9 แสดงการออกแบบการทำงานของซอฟต์แวร์บนแพลตฟอร์ม Node-RED ที่ติดตั้งอยู่ใน AWS Cloud EC2 โดยระบบจะเริ่มต้นด้วยการรับข้อมูล (Subscribe) จากเซนเซอร์ผ่าน MQTT Protocol ซึ่งเชื่อมต่อกับ AWS IoT Core โดยกำหนดหัวข้อ (Topic) เป็น gateway1/PH_Temp_sensor ผ่าน MQTT-in Node เพื่อดึงข้อมูลที่ส่งมาจาก IoT Gateway ซึ่งประกอบด้วยค่าจากเซนเซอร์วัดค่า pH และอุณหภูมิ ข้อมูลที่ได้รับจะถูกแยกออกเป็นพารามิเตอร์ต่างๆ เช่น ชื่ออุปกรณ์ (Device name), ค่า pH (PH value), และค่าอุณหภูมิ (Temp value) จากนั้นส่งไปยัง Function Node เพื่อประมวลผลเพิ่มเติม เช่น การแปลงค่าหรือจัดรูปแบบข้อมูลให้เหมาะสม ก่อนจะนำไปแสดงผลบน Dashboard ผ่าน Widget ประเภทต่างๆ เช่น gauge และ chart การใช้งาน Dashboard ของ Node-RED ทำให้สามารถแสดงผลข้อมูลแบบ Real-time ได้ทันที ทั้งในรูปแบบตัวเลขและกราฟ เพื่อช่วยให้ผู้ใช้สามารถติดตามสถานะของเซนเซอร์ได้อย่างมีประสิทธิภาพ โดยระบบนี้เหมาะสำหรับการประยุกต์ใช้ในงาน IoT ที่ต้องการติดตามข้อมูลจากอุปกรณ์ระยะไกลผ่านระบบ Cloud อย่างต่อเนื่อง

3.5 การออกแบบระบบการพยากรณ์

การพัฒนาแบบจำลองพยากรณ์ข้อมูลคุณภาพน้ำแม่น้ำเจ้าพระยาแบบอนุกรมเวลาในครั้งนี้ ได้ทำการรวบรวมข้อมูลพารามิเตอร์ด้านคุณภาพน้ำจากพื้นที่ต่างๆ ที่มีการติดตั้งเซนเซอร์ตรวจวัดแบบต่อเนื่อง ได้แก่ วัน-เวลา อุณหภูมิของน้ำ การนำไฟฟ้า (EC), ปริมาณออกซิเจนที่ละลายในน้ำ (DO) และค่าความเป็นกรด-เบส (pH) ซึ่งเป็นตัวแปรสำคัญในการบ่งชี้คุณภาพของน้ำ โดยข้อมูลดังกล่าวถูกจัดเตรียมและประมวลผลเบื้องต้น (Data Preprocessing) เช่น การเติมค่าที่หายไป การปรับสเกลข้อมูล และการจัดเรียงตามลำดับเวลา ก่อนจะนำเข้าสู่กระบวนการเรียนรู้ของโมเดล

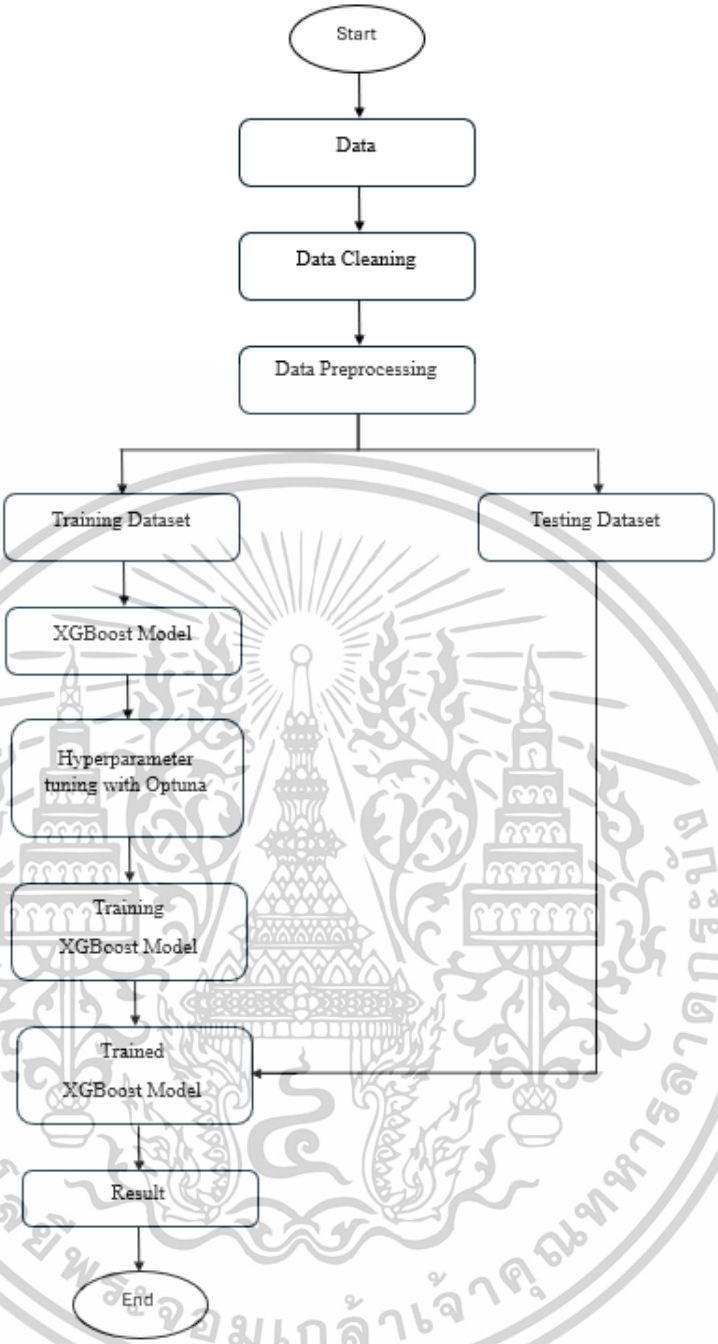
การสร้างแบบจำลองนั้นได้เลือกใช้สองเทคนิคหลัก ได้แก่ แบบจำลอง LSTM (Long Short-Term Memory) และ XGBoost (Extreme Gradient Boosting) เพื่อเปรียบเทียบประสิทธิภาพในการพยากรณ์ โดย LSTM นั้นสามารถจัดการกับข้อมูลอนุกรมเวลาได้ดีเยี่ยม เนื่องจากมีความสามารถในการเรียนรู้ความสัมพันธ์ในช่วงเวลาที่ยาวนาน ทำให้สามารถคาดการณ์แนวโน้มของคุณภาพน้ำในอนาคตได้อย่างแม่นยำ

ในขณะที่ XGBoost มีความสามารถในการจัดการกับชุดข้อมูลที่มีความซับซ้อนและไม่ต่อเนื่องได้ดี ด้วยการใช้เทคนิคของการเพิ่มประสิทธิภาพแบบต่อเนื่อง (Boosting) และการเรียนรู้จากข้อผิดพลาดของต้นไม้ก่อนหน้าอย่างเป็นระบบ นอกจากนี้ ยังได้ใช้เทคนิค Optuna ซึ่งเป็นเครื่องมือสำหรับการปรับแต่งค่าพารามิเตอร์แบบอัตโนมัติ (Hyperparameter Optimization) เพื่อหาเซตค่าที่เหมาะสมที่สุดสำหรับทั้งสองแบบจำลอง ซึ่งจะช่วยให้ผลลัพธ์ของการพยากรณ์มีความแม่นยำมากยิ่งขึ้น

โดยประสิทธิภาพของแบบจำลองถูกประเมินด้วยตัวชี้วัดมาตรฐาน ได้แก่:

1) Coefficient of Determination (R^2 Score): วัดความสามารถของแบบจำลองในการอธิบายความแปรปรวนของข้อมูล
2) Mean Absolute Error (MAE): ค่าความผิดพลาดเฉลี่ยแบบสัมบูรณ์
3) Mean Squared Error (MSE): ค่าความผิดพลาดเฉลี่ยยกกำลังสอง
4) Root Mean Squared Error (RMSE): ค่ารากที่สองของ MSE ซึ่งแสดงความคลาดเคลื่อนของการพยากรณ์ในหน่วยเดียวกับข้อมูลจริง

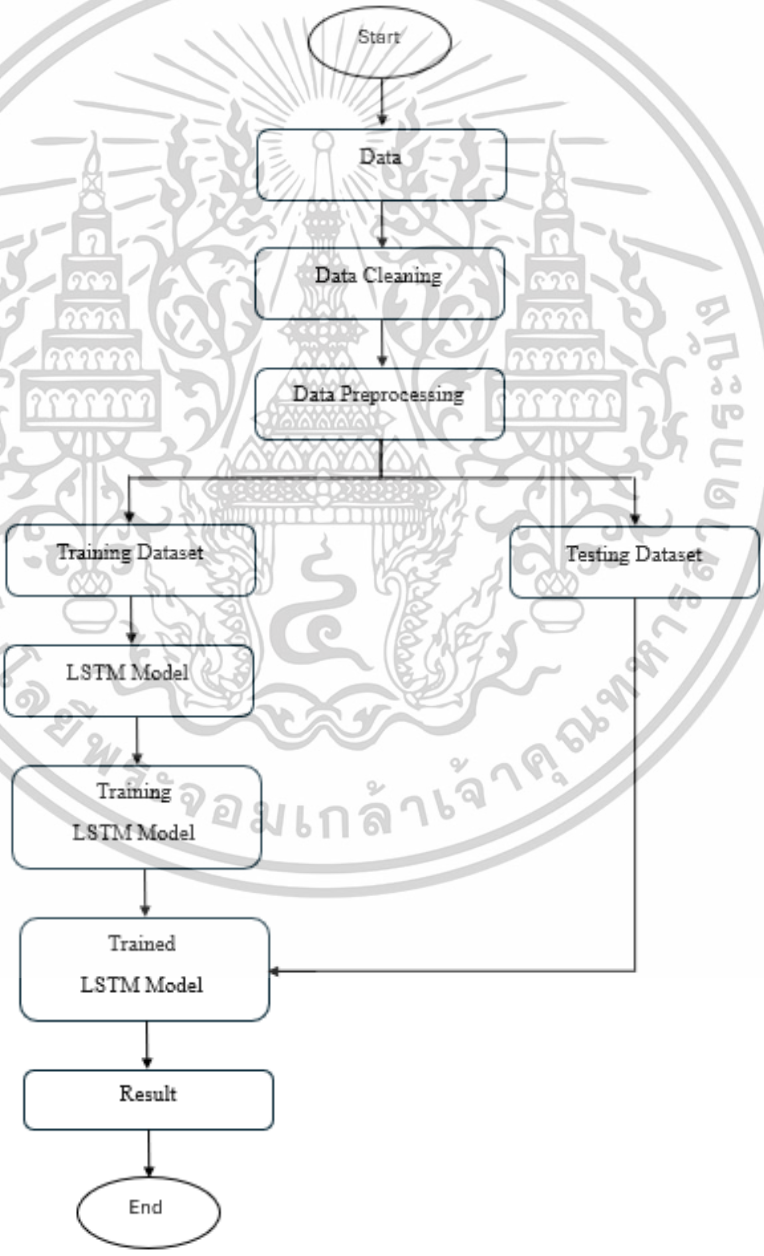
ผลลัพธ์จากการเปรียบเทียบแบบจำลองทั้งสองจะแสดงให้เห็นว่าแบบจำลองใดมีความเหมาะสมกับลักษณะของข้อมูลคุณภาพน้ำแม่น้ำเจ้าพระยามากที่สุด และสามารถนำไปประยุกต์ใช้ในการติดตามและพยากรณ์คุณภาพน้ำในอนาคตได้อย่างมีประสิทธิภาพ โดยเฉพาะในบริเวณที่เสี่ยงต่อการปนเปื้อน หรือมีการเปลี่ยนแปลงของสภาพน้ำอย่างรวดเร็ว อันจะช่วยสนับสนุนการบริหารจัดการทรัพยากรน้ำอย่างยั่งยืนและแม่นยำมากยิ่งขึ้น



รูปที่ 3.10 ผังงานการสร้างแบบจำลอง Extreme Gradient Boosting

จากรูปที่ 3.10 เป็นขั้นตอนการสร้างแบบจำลองเพื่อทำนายค่าด้วยอัลกอริทึม XGBoost เริ่มต้นจากการจัดการข้อมูลที่รวบรวมมาจากแหล่งต่าง ๆ โดยกระบวนการเตรียมข้อมูล (Data Preparation) ซึ่งครอบคลุมการทำความสะอาดข้อมูล เช่น การจัดการกับค่าที่หายไป ข้อมูลผิดปกติ และการแปลงข้อมูลให้อยู่ในรูปแบบที่เหมาะสมสำหรับการนำไปวิเคราะห์ หลังจากนั้น ทำการแบ่งชุดข้อมูลออกเป็น 2 ส่วน ได้แก่ ชุดข้อมูลฝึกสอน (Training Dataset) และ ชุดข้อมูลทดสอบ (Testing Dataset) โดยในส่วนของ Training Dataset จะนำมาใช้เพื่อฝึกสอนแบบจำลอง ซึ่งก่อนจะเข้าสู่กระบวนการฝึกสอน จะมีการเลือกคุณลักษณะ (Feature Selection) ที่มีความสัมพันธ์กับตัวแปรเป้าหมาย เพื่อเพิ่มประสิทธิภาพของแบบจำลอง จากนั้นนำข้อมูลที่ผ่านการเตรียมเข้าสู่แบบจำลอง

XGBoost (Extreme Gradient Boosting) ซึ่งเป็นอัลกอริทึมประเภท Machine Learning ที่มีประสิทธิภาพสูงทั้งในงาน Classification และ Regression ด้วยจุดเด่นเรื่องความแม่นยำ และความสามารถในการจัดการกับข้อมูลที่ซับซ้อน การปรับแต่งค่าพารามิเตอร์ (Hyperparameter Tuning) เป็นขั้นตอนสำคัญที่มีผลต่อคุณภาพของแบบจำลอง โดยแทนที่จะใช้วิธีการปรับแบบดั้งเดิม ซึ่งใช้เวลานานในการประมวลผล ทางผู้วิจัยเลือกใช้ Optuna ซึ่งเป็นไลบรารีที่ช่วยค้นหาค่าพารามิเตอร์ที่ดีที่สุดโดยอัตโนมัติ ผ่านการเรียนรู้จากรอบก่อนหน้า (Sequential Optimization) ทำให้ได้ค่าที่เหมาะสมที่สุดอย่างรวดเร็ว เมื่อได้แบบจำลองที่ผ่านการฝึกด้วยค่าพารามิเตอร์ที่ดีที่สุดแล้ว จึงนำไปใช้ในการ ทำนายผลลัพธ์กับชุดข้อมูลทดสอบ (Testing Dataset) ซึ่งในกรณีนี้อาจเลือกข้อมูลช่วงเวลาสุดท้ายของชุดข้อมูล เพื่อประเมินความแม่นยำและประสิทธิภาพของแบบจำลองก่อนนำไปใช้งานจริง



รูปที่ 3.11 ผังงานการสร้างแบบจำลอง Long Short-Term Memory

จากรูปที่ 3.11 เป็นเริ่มต้นจากการรวบรวมข้อมูลดิบที่ได้จากแหล่งข้อมูลต่าง ๆ และนำมาผ่านกระบวนการเตรียมข้อมูล (Data Preparation) ซึ่งประกอบด้วยการทำความสะอาดข้อมูล (Data Cleaning) เช่น การจัดการกับค่าที่หายไป การลบข้อมูลซ้ำ หรือการจัดรูปแบบให้อยู่ในลักษณะที่เหมาะสมสำหรับการนำเข้าสู่แบบจำลอง หลังจากข้อมูลได้รับการทำความสะอาดแล้ว จะเข้าสู่ขั้นตอนการ เตรียมข้อมูลล่วงหน้า (Data Preprocessing) ซึ่งอาจประกอบด้วยการทำ normalization หรือ standardization เพื่อปรับค่าข้อมูลให้อยู่ในช่วงที่เหมาะสม ตลอดจนการแปลงข้อมูลให้อยู่ในรูปแบบลำดับเวลา (Sequential Input) ซึ่งจำเป็นสำหรับการป้อนเข้าสู่โมเดล LSTM เมื่อข้อมูลพร้อมแล้ว จะมีการแบ่งข้อมูลออกเป็น 2 ชุด ได้แก่ ชุดข้อมูลฝึกสอน (Training Dataset) สำหรับการสร้างแบบจำลอง และชุดข้อมูลทดสอบ (Testing Dataset) สำหรับการประเมินประสิทธิภาพของแบบจำลองหลังการฝึก ในขั้นตอนของ Training Dataset จะทำการสร้างแบบจำลอง LSTM (Long Short-Term Memory) ซึ่งเป็นโครงข่ายประสาทเทียมแบบ Recurrent Neural Network (RNN) ที่มีความสามารถในการจดจำข้อมูลจากอดีตได้ในช่วงเวลานาน และเหมาะสมกับปัญหาที่เป็นลำดับเวลา (time-series) เช่น การพยากรณ์หรือการจำแนกรูปแบบในข้อมูลต่อเนื่อง แบบจำลอง LSTM จะถูกฝึก (Training) ด้วยข้อมูลฝึกสอน พร้อมการตั้งค่าพารามิเตอร์ต่าง ๆ เช่น จำนวนหน่วยความจำ (LSTM Units), จำนวนชั้น (Layers), dropout, batch size และ learning rate ซึ่งสามารถปรับให้เหมาะสมเพื่อให้โมเดลมีประสิทธิภาพสูงสุด เมื่อฝึกเสร็จแล้ว จะได้ แบบจำลองที่ผ่านการฝึก (Trained LSTM Model) ซึ่งจะนำไปใช้ในการทำนายกับข้อมูลทดสอบ โดยใช้ Testing Dataset ที่อาจเป็นข้อมูลในช่วงเวลาสุดท้าย เช่น สัปดาห์สุดท้ายของเดือน เพื่อดูว่าแบบจำลองสามารถทำนายได้แม่นยำเพียงใด ผลลัพธ์ที่ได้จากการทดสอบจะถูกนำมาวิเคราะห์ เพื่อประเมินความแม่นยำของแบบจำลอง และอาจนำไปใช้ในงานทำนายจริงต่อไปในอนาคต

บทที่ 4
การดำเนินงานและผลการวิจัย

4.1 การพัฒนาและการทดสอบการทำงานของสถานี

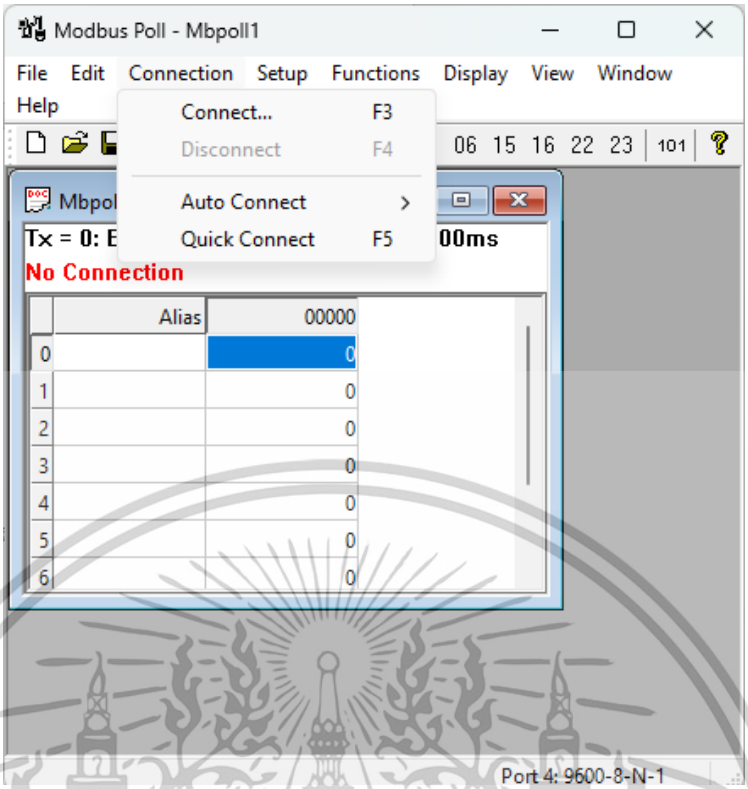
4.1.1 การตั้งค่าอุปกรณ์วัดด้วย Modbus Poll

4.1.1.1 การอ่านค่าอุปกรณ์วัดด้วย Modbus Poll

เริ่มต้นโดยเชื่อมต่ออุปกรณ์เซนเซอร์เข้ากับอุปกรณ์แปลงสัญญาณ RS485 to USB Serial และต่อเข้ากับแหล่งจ่ายไฟ (Power Supply) ให้เรียบร้อย จากนั้นนำสาย USB จากอุปกรณ์แปลงสัญญาณ RS485 to USB Serial มาเชื่อมต่อเข้ากับคอมพิวเตอร์ที่ติดตั้งโปรแกรม Modbus Poll เพื่อใช้ในการสื่อสารและอ่านค่าข้อมูลจากเซนเซอร์ผ่านโปรโตคอล Modbus RTU

รูปที่ 4.1 อุปกรณ์แปลงสัญญาณ RS485 to USB Serial

หลังจากเชื่อมต่ออุปกรณ์เรียบร้อยแล้ว การอ่านค่าจากอุปกรณ์เซนเซอร์สามารถทำได้โดยใช้โปรแกรม Modbus Poll โดยเริ่มจากการเชื่อมต่อโปรแกรมกับอุปกรณ์ผ่านพอร์ตอนุกรม โดยคลิกที่แถบ "Connection" จากนั้นเลือก "Connect..." เพื่อกำหนดพอร์ตและพารามิเตอร์ที่ใช้ในการสื่อสาร เช่น COM Port, Baud Rate, Parity, Data Bits และ Stop Bits ให้ตรงกับการตั้งค่าของอุปกรณ์ เมื่อเชื่อมต่อสำเร็จ โปรแกรมจะสามารถอ่านค่าจากเซนเซอร์ได้ตามที่กำหนดไว้ในโปรโตคอล Modbus RTU

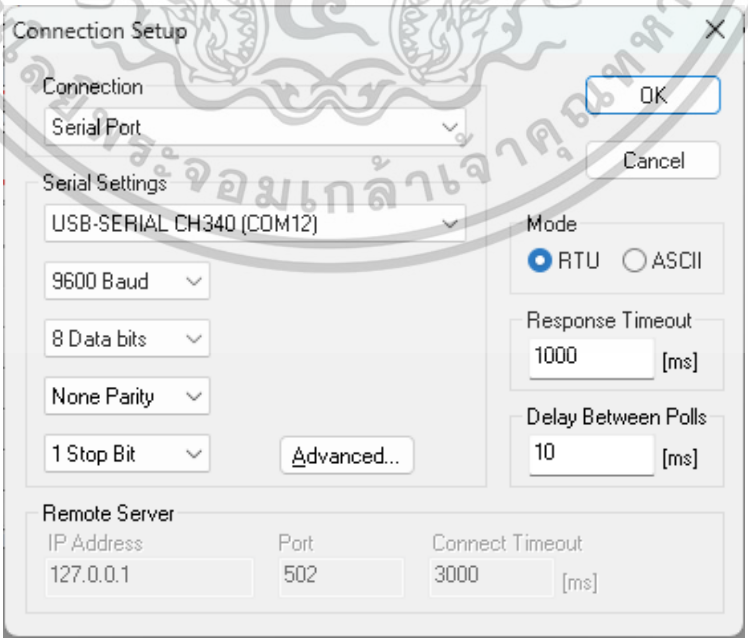


รูปที่ 4.2 การเชื่อมต่ออุปกรณ์วัดกับโปรแกรม Modbus Poll

หลังจากนั้นจะมีหน้าต่างการตั้งค่า ให้ตั้งค่าตามดังนี้

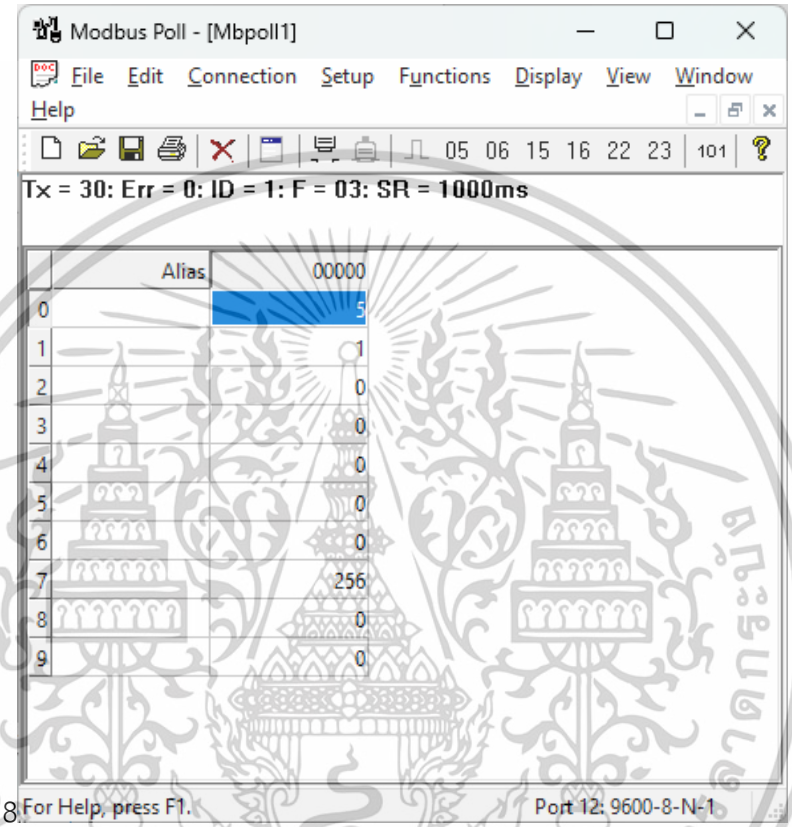
1) เลือก Serial Port: USB-SERIAL CH340 (COM12)
2) Baud Rate: 9600

ตั้งค่าเรียบร้อยกด OK



รูปที่ 4.3 การตั้งค่าอุปกรณ์วัดด้วยโปรแกรม Modbus Poll

เมื่อเชื่อมต่อสำเร็จและเริ่มการอ่านค่าด้วยโปรแกรม Modbus Poll จะปรากฏค่าที่อ่านได้จากอุปกรณ์เซนเซอร์ตามรายละเอียดที่ระบุไว้ใน Data Sheet โดยในกรณีของเซนเซอร์นี้ข้อมูลที่ต้องการจะอยู่ที่ Address 0 ซึ่งเป็นตำแหน่งเริ่มต้นของข้อมูลในหน่วยความจำของอุปกรณ์ตามมาตรฐานของโปรโตคอล Modbus RTU

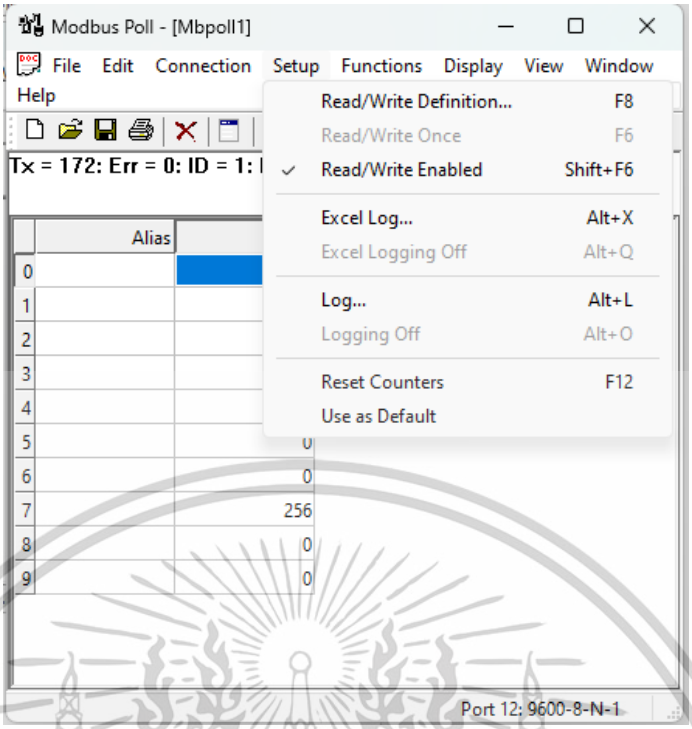


รูปที่ 4.4 การอ่านค่าอุปกรณ์วัดด้วยโปรแกรม Modbus Poll

4.1.1.2 การเปลี่ยนการตั้งค่าอุปกรณ์วัดต่างๆ ใหม่ด้วย Modbus Poll

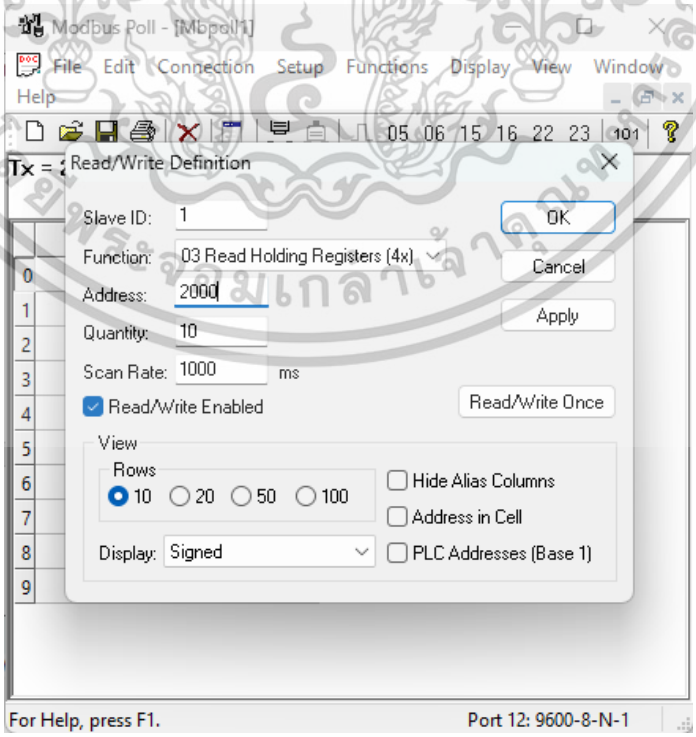
เพื่อให้สามารถใช้งานเซนเซอร์ได้มากกว่าหนึ่งตัวบนเครือข่ายเดียวกันโดยไม่มีปัญหาการสื่อสารชนกัน จำเป็นต้องเปลี่ยนค่าการตั้งค่า Slave ID และ Baud Rate ของอุปกรณ์เซนเซอร์แต่ละตัวให้แตกต่างกันอย่างเหมาะสม การตั้งค่าดังกล่าวช่วยให้สามารถจัดระเบียบการสื่อสารบนสาย RS485 ได้อย่างมีประสิทธิภาพ และรองรับการใช้งานร่วมกับอุปกรณ์หลายตัวในระบบเดียว

การเปลี่ยนค่า Slave ID และ Baud Rate สามารถทำได้ผ่านโปรแกรม Modbus Poll โดยไปที่แถบเมนู "Setup" จากนั้นเลือก "Read/Write Definition" เพื่อเข้าสู่หน้าต่างสำหรับกำหนดค่าพารามิเตอร์ต่าง ๆ ของเซนเซอร์ แล้วดำเนินการปรับเปลี่ยนค่าตามต้องการ



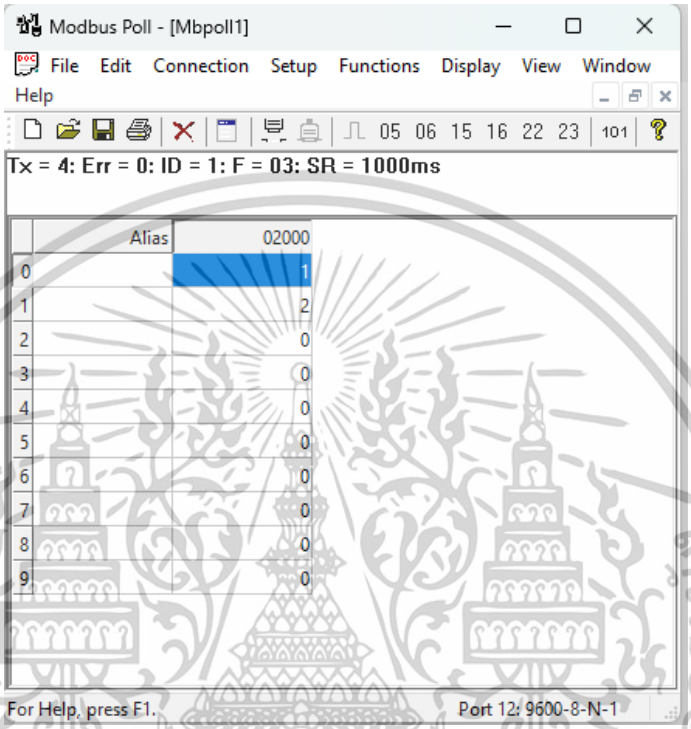
รูปที่ 4.5 การตั้งค่าอุปกรณ์วัดใหม่ด้วยโปรแกรม Modbus Poll

เมื่อเข้าสู่หน้าต่างการตั้งค่าแล้ว คุณจะสามารถปรับค่า Slave ID และ Address ที่ต้องการอ่านได้ จากนั้นให้เลื่อนหน้าต่างไปยัง Address 2000 เพื่อกำหนดตำแหน่งที่ต้องการอ่านข้อมูลจากอุปกรณ์เซนเซอร์ ซึ่งจะช่วยให้สามารถเข้าถึงข้อมูลจากตำแหน่งที่ต้องการได้อย่างถูกต้องและแม่นยำ



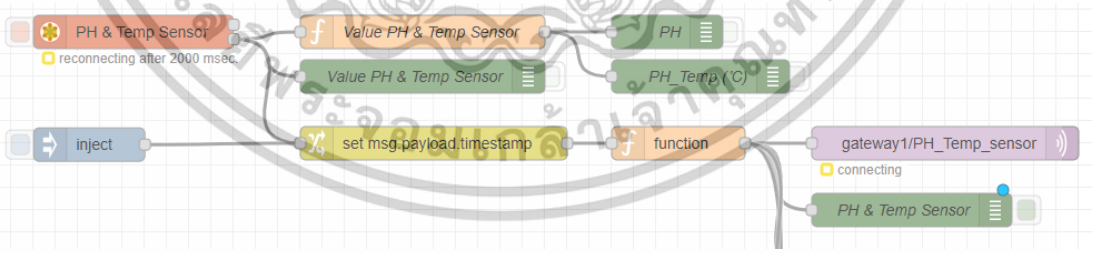
รูปที่ 4.6 การกำหนดการตั้งค่าอุปกรณ์วัดด้วยโปรแกรม Modbus Poll

ให้ทำการ Double Click ที่ช่องข้อมูลตรง Address 2000 เพื่อทำการตั้งค่า Slave ID จากนั้นทำการ Double Click ที่ช่องข้อมูลตรง Address 2001 เพื่อทำการตั้งค่า Baud Rate ตามค่าที่ต้องการ ซึ่งจะช่วยให้การสื่อสารระหว่างอุปกรณ์เซนเซอร์และระบบสามารถทำงานได้ตามที่กำหนดไว้



รูปที่ 4.7 ผลที่ได้จากการตั้งค่าอุปกรณ์วัดด้วยโปรแกรม Modbus Poll

4.1.2 การตั้งค่า node-red ในเกตเวย์



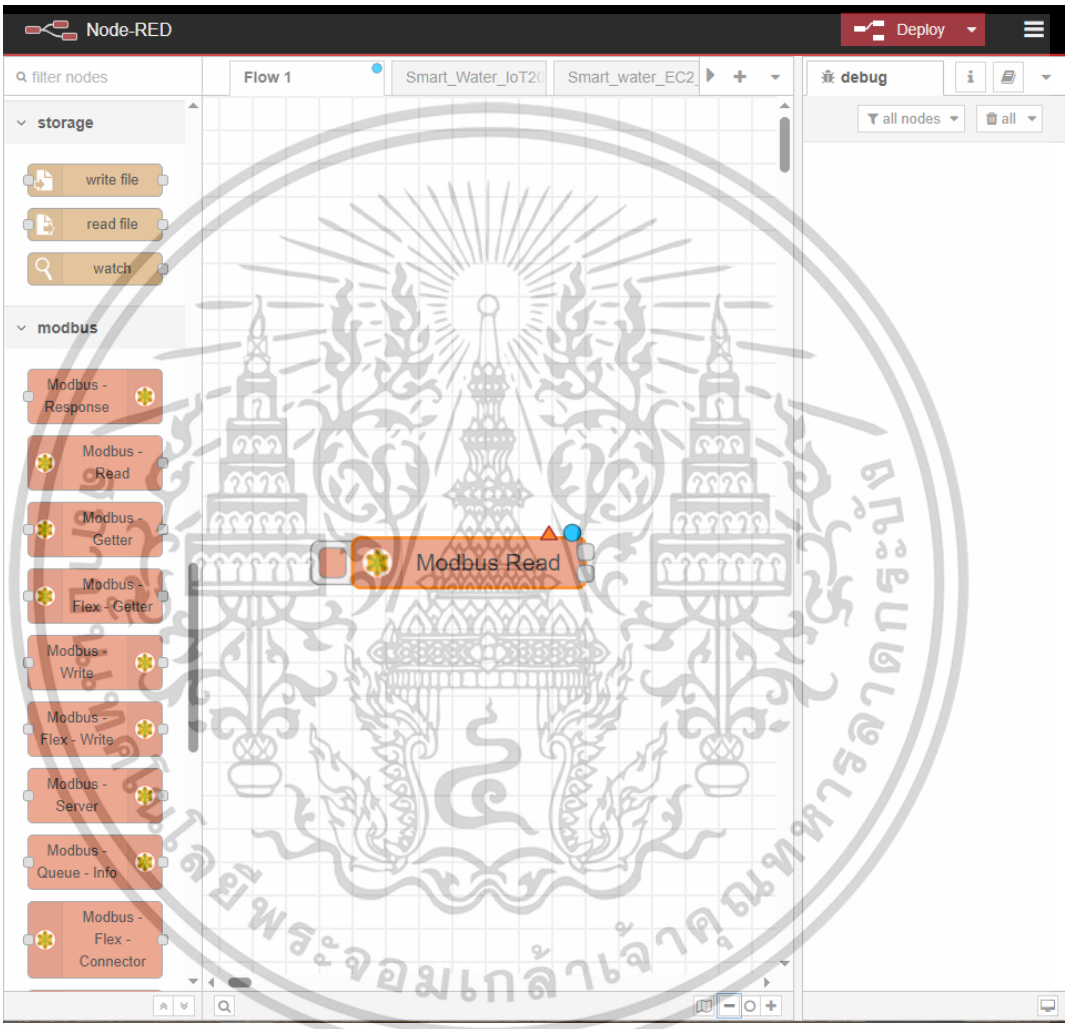
รูปที่ 4.8 ภาพรวมการอ่านข้อมูลอุปกรณ์วัดบนเกตเวย์

การตั้งค่า Node-RED ใน IoT Gateway เพื่ออ่านข้อมูลจากอุปกรณ์เซนเซอร์จะเริ่มต้นด้วยการตั้งค่าการใช้ Modbus Read Node เพื่อเชื่อมต่อและอ่านข้อมูลจากอุปกรณ์เซนเซอร์ โดยในขั้นตอนถัดไปจะทำการตั้งค่า Function Node เพื่อเขียนฟังก์ชันที่ทำหน้าที่อ่านข้อมูลจากเซนเซอร์และเก็บค่าไว้ในตัวแปรที่กำหนด จากนั้นทำการตั้งค่าตัวแปรที่จำเป็นเพื่อใช้ในการสื่อสารระหว่าง IoT Gateway กับ AWS IoT Core ผ่านโปรโตคอล MQTT ซึ่งเป็นมาตรฐานการสื่อสารที่ใช้ในระบบ IoT เมื่อการตั้งค่าการสื่อสารเสร็จสิ้นแล้ว ข้อมูลที่ได้จากเซนเซอร์จะถูกส่งผ่าน MQTT out Node เพื่อทำ

การ publish ข้อมูลไปยัง AWS IoT Core ซึ่งจะช่วยให้ข้อมูลสามารถนำไปใช้งานในการประมวลผลหรือจัดเก็บในระบบคลาวด์ได้ตามต้องการ

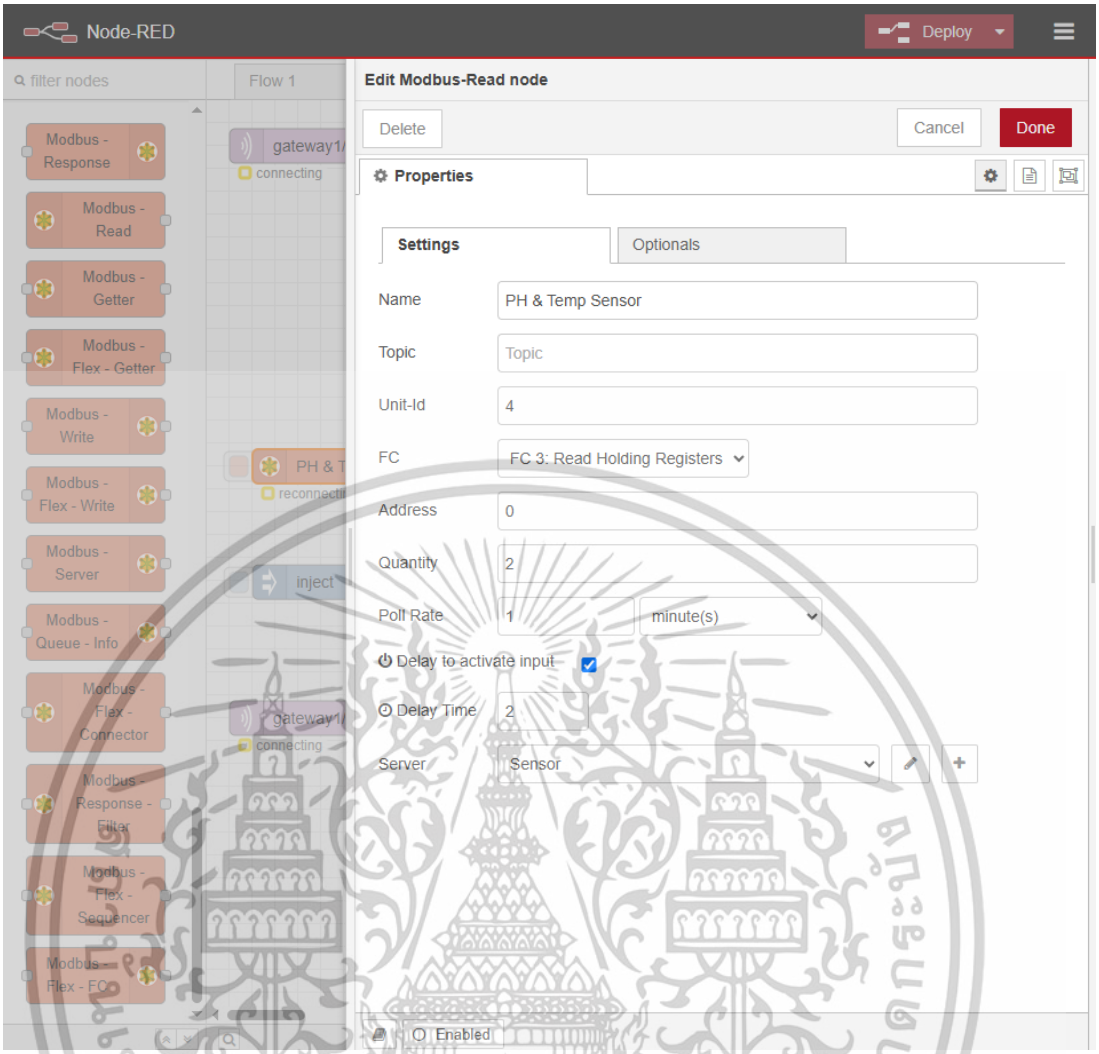
4.1.2.1 อ่านข้อมูลอุปกรณ์ sensor ด้วย Modbus RTU Protocol

โดยเริ่มจากการเชื่อมต่ออุปกรณ์เซนเซอร์เข้ากับ IoT Gateway ผ่าน COM interfaces (RS232/422/485) จากนั้นทำการอ่านข้อมูลจากอุปกรณ์เซนเซอร์ด้วย Modbus RTU Protocol โดยการตั้งค่าโปรแกรม Node-RED ใน IoT Gateway ดังนี้



รูปที่ 4.9 Modbus Read Node ในการอ่านข้อมูล

เลือก Modbus Read Node จาก Modbus Node และลากมาวางลงใน Flow ของโปรแกรม Node-RED จากนั้นทำการ Double Click ที่ตัว Node เพื่อทำการตั้งค่าคอนฟิกและปรับแต่งค่าต่าง ๆ ตามความต้องการ

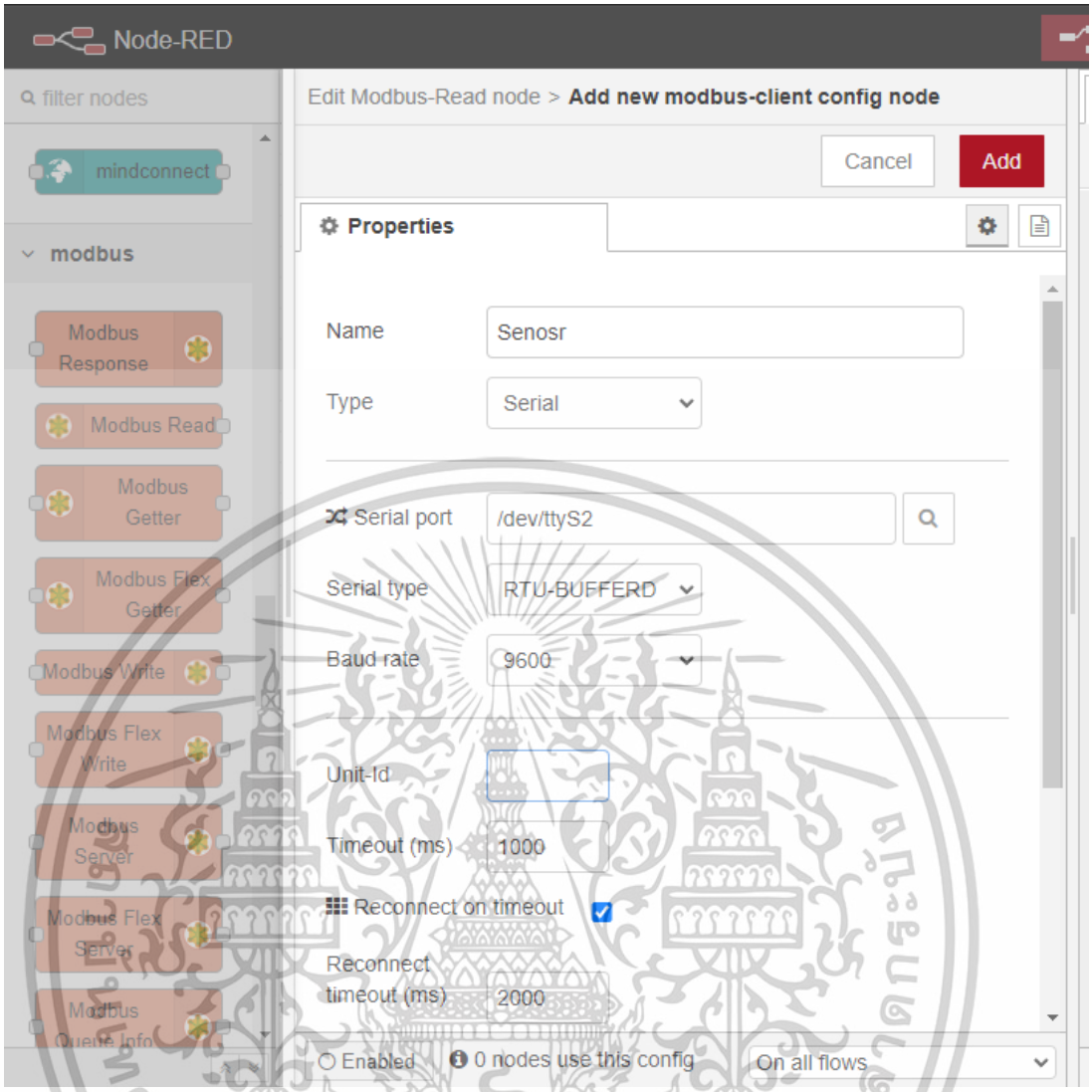


รูปที่ 4.10 การตั้งค่า Modbus Read Node ในการอ่านข้อมูล

หลังจากนั้นจะปรากฏหน้าต่างการตั้งค่า Modbus-Read Node ให้ทำการตั้งค่าตามข้อมูลเฉพาะของอุปกรณ์เซนเซอร์ดังนี้:

1) ตั้งชื่อ: PH & Temp Sensor
2) Unit-Id / Slave ID: 4
3) FC 3: Read Holding Registers
4) Address: 0
5) Quantity: 2

เมื่อทำการตั้งค่าต่าง ๆ เสร็จแล้ว ให้กดปุ่ม แก้ไข ตรง Server เพื่อสร้างมาตรฐานการสื่อสารโดยใช้ Modbus RTU Protocol ตามที่ต้องการ

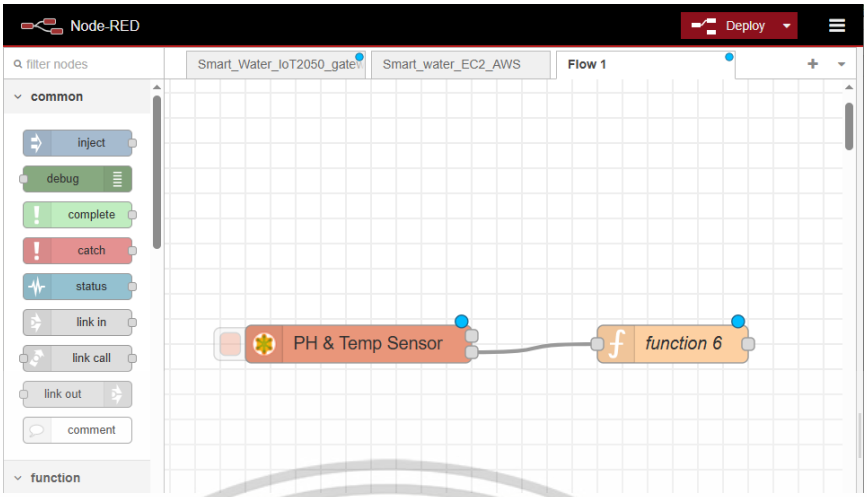


รูปที่ 4.11 การตั้งค่า Modbus-client config node ในการอ่านข้อมูล

หลังจากนั้นจะปรากฏหน้าต่างการตั้งค่า Modbus-client config node ให้ทำการตั้งค่าดังนี้

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

เมื่อทำการตั้งค่าทั้งหมดเสร็จแล้ว ให้กดปุ่ม Add เพื่อบันทึกการตั้งค่า



รูปที่ 4.12 การเชื่อมต่อ Function Node กับ Modbus Read Node ในการอ่านข้อมูล

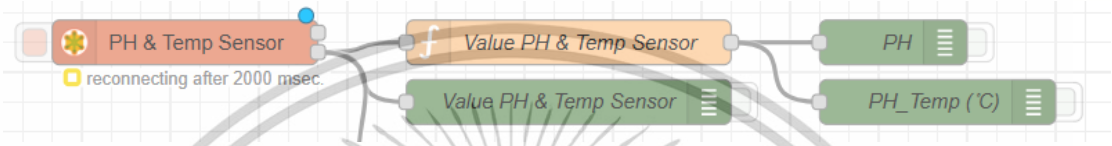
เลือก Function Node จากหมวด Function และลากมาวางลงใน Flow จากนั้นทำการลากสายเชื่อมต่อระหว่าง Function Node กับ Modbus-Read Node เพื่อสร้างการเชื่อมต่อ เมื่อเชื่อมต่อเรียบร้อยแล้ว ให้ทำการ Double Click ที่ตัว Function Node เพื่อเข้าสู่หน้าต่างการตั้งค่าและการเขียนฟังก์ชัน

รูปที่ 4.13 การตั้งค่า Function node ในการอ่านข้อมูล

หลังจากนั้นจะปรากฏหน้าต่างการตั้งค่า Function Node ให้ทำการเขียนโปรแกรมตามคำสั่งดังนี้:

1) msg.payload.PH_Temp = msg.payload.data[0]/10;
2) msg.payload.PH = msg.payload.data[1]/10;
3) global.set('PH_Temp',msg.payload.PH_Temp);
4) global.set('PH',msg.payload.PH_Temp);
5) return msg;

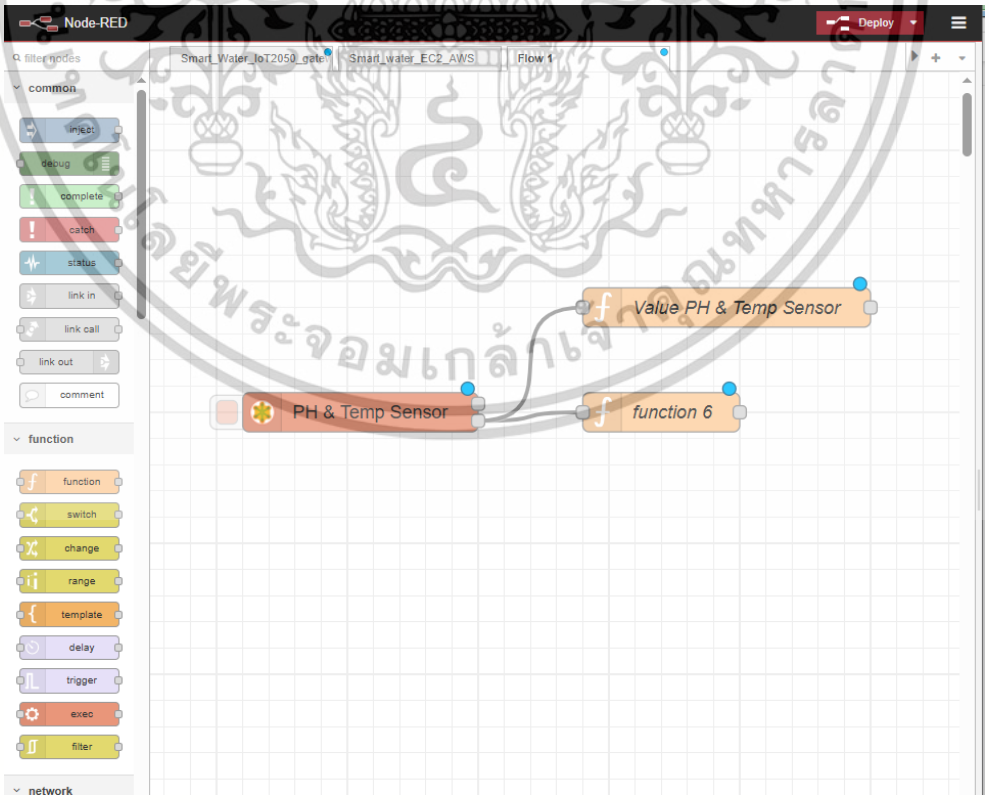
เมื่อเขียนโปรแกรมเสร็จแล้ว ให้กดปุ่ม Done เพื่อบันทึกการตั้งค่า



รูปที่ 4.14 การอ่านข้อมูลอุปกรณ์วัดบนเกตเวย์

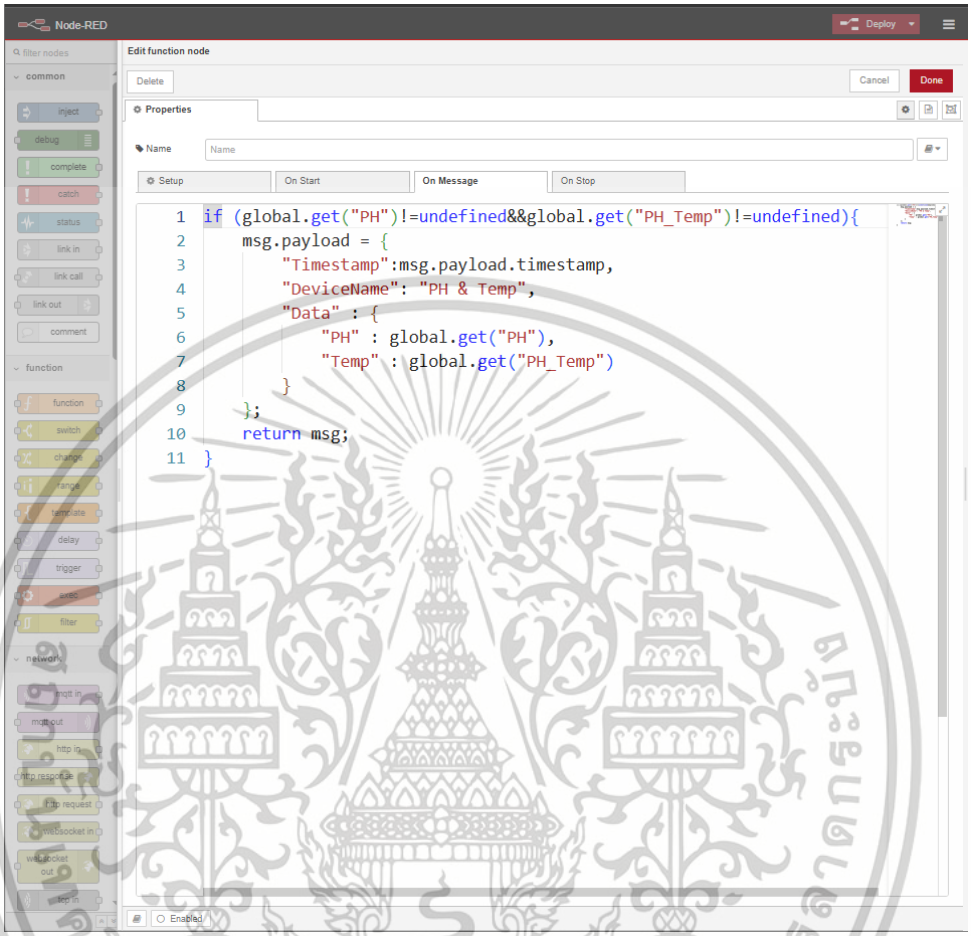
4.1.2.2 ส่งข้อมูลอุปกรณ์วัดไปยัง AWS IoT Core ด้วย MQTT Protocol

เริ่มจากการเชื่อมต่ออุปกรณ์เซนเซอร์เข้ากับ IoT Gateway ผ่าน COM interfaces (RS232/422/485) และแหล่งจ่ายไฟ (Power Supply) ที่เหมาะสม จากนั้นทำการอ่านข้อมูลจากอุปกรณ์เซนเซอร์โดยใช้ Modbus RTU Protocol โดยการตั้งค่าโปรแกรม Node-RED ใน IoT Gateway ตามขั้นตอนดังนี้:



รูปที่ 4.15 การเชื่อมต่อ Function Node ในการส่งข้อมูล

เลือก Function Node จากหมวด Function และลากมาวางลงใน Flow จากนั้นลากสายเชื่อมต่อระหว่าง Function Node กับ Modbus-Read Node ที่สร้างขึ้น หลังจากนั้นทำการ Double Click ที่ Function Node ที่เพิ่งสร้างขึ้นเพื่อเข้าสู่หน้าต่างการตั้งค่าและการปรับแต่งฟังก์ชัน

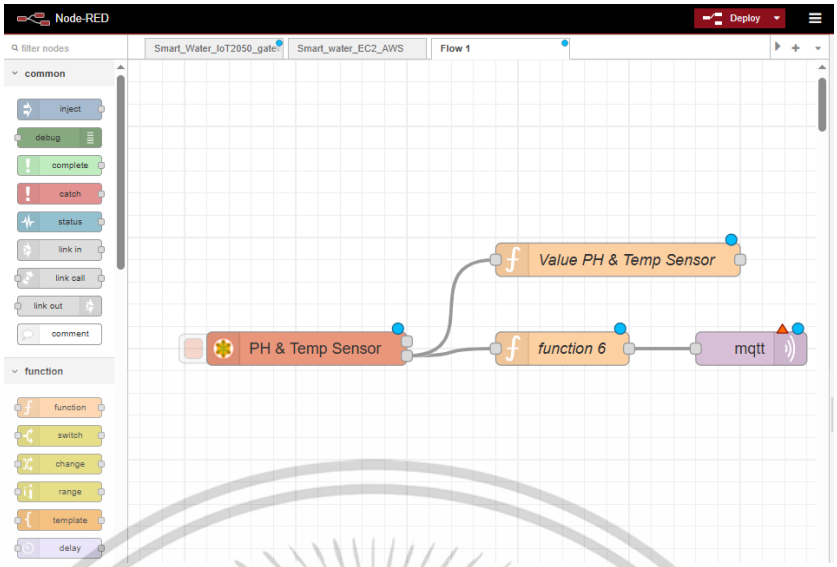


รูปที่ 4.16 การตั้งค่า Function node ในการส่งข้อมูล

หลังจากนั้นจะปรากฏหน้าต่างการตั้งค่า Function Node ให้ทำการเขียนโปรแกรมตามคำสั่งดังนี้:

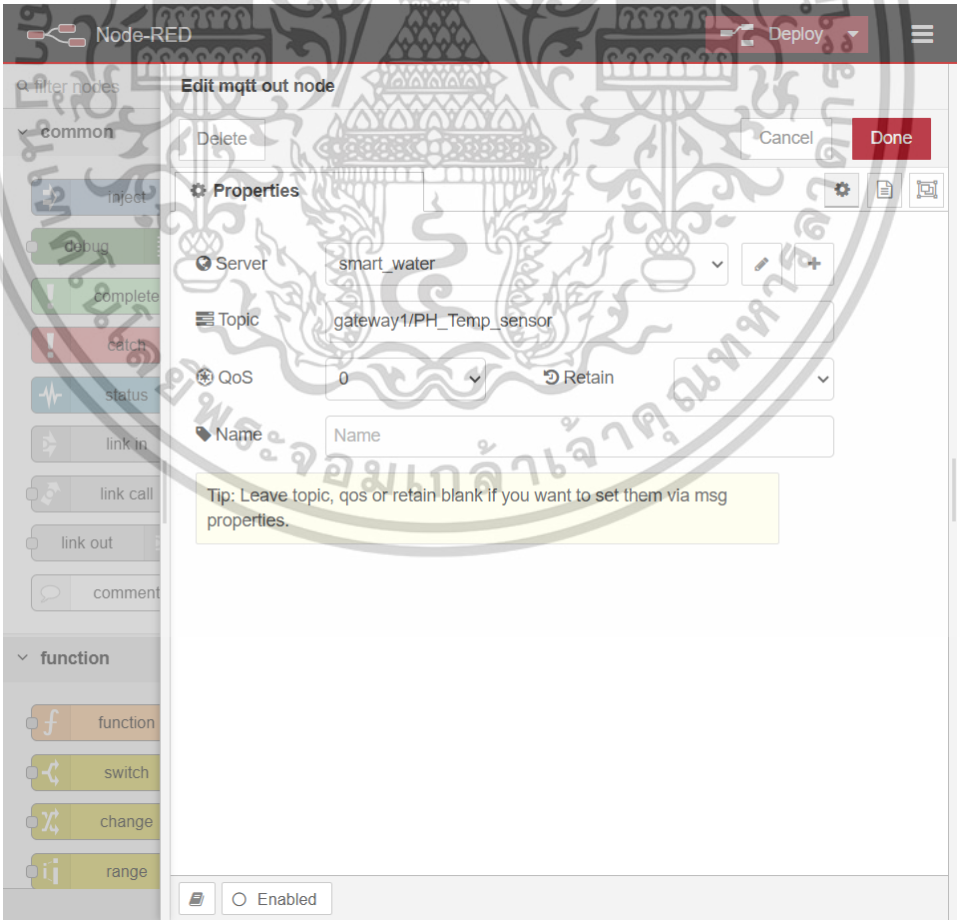
```
1)  if (global.get("PH")!=undefined && global.get("PH_Temp")!=undefined){
2)      msg.payload = {
3)          "Timestamp":msg.payload.timestamp,
4)          "DeviceName": "PH & Temp",
5)          "Data": {
6)              "PH" : global.get("PH"),
7)              "Temp" : global.get("PH_Temp")
8)          }
9)      };
10)     return msg;
11) }
```

เมื่อเขียนโปรแกรมเสร็จแล้ว ให้กดปุ่ม Done เพื่อบันทึกการตั้งค่า



รูปที่ 4.17 การเชื่อมต่อ mqtt out Node กับ Function Node ในการส่งข้อมูล

เลือก mqtt out Node จากหมวด Network Node และลากมาวางลงใน Flow จากนั้นลากสายเชื่อมต่อระหว่าง mqtt out Node กับ Function Node ที่สร้างขึ้น หลังจากนั้นทำการ Double Click ที่ mqtt out Node ที่เพิ่งสร้างขึ้นเพื่อเข้าสู่หน้าต่างการตั้งค่าและการปรับแต่ง

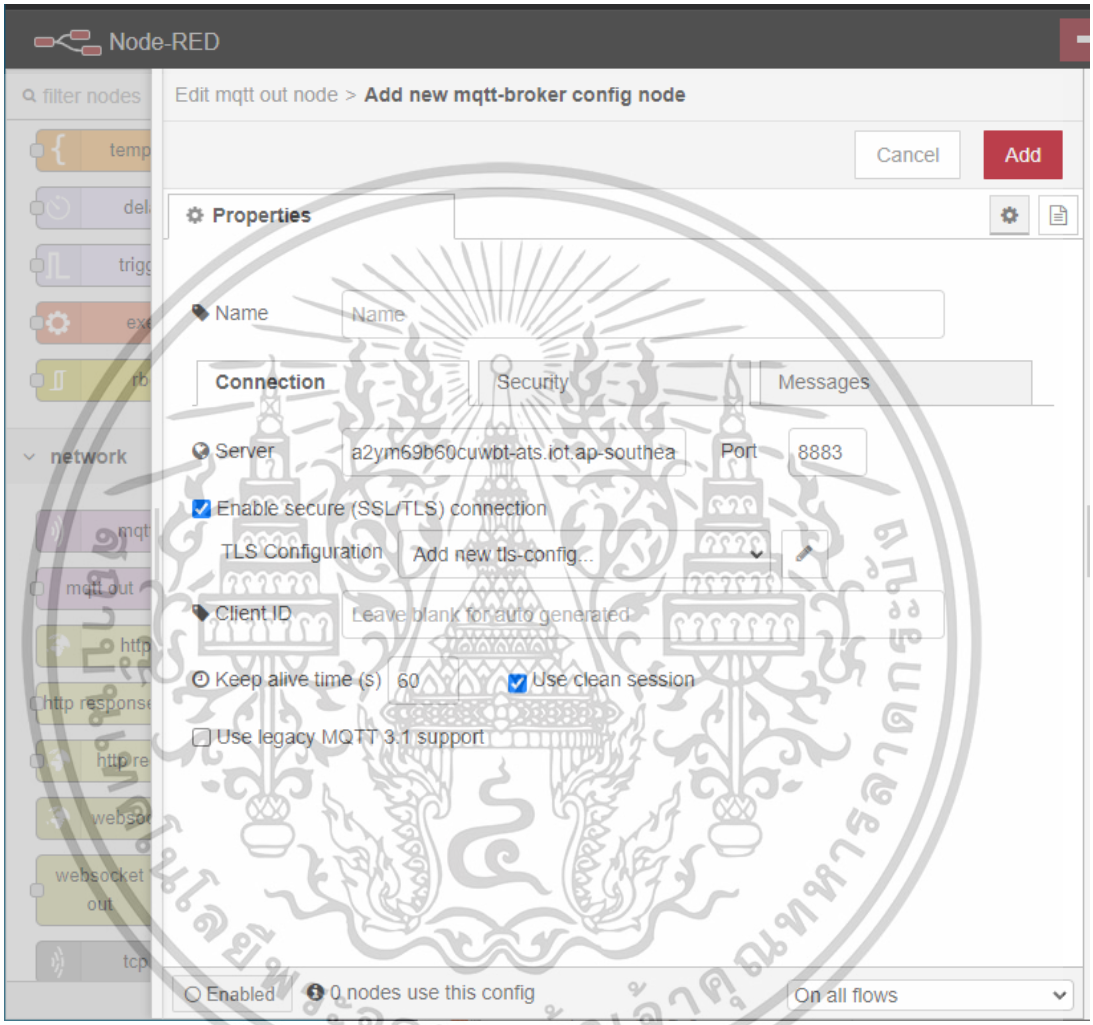


รูปที่ 4.18 การตั้งค่า mqtt out node ในการส่งข้อมูล

หลังจากนั้นจะปรากฏหน้าต่างการตั้งค่า mqtt out node ให้ทำการตั้งค่าตามรายละเอียดดังนี้:

1) Topic: gateway1/PH_Temp_sensor

เมื่อทำการตั้งค่าเสร็จแล้ว ให้กดปุ่ม แก้ไข ตรง Server เพื่อสร้างมาตรฐานการสื่อสารโดยใช้ MQTT Protocol ตามที่กำหนด

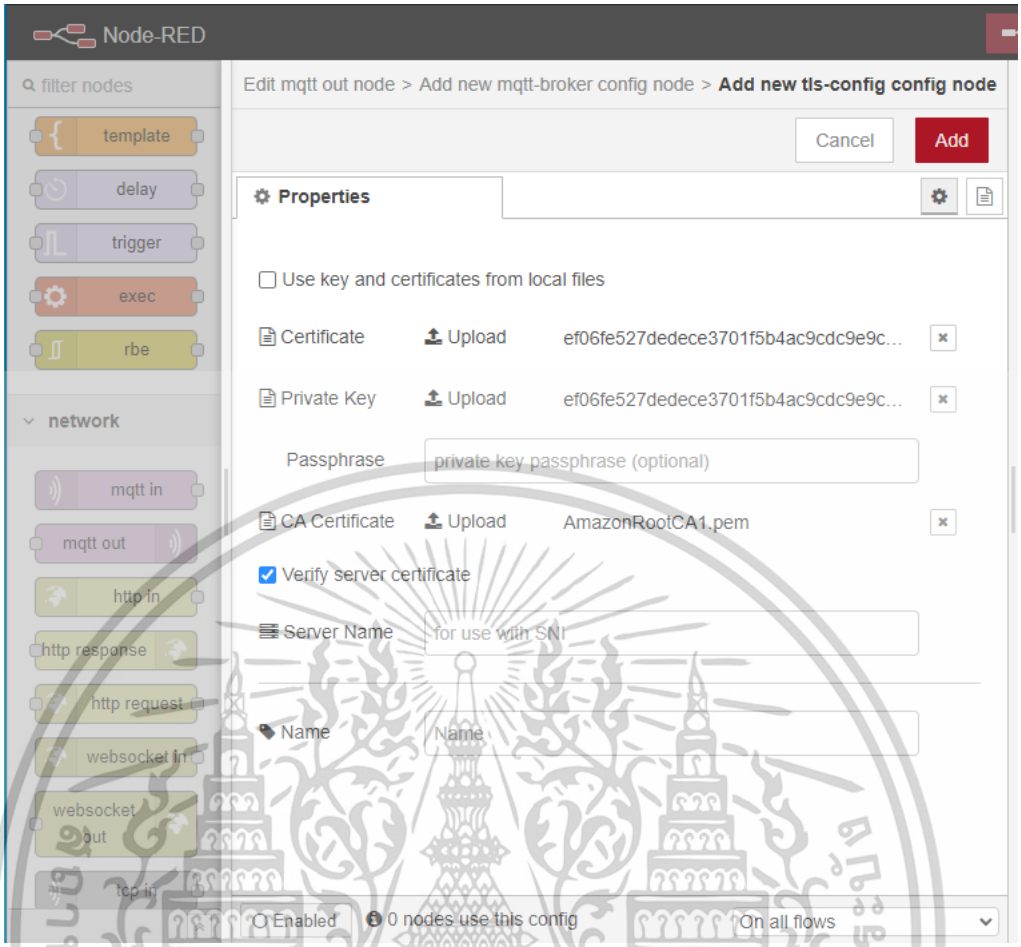


รูปที่ 4.19 การตั้งค่า mqtt-broker node ในการส่งข้อมูล

หลังจากนั้นจะปรากฏหน้าต่างการตั้งค่า mqtt-broker node ให้ทำการตั้งค่าดังนี้:

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

หลังจากนั้นให้กดปุ่ม Enable secure (SSL/TLS) connection และกดปุ่ม แก้ไข ตรง TLS Configuration เพื่อทำการยืนยันตัวตนในการเชื่อมต่อกับ AWS IoT Core

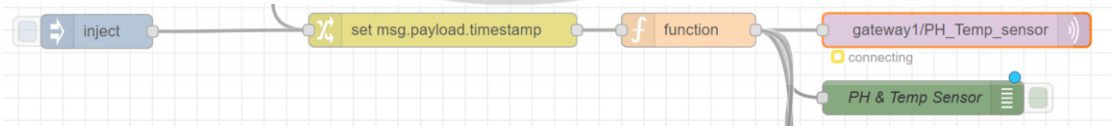


รูปที่ 4.20 การตั้งค่า tls-config node ในการส่งข้อมูลอุปกรณ์ sensor

หลังจากนั้นจะปรากฏหน้าต่างการตั้งค่า tls-config node ให้ทำการอัพโหลดไฟล์ตามรายละเอียดดังนี้:

1) Certificate
2) Private Key
3) CA Certificate

เมื่ออัพโหลดไฟล์ครบถ้วนแล้ว ให้กดปุ่ม Add เพื่อบันทึกการตั้งค่า

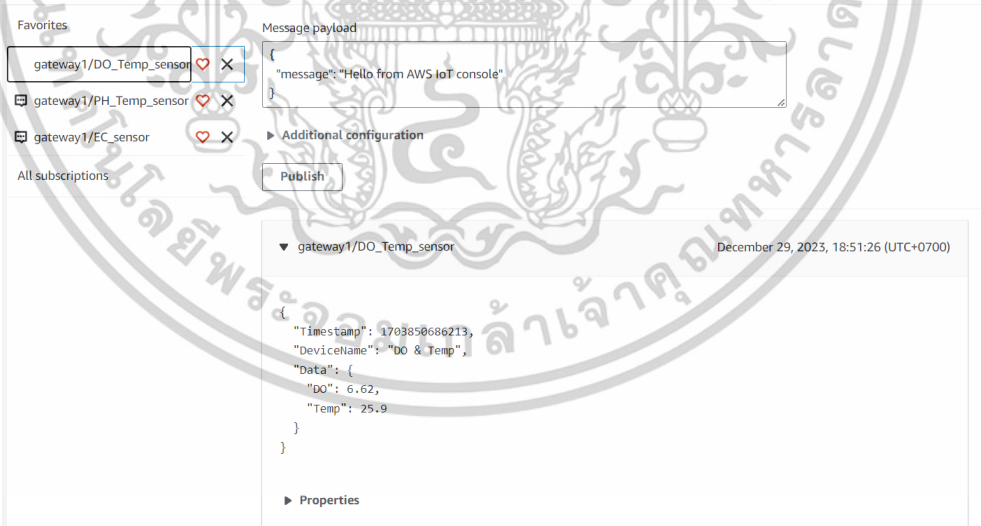


รูปที่ 4.21 การส่งข้อมูลอุปกรณ์วัดไปยัง AWS IoT Core

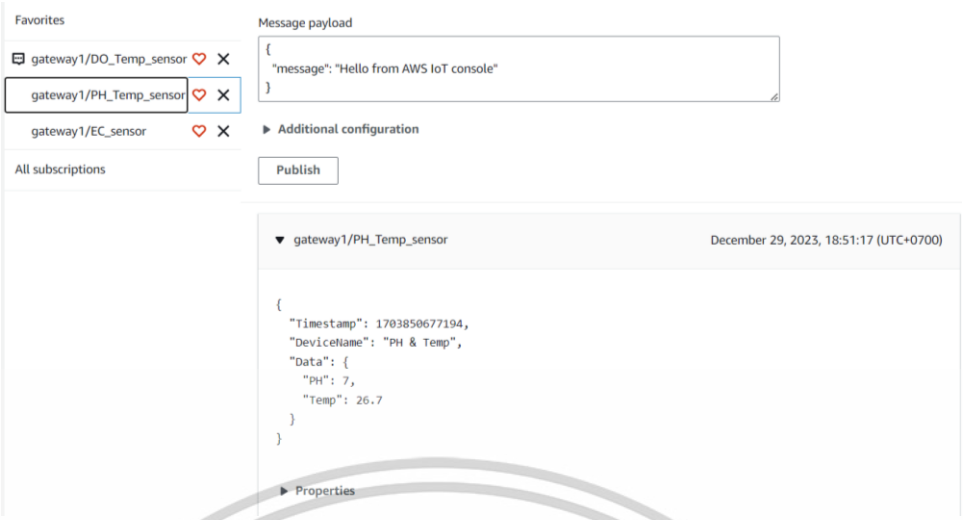
4.1.3 การทดสอบการทำงานของสถานี

การทดสอบการทำงานของสถานีตรวจวัดคุณภาพน้ำ เริ่มจากอ่านข้อมูล (Subscribe) จาก Server Broker ของ AWS IoT Core ผ่าน MQTT Protocol ที่ใน Cloud โดยเริ่มจาก DO and Temperature Sensor, PH and Temperature Sensor และ EC Sensor ตามลำดับ โดยข้อมูลประกอบไปด้วย Timestamp, DevicdeName และ Data ตามลำดับ และมีรูปแบบเป็น JavaScript Object Notation (JSON) ซึ่งเป็นมาตรฐานในการแลกเปลี่ยนข้อมูล (Data Interchange Format) แสดงไว้ในรูปที่ 4.22-4.28

รูปที่ 4.22 การทดสอบการทำงานของสถานี

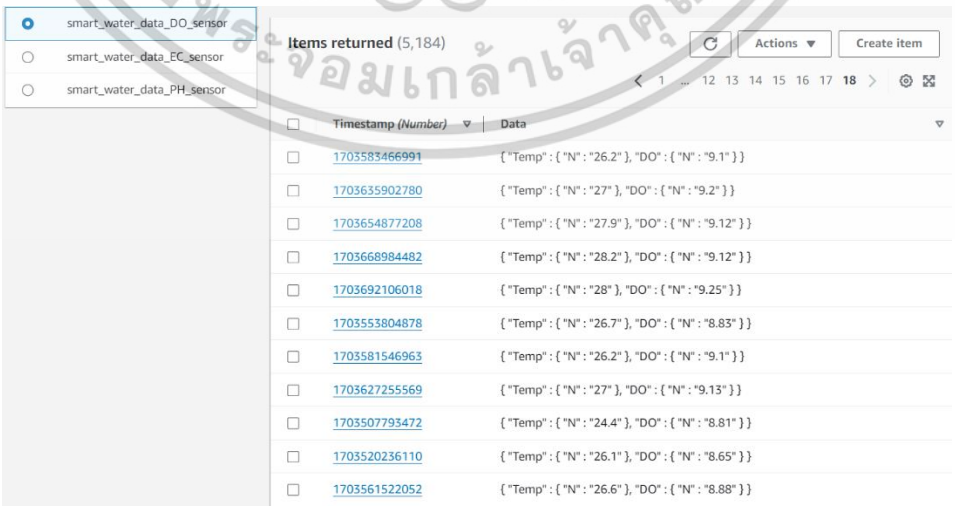


รูปที่ 4.23 ตัวอย่างข้อมูล DO and Temperature Sensor จาก AWS IoT Core

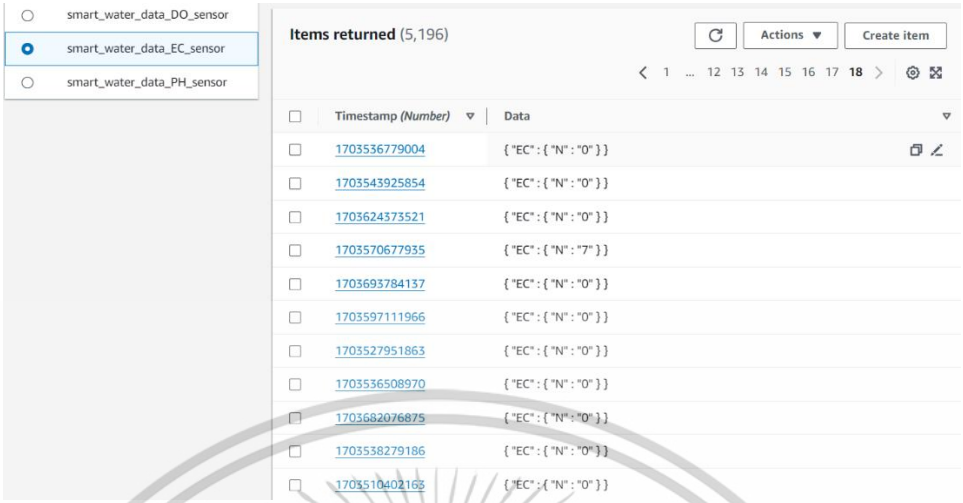


รูปที่ 4.24 ตัวอย่างข้อมูล PH and Temperature Sensor จาก AWS IoT Core

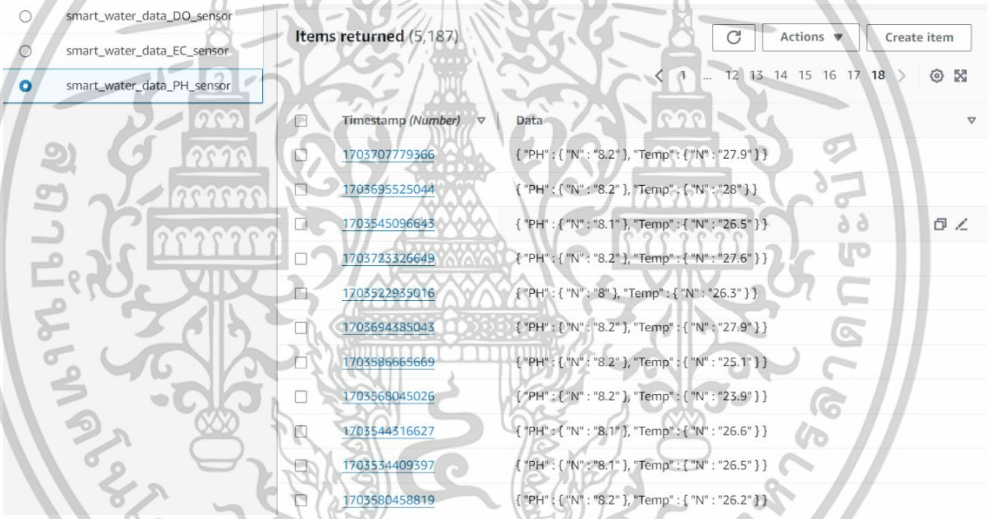
รูปที่ 4.25 ตัวอย่างข้อมูล EC Sensor จาก AWS IoT Core



รูปที่ 4.26 ตัวอย่างข้อมูล DO and Temperature Sensor จาก AWS DynamoDB

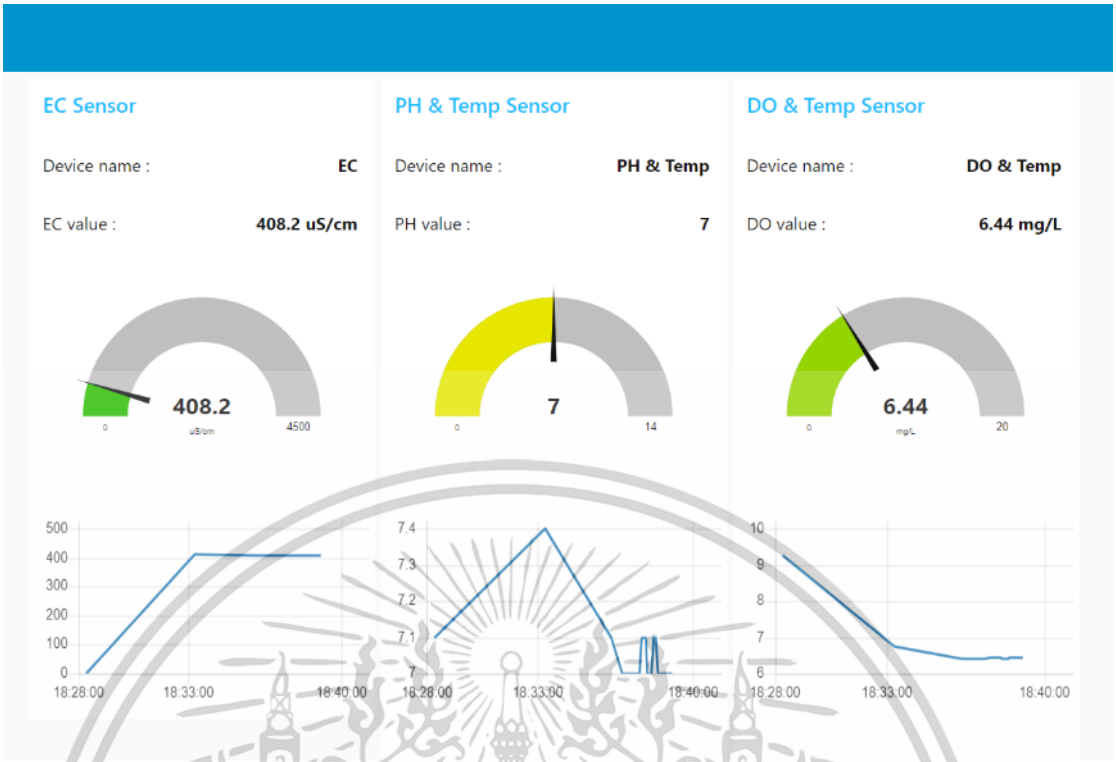


รูปที่ 4.27 ตัวอย่างข้อมูล EC Sensor จาก AWS DynamoDB



รูปที่ 4.28 ตัวอย่างข้อมูล PH and Temperature Sensor จาก AWS DynamoDB

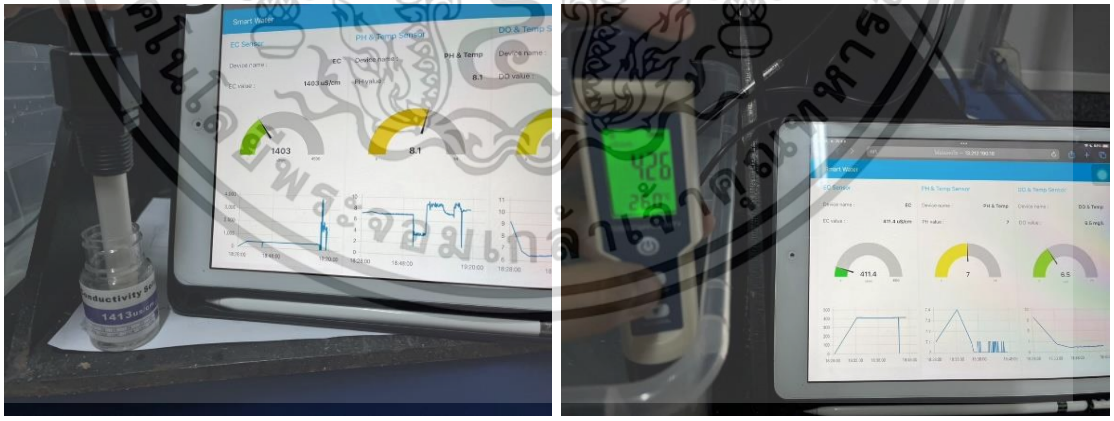
จากรูป 4.23-4.28 เป็นการทดลองอ่านข้อมูลจาก Sensor ที่เชื่อมต่อผ่านมาตรฐาน RS485 Modbus RTU และส่งข้อมูล (Publish) จาก IIoT Gateway ไปยัง Server Broker ของ AWS IoT Core ผ่าน MQTT Protocol โดยใช้ระบบอินเทอร์เน็ตผ่านเครือข่ายเซลลูลาร์ 4G เก็บข้อมูลในฐานข้อมูล AWS DynamoDB จากการเรียกใช้งานฟังก์ชั่น (Triggers) เขียนข้อมูลที่ได้รับจาก Sensor ในตู้ควบคุม ไปเก็บใน Database AWS DynamoDB ผ่านทาง Sevices ของ AWS IoT Core และแสดงข้อมูลผ่าน Dashboard บน AWS EC2 (Cloud) ที่ใช้เครื่องมือสำหรับนักพัฒนาโปรแกรมในการเชื่อมต่ออุปกรณ์ Node-red ทำงานบน Node.js ภายในระบบปฏิบัติการ Amazon Linux โดยโปรแกรมจะเริ่มการทำงานอ่านข้อมูล Sensor (Subscribe) แต่ละ Sensor จาก Server Broker ของ AWS IoT Core ผ่าน MQTT Protocol โดยเขียนฟังก์ชั่นและประมวลผลให้แสดงข้อมูลผ่าน Dashboard บน Cloud ของ AWS EC2 แสดงไว้ในรูปที่ 4.29



รูปที่ 4.29 แดชบอร์ดแสดงคุณภาพน้ำของสถานี

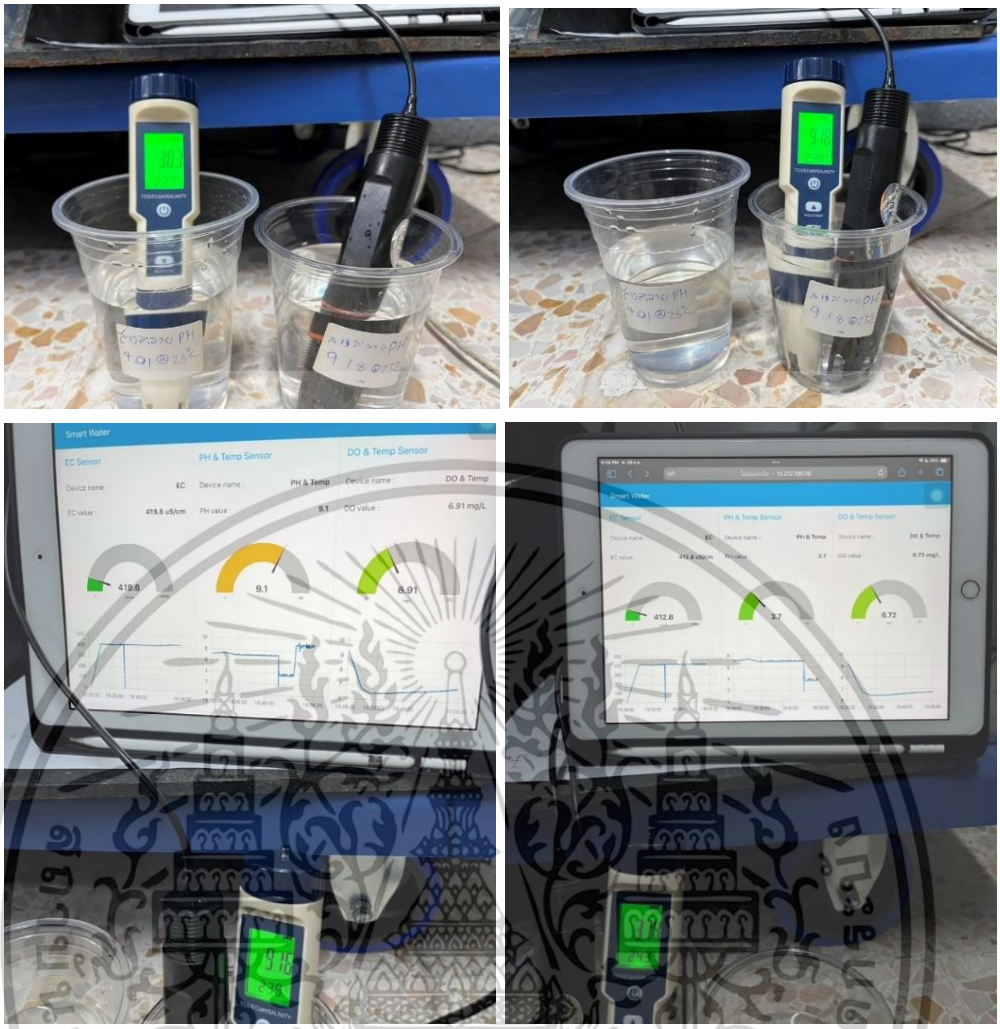
4.1.4 การทดสอบข้อมูลเปรียบเทียบกับเครื่องมือวัด

การทดสอบเปรียบเทียบระหว่างข้อมูลที่ได้จาก EC Sensor และ PH and Temperature Sensor กับเครื่องมือวัดทางวิทยาศาสตร์ที่ใช้วัดค่าความเป็นกรด-ด่างในสารละลายและความสามารถของของเหลวในการนำกระแสไฟฟ้า แสดงไว้ในรูปที่ 4.30-4.31



รูปที่ 4.30 การเปรียบเทียบเซนเซอร์วัดการนำไฟฟ้ากับเครื่องมือวัด

จากรูปที่ 4.30 เป็นการทดสอบเปรียบเทียบระหว่างข้อมูลที่ได้จาก EC Sensor กับเครื่องมือวัดทางวิทยาศาสตร์ที่ใช้วัดค่าความสามารถของของเหลวในการนำกระแสไฟฟ้าและสารละลายมาตรฐานค่าความนำไฟฟ้า 1413 uS/cm



รูปที่ 4.31 การเปรียบเทียบเซนเซอร์วัดความเป็นกรดเป็นเบสกับเครื่องมือวัด

จากรูปที่ 4.31 เป็นการทดสอบเปรียบเทียบระหว่างข้อมูลที่ได้จาก PH and Temperature Sensor กับเครื่องมือวัดทางวิทยาศาสตร์ที่ใช้วัดค่าความเป็นกรด-ด่างและอุณหภูมิในน้ำและสารละลายบัฟเฟอร์กรด-ด่าง

4.2 กาพัฒนาและการทดสอบระบบการพยากรณ์

4.2.1 ชุดข้อมูล

จากการศึกษานี้ใช้ชุดข้อมูลคุณภาพน้ำของแม่น้ำเจ้าพระยา ณ สถานีวัดมะขาม ย้อนหลัง 10 ปี จากหน่วยงานการประปานครหลวง [51] โดยมีพารามิเตอร์ได้แก่ วัน-เวลา อุณหภูมิของน้ำ การนำไฟฟ้าของน้ำ ปริมาณของแข็งที่ละลายเจือปนอยู่ในน้ำ ปริมาณของเกลือที่ละลายในน้ำ ปริมาณออกซิเจนที่มีอยู่ในน้ำ ความลึก ความเป็นกรด-เบส ปริมาณคลอโรฟิลล์ และความขุ่นของน้ำ ตามลำดับ โดยตัวอย่างชุดข้อมูลคุณภาพน้ำแม่น้ำเจ้าพระยาสถานีวัดมะขามของการศึกษาที่นำเสนอนี้แสดงไว้ในรูปที่ 4.32

date_time	Temper...	Conduct...	TDS (m...	Salinity ...	DO (mg/L)	Sensor ...	pH	Chlorop...	Turbid (...
2014-07-05T23:20:00	31.59	330	210	0.16	4.54	1.25	7.15	13.3	17.4
2014-07-05T23:30:00	31.6	330	220	0.16	4.48	1.26	7.15	12.1	18.9
2014-07-05T23:40:00	31.6	333	220	0.16	4.44	1.26	7.14	12.1	16.7
2014-07-05T23:50:00	31.6	334	220	0.16	4.27	1.27	7.11	12.6	15.6
2014-07-06T00:00:00	31.62	338	220	0.16	4.25	1.28	7.11	14	16.9
2014-07-06T00:10:00	31.61	340	220	0.16	4.15	1.28	7.1	13.9	17.9
2014-07-06T00:20:00	31.6	341	220	0.16	3.99	1.29	7.07	13	15.9
2014-07-06T00:30:00	31.59	345	220	0.16	3.92	1.29	7.06	14.4	18.5
2014-07-06T00:40:00	31.57	346	230	0.16	3.8	1.3	7.05	15.1	18
2014-07-06T00:50:00	31.56	348	230	0.16	3.62	1.3	7.03	14	17.7
2014-07-06T01:00:00	31.54	349	230	0.16	3.49	1.31	7.01	13.9	16
2014-07-06T01:10:00	31.53	351	230	0.17	3.35	1.31	7	13.1	16.7
2014-07-06T01:20:00	31.52	354	230	0.17	3.24	1.31	6.99	13.4	18.1
2014-07-06T01:30:00	31.51	357	230	0.17	3.15	1.31	6.98	13.6	15.4
2014-07-06T01:40:00	31.5	358	230	0.17	3.04	1.32	6.97	13.6	15.1
2014-07-06T01:50:00	31.5	362	240	0.17	2.97	1.32	6.96	13.2	16.4
2014-09-30T03:30:00	0	0	0	0	0	0	0	0	0
2014-07-06T02:00:00	31.49	365	240	0.17	2.91	1.32	6.96	13.3	14.9
2014-07-06T02:10:00	31.49	366	240	0.17	2.89	1.32	6.95	12.9	14.3
2014-07-06T02:20:00	31.49	371	240	0.18	2.86	1.32	6.95	13	14.6
2014-07-06T02:30:00	31.49	373	240	0.18	2.79	1.32	6.94	13.7	15.2
2014-07-06T02:40:00	31.49	376	250	0.18	2.8	1.31	6.95	12.3	10.6
2014-07-06T02:50:00	31.48	380	250	0.18	2.81	1.31	6.94	12.1	13.6

รูปที่ 4.32 ตัวอย่างชุดข้อมูลคุณภาพน้ำแม่น้ำเจ้าพระยาสถานีวัดมะขาม

4.2.2 การจัดรูปแบบชุดข้อมูลที่จะศึกษา

การจัดรูปแบบชุดข้อมูลเริ่มจากนำพารามิเตอร์ที่ไม่ได้ศึกษาออกได้แก่ ปริมาณของแข็งที่ละลายเจือปนอยู่ในน้ำ ปริมาณของเกลือที่ละลายในน้ำ ความลึก ปริมาณคลอโรฟิลล์ และความขุ่นของน้ำ ตามลำดับ ให้เหลือแค่ วัน-เวลา อุณหภูมิของน้ำ การนำไฟฟ้าของน้ำ ปริมาณออกซิเจนที่มีอยู่ในน้ำ และความเป็นกรด-เบส ตามลำดับ และตรวจสอบค่าข้อมูลที่เป็นไปไม่ได้สำหรับทุกพารามิเตอร์และบางพารามิเตอร์ เช่น ค่า 0, ค่าว่าง, ค่าติดลบ และค่าที่เกินความเป็นจริงที่จะเป็นได้ โดยแสดงไว้ในรูปที่ 4.33-4.36

```
df = df.drop(['TDS\n(mg/L)'], axis=1)
df = df.drop(['Salinity\n(g/L)'], axis=1)
df = df.drop(['Sensor Depth\n(m)'], axis=1)
df = df.drop(['Chlorophyll\n(ug/L)'], axis=1)
df = df.drop(['Turbid\n(NTU)'], axis=1)
```

รูปที่ 4.33 การนำพารามิเตอร์ที่ไม่ได้ศึกษาออก

```
df.describe()
```

	date_time	Temperature\n(C)	Conductivity\n(uS/cm)	DO\n(mg/L)	pH
count	538753	428220.000000	428241.000000	423665.000000	415368.000000
mean	2019-06-12 06:39:59.999998464	30.187435	462.158614	2.691163	7.193080
min	2014-04-28 14:40:00	0.000000	-3999.000000	0.000000	-26.300000
25%	2016-11-18 22:40:00	29.190000	297.000000	1.730000	7.130000
50%	2019-06-12 06:40:00	30.570000	358.000000	2.760000	7.240000
75%	2022-01-02 14:40:00	31.420000	442.000000	3.620000	7.350000
max	2024-07-25 22:40:00	99.990000	7816.000000	11.470000	50.840000
std	NaN	2.339895	462.215793	1.578567	0.650542

รูปที่ 4.34 ข้อมูลโดยรวมของพารามิเตอร์ก่อนการจัดรูปแบบ

```
neg9999_counts = (df == -9999).sum()
print(neg9999_counts)
```

```
date_time              0
Temperature\n(C)    4679
Conductivity\n(uS/cm)    4658
DO\n(mg/L)          9234
pH                 17531
dtype: int64
```

```
df = df.replace(-9999, np.nan)
```

```
zero_counts = (df == 0).sum()
print(zero_counts)
```

```
date_time              0
Temperature\n(C)     597
Conductivity\n(uS/cm)    2263
DO\n(mg/L)          42510
pH                  2794
dtype: int64
```

```
df = df.replace(0, np.nan)
```

```
zero_counts = (df == 50.840000).sum()
print(zero_counts)
```

```
date_time              0
Temperature\n(C)       0
Conductivity\n(uS/cm)    0
DO\n(mg/L)             0
pH                     1
dtype: int64
```

```
df = df.replace(50.840000, np.nan)
```

```
zero_counts = (df == -26.300000).sum()
print(zero_counts)
```

```
date_time              0
Temperature\n(C)       0
Conductivity\n(uS/cm)    0
DO\n(mg/L)             0
pH                     1
dtype: int64
```

```
df = df.replace(-26.300000, np.nan)
```

```
zero_counts = (df == -3999.000000).sum()
print(zero_counts)
```

```
date_time              0
Temperature\n(C)       0
Conductivity\n(uS/cm)     87
DO\n(mg/L)             0
pH                     0
dtype: int64
```

```
df = df.replace(-3999.000000, np.nan)
```

```
zero_counts = (df == 99.990000).sum()
print(zero_counts)
```

```
date_time              0
Temperature\n(C)     105
Conductivity\n(uS/cm)       0
DO\n(mg/L)             0
pH                     0
dtype: int64
```

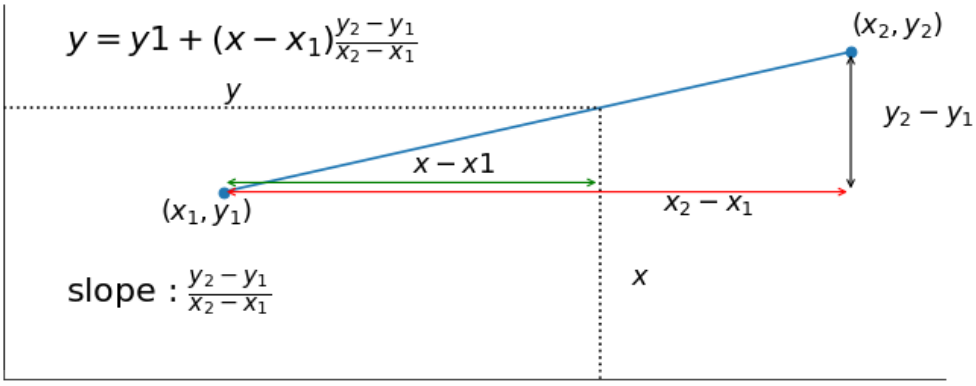
```
df = df.replace(99.990000, np.nan)
```

รูปที่ 4.35 ตัวอย่างการตรวจสอบค่าข้อมูลที่เป็นไปไม่ได้สำหรับทุกพารามิเตอร์

```
df['Conductivity\n(uS/cm)'][df['Conductivity\n(uS/cm)'] < 100] = np.nan
df['pH'][df['pH'] < 1] = np.nan
df['pH'][df['pH'] > 14] = np.nan
```

รูปที่ 4.36 ตัวอย่างการตรวจสอบค่าข้อมูลที่เป็นไปไม่ได้สำหรับบางพารามิเตอร์

หลังจากการจัดรูปแบบชุดข้อมูลเริ่มจากนำพารามิเตอร์ที่ไม่ได้ศึกษาออกและตรวจสอบค่าข้อมูลที่เป็นไปไม่ได้สำหรับทุกพารามิเตอร์และบางพารามิเตอร์ ยังมีข้อมูลที่ขาดหายไปทำให้ต้องใช้หลักการการประมาณค่าในช่วงโดยใช้ค่าเฉลี่ยของข้อมูลก่อน-หลัง โดยแสดงไว้ในรูปที่ 4.37-4.38



รูปที่ 4.37 หลักการการประมาณค่าในช่วงโดยใช้ค่าเฉลี่ยของข้อมูลก่อน-หลัง

```
df['Temperature\n(C)'].interpolate(method='linear', inplace=True)
df['Conductivity\n(uS/cm)'].interpolate(method='linear', inplace=True)
df['DO\n(mg/L)'].interpolate(method='linear', inplace=True)
df['pH'].interpolate(method='linear', inplace=True)
```

รูปที่ 4.38 ตัวอย่างการใช้การประมาณค่าในช่วงโดยใช้ค่าเฉลี่ยในพารามิเตอร์

หลังจากการใช้หลักการการประมาณค่าในช่วงโดยใช้ค่าเฉลี่ยของข้อมูลก่อน-หลัง จะทำให้ได้ชุดข้อมูลที่พร้อมที่จะนำไปศึกษาแบบจำลอง LSTM และ XGBoost โดยแสดงไว้ในรูปที่ 4.39 นอกจากนี้แต่ละแบบจำลองมีการใช้เทคนิคการจัดรูปแบบข้อมูลไม่เหมือนกัน เช่น LSTM การสร้างชุดข้อมูลความสัมพันธ์ระหว่าง x และ y ส่วน XGBoost เพิ่มความสัมพันธ์กับฤดูกาลหรือวันหยุด การทำ Time-based Interaction เป็นต้น เพื่อเพิ่มประสิทธิภาพให้แต่ละแบบจำลองสามารถคาดการและพยากรณ์ได้แม่นยำและถูกต้องมากขึ้น

	date_time	Temperature\n(C)	Conductivity\n(uS/cm)	DO\n(mg/L)	pH
count	538753	538753.000000	538753.000000	538753.000000	538753.000000
mean	2019-06-12 06:39:59.999998464	30.083002	501.199898	2.964049	7.236434
min	2014-04-28 14:40:00	12.830000	120.000000	0.010000	1.000000
25%	2016-11-18 22:40:00	29.151753	315.000000	2.290000	7.160000
50%	2019-06-12 06:40:00	30.200000	381.654009	2.905802	7.260000
75%	2022-01-02 14:40:00	31.250000	523.263959	3.490000	7.340000
max	2024-07-25 22:40:00	36.730000	7816.000000	11.470000	9.690000
std	NaN	1.616849	425.558691	1.217890	0.353856

รูปที่ 4.39 ข้อมูลโดยรวมของพารามิเตอร์หลังการจัดรูปแบบ

และทำการเรียงข้อมูลลำดับเวลาโดยปรับขนาดข้อมูลให้อยู่ในช่วง 0–1 (Min-Max Normalization) โดยแสดงไว้ในรูปที่ 4.40

```
scaler = MinMaxScaler(feature_range=(0, 1))
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1))
```

```
X = pd.DataFrame(X_scaled, columns=X.columns, index=X.index)
y = pd.DataFrame(y_scaled, columns=[y.name], index=y.index)
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(df[['Temperature\n(C)']])
```

รูปที่ 4.40 ตัวอย่างการใช้ Min-Max Normalization

4.2.2.1 การจัดรูปแบบชุดข้อมูลที่จะศึกษาเพิ่มเติมในแบบจำลอง LSTM

การจัดรูปแบบชุดข้อมูลที่จะศึกษาในแบบจำลอง LSTM จะเปลี่ยนแปลงพารามิเตอร์ของวัน-เวลา ที่เป็นข้อความ ให้เป็นรูปแบบ วัน-เวลา โดยแสดงไว้ในรูปที่ 4.41-4.42

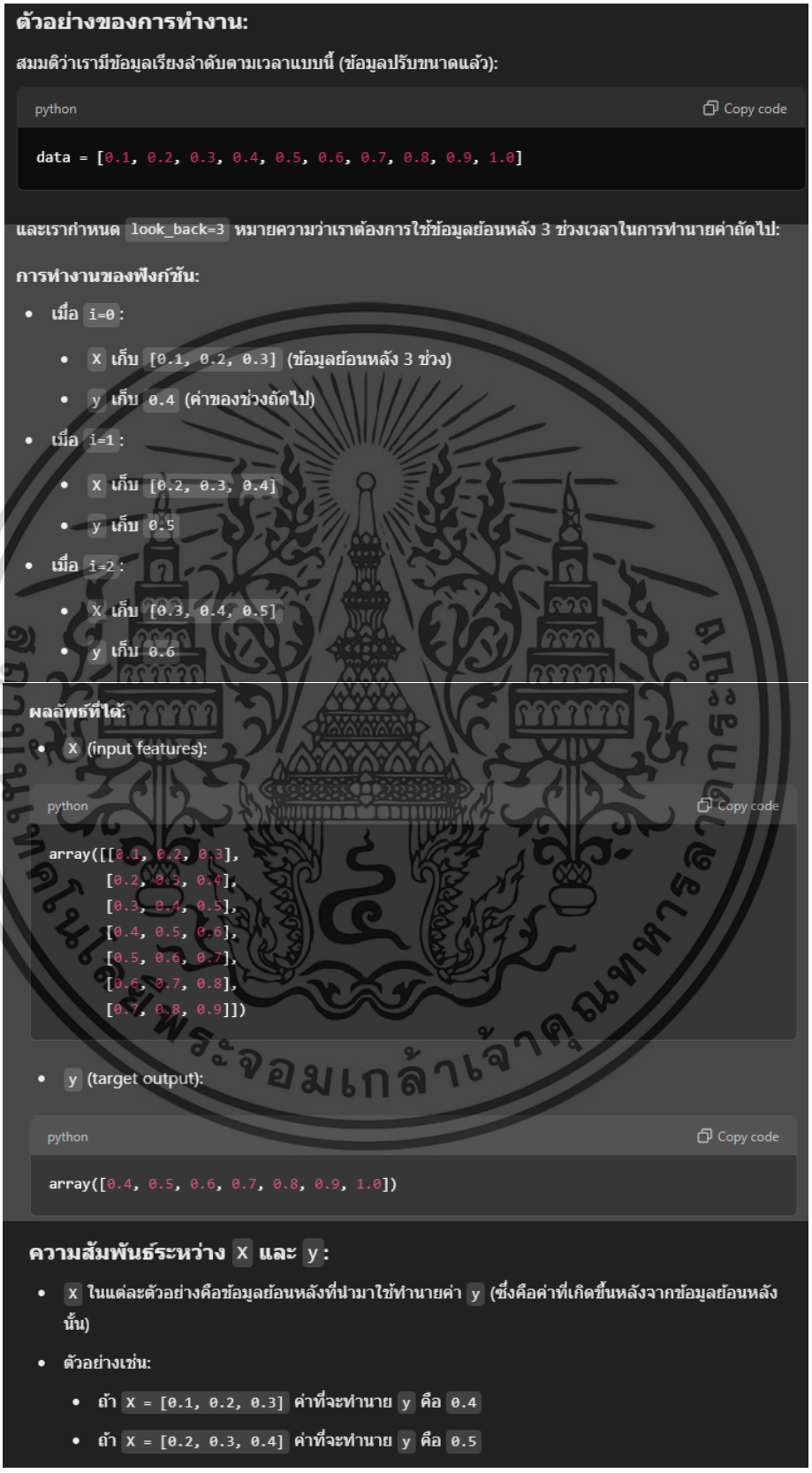
```
df['date_time'] = pd.to_datetime(df['date_time'])

df = df.set_index('date_time')

df = df.resample('D').mean()
```

รูปที่ 4.41 การเปลี่ยนแปลงพารามิเตอร์ของ วัน-เวลา ที่เป็นข้อความ ให้เป็นรูปแบบ วัน-เวลา

date_time	Temperature\n(C)	Conductivity\n(uS/cm)	DO\n(mg/L)	pH
2014-04-28	29.038523	501.199898	7.757047	9.276510
2014-04-29	29.433154	501.199898	7.346309	9.045638
2014-04-30	30.001423	501.199898	6.754846	8.713181
2014-05-01	30.569691	501.199898	6.163383	8.380725
2014-05-02	31.137960	501.199898	5.571919	8.048268
...	...	...	...	...
2024-07-21	31.365139	351.027778	2.146181	3.080000
2024-07-22	31.287986	345.090278	2.114375	3.080000
2024-07-23	31.137361	338.798611	2.126111	3.080000
2024-07-24	30.967847	322.312500	2.622986	3.080000
2024-07-25	30.821606	288.642336	3.139635	3.080000

รูปที่ 4.42 ข้อมูลโดยรวมของพารามิเตอร์หลังเปลี่ยนแปลงพารามิเตอร์ของรูปแบบวัน-เวลา

และสร้างฟังก์ชันเพื่อกำหนด Lookback การใช้ข้อมูลย้อนหลัง 7 ช่วงเวลาในการพยากรณ์ค่าถัดไป และความสัมพันธ์ระหว่าง x และ y โดยมีหลักการทำงานแสดงดังรูปที่ 4.43-4.45



รูปที่ 4.43 ตัวอย่างของการทำงานเรียงข้อมูลลำดับเวลาและความสัมพันธ์ของ x และ y

```python
def create_dataset(data, look_back=1):
    X, y = [], []
    for i in range(len(data) - look_back):
        X.append(data[i:(i + look_back), 0])
        y.append(data[i + look_back, 0])
    return np.array(X), np.array(y)
```

```python
look_back = 7 #ย้อนหลัง 7 วันมาเทรนข้อมูล
X, y = create_dataset(data_scaled, look_back)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))
```

รูปที่ 4.44 การสร้างฟังก์ชันกำหนด Lookback และความสัมพันธ์ระหว่าง x และ y

```python
print("X:")
print(X[:5])

# แสดงค่า y แถวแรก 5 แถว
print("\n y:")
print(y[:5])
```

```
X:
[[[0.62912608]
  [0.65733737]
  [0.69796161]
  [0.73858586]
  [0.77921011]
  [0.81784608]
  [0.81894232]]

 [[0.65733737]
  [0.69796161]
  [0.73858586]
  [0.77921011]
  [0.81784608]
  [0.81894232]
  [0.82281457]]

 [[0.69796161]
  [0.73858586]
  [0.77921011]
  [0.81784608]
  [0.81894232]
  [0.82281457]
  [0.82653541]]

 [[0.73858586]
  [0.77921011]
  [0.81784608]
  [0.81894232]
  [0.82281457]
  [0.82653541]
  [0.82480531]]

 [[0.77921011]
  [0.81784608]
  [0.81894232]
  [0.82281457]
  [0.82653541]
  [0.82480531]
  [0.83715681]]]

y:
[0.82281457 0.82653541 0.82480531 0.83715681 0.84707574]
```

รูปที่ 4.45 ตัวอย่างความสัมพันธ์ของ x และ y

4.2.2.2 การจัดรูปแบบชุดข้อมูลที่จะศึกษาเพิ่มเติมในแบบจำลอง XGBoost

การจัดรูปแบบชุดข้อมูลที่จะศึกษาในแบบจำลอง XGBoost จะแยกพารามิเตอร์วัน-เวลาออก โดยแสดงไว้ในรูปที่ 4.46

```
df['date_time'] = pd.to_datetime(df['date_time'])
```

```
df['day'] = df['date_time'].dt.day
df['month'] = df['date_time'].dt.month
df['year'] = df['date_time'].dt.year
```

```
df
```

	date_time	Temperature\n(C)	Conductivity\n(uS/cm)	DO\n(mg/L)	pH	day	month	year
0	2014-04-28 14:40:00	28.930000	501.199898	7.870000	9.340000	28	4	2014
1	2014-04-28 14:50:00	28.933946	501.199898	7.865893	9.337691	28	4	2014
2	2014-04-28 15:00:00	28.937893	501.199898	7.861785	9.335383	28	4	2014
3	2014-04-28 15:10:00	28.941839	501.199898	7.857678	9.333074	28	4	2014
4	2014-04-28 15:20:00	28.945785	501.199898	7.853570	9.330765	28	4	2014
...	...	...	...	...	...	...	...	...
538748	2024-07-25 22:00:00	30.720000	260.000000	3.270000	3.080000	25	7	2024
538749	2024-07-25 22:10:00	30.710000	260.000000	3.350000	3.080000	25	7	2024
538750	2024-07-25 22:20:00	30.710000	258.000000	3.240000	3.080000	25	7	2024
538751	2024-07-25 22:30:00	30.710000	258.000000	3.300000	3.080000	25	7	2024
538752	2024-07-25 22:40:00	30.720000	258.000000	3.330000	3.080000	25	7	2024

538753 rows × 8 columns

รูปที่ 4.46 ตัวอย่างการแยกพารามิเตอร์วัน-เวลา

ทำการเพิ่มความสัมพันธ์กับฤดูกาลหรือวันหยุด เพื่อช่วยให้แบบจำลองสามารถจับพฤติกรรมที่อาจจะเกิดในช่วงวัน-เวลาต่างๆของปี โดยแสดงไว้ในรูปที่ 4.47

```
df['season'] = df['month'].apply(lambda x: (x%12 + 3)//3)
```

```
df
```

	date_time	Temperature\n(C)	Conductivity\n(uS/cm)	DO\n(mg/L)	pH	day	month	year	season
0	2014-04-28 14:40:00	28.930000	501.199898	7.870000	9.340000	28	4	2014	2
1	2014-04-28 14:50:00	28.933946	501.199898	7.865893	9.337691	28	4	2014	2
2	2014-04-28 15:00:00	28.937893	501.199898	7.861785	9.335383	28	4	2014	2
3	2014-04-28 15:10:00	28.941839	501.199898	7.857678	9.333074	28	4	2014	2
4	2014-04-28 15:20:00	28.945785	501.199898	7.853570	9.330765	28	4	2014	2
...	...	...	...	...	...	...	...	...	...
538748	2024-07-25 22:00:00	30.720000	260.000000	3.270000	3.080000	25	7	2024	3
538749	2024-07-25 22:10:00	30.710000	260.000000	3.350000	3.080000	25	7	2024	3
538750	2024-07-25 22:20:00	30.710000	258.000000	3.240000	3.080000	25	7	2024	3
538751	2024-07-25 22:30:00	30.710000	258.000000	3.300000	3.080000	25	7	2024	3
538752	2024-07-25 22:40:00	30.720000	258.000000	3.330000	3.080000	25	7	2024	3

538753 rows × 9 columns

รูปที่ 4.47 ตัวอย่างการเพิ่มความสัมพันธ์กับฤดูกาลหรือวันหยุด

ทำการเพิ่ม Time-based Interaction เป็นการเพิ่มความสัมพันธ์ระหว่างชั่วโมงกับเดือน หรือวันกับดือน เพื่อช่วยให้แบบจำลองสามารถจับพฤติกรรมของพารามิเตอร์ในช่วงเวลาต่างๆ โดยแสดงไว้ในรูปที่ 4.48

```
df['day_month_interaction'] = df['day'] * df['month']
```

```
df = df.set_index('date_time')
df = df.resample('D').mean()
```

```
df
```

date_time	Temperature\n(C)	Conductivity\n(uS/cm)	DO\n(mg/L)	pH	day	month	year	season	day_month_interaction
2014-04-28	29.038523	501.199898	7.757047	9.276510	28.0	4.0	2014.0	2.0	112.0
2014-04-29	29.433154	501.199898	7.346309	9.045638	29.0	4.0	2014.0	2.0	116.0
2014-04-30	30.001423	501.199898	6.754846	8.713181	30.0	4.0	2014.0	2.0	120.0
2014-05-01	30.569691	501.199898	6.163383	8.380725	1.0	5.0	2014.0	2.0	5.0
2014-05-02	31.137960	501.199898	5.571919	8.048268	2.0	5.0	2014.0	2.0	10.0
...	...	...	...	...	...	...	...	...	...
2024-07-21	31.365139	351.027778	2.146181	3.080000	21.0	7.0	2024.0	3.0	147.0
2024-07-22	31.287986	345.090278	2.114375	3.080000	22.0	7.0	2024.0	3.0	154.0
2024-07-23	31.137361	338.798611	2.126111	3.080000	23.0	7.0	2024.0	3.0	161.0
2024-07-24	30.967847	322.312500	2.622986	3.080000	24.0	7.0	2024.0	3.0	168.0
2024-07-25	30.821606	288.642336	3.139635	3.080000	25.0	7.0	2024.0	3.0	175.0

3742 rows × 9 columns

รูปที่ 4.48 ตัวอย่างการเพิ่ม Time-based Interaction

และกำหนดการใช้ข้อมูลย้อนหลัง 7 ช่วงข้อมูลในการพยากรณ์ค่าถัดไป โดยแสดงไว้ในรูปที่ 4.49-4.50

```
df['temp_lag_1'] = df['Temperature\n(C)'].shift(1)
df['temp_lag_2'] = df['Temperature\n(C)'].shift(2)
df['temp_lag_3'] = df['Temperature\n(C)'].shift(3)
df['temp_lag_4'] = df['Temperature\n(C)'].shift(4)
df['temp_lag_5'] = df['Temperature\n(C)'].shift(5)
df['temp_lag_6'] = df['Temperature\n(C)'].shift(6)
df['temp_lag_7'] = df['Temperature\n(C)'].shift(7)
```

```
df['DO_lag_1'] = df['DO\n(mg/L)'].shift(1)
df['DO_lag_2'] = df['DO\n(mg/L)'].shift(2)
df['DO_lag_3'] = df['DO\n(mg/L)'].shift(3)
df['DO_lag_4'] = df['DO\n(mg/L)'].shift(4)
df['DO_lag_5'] = df['DO\n(mg/L)'].shift(5)
df['DO_lag_6'] = df['DO\n(mg/L)'].shift(6)
df['DO_lag_7'] = df['DO\n(mg/L)'].shift(7)
```

```
df['EC_lag_1'] = df['Conductivity\n(uS/cm)'].shift(1)
df['EC_lag_2'] = df['Conductivity\n(uS/cm)'].shift(2)
df['EC_lag_3'] = df['Conductivity\n(uS/cm)'].shift(3)
df['EC_lag_4'] = df['Conductivity\n(uS/cm)'].shift(4)
df['EC_lag_5'] = df['Conductivity\n(uS/cm)'].shift(5)
df['EC_lag_6'] = df['Conductivity\n(uS/cm)'].shift(6)
df['EC_lag_7'] = df['Conductivity\n(uS/cm)'].shift(7)
```

```
df['pH_lag_1'] = df['pH'].shift(1)
df['pH_lag_2'] = df['pH'].shift(2)
df['pH_lag_3'] = df['pH'].shift(3)
df['pH_lag_4'] = df['pH'].shift(4)
df['pH_lag_5'] = df['pH'].shift(5)
df['pH_lag_6'] = df['pH'].shift(6)
df['pH_lag_7'] = df['pH'].shift(7)
```

รูปที่ 4.49 การกำหนดการใช้ข้อมูลย้อนหลัง 7 ช่วงข้อมูล

```
df.dropna(inplace=True)
```

```
df.describe()
```

temp_lag_1	...	DO_lag_5	DO_lag_6	DO_lag_7	pH_lag_1	pH_lag_2	pH_lag_3	pH_lag_4	pH_lag_5	pH_lag_6	
3735.000000	...	3735.000000	3735.000000	3735.000000	3735.000000	3735.000000	3735.000000	3735.000000	3735.000000	3735.000000	37
30.082278	...	2.963094	2.964452	2.965889	7.235749	7.236989	7.238320	7.239739	7.241247	7.242844	
1.605182	...	1.108521	1.110782	1.113507	0.334666	0.327774	0.320908	0.314164	0.307642	0.301456	
20.238053	...	0.233403	0.233403	0.233403	2.082611	2.082611	2.082611	2.082611	2.082611	2.082611	
29.155264	...	2.333437	2.333933	2.333933	7.160399	7.160450	7.160693	7.160972	7.161042	7.161042	
30.205035	...	2.907847	2.908403	2.908436	7.261667	7.261701	7.261806	7.261883	7.262083	7.262099	
31.246771	...	3.466076	3.469230	3.470827	7.339826	7.339896	7.339966	7.340035	7.340143	7.340234	
34.226458	...	7.837917	7.837917	7.837917	8.270000	8.270000	8.270000	8.380725	8.713181	9.045638	

รูปที่ 4.50 ข้อมูลโดยรวมของพารามิเตอร์หลังการกำหนดการใช้ข้อมูลย้อนหลัง 7 ช่วงข้อมูล

4.2.3 การสร้างแบบจำลองต่างๆ

4.2.3.1 แบบจำลอง LSTM (Long Short-Term Memory)

การสร้างฟังก์ชันแบบจำลอง LSTM โดยที่สร้าง 2 ชั้นประกอบไปด้วย L2 Regularization และ Dropout หลังจาก LSTM ชั้นแรกจะเพิ่มการ Batch Normalization และผลลัพธ์จะอยู่ใน Output Layer นอกจากนี้ ยังเพิ่ม Early Stopping เพื่อตรวจจับการ overfitting และยังมีการสร้างพารามิเตอร์ Coefficient of determination Score (R-Squared), Mean Squared Error (MSE), Mean Absolute Error (MAE) และ Root Mean Squared Error (RMSE) เพื่อตรวจวัดประสิทธิภาพของแบบจำลอง โดยแสดงไว้ในรูปที่ 4.51

```
def train_and_evaluate_lstm(X_train, y_train, X_test, y_test, look_back, epochs=20, batch_size=16):

    # สร้างโมเดล LSTM
    model = Sequential()

     # LSTM ชั้นแรก พร้อม L2 Regularization และ Dropout
     model.add(LSTM(50, return_sequences=True, input_shape=(look_back, 1),
                    kernel_regularizer=regularizers.l2(0.001)))  # L2 Regularization
     model.add(Dropout(0.4))  # Dropout ที่ 0.3

     # Batch Normalization หลังจาก LSTM ชั้นแรก
     model.add(BatchNormalization())

     # LSTM ชั้นที่สอง พร้อม L2 Regularization และ Dropout
     model.add(LSTM(50, return_sequences=False, kernel_regularizer=regularizers.l2(0.001)))
     model.add(Dropout(0.4))  # Dropout ที่ 0.3

      # Dense ชั้นที่มี Dropout
     model.add(Dense(20, activation='relu'))
     model.add(Dropout(0.4))
    model.add(Dense(1))  # Output layer
    optimizer = RMSprop(learning_rate=0.0005)
    model.compile(optimizer='adam', loss='mean_squared_error')

    # เพิ่ม EarlyStopping เพื่อตรวจจับการ overfitting
    early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
    history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
                        validation_data=(X_test, y_test), callbacks=[early_stop])

    # แสดงกราฟ Loss สำหรับ Training และ Validation
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.legend()
    plt.title('Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.show()


    predictions = model.predict(X_test)


    r2 = r2_score(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    mae = mean_absolute_error(y_test, predictions)
    rmse = np.sqrt(mse)

    print(f"R2: {r2}")
    print(f"MAE: {mae}")
    print(f"RMSE: {rmse}")
    print(f"MSE: {mse}")
    #print(f"y_test: {y_test}")
    #print(f"predictions: {predictions}")
    # เก็บค่า metrics ไว้ในดิกชันนารี
    metrics = {
        'R2': r2,
        'MAE': mae,
        'RMSE': rmse,
        'MSE': mse,
        'predictions': predictions,
        'y_test': y_test
    }
    return model, predictions, metrics
```

รูปที่ 4.51 การสร้างฟังก์ชันแบบจำลอง LSTM

4.2.3.2 การสร้างแบบจำลอง XGBoost (Extreme Gradient Boosting)

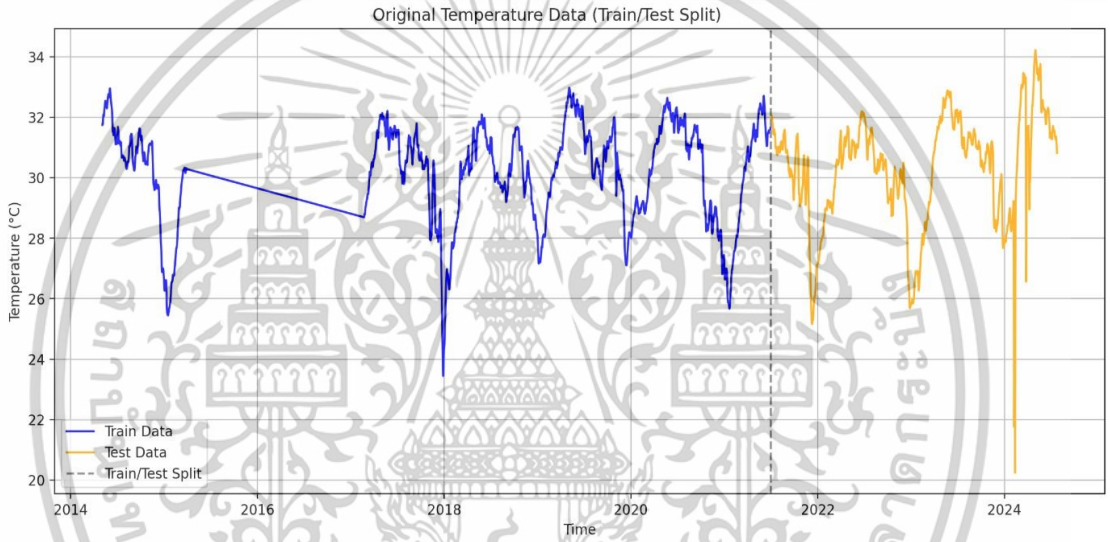
การสร้างฟังก์ชันแบบจำลอง XGBoost ด้วย Custom Huber Loss ในกรณีที่ข้อมูลใกล้เคียงกันมากจะหา Loss Function ที่เหมาะสม เพื่อแบบจำลองใส่ใจกับการพยากรณ์ที่ผิดพลาดมากกว่าค่าที่ใกล้เคียง และการปรับจูนพารามิเตอร์เพื่อให้ค่าการพยากรณ์ของน้ำมีความแม่นยำมากที่สุดด้วย Optuna และยังมีการสร้างพารามิเตอร์ Coefficient of determination Score (R-Squared), Mean Squared Error (MSE), Mean Absolute Error (MAE) และ Root Mean Squared Error (RMSE) เพื่อตรวจวัดประสิทธิภาพของแบบจำลอง โดยแสดงไว้ในรูปที่ 4.52

```python
import optuna
from optuna.samplers import TPESampler
def regression_model(clf, X_train, y_train, X_test, y_test, show_feature_importance=True):
    # ฝึกโมเดล
    clf_model = clf.fit(X_train, y_train)

    # ทำนายผลบนชุดข้อมูลฝึก
    y_hat_train = clf_model.predict(X_train)

    # ทำนายผลบนชุดข้อมูลทดสอบ
    y_hat_test = clf_model.predict(X_test)

    # คำนวณค่า MAE, RMSE, MSE สำหรับชุดข้อมูลทดสอบ
    mae = mean_absolute_error(y_test, y_hat_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_hat_test))
    mse = mean_squared_error(y_test, y_hat_test)
    train_r2 = r2_score(y_test, y_hat_test)

    print('Test set performance:')
    print('\n')
    print(f'R2: {train_r2}')
    print(f'MAE: {mae}')
    print(f'RMSE: {rmse}')
    print(f'MSE: {mse}')
    print('\n')

    # ความสำคัญของตัวแปร
    if show_feature_importance:
        feature_importances = clf_model.feature_importances_
        feature_df = pd.DataFrame({
            "Feature": pd.DataFrame(X_train).columns,
            "Importance": feature_importances
        }).sort_values(by="Importance", ascending=False)
        print("\nความสำคัญของตัวแปร:\n", feature_df)

# Loss Function
def huber_approx_obj(preds, dtrain):
    d = preds - dtrain.get_label()
    h = 1
    scale = 1 + (d / h)**2
    grad = d / scale
    hess = 1 / scale
    return grad, hess

# Objective สำหรับ Optuna
def objective(trial):
    param = {
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 0.05),  # ค่า learning rate
        'max_depth': trial.suggest_int('max_depth', 3, 6),  # ความลึกของต้นไม้
        'n_estimators': trial.suggest_int('n_estimators', 50, 200),  # จำนวนต้นไม้
        'min_child_weight': trial.suggest_int('min_child_weight', 5, 10),  #  min_child_weight
        'subsample': trial.suggest_uniform('subsample', 0.4, 0.6),  #  subsample
        'colsample_bytree': trial.suggest_uniform('colsample_bytree', 0.4, 0.6),  #  colsample_bytree
        'gamma': trial.suggest_loguniform('gamma', 0.5, 1.0),  #  gamma
        'alpha': trial.suggest_loguniform('alpha', 1e-5, 1e-2),  #  L1 regularization
        'lambda': trial.suggest_loguniform('lambda', 1e-5, 1e-2),  # L2 regularization
        'random_state': 0
    }

 # สร้างโมเดล XGBoost ด้วย Custom Huber Loss
 model = xgb.train(param, xgb.DMatrix(X_train1, label=y_train1), obj=huber_approx_obj)
 y_pred = model.predict(xgb.DMatrix(X_test1))
 mse = mean_squared_error(y_test1, y_pred)
 return mse
```

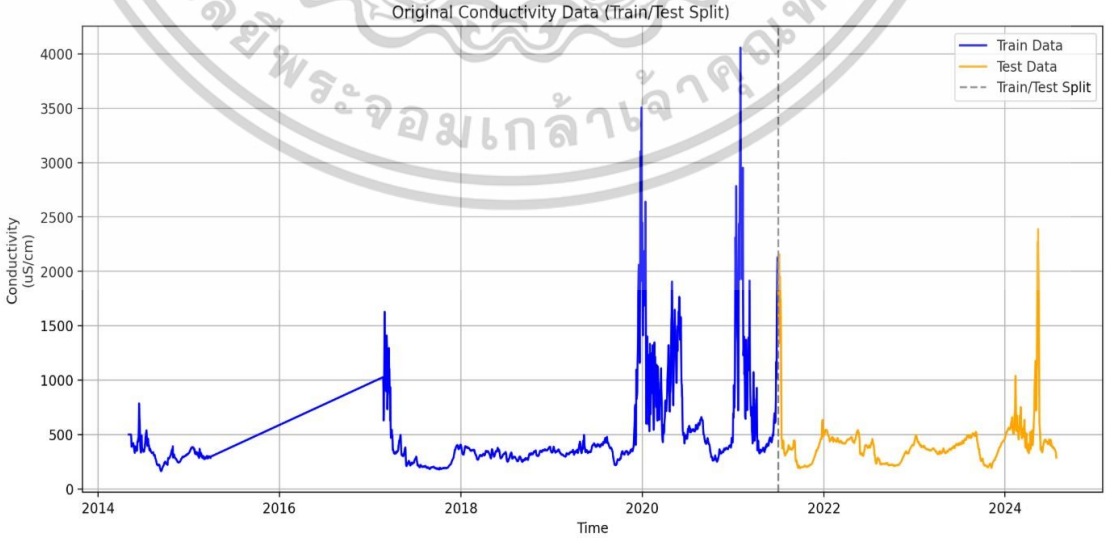
รูปที่ 4.52 การสร้างฟังก์ชันแบบจำลอง XGBoost

4.2.4 การทดสอบระบบการพยากรณ์

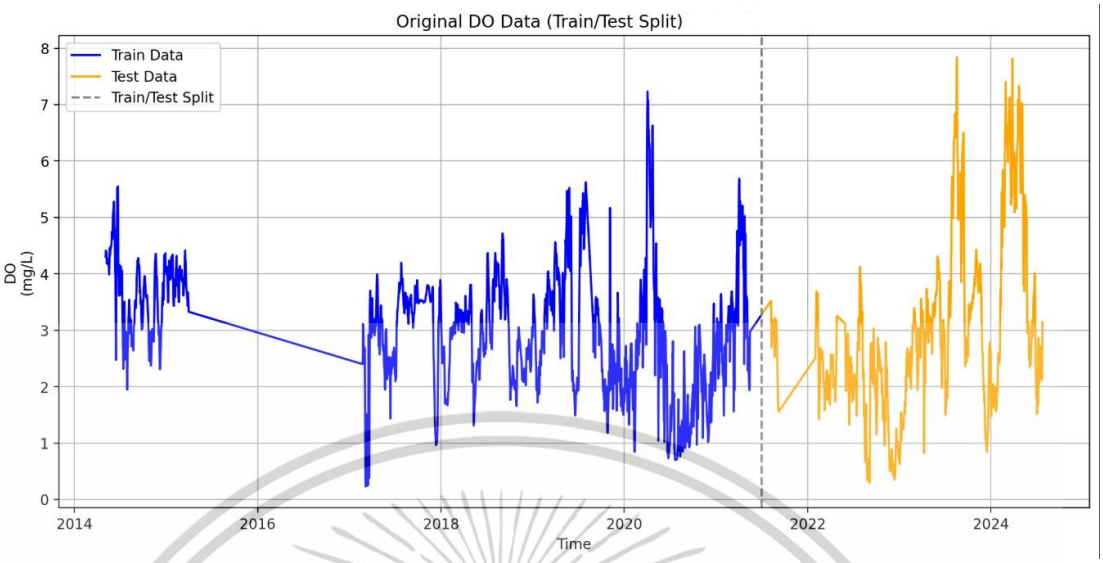
ภายหลังการจัดเตรียมรูปแบบของชุดข้อมูลที่ใช้ในการศึกษา ผู้วิจัยได้ดำเนินการพล็อตกราฟเพื่อแสดงการแบ่งช่วงข้อมูลระหว่างข้อมูลฝึกสอนโดยคิดเป็น 70% ของชุดข้อมูลทั้งหมดแต่ละพารามิเตอร์ และข้อมูลทดสอบโดยคิดเป็น 30% ของข้อมูลที่เหลืออยู่ในชุดข้อมูลทั้งหมด โดยมีแกนแนวนอนแสดงเวลา และแกนแนวตั้งแสดงค่าของพารามิเตอร์ต่าง ๆ ได้แก่ อุณหภูมิของน้ำ การนำไฟฟ้าของน้ำ ปริมาณออกซิเจนที่ละลายในน้ำ และความเป็นกรด-ด่าง ตามลำดับ โดยเส้นกราฟสีน้ำเงินแทนข้อมูลที่ใช้สำหรับฝึกสอนแบบจำลอง เส้นกราฟสีเหลืองแทนข้อมูลที่ใช้สำหรับทดสอบแบบจำลอง และมีเส้นประสีเทาเป็นตัวแบ่งช่วงข้อมูลทั้งสองส่วนอย่างชัดเจน โดย 70% ของชุดข้อมูลทั้งหมดแต่ละพารามิเตอร์ ได้แก่ อุณหภูมิของน้ำ การนำไฟฟ้าของน้ำ ปริมาณออกซิเจนที่มีอยู่ในน้ำ และความเป็นกรด-เบส ตามลำดับ แสดงไว้ในรูปที่ 4.53-4.56



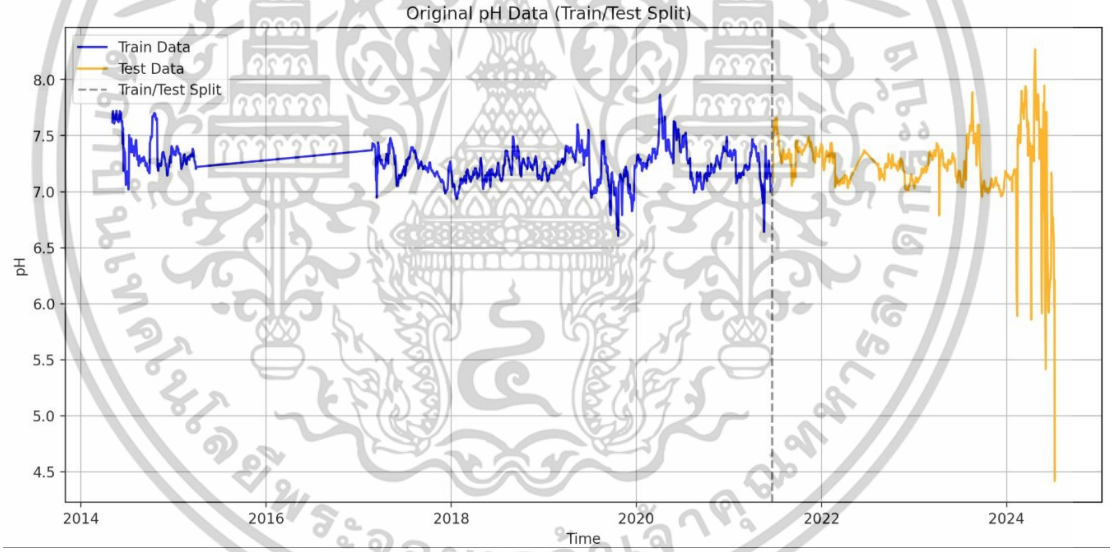
รูปที่ 4.53 กราฟชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบของอุณหภูมิของน้ำ



รูปที่ 4.54 กราฟชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบของการนำไฟฟ้าของน้ำ

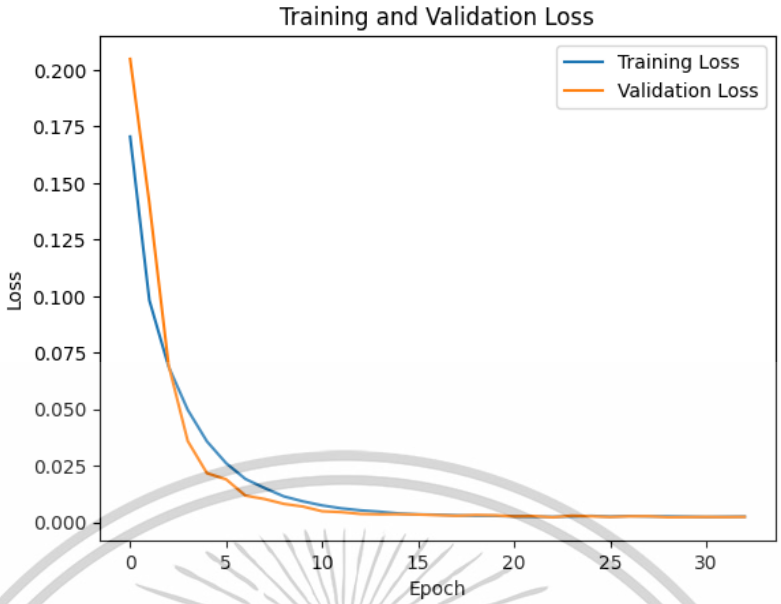


รูปที่ 4.55 กราฟชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบของปริมาณออกซิเจนที่มีอยู่ในน้ำ

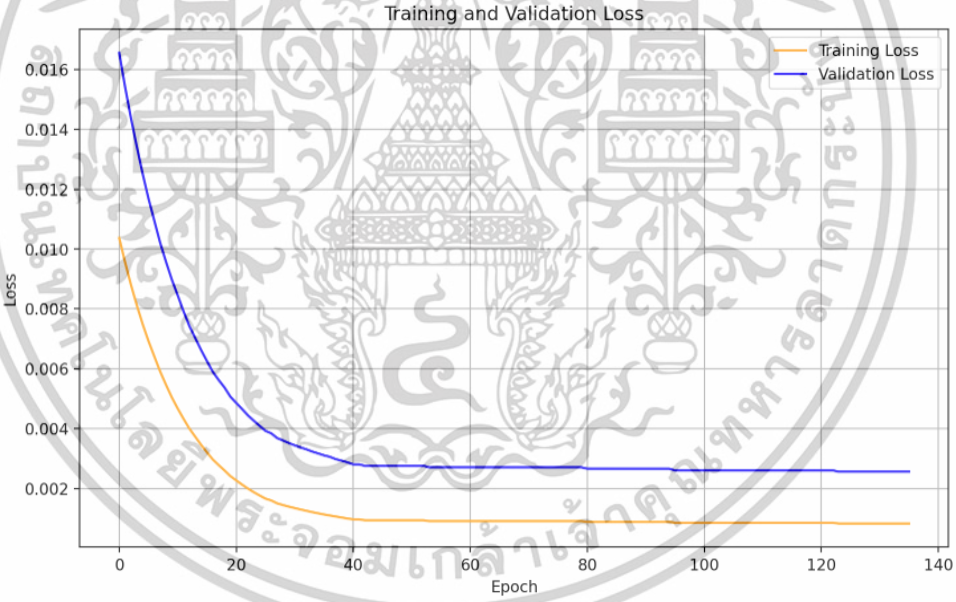


รูปที่ 4.56 กราฟชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบของความเป็นกรด-เบส

การทดสอบ (Test) ระบบการพยาณกรณ์แต่ละแบบจำลองทดสอบด้วยข้อมูลที่เหลืออยู่ 30% ของชุดข้อมูลทั้งหมด โดยเปรียบเทียบแต่ละพารามิเตอร์ของแต่ละแบบจำลองใดให้ประสิทธิภาพในการพยากรณ์ได้ดีกว่ากัน ทางผู้วิจัยได้แสดงการวัดค่าประสิทธิภาพและกราฟเพื่อจัดทำแสดงการแบ่งช่วงข้อมูลระหว่างข้อมูลฝึกสอน (training data) และข้อมูลทดสอบ (testing data) อย่างชัดเจน เพื่อใช้ในการเปรียบเทียบระหว่างข้อมูลดิบกับข้อมูลที่ได้จากกระบวนการพยากรณ์ ไว้ในรูปที่ 4.57-4.72

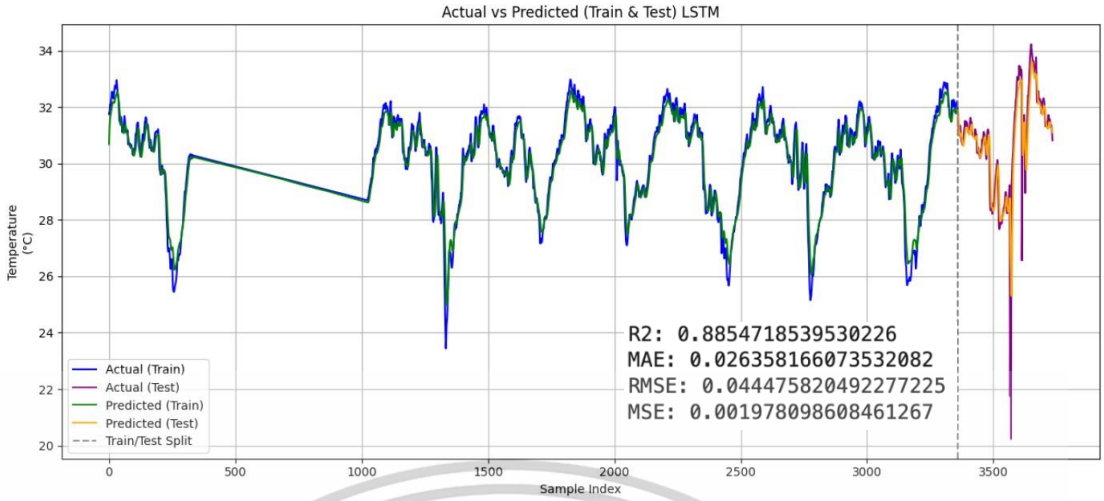


รูปที่ 4.57 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของอุณหภูมิของน้ำแบบจำลอง LSTM

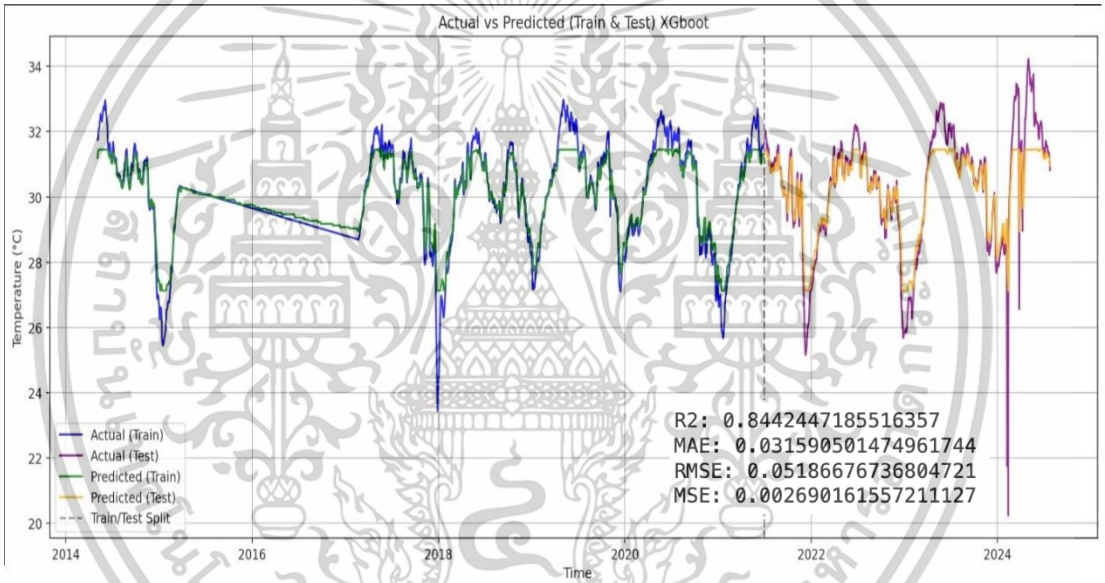


รูปที่ 4.58 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของอุณหภูมิของน้ำแบบจำลอง XGBoost

จากรูปที่ 4.57-4.58 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์อุณหภูมิของน้ำแบบจำลอง LSTM มีค่าความผิดพลาด (Loss) ของข้อมูลชุดฝึก (Training) และข้อมูลชุดตรวจสอบ (Validation) ซึ่งค่าความผิดพลาดลดลงอย่างรวดเร็วและมีความใกล้เคียงกัน แสดงให้เห็นว่าแบบจำลองสามารถเรียนรู้ลักษณะของข้อมูลได้ดีและไม่มีแนวโน้มของการเกิด Overfitting ส่วนในแบบจำลอง XGBoost ค่าความผิดพลาดของชุดฝึกมีแนวโน้มลดลงอย่างต่อเนื่อง ในขณะที่ค่าความผิดพลาดของชุดตรวจสอบเริ่มคงที่หลังจากประมาณ 50 epoch ซึ่งอาจแสดงถึงการเกิด Overfitting



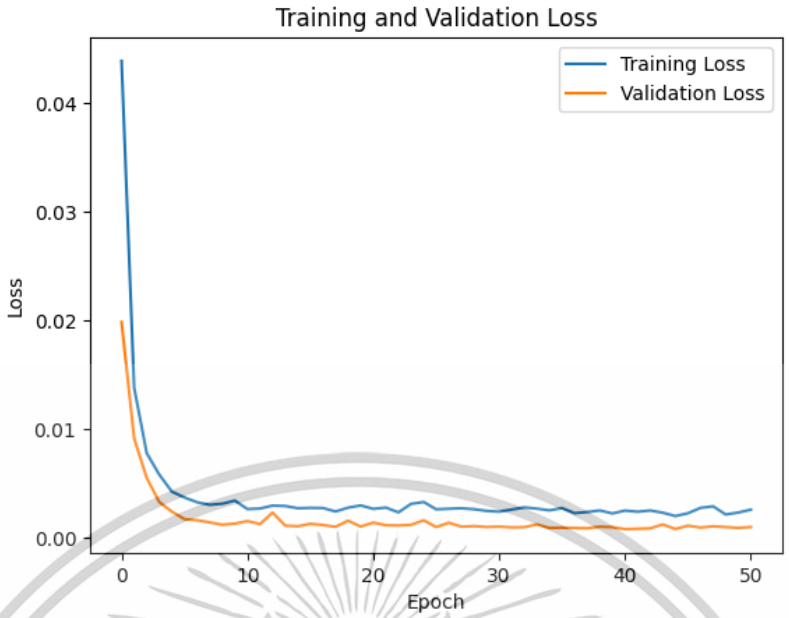
รูปที่ 4.59 กราฟประสิทธิภาพในการพยากรณ์ของอุณหภูมิของน้ำแบบจำลอง LSTM



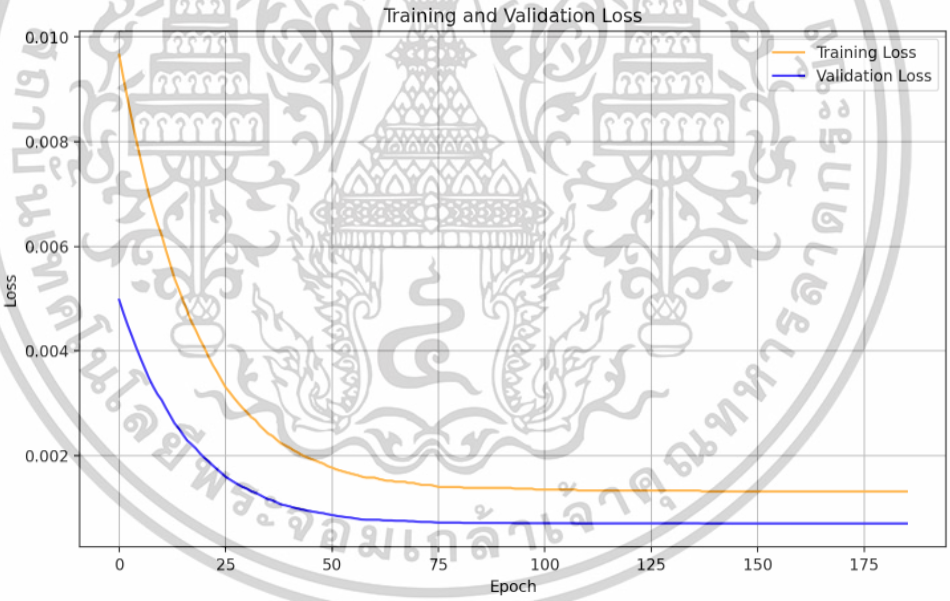
รูปที่ 4.60 กราฟประสิทธิภาพในการพยากรณ์ของอุณหภูมิของน้ำแบบจำลอง XGBoost

จากรูปที่ 4.59-4.60 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์อุณหภูมิของน้ำแบบจำลอง LSTM มีค่า R-Squared ประมาณ 0.8855 ซึ่งค่าเข้าใกล้ 1 มากกว่าแบบจำลอง XGBoost ประมาณ 0.8442 ซึ่งมีความแตกต่างอยู่ระหว่างแบบจำลองทั้ง 2 แบบจำลองประมาณ 0.0413 และค่า MAE, RMSE และ MSE ของแบบจำลอง LSTM มีค่าประมาณ 0.0264, 0.0445 และ 0.0012 ตามลำดับ ซึ่งค่าทั้งหมดที่กล่าวมานั้นเข้าใกล้ 0 มากกว่าแบบจำลอง XGBoost ที่มีค่า MAE, RMSE และ MSE ประมาณ 0.0316, 0.0519 และ 0.0027 ตามลำดับ ซึ่งมีความแตกต่างของค่า MAE, RMSE และ MSE ระหว่างแบบจำลองทั้ง 2 แบบจำลองอยู่ประมาณ 0.0052, 0.0074 และ 0.0015 ตามลำดับ

ดังนั้นการพยากรณ์ของพารามิเตอร์อุณหภูมิน้ำแบบอนุกรมเวลของชุดข้อมูลแม่น้ำเจ้าพระยา ณ สถานีวัดมะขาม ย้อนหลัง 10 ปี จากหน่วยงานการประปานครหลวง โดยใช้แบบจำลอง LSTM นั้นดีกว่าแบบจำลอง XGBoost

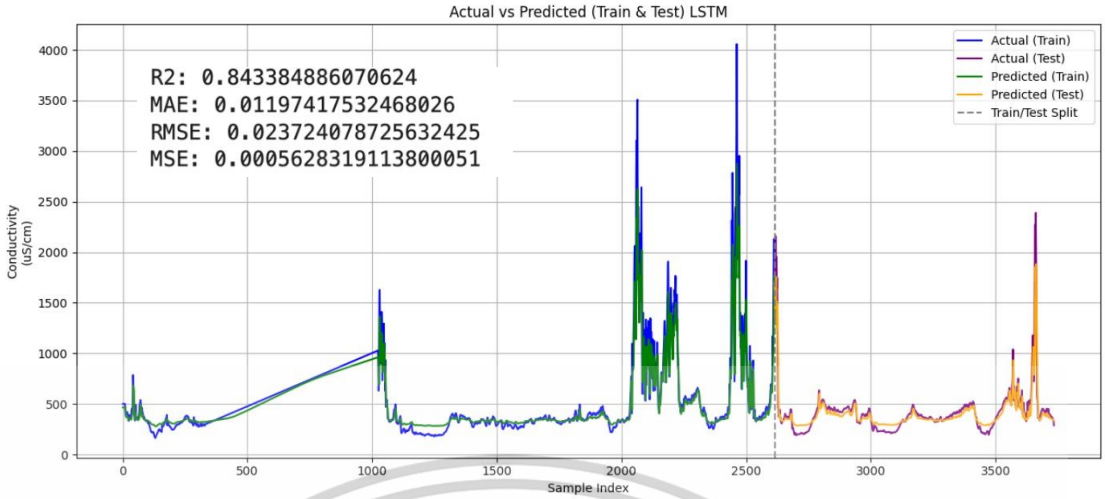


รูปที่ 4.61 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของการนำไฟฟ้าของน้ำแบบจำลอง LSTM

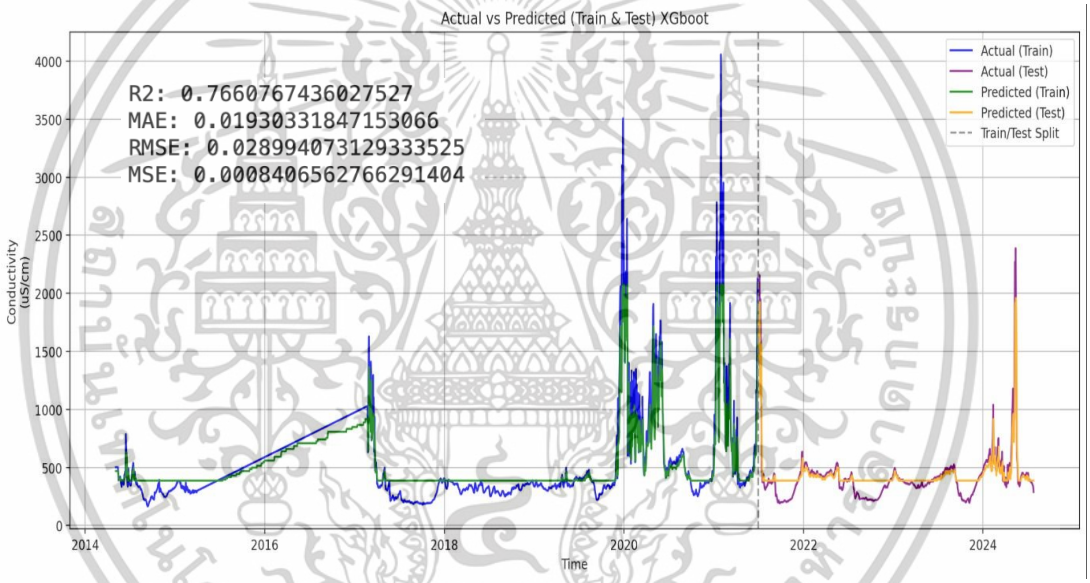


รูปที่ 4.62 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของการนำไฟฟ้าของน้ำแบบจำลอง XGBoost

จากรูปที่ 4.61-4.62 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์การนำไฟฟ้าของน้ำแบบจำลอง LSTM มีค่าความผิดพลาด (Loss) ของข้อมูลชุดฝึก (Training) และข้อมูลชุดตรวจสอบ (Validation) ซึ่งค่าความผิดพลาดลดลงอย่างรวดเร็วและมีความใกล้เคียงกัน แสดงให้เห็นว่าแบบจำลองสามารถเรียนรู้ลักษณะของข้อมูลได้ดีและไม่มีแนวโน้มของการเกิด Overfitting ส่วนในแบบจำลอง XGBoost ค่าความผิดพลาดของชุดฝึกมีแนวโน้มลดลงอย่างต่อเนื่อง ในขณะที่ค่าความผิดพลาดของชุดตรวจสอบเริ่มคงที่ ซึ่งอาจแสดงถึงการเกิด Overfitting



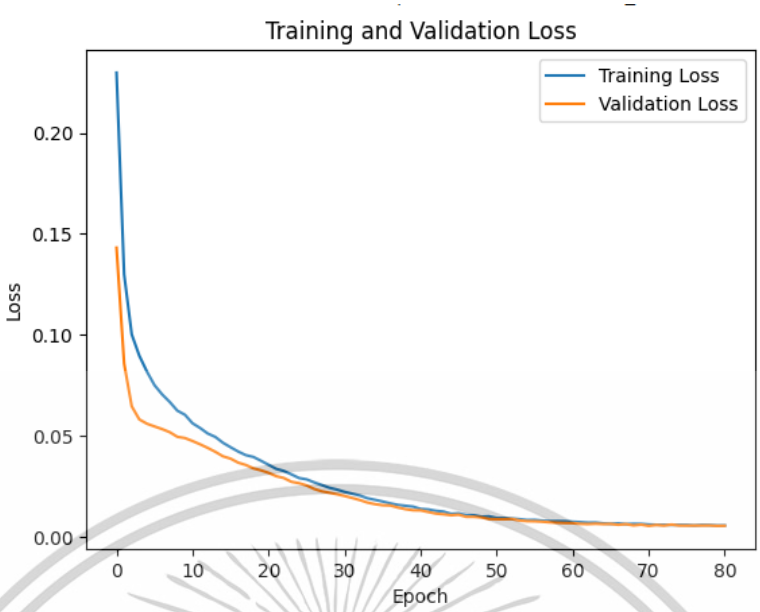
รูปที่ 4.63 กราฟประสิทธิภาพในการพยากรณ์ของการนำไฟฟ้าของน้ำแบบจำลอง LSTM



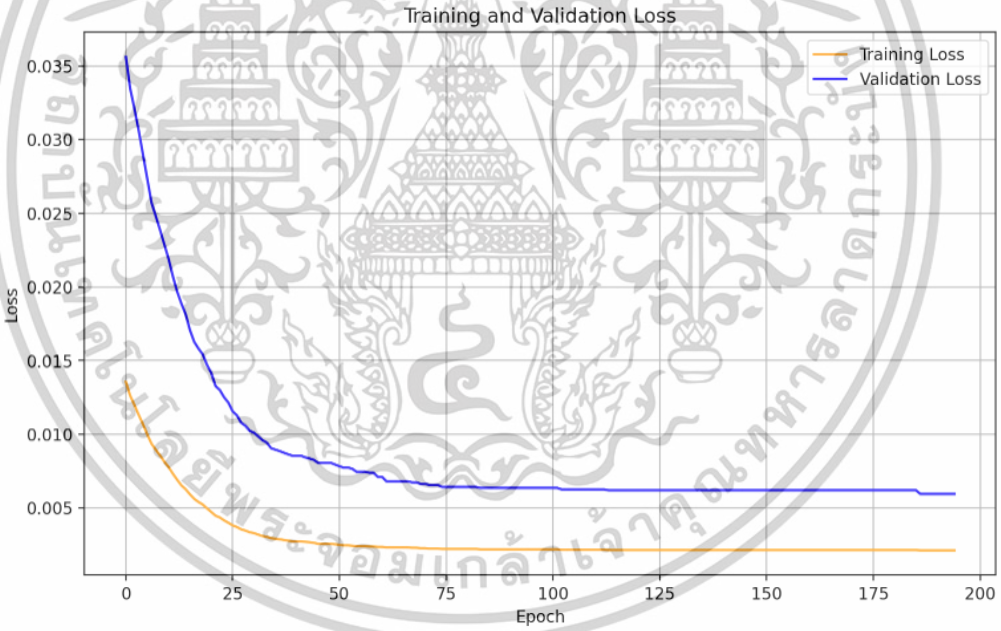
รูปที่ 4.64 กราฟประสิทธิภาพในการพยากรณ์ของการนำไฟฟ้าของน้ำแบบจำลอง XGBoost

จากรูปที่ 4.63-4.64 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์การนำไฟฟ้าของน้ำแบบจำลอง LSTM มีค่า R-Squared ประมาณ 0.8434 ซึ่งค่าเข้าใกล้ 1 มากกว่าแบบจำลอง XGBoost ประมาณ 0.7661 ซึ่งมีความแตกต่างอยู่ระหว่างแบบจำลองทั้ง 2 แบบจำลองประมาณ 0.0773 และค่า MAE, RMSE และ MSE ของแบบจำลอง LSTM มีค่าประมาณ 0.0120, 0.0237 และ 0.0006 ตามลำดับ ซึ่งค่าทั้งหมดที่กล่าวมานั้นเข้าใกล้ 0 มากกว่าแบบจำลอง XGBoost ที่มีค่า MAE, RMSE และ MSE ประมาณ 0.0193, 0.0290 และ 0.0008 ตามลำดับ ซึ่งมีความแตกต่างของค่า MAE, RMSE และ MSE ระหว่างแบบจำลองทั้ง 2 แบบจำลองอยู่ประมาณ 0.0073, 0.0053 และ 0.0002 ตามลำดับ

ดังนั้นการพยากรณ์ของพารามิเตอร์การนำไฟฟ้าของน้ำแบบอนุกรมเวลของชุดข้อมูลแม่น้ำเจ้าพระยา ณ สถานีวัดมะขาม ย้อนหลัง 10 ปี จากหน่วยงานการประปานครหลวง โดยใช้แบบจำลอง LSTM นั้นดีกว่าแบบจำลอง XGBoost

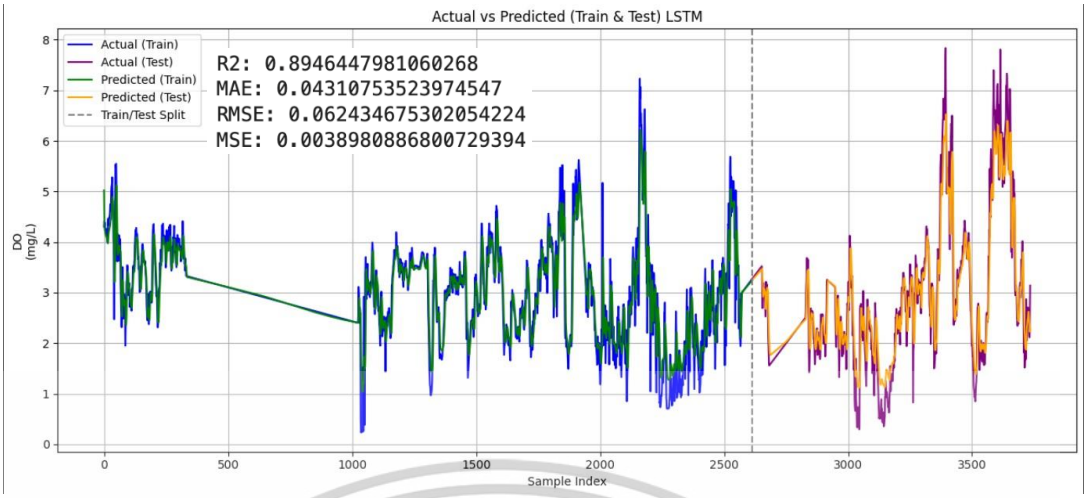


รูปที่ 4.65 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของปริมาณออกซิเจนที่มีอยู่ในน้ำแบบจำลอง LSTM

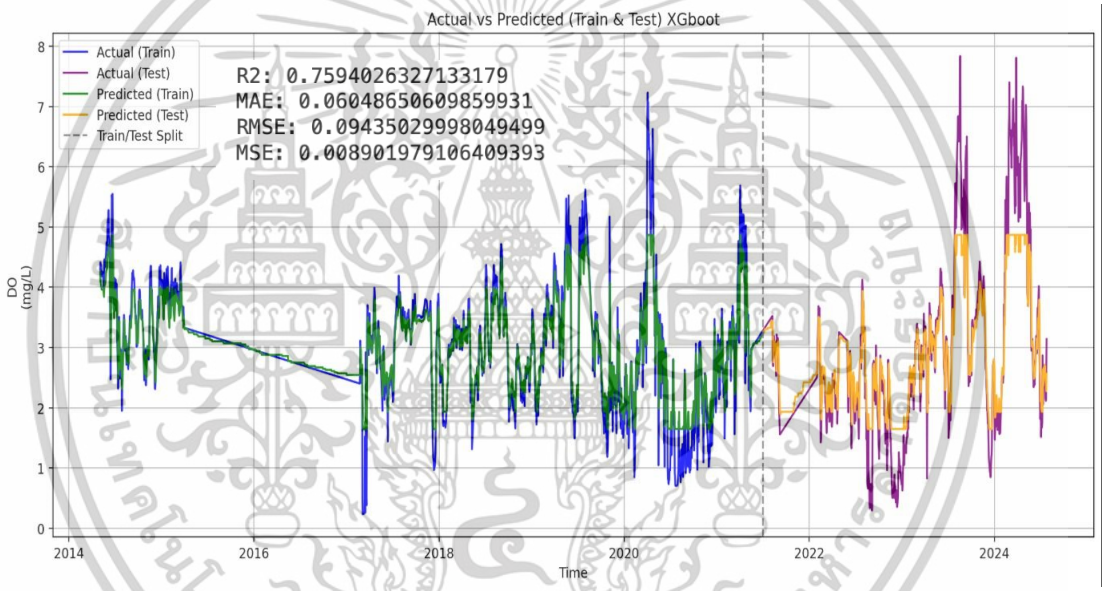


รูปที่ 4.66 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของปริมาณออกซิเจนที่มีอยู่ในน้ำแบบจำลอง XGBoost

จากรูปที่ 4.65-4.66 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์การนำไฟฟ้าของน้ำแบบจำลอง LSTM มีค่าความผิดพลาด (Loss) ของข้อมูลชุดฝึก (Training) และข้อมูลชุดตรวจสอบ (Validation) ซึ่งค่าความผิดพลาดลดลงอย่างรวดเร็วและมีความใกล้เคียงกัน แสดงให้เห็นว่าแบบจำลองสามารถเรียนรู้ลักษณะของข้อมูลได้ดีและไม่มีแนวโน้มของการเกิด Overfitting ส่วนในแบบจำลอง XGBoost ค่าความผิดพลาดของชุดฝึกมีแนวโน้มลดลงอย่างต่อเนื่อง ในขณะที่ค่าความผิดพลาดของชุดตรวจสอบเริ่มคงที่ ซึ่งอาจแสดงถึงการเกิด Overfitting



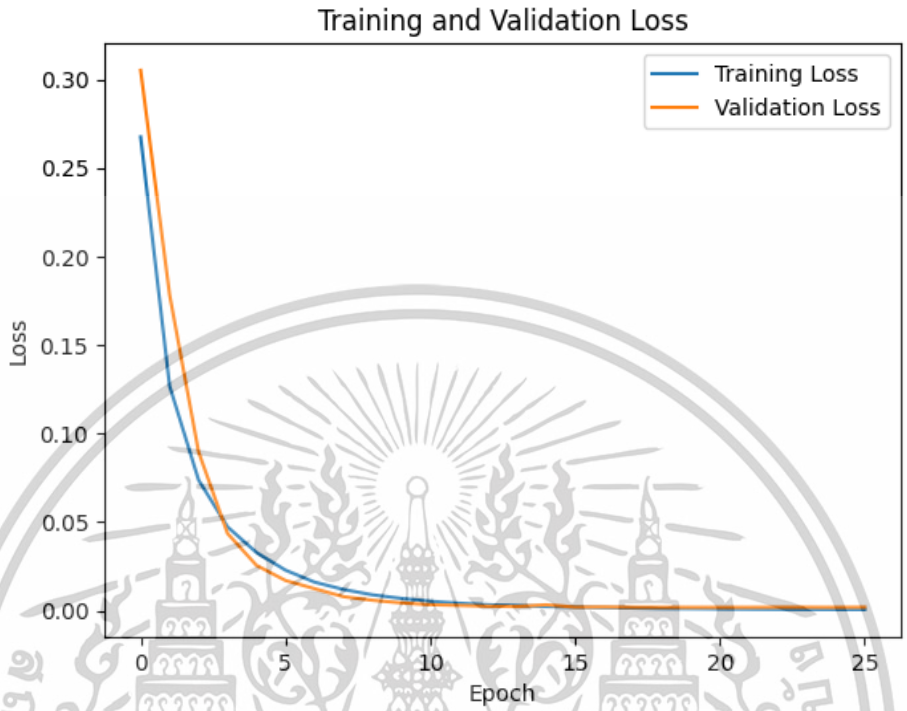
รูปที่ 4.67 กราฟประสิทธิภาพในการพยากรณ์ของปริมาณออกซิเจนที่มีอยู่ในน้ำแบบจำลอง LSTM



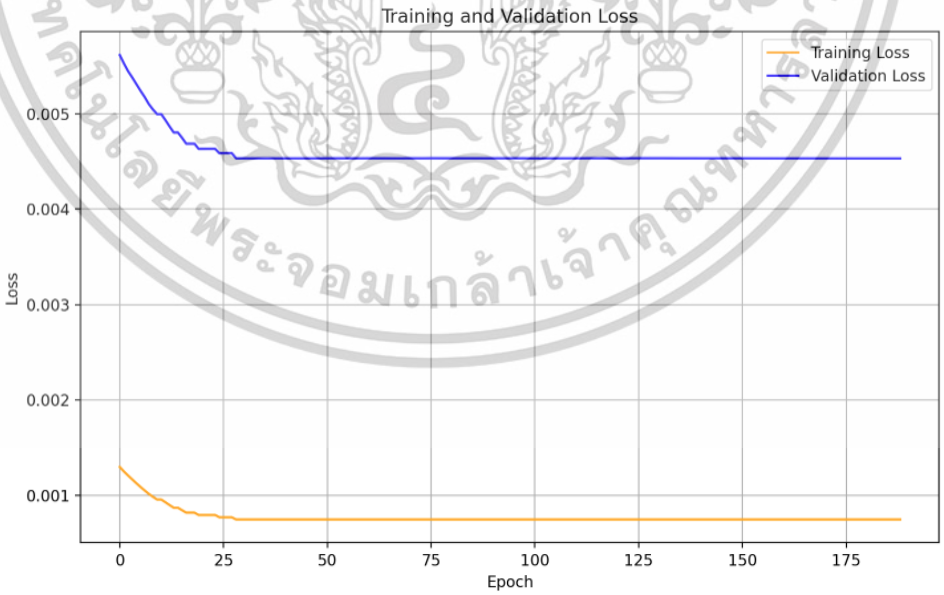
รูปที่ 4.68 กราฟประสิทธิภาพในการพยากรณ์ของปริมาณออกซิเจนที่มีอยู่ในน้ำแบบจำลอง XGBoost

จากรูปที่ 4.67-4.68 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์ปริมาณออกซิเจนที่มีอยู่ในน้ำแบบจำลอง LSTM มีค่า R-Squared ประมาณ 0.8946 ซึ่งค่าเข้าใกล้ 1 มากกว่าแบบจำลอง XGBoost ประมาณ 0.7594 ซึ่งมีความแตกต่างอยู่ระหว่างแบบจำลองทั้ง 2 แบบจำลองประมาณ 0.1352 และค่า MAE, RMSE และ MSE ของแบบจำลอง LSTM มีค่าประมาณ 0.0431, 0.0624 และ 0.0039 ตามลำดับ ซึ่งค่าทั้งหมดที่กล่าวมานั้นเข้าใกล้ 0 มากกว่าแบบจำลอง XGBoost ที่มีค่า MAE, RMSE และ MSE ประมาณ 0.0605, 0.0944 และ 0.0089 ตามลำดับ ซึ่งมีความแตกต่างของค่า MAE, RMSE และ MSE ระหว่างแบบจำลองทั้ง 2 แบบจำลองอยู่ประมาณ 0.0174, 0.0320 และ 0.0050 ตามลำดับ

ดังนั้นการพยากรณ์ของพารามิเตอร์ปริมาณออกซิเจนที่มีอยู่ในน้ำแบบอนุกรมเวลของชุดข้อมูลแม่น้ำเจ้าพระยา ณ สถานีวัดมะขาม ย้อนหลัง 10 ปี จากหน่วยงานการประปานครหลวง โดยใช้แบบจำลอง LSTM นั้นดีกว่าแบบจำลอง XGBoost

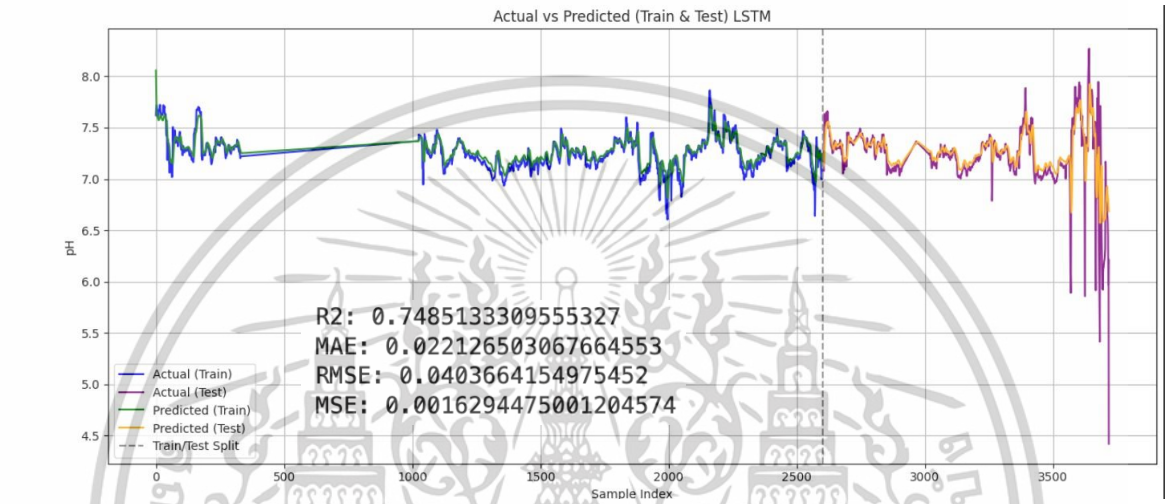


รูปที่ 4.69 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของความเป็นกรด-เบส แบบจำลอง LSTM

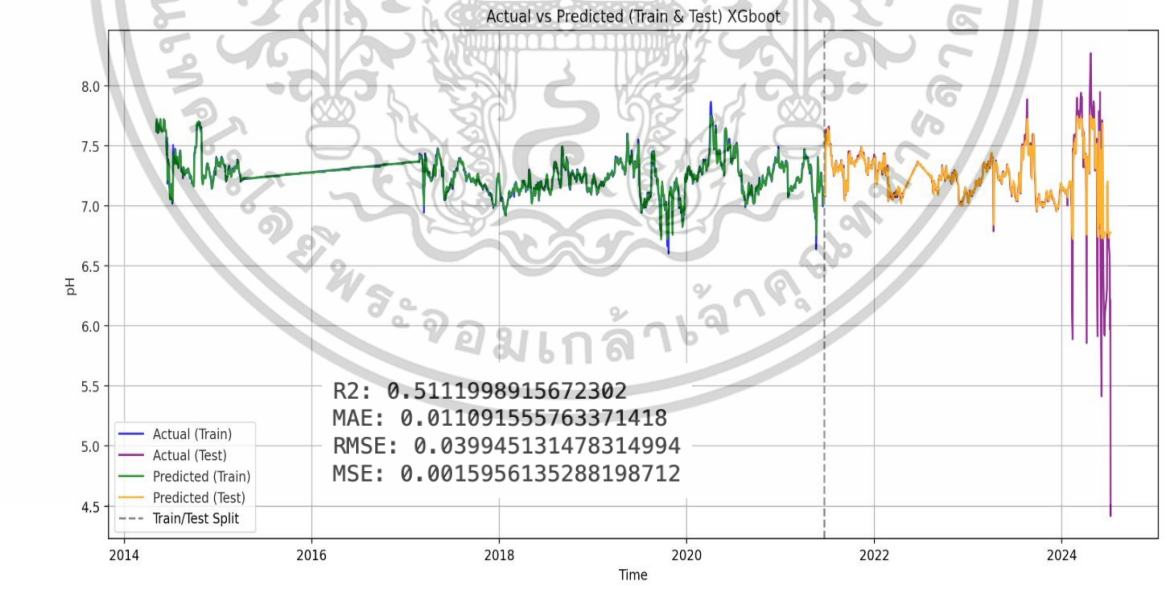


รูปที่ 4.70 กราฟค่าความผิดพลาดเมื่อเทียบกับ Epoch ในการพยากรณ์ของความเป็นกรด-เบส แบบจำลอง XGBoost

จากรูปที่ 4.69-4.70 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์การนำไฟฟ้าของน้ำแบบจำลอง LSTM มีค่าความผิดพลาด (Loss) ของข้อมูลชุดฝึก (Training) และข้อมูลชุดตรวจสอบ (Validation) ซึ่งค่าความผิดพลาดลดลงอย่างรวดเร็วและมีความใกล้เคียงกัน แสดงให้เห็นว่าแบบจำลองสามารถเรียนรู้ลักษณะของข้อมูลได้ดีและไม่มีแนวโน้มของการเกิด Overfitting ส่วนในแบบจำลอง XGBoost ค่าความผิดพลาดทั้งสำหรับข้อมูลฝึก (Training Loss) และข้อมูลตรวจสอบ (Validation Loss) ลดลงอย่างรวดเร็วและคงที่ในระดับต่ำ แสดงให้เห็นถึงความสามารถของแบบจำลองในการเรียนรู้ข้อมูลได้อย่างมีประสิทธิภาพและแม่นยำ



รูปที่ 4.71 กราฟประสิทธิภาพในการพยากรณ์ของความเป็นกรด-เบสแบบจำลอง LSTM



รูปที่ 4.72 กราฟประสิทธิภาพในการพยากรณ์ของความเป็นกรด-เบสแบบจำลอง XGBoost

จากรูปที่ 4.71-4.72 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์ความเป็นกรด-เบสแบบจำลอง LSTM มีค่า R-Squared ประมาณ 0.7485 ซึ่งค่าเข้าใกล้ 1 มากกว่าแบบจำลอง XGBoost ประมาณ 0.5112 ซึ่งมีความแตกต่างอยู่ระหว่างแบบจำลองทั้ง 2 แบบจำลองประมาณ 0.2372 แต่ค่า MAE, RMSE และ MSE ของแบบจำลอง XGBoost มีค่าประมาณ 0.0111, 0.0400 และ 0.0016 ตามลำดับ ซึ่งค่าทั้งหมดที่กล่าวมานั้นเข้าใกล้ 0 มากกว่าแบบจำลอง LSTM ที่มีค่า MAE, RMSE และ MSE ประมาณ 0.0221, 0.0404 และ 0.0016 ตามลำดับ ซึ่งมีความแตกต่างของค่า MAE, RMSE และ MSE ระหว่างแบบจำลองทั้ง 2 แบบจำลองอยู่ประมาณ 0.0110, 0.0004 และ 0.0000 ตามลำดับ

ดังนั้นการพยากรณ์ของพารามิเตอร์ความเป็นกรด-เบสแบบอนุกรมเวลของชุดข้อมูลแม่น้ำเจ้าพระยา ณ สถานีวัดมะขาม ย้อนหลัง 10 ปี จากหน่วยงานการประปานครหลวง โดยใช้แบบจำลอง XGBoost นั้นดีกว่าแบบจำลอง LSTM เนื่องจากพารามิเตอร์การวัดค่าประสิทธิภาพของแบบจำลองทั้ง 3 พารามิเตอร์ได้แก่ MAE, RMSE และ MSE ให้ผลลัพธ์ไปในทิศทางเดียวกัน

ตารางที่ 4.1 ประสิทธิภาพในการพยากรณ์พารามิเตอร์ทั้ง 4 ชนิดของทั้ง 2 แบบจำลอง

Efficiency	Temperature		Electrical Conductivity		Dissolved Oxygen		pH	
	LSTM	XGBoost	LSTM	XGBoost	LSTM	XGBoost	LSTM	XGBoost
R-Squared	0.8854	0.8442	0.8433	0.7660	0.8946	0.7594	0.7485	0.5111
MAE	0.0263	0.0315	0.0119	0.0193	0.0431	0.0604	0.022	0.0110
RMSE	0.0444	0.0518	0.0237	0.0289	0.0624	0.094	0.0403	0.0399
MSE	0.0019	0.0026	0.0005	0.0008	0.0038	0.0089	0.0016	0.0015

จากตารางที่ 4.1 จะเห็นได้ว่าในการพยากรณ์ของพารามิเตอร์อุณหภูมิของน้ำ การนำไฟฟ้าของน้ำ และปริมาณออกซิเจนที่มีอยู่ในน้ำของแบบจำลอง LSTM มีค่า R-Squared ซึ่งค่าเข้าใกล้ 1 มากกว่าแบบจำลอง XGBoost และค่า MAE, RMSE และ MSE ของแบบจำลอง LSTM ซึ่งค่าทั้งหมดนั้นเข้าใกล้ 0 มากกว่าแบบจำลอง XGBoost ดังนั้นการพยากรณ์ของพารามิเตอร์อุณหภูมิของน้ำ การนำไฟฟ้าของน้ำ และปริมาณออกซิเจนที่มีอยู่ในน้ำของชุดข้อมูลโดยใช้แบบจำลอง LSTM นั้นดีกว่าแบบจำลอง XGBoost ส่วนการพยากรณ์ของพารามิเตอร์ความเป็นกรด-เบสแบบจำลอง LSTM มีค่า R-Squared ซึ่งค่าเข้าใกล้ 1 มากกว่าแบบจำลอง XGBoost แต่ค่า MAE, RMSE และ MSE ของแบบจำลอง XGBoost ซึ่งค่าทั้งหมดนั้นเข้าใกล้ 0 มากกว่าแบบจำลอง LSTM ดังนั้นการพยากรณ์ของพารามิเตอร์ความเป็นกรด-เบสของชุดข้อมูล โดยใช้แบบจำลอง XGBoost นั้นดีกว่าแบบจำลอง LSTM

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผล

วิทยานิพนธ์ฉบับนี้นำเสนอระบบตรวจสอบและพยากรณ์คุณภาพน้ำแบบเรียลไทม์ โดยใช้เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง (IoT) เพื่อประยุกต์ใช้ในการติดตามและควบคุมคุณภาพน้ำในพื้นที่ต่าง ๆ อย่างต่อเนื่อง ครอบคลุม และมีประสิทธิภาพสูง ช่วยเพิ่มความแม่นยำในการวิเคราะห์ข้อมูล และลดความล่าช้าในการรวบรวมข้อมูลเพื่อการบริหารจัดการคุณภาพน้ำอย่างมีประสิทธิผล ระบบที่พัฒนาขึ้นประกอบด้วยสถานีตรวจวัดคุณภาพน้ำที่สามารถทำงานได้ตลอดเวลา ใช้งานง่าย ประหยัดเวลา และต้นทุนต่ำ โดยเชื่อมต่อกับระบบอินเทอร์เน็ตผ่านเครือข่ายเซลลูลาร์ ไปยังฐานข้อมูลบนคลาวด์ จากนั้นข้อมูลจะถูกส่งต่อไปยังเว็บไซต์แสดงผล ซึ่งผู้ใช้งานสามารถเข้าถึงและตรวจสอบคุณภาพน้ำได้แบบเรียลไทม์ผ่านอุปกรณ์สื่อสารต่าง ๆ เช่น คอมพิวเตอร์ แท็บเล็ต หรือสมาร์ตโฟน โดยระบบยังสามารถเก็บข้อมูลย้อนหลังเพื่อการวิเคราะห์ระยะยาว

นอกจากนี้ วิทยานิพนธ์ยังได้ศึกษาและพัฒนาระบบพยากรณ์คุณภาพน้ำในอนาคต โดยใช้ข้อมูลอนุกรมเวลาของแม่น้ำเจ้าพระยาจากสถานีวัดมะขาม ซึ่งจัดเก็บโดยการประปานครหลวง ย้อนหลังเป็นระยะเวลา 10 ปี เพื่อนำมาใช้สร้างแบบจำลองพยากรณ์ด้วยวิธีการถดถอยเชิงเส้นหลายระดับ (Multiple Linear Regression) ด้วยแบบจำลอง Extreme Gradient Boosting (XGBoost) ที่ปรับแต่งพารามิเตอร์ด้วย Optuna และการเรียนรู้เชิงลึกด้วยโครงข่ายประสาทเทียม Long Short-Term Memory (LSTM) จากผลการทดลองพบว่า แบบจำลอง LSTM สามารถพยากรณ์ค่าพารามิเตอร์คุณภาพน้ำ ได้แก่ อุณหภูมิของน้ำ, การนำไฟฟ้าของน้ำ และปริมาณออกซิเจนละลายในน้ำ ได้อย่างแม่นยำ โดยมีค่าประสิทธิภาพการวัด R-Squared, Mean Absolute Error (MAE), Root Mean Square Error (RMSE) และ Mean Square Error (MSE) เฉลี่ยแต่ละพารามิเตอร์โดยประมาณ 0.8744, 0.0271, 0.0435 และ 0.0020 ตามลำดับ ที่ให้ผลลัพธ์ไปในทิศทางเดียวกันว่าแบบจำลอง LSTM มีประสิทธิภาพเหนือกว่าแบบจำลอง XGBoost อย่างไรก็ตาม สำหรับพารามิเตอร์ค่าความเป็นกรด-เบส (pH) แบบจำลอง XGBoost ให้ผลลัพธ์ที่แม่นยำกว่าด้วยค่าประสิทธิภาพการวัด R-Squared, MAE, RMSE และ MSE โดยประมาณ 0.5111, 0.0110, 0.0399 และ 0.015 ตามลำดับ ที่ดีกว่าแบบจำลอง LSTM ซึ่งแสดงให้เห็นถึงศักยภาพในการประยุกต์ใช้เทคโนโลยีการเรียนรู้ของเครื่องร่วมกับระบบเทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง IoT สามารถเพิ่มศักยภาพในการติดตาม ตรวจสอบ และคาดการณ์คุณภาพน้ำได้อย่างมีประสิทธิภาพ ทั้งในเชิงเวลาจริงและการวางแผนเชิงอนาคต ซึ่งเหมาะสมอย่างยิ่งต่อการนำไปประยุกต์ใช้ในงานด้านสิ่งแวดล้อม การบริหารจัดการทรัพยากรน้ำ และการเตือนภัยล่วงหน้าในระบบนิเวศแหล่งน้ำต่าง ๆ

5.2 ข้อเสนอแนะและแนวทางวิจัยเพิ่มเติม

แนวทางวิจัยสำหรับการพัฒนางานวิจัยในอนาคต จะดำเนินการโดยเพิ่มการทำงานของระบบ โดยจะเพิ่มเซ็นเซอร์เพิ่มเติมสำหรับวัดและตรวจสอบคุณภาพน้ำที่หลากหลาย เช่น เซ็นเซอร์การวัดปริมาณของแข็ง สารอนินทรีย์และอินทรีย์ทั้งหมดที่ละลายอยู่ในน้ำ เซนเซอร์วัดความขุ่นในน้ำ และเซ็นเซอร์วัดความเค็มในน้ำ เพื่อให้แต่ละสถานีสามารถวัดและตรวจสอบคุณภาพน้ำได้หลากหลายช่วง ซึ่งอาจส่งผลต่อคุณภาพน้ำ และเป็นแบบจำลองสำหรับการใช้งานในพื้นที่อื่นๆ ที่ต้องคำนึงถึง และการใช้งานอื่นๆ เช่น การแจ้งเตือนและการแสดงผลบนแดชบอร์ดผ่านแอปพลิเคชันบนโทรศัพท์มือถือ เพื่อให้เข้าถึงการแสดงผลข้อมูลที่รวดเร็วและสามารถใช้งานได้ง่ายอย่างต่อเนื่อง และนำข้อมูลที่ได้จากสถานีตรวจวัดคุณภาพน้ำมาคาดการณ์การตรวจสอบคุณภาพน้ำในอนาคต

เอกสารอ้างอิง

[1] Economic and social commission for Asia and the Pacific, "A Case Study on Urban Green Growth: Conserving the Bang Kachao Green Area," **Workshop on Water and Green Growth in Asia and the Pacific,** Bangkok, Thailand, Feb. 23–25, 2015.

[2] S. Madakam, R. Ramaswamya and S. Tripathi, " Internet of Things (IoT): A Literature Review," **Journal of Computer and Communications,** Vol. 3, No. 5, pp.164-173, May 2015.

[3] J. A. Smith and P. R. Jones, "Internet of Things: A Survey of Technologies, Applications, and Challenges," **IEEE Internet of Things Journal**, vol. 3, no. 4, pp. 456-463, Aug. 2016, doi: 10.1109/JIOT.2016.2548623.

[4] R. Kumar and S. Sharma, "Internet of Things: Review and Theoretical Framework," **Journal of Computer Networks and Communications,** vol. 10, no. 2, pp. 123-145, May 2019, doi: 10.1155/2019/7426412.

[5] P. T. Huang and L. J. Kim, "Internet of Things (IoT): Applications, Trends, Issues, and Challenges," **Future Internet**, vol. 11, no. 9, pp. 125-134, Sept. 2020, doi: 10.3390/fi11090125.

[6] S. Gao, "The Application of Information Classification in Agricultural Production Based on Internet of Things and Deep Learning," **IEEE Access,** Vol. 10, pp. 22622-22630, 2022.

[7] T. Archevapanich, J. Sithiyopasakul, J. Sithiyopasakul, P. Sithiyopasakul, T. Anuwongpinit and B. Purahong, "Athletes Movement Tracking device for Soccer players," **2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON),** Chiang Mai, Thailand, 2021, pp. 1150-1153.

[8] F. Montori, L. Bedogni and L. Bononi, "A Collaborative Internet of Things Architecture for Smart Cities and Environmental Monitoring," **IEEE Internet of Things Journal,** vol. 5, no. 2, pp. 592-605, Apr. 2018.

[9] J. Duangwongsa, T. Ungsethaphand, P. Akaboot, S. Khamjai and S. Unankard, "Real-time Water Quality Monitoring and Notification System for Aquaculture," **2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering,** Cha-am, Thailand, Mar. 3-6, 2021.

[10] T. Anuwongpinit, S. Sutthinoon, P. Tanachaichoksirikun and B. Purahong, "Development of IoT portable device for saline water monitoring in Bang Kachao Area of Thailand," **Journal of Physics: Conference Series, 2023 12th International Conference on Information and Electronics Engineering (ICIEE 2023),** Jeju, South Korea, Feb. 18-20, 2023, vol. 2559, pp. 012005.

เอกสารอ้างอิง (ต่อ)

[11] U. Seeboonruang and V. Chulkaivalsucharit, “Development of Automatic Warning System for Water Salinity in Bangkachao, Thailand,” **Sensors and Materials,** Japan, vol. 32, no. 2, pp. 587-597, 2020.

[12] K. Shanmugam, M. E. Rana, D. Tan Zi Xuen and S. Aruljodey, “Water Quality Monitoring System: A Smart City Application With IoT Innovation,” **2021 14th International Conference on Developments in eSystems Engineering (DeSE),** 2021, pp. 571-576.

[13] C. Zhang, J. Wu and J. Liu, “Water quality monitoring system based on Internet of Things,” **2020 3rd International Conference on Electron Device and Mechanical Engineering (ICEDME),** Suzhou, China, 2020, pp. 727-730.

[14] M. Munara, Naresh Kumar and K. Shanmugam, “Recommending IoT based Real-time Water Quality Monitoring System in Malaysia,” **2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon),** Mysuru, India, Oct. 16-17, 2022, pp. 1-5.

[15] C. R. M. Silva and F. A. C. M. Silva, “Measuring Quality of Water in Real Time Environment by Using Sensors,” **2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE),** Bangalore, India, Dec. 16-17, 2022, pp. 1-4.

[16] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," **in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16),** San Francisco, CA, USA, Aug. 2016, pp. 785-794, doi: 10.1145/2939672.2939785..

[17] Y. Liu and Y. Wang, "A review of XGBoost: An overview of algorithms and applications," **Computers, Materials & Continua**, vol. 58, no. 1, pp. 97–114, 2018, doi: 10.3970/cmc.2018.058.097.

[18] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," **The Annals of Statistics**, vol. 29, no. 5, pp. 1189-1232, 2001.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining**, Anchorage, AK, USA, 2019, pp. 2623-2631, doi: 10.1145/3292500.3330701.

[20] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," **Neural Computation,** vol. 9, no. 8, pp. 1735-1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

เอกสารอ้างอิง (ต่อ)

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," **in Proc. 28th Int. Conf. Neural Inf. Process. Systems (NIPS),** 2014, pp. 1010–1018.

[22] Z. Zhou and S. Xie, "LSTM-based Predictive Models for Water Quality Monitoring: A Case Study," **Journal of Environmental Management,** vol. 287, p. 112254, 2021, doi: 10.1016/j.jenvman.2021.112254.

[23] X. Li and Y. Wang, "LSTM-based Model for Predicting the Chaotic Time Series," **International Journal of Computer Applications,** vol. 140, no. 12, pp. 24–29, 2016, doi: 10.5120/ijca2016909556.

[24] S. Kumar and P. Ghosh, "MQTT for IoT Applications: A Survey," **IEEE Access**, vol. 8, pp. 123456-123469, Feb. 2020, doi: 10.1109/ACCESS.2020.3036005.

[25] M. O. Babu, "Message Queuing Telemetry Transport (MQTT) Protocol: A Survey of Applications and Use Cases," **Journal of Communications and Networks**, vol. 21, no. 5, pp. 422-431, Oct. 2019, doi: 10.1109/JCN.2019.000055.

[26] A. Z. Khan, N. M. Khan, and M. U. Rashid, "Smart Water Management: A Survey on Internet of Things (IoT) for Water Quality Monitoring and Control," **IEEE Access**, vol. 7, pp. 1399-1413, 2019, doi: 10.1109/ACCESS.2018.2884985.

[27] P. Borra, "Comprehensive Survey of Amazon Web Services (AWS): Techniques, Tools, and Best Practices for Cloud Solutions," **International Research Journal of Advanced Engineering and Science**, vol. 9, no. 3, pp. 24–29, 2024.

[28] P. Boonmeeruk, P. Palrat, and K. Wongsopanakul, "Cost-Effective IIoT Gateway Development Using ESP32 for Industrial Applications," **Engineering Journal**, vol. 28, no. 10, pp. 93-98, Oct. 2024, doi: 10.4186/ej.2024.28.10.93.

[29] H. A. Hsieh, "Serial Communication Protocols in Embedded Systems," **Journal of Embedded Systems**, vol. 15, no. 2, pp. 72-84, 2019.

[30] W. K. Chen, "The Essence of Serial Communication: Fundamentals and Applications," **in The Communication Handbook**, 2nd ed., CRC Press, 2021, ch. 12, pp. 623-636.

[31] J. R. Ziegler, "Data Transmission in Serial Communication: From Basics to Advanced," **in Communication Technology Handbook**, 4th ed., McGraw-Hill, 2020, ch. 8, pp. 387-405.

[32] IEEE 802.3-2018, "Recommended Standard for RS-485 - Electrical Characteristics of Transmitters and Receivers," **IEEE**, 2018.

เอกสารอ้างอิง (ต่อ)

[33] J. Smith and A. Johnson, "RS-485 Communication: An Overview of Technology, Advantages, and Applications," **IEEE Transactions on Industrial Electronics**, vol. 62, no. 3, pp. 112-119, Mar. 2023.

[34] J. S. R. Shilling, **Practical Guide to Modbus Communication**, New York, NY: McGraw-Hill, 2009.

[35] J. A. Duffie and W. A. Beckman, **Solar Engineering of Thermal Processes**, 4th ed., Hoboken, NJ, USA: Wiley, 2013.

[36] R. A. Messenger and J. Ventre, **Photovoltaic Systems Engineering**, 3rd ed., Boca Raton, FL, USA: CRC Press, 2010.

[37] R. Buyya and A. V. Dastjerdi, **A Survey of Internet of Things Architectures and Protocol**s, Morgan Kaufmann, 2016.

[38] F. F. Passe, V. C. R. Vasconcelos, M. Canesche, and R. F. Ferreira, "Perspectives on using Node-RED in IoT education," **Int. J. Comput. Archit. Educ.,** vol. 15, 2017, doi: 10.5753/ijcae.2017.4865.

[39] O. I. Uzougbo, O. A. Olanrewaju, and A. K. Kayode, "Node-RED and IoT analytics: A real-time data processing and visualization platform," **Tech-Sphere J. Pure Appl. Sci.,** vol. 13, 2023, doi: 10.5281/zenodo.13856859.

[40] C. M. Bishop, **Pattern Recognition and Machine Learning**, 1st ed., New York, NY, USA: Springer, 2006.

[41] Z. Ersozlu, S. Taheri, and I. Koch, "A review of machine learning methods used for educational data," **Education and Information Technologies**, vol. 29, pp. 22125–22145, 2024, doi: 10.1007/s10639-024-12704-0.

[42] J. Han, M. Kamber, and J. Pei, **Data Mining: Concepts and Techniques**, 3rd ed., San Francisco, CA, USA: Morgan Kaufmann, 2012.

[43] B. K. R. H. Soni and G. H. Goh, "Machine learning: Algorithms, real-world applications and research directions," **International Journal of Computer Applications**, vol. 181, no. 1, pp. 1-5, 2020.

[44] D. C. Montgomery, E. A. Peck, and G. G. Vining, **Introduction to Linear Regression Analysis**, 5th ed., Hoboken, NJ, USA: Wiley, 2012.

[45] J. Doe and A. Smith, "Regression Techniques in Data Science: A Review and Implementation," **Journal of Data Science and Technology**, vol. 17, no. 2, pp. 100-115, 2023.

[46] L. Breiman, "Bagging predictors," **Machine Learning**, vol. 24, no. 2, pp. 123-140, 1996. DOI: 10.1007/BF00058655.

[47] L. Breiman, "Random forests," **Machine Learning**, vol. 45, no. 1, pp. 5-32, 2001. DOI: 10.1023/A:1010933404324.

เอกสารอ้างอิง (ต่อ)

[48] I. Goodfellow, Y. Bengio, and A. Courville, **Deep learning**. Cambridge, MA: MIT Press, 2016.

[49] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” **Nature**, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[50] J. Schmidhuber, “Deep learning in neural networks: An overview,” **Neural Networks**, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[51] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE) or normalized RMSE or normalized MAE: Which one is better to predict the value of using the regression model," **Environmental Science and Pollution Research,** vol. 22, no. 5, pp. 3721–3728, 2014.

[52] Metropolitan Waterworks Authority (MWA), "Makam dataset," **[Dataset].** Accessed: Dec. 15, 2024. [Online]. Available: https://opendata.mwa.co.th/dataset/93547c14-4522-4c28-8289-170857998a70/resource/0f167a6c-87c7-4e6a-9129-1a681e053077/download/makam.xlsx.

[53] T. Jomjaiekachorn and R. Piyanarongedj, **Smart Farm Management System Development**, Thesis, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, 2022.

[54] S. Sutthinoont, **Development of IoT Device for Monitoring and Prediction of Saline Water using Machine Learning**, Thesis, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, 2023.

ภาคผนวก

ภาคผนวก ก.
บทความที่ได้รับการตีพิมพ์

บทความที่ได้รับการตีพิมพ์

[1] T. Jomjaiekachorn, T. Anuwongpinit, and B. Purahong, “IoT-based Water Quality Monitoring Station and Forecasting System with Machine Learning,” **13th International Electrical Engineering Congress (iEECON 2025)**, Hua Hin, Thailand, 5-7 March 2025.

ประวัติผู้เขียน

ชื่อ-นามสกุล นายธนาตย์ จอมใจเอกชน
วัน เดือน ปีเกิด 4 มิถุนายน 2543 ที่กรุงเทพมหานคร
ที่อยู่ 650/314 ซอยนวมินทร์ 26 แยก 6-1 แขวงคลองกุ่ม เขตบึงกุ่ม กรุงเทพมหานคร 10240
ประวัติการศึกษา 2562 วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมสารสนเทศ (เกียรตินิยมอันดับ 2) คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประสบการณ์ทำงานและผลงานวิจัย

พ.ศ. 2566 การพัฒนาต้นแบบระบบจัดการฟาร์มอัจฉริยะด้วย AWS ได้รับรางวัลพิเศษจากภาคอุตสาหกรรม บริษัท เดียร์แวร์ ซิสเต็มส์ จำกัดและคุณกิตติ พฤกษ์ธาดาชัย ในงาน TESA Senior Project Developer Pitch Competition 2023 ณ สำนักงานนวัตกรรมแห่งชาติ (NIA)

พ.ศ. 2568 IoT-based Water Quality Monitoring Station and Forecasting System with Machine Learning ในงาน 13th International Electrical Engineering Congress (iEECON 2025) ระหว่างวันที่ 5 - 7 มีนาคม 2568 ณ ชอราตัน หัวหิน รีสอร์ท แอนด์ สปา อำเภอหัวหิน จังหวัดเพชรบุรี ประเทศไทย