



รายงานปฏิบัติงานสหกิจศึกษา

เรื่อง

ระบบตรวจวัดและติดตามปริมาณมลพิษทางอากาศ
Continuous Emission Monitoring System

โดย

นายยุทธชัย คำมีภักดิ์
รหัสนักศึกษา 651540005011-9

ปฏิบัติงาน ณ

บริษัท อะนาไลติกอล ซิสเต็ม เอ็นจิเนียริง

บริษัท อะนาไลติกอล
ซิสเต็ม เอ็นจิเนียริง
(ประเทศไทย) จำกัด

1/9 ซอยสามแยกบายพาส ถนนสุขุมวิท ตำบลเนินพระ อำเภอเมืองระยอง จังหวัดระยอง 21150

รายงานนี้เป็นส่วนหนึ่งของการเรียนรายวิชาสหกิจศึกษา
สาขาวิชา วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยกาฬสินธุ์
ปีการศึกษา 2568



รายงานปฏิบัติงานสหกิจศึกษา

เรื่อง

ระบบตรวจวัดและติดตามปริมาณมลพิษทางอากาศ
Continuous Emission Monitoring System

โดย

นายยุทธชัย คำมีภักดิ์
รหัสนักศึกษา 651540005011-9

ปฏิบัติงาน ณ

บริษัท อะนาไลติกอล ซิสเต็ม เอ็นจิเนียริง

บริษัท อะนาไลติกอล
ซิสเต็ม เอ็นจิเนียริง
(ประเทศไทย) จำกัด

1/9 ซอยสามแยกบายพาส ถนนสุขุมวิท ตำบลเนินพระ อำเภอเมืองระยอง จังหวัดระยอง 21150

รายงานนี้เป็นส่วนหนึ่งของการเรียนรายวิชาสหกิจศึกษา
สาขาวิชา วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยกาฬสินธุ์
ปีการศึกษา 2568

กิตติกรรมประกาศ (Acknowledgement)

การที่ข้าพเจ้าได้มาปฏิบัติงานสหกิจศึกษา ณ บริษัทอะนาไลติกอล ซิสเต็ม เอ็นจิเนียริง ตั้งแต่วันที่ 23 เดือน มิถุนายน พ.ศ. 2568 ถึง วันที่ 17 เดือน ตุลาคม พ.ศ. 2568 ส่งผลให้ข้าพเจ้าได้รับความรู้และประสบการณ์ต่างๆ ที่มีค่ามากมาย สำหรับรายงานวิชาสหกิจศึกษา ฉบับนี้ สำเร็จลงได้ด้วยดีจากความร่วมมือและสนับสนุนจากหลายฝ่าย ดังนี้

1. ชื่อ-สกุล นายคมกฤษ ตำแหน่ง..... นายคมกฤษ พรหมรักษา
ผู้จัดการฝ่ายวิศวกรรม
2. ชื่อ-สกุล.....ตำแหน่ง..... อาจจะขอพี่แชมป์ พี่โต้ง
พี่ดริน พี่เดี่ยวใส่ชื่อดู
3. ชื่อ-สกุล.....ตำแหน่ง.....
4. ชื่อ-สกุล.....ตำแหน่ง.....
5. ชื่อ-สกุล.....ตำแหน่ง.....
6. ระบุคนอื่น ๆ ที่ได้ให้คำแนะนำ ช่วยเหลือ ในการจัดทำรายงาน

นอกจากนี้ขอขอบคุณบิดา - มารดา อาจารย์ และบุคคลท่านอื่นๆ ที่ไม่ได้กล่าวนามไว้ ณ ที่นี้ ซึ่งท่านเหล่านี้ได้กรุณาให้คำแนะนำช่วยเหลือในการจัดทำรายงานฉบับนี้ ข้าพเจ้าใคร่ขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่มีส่วนร่วมในการให้ข้อมูล คำแนะนำ และวิธีการของการปฏิบัติงาน รวมถึงเป็นที่ปรึกษาในการจัดทำรายงานฉบับนี้จนเสร็จสมบูรณ์

ลงชื่อ

(.....)

นักศึกษาผู้ปฏิบัติงานสหกิจศึกษา

วันที่.....เดือน.....พ.ศ.....

บทคัดย่อ (Abstract)

การเข้าร่วมโครงการสหกิจศึกษาเป็นการเปิดโอกาสให้นักศึกษาได้เพิ่มพูนความรู้และประสบการณ์วิชาชีพที่ตรงตามสาขาวิชาที่เรียน และสามารถนำความรู้ที่ได้เรียนไปใช้ให้เกิดประโยชน์ ที่ก่อให้เกิดการเรียนรู้และพัฒนาศักยภาพให้มีความพร้อมสำหรับการทำงานที่สอดคล้องกับความต้องการของสถานการณ์ประกอบการ ตามความต้องการของประเทศและพร้อมที่จะทำงานในระดับสังคมโลกในอนาคตเพื่อผลิตบัณฑิตที่มีศักยภาพในการทำงานมีจรรยาบรรณที่ดีในสาขาวิชาชีพอันจะส่งผลต่อการพัฒนาประเทศทั้งด้านเศรษฐกิจและสังคม

ข้าพเจ้าได้รับมอบหมายให้ปฏิบัติงานในตำแหน่ง นักศึกษาฝึกงาน บริษัท อะนาไลติกคอล ซิสเต็มเอ็นจิเนียริง จำกัด ซึ่งมี นายคมกฤช เป็นผู้มอบหมายงาน โดยงานที่ข้าพเจ้าได้รับมอบหมายเกี่ยวข้องกับ งานโปรแกรมแดชบอร์ด CEMS เป็นต้น จากการปฏิบัติงานดังกล่าวข้างต้นทำให้ข้าพเจ้าได้รับประโยชน์มากมาย ไม่ว่าจะเป็นการวางแผนก่อนการทำงาน การบริหารจัดการเวลาที่มีอยู่อย่างจำกัด การปรับตัวให้กับผู้อื่น การค้นพบข้อบกพร่องของตนเองในด้านความรู้และทักษะการทำงาน ซึ่งสิ่งต่างๆ เหล่านี้เป็นบทเรียนแรกที่ใช้เป็นแนวทางในการพัฒนาด้านการทำงานในอนาคตของข้าพเจ้า

บริษัท อะนาไลติกคอล
ซิสเต็ม เอ็นจิเนียริง
(ประเทศไทย) จำกัด

นายคมกฤช พรหมรักษา

ให้เข้ากับผู้อื่น

คำสำคัญ CEMS, Dashboard, Modbus TCP/IP, FastAPI, InfluxDB, React

สารบัญ

บทที่ 1 บทนำ	1
1.1 ชื่อและที่ตั้งสถานประกอบการ	1
1.2 ลักษณะการประกอบการ ผลิตภัณฑ์ การให้บริการหลักขององค์กร	1
1.3 รูปแบบการจัดองค์การและการบริหารงานขององค์กร	2
1.4 ตำแหน่งและลักษณะงานที่ได้รับมอบหมาย	2
1.5 ชื่อและตำแหน่งของพนักงานที่ปรึกษา	3
1.6 ระยะเวลาที่ปฏิบัติงาน	3
บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง	4
2.1 ความเป็นมาและความสำคัญของปัญหา	4
2.2 วัตถุประสงค์การปฏิบัติงาน	4
2.3 ขอบเขตของโครงการ	5
2.3.1 ระบบย่อยและหน้าที่ (System Components)	5
2.4 ประโยชน์ที่คาดว่าจะได้รับ	5
2.5 ขั้นตอนและวิธีการดำเนินงาน	6
2.6 อุปกรณ์และเครื่องมือที่ใช้	8
2.6.10 Influx DB	17
2.7 รายละเอียดของโครงการ	22
บทที่ 3 สรุปผลการปฏิบัติงาน	34
3.1 สิ่งที่คาดหวัง	34
3.2 ประโยชน์ที่ได้รับจากการปฏิบัติงาน	34
3.3 วิเคราะห์จุดเด่น จุดด้อย โอกาส อุปสรรค (Swot Analysis)	34
บรรณานุกรม	38
ภาคผนวก	40

สารบัญภาพ

ภาพที่ 1 โลโก้บริษัท	1
ภาพที่ 2 ไดอะแกรมขั้นตอนและวิธีดำเนินการ	6
ภาพที่ 3 ไดอะแกรมออกแบบส่วนติดต่อผู้ใช้	7
ภาพที่ 4 วิชวล สตูดิโอ โค้ด	9
ภาพที่ 5 กิต	10
ภาพที่ 6 รีแอคต์ + ไวท์	11
ภาพที่ 7 เทลวินด์	12
ภาพที่ 8 เว็บช็อคเก็ต	13
ภาพที่ 9 เรสท์ เอพีไอ	13
ภาพที่ 10 ไพธอน	14
ภาพที่ 11 ฟาสต์ เอ พี ไอ	15
ภาพที่ 12 มอดบัส	17
ภาพที่ 13 อินฟลัก ดี ปี	18
ภาพที่ 14 ด็อกเกอร์	19
ภาพที่ 15 อิเล็กตรอน บิลด์เดอร์	20
ภาพที่ 16 มอดบัส สเลฟ	21
ภาพที่ 17 วีเอ็ม แวร์	22
ภาพที่ 18 ภาพรวมของระบบ	22
ภาพที่ 19 สถาปัตยกรรมการไหลของข้อมูล	23
ภาพที่ 20 หน้าหลัก	25
ภาพที่ 21 หน้ากราฟ	25
ภาพที่ 22 หน้าดูข้อมูลย้อนหลัง	26
ภาพที่ 23 หน้าสถิติ	26
ภาพที่ 24 หน้าโบลแบ็ค	27
ภาพที่ 25 แท็บการเชื่อมต่ออุปกรณ์	28
ภาพที่ 26 กำหนดสัญญาณ	28
ภาพที่ 27 แท็บกำหนดการ์ดที่จะแสดงในหน้าโฮม	29
ภาพที่ 28 แท็บเชื่อมต่อสัญญาณสถิติ/อะลาม	29
ภาพที่ 29 หน้าแรกที่ยังไม่มีการล็อกอิน	30

ภาพที่ 30 หน้าเลือกสื่ออื่น	30
ภาพที่ 31 การเข้าถึงแบบแอดมิน	31
ภาพที่ 32 การเข้าถึงแบบผู้ใช้หรือผู้เข้าชม	31
ภาพที่ 33 หน้าฐานข้อมูล	32
ภาพที่ 34 การรัน ด็อกเกอร์ ของ ฐานข้อมูล	32

บทที่ 1

บทนำ

1.1 ชื่อและที่ตั้งสถานประกอบการ

ชื่อสถานประกอบการ บริษัท อะนาไลติกอล ซิสเต็ม เอ็นจิเนียริง (ประเทศไทย) จำกัด

ที่ตั้งสถานประกอบการ 1/9 ซอยสาม, แยกบายพาส, ต.เนินพระ, อำเภอเมืองระยอง ระยอง 21150

ปรับอัตราส่วนเป็น
1.877:1

1/9 ซอยสามแยกบายพาส
ถนนสุขุมวิท ตำบลเนินพระ
อำเภอเมืองระยอง จังหวัดระยอง
21150



ภาพที่ 1 โลโก้บริษัท

1.2 ลักษณะการประกอบการ ผลิตภัณฑ์ การให้บริการหลักขององค์กร

บริษัท อะนาไลติกอล ซิสเต็ม เอ็นจิเนียริง (ประเทศไทย) จำกัด เป็น System Integrator/All-in-one solution provider ด้านเครื่องมือวิเคราะห์ (Analyzer) และระบบวิเคราะห์กระบวนการ/สิ่งแวดล้อมสำหรับโรงงานอุตสาหกรรมในประเทศไทย มีทีมผู้เชี่ยวชาญและประสบการณ์มากกว่า 20 ปี ให้บริการตั้งแต่ศึกษาความเป็นไปได้ ออกแบบวิศวกรรม ประกอบระบบ ทดสอบ-เดินระบบ จนถึงบริการหลังการขายครบวงจร

กระบวนการผลิต

1.2.1 ผลิตภัณฑ์/โซลูชันหลัก

- 1 CEMS Continuous Emission Monitoring System
- 2 SWAS Steam & Water Analysis System
- 3 โซลูชัน COD/BOD, Gas Chromatography (GC), Gas/Water Analyzer และ Analyzer House/Shelter

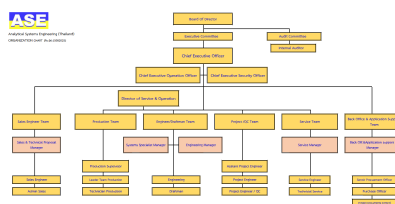
- 4 ระบบ Data Acquisition/Reporting (DAS) และแดชบอร์ดติดตามแบบเรียลไทม์(ตาม ~~หมวด “Products & Services / System Solution” บนเว็บไซต์บริษัท~~)

1.2.2 การให้บริการหลัก

- 1 Consulting & Engineering เก็บ บัชข้อมูลกระบวนการ ออกแบบตามมาตรฐาน DEP/Licenser และความต้องการลูกค้า
- 2 Turnkey Installation & Commissioning: ผลิต/ประกอบภายในบริษัท ทดสอบ-เดินระบบโดยทีมวิศวกร ASE
- 3 After-sales & Maintenance: บำรุงรักษาเชิงป้องกัน (PM), ตรวจสอบ-แก้ไขปัญหา (Troubleshooting) และดูแลระยะยาว
- 4 Training/Knowledge Transfer และบริการสนับสนุนอื่น ๆ ตามหน้างาน(อ้างอิง ~~“เราดูแลทุกขั้นตอน” และหน้า Service Business~~)

ใส่ผังองค์กร

1.3 รูปแบบการจัดองค์กรและการบริหารงานขององค์กร



1.4 ตำแหน่งและลักษณะงานที่ได้รับมอบหมาย

ตำแหน่งที่ได้รับมอบหมาย นักศึกษาฝึกงาน

1.4.1 บทบาทและหน้าที่(Roles and Responsibilities)

- 1 พัฒนาโปรแกรม Continuous Emission Monitoring System ให้เทียบเท่าโปรแกรมเชิงพาณิชย์
- 2 ออกแบบและพัฒนา โมดูลสื่อสาร Modbus TCP/IP (ใช้ Pymodbus) เพื่ออ่าน/แปลค่ารีจิสเตอร์จากเครื่องมือวิเคราะห์
- 3 พัฒนา บริการกลาง (Backend API) ด้วย Python FastAPI รวมถึงงาน ETL/แปลงหน่วย/คำนวณค่า (เช่น O₂ Corrected)
- 4 ออกแบบสคีมาบันทึกข้อมูลและเขียนข้อมูลแบบเรียลไทม์ลง InfluxDB พร้อมกำหนด Retention/Index ที่เหมาะสม
- 5 พัฒนา Dashboard แบบ Real-time ด้วย React + Ant Design (กราฟแนวเวลา, การ์ดสถานะ, หน้า Trend/Report)
- 6 สร้าง แจ้งเตือน (Alert/Notification) และ Export ตามรูปแบบที่ใช้ส่ง DIW

- 7 จัดทำ เอกสาร: API Spec, Modbus Mapping, คู่มือใช้งาน/ติดตั้ง, Test Plan & Test Report
- 8 ร่วม ทดสอบ (Unit/Integration/UAT) ปรับปรุงคุณภาพ และสนับสนุนการ Demo/ส่งมอบ

1.5 ชื่อและตำแหน่งของพนักงานที่ปรึกษา

นาย คมกฤษ ตำแหน่ง หัวหน้า Engineering

นายคมกฤษ พรหมรักษา
ผู้จัดการฝ่ายวิศวกรรม



1.6 ระยะเวลาที่ปฏิบัติงาน

ตั้งแต่วันที่ 23 เดือน มิถุนายน พ.ศ. 2568 ถึง 17 เดือน ตุลาคม พ.ศ. 2568

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ความเป็นมาและความสำคัญของปัญหา

อุตสาหกรรมการผลิตในประเทศไทยต้องปฏิบัติตามข้อกำหนดด้านสิ่งแวดล้อมและความปลอดภัยอย่างเคร่งครัด โดยเฉพาะการติดตามและรายงานการปล่อยอากาศเสียจากปล่องระบาย (Stack/Chimney) ให้เป็นไปตามมาตรฐานของหน่วยงานกำกับดูแล การมีระบบ เก็บ ประมวลผล แสดงผล รายงาน ข้อมูลการปล่อยอย่างต่อเนื่อง (Continuous Emission Monitoring System CEMS) และระบบจัดเก็บ/รายงานข้อมูล (Data Acquisition System DAS) ที่เชื่อถือได้ จึงเป็นปัจจัยสำคัญต่อความต่อเนื่องทางธุรกิจและการบริหารความเสี่ยงด้านกฎหมาย ทว่าในทางปฏิบัติ องค์กรมักพึ่งพาโปรแกรมเชิงพาณิชย์ซึ่ง มีราคาสูง ทั้งค่าลิขสิทธิ์ แรกเข้า ค่าบำรุงรักษารายปี และค่าโมดูลเสริม อีกทั้งยังไม่ตอบโจทย์ฟังก์ชันเฉพาะ ที่จำเป็นต่อหน่วยงาน เช่น การปรับสูตรคำนวณ/หน่วย การกำหนดรูปแบบไฟล์รายงานสำหรับส่งหน่วยงานรัฐ การออกแบบแดชบอร์ดแบบยืดหยุ่น หรือการเชื่อมต่อกับอุปกรณ์/ฐานข้อมูลที่หลากหลาย เมื่อความต้องการเปลี่ยนเพียงเล็กน้อย องค์กรจึงต้อง ซื้อเพิ่มหรือเปลี่ยนโปรแกรมใหม่ซ้ำ ๆ ทำให้เกิดค่าใช้จ่ายต่อเนื่องโดยไม่จำเป็น ทั้งค่าใบอนุญาต อัปเดต และการฝึกอบรม และเสี่ยงต่อภาวะพึ่งพาผู้ขาย (vendor lock-in) ซึ่งชะลอการตอบสนองต่อข้อกำหนดและงานภาคสนาม จาก Pain Points ดังกล่าว ปัญหาจริงของโครงการนี้ คือองค์กร ต้องการพัฒนาโปรแกรม DAS ที่มีความสามารถเทียบเท่าโปรแกรมในตลาด แต่สามารถปรับแต่งได้เองตามบริบทงาน ทั้งด้านการเชื่อมต่อ (เช่น Modbus TCP/IP), การคำนวณ/แปลงหน่วย, การจัดเก็บแบบ time-series, แดชบอร์ดเรียลไทม์ และการส่งออกรายงานตามรูปแบบมาตรฐาน—เพื่อลดต้นทุนระยะยาว เพิ่มความยืดหยุ่น และหลีกเลี่ยงการ ต้องซื้อใหม่และจ่ายค่าโปรแกรมทุกครั้ง เมื่อมีการเปลี่ยนแปลงความต้องการในอนาคต

2.2 วัตถุประสงค์การปฏิบัติงาน

2.2.1 เพื่อพัฒนาโปรแกรม DAS/CEMS ที่มีความสามารถเทียบเท่าโปรแกรมในตลาด ที่สามารถปรับแต่งได้เองให้เหมาะกับงานจริง

2.2.2 เพื่อออกแบบสถาปัตยกรรมและเป็นเจ้าของโปรแกรมได้เอง

2.2.3 เพื่อจัดทำฟังก์ชันหลักที่จำเป็นต่อการใช้งานจริง ได้แก่ การคำนวณ/แปลงหน่วยม Dashboard แบบเรียลไทม์, ระบบแจ้งเตือน

2.2.4 เพื่อลดต้นทุนระยะยาวและหลีกเลี่ยงการต้องซื้อใหม่ทุกครั้ง ด้วยการพัฒนา-ดูแลในองค์กร ตั้งค่าปรับแต่งได้เอง และใช้ซอฟต์แวร์/มาตรฐานที่แพร่หลาย



2.3 ขอบเขตของโครงการ

2.3.1 ระบบย่อยและหน้าที่ (System Components)

- 1 Data Acquisition เชื่อมต่ออุปกรณ์ผ่าน Modbus TCP/IP (ใช้ Pymodbus) ตาม Tag/Address Map ที่กำหนด
- 2 Processing & API ให้บริการ Backend API (FastAPI) สำหรับคำนวณ/แปลงหน่วย (เช่น Scaling, Averaging, O₂ Corrected), ตรวจสอบความถูกต้องของข้อมูล และให้บริการกับ แดชบอร์ด/การส่งออก
- 3 Time-series Storage จัดเก็บข้อมูลใน InfluxDB 2.x กำหนด Measurement/Tag/Field, นโยบาย Retention และวิธี Backup/Restore สำหรับสภาพแวดล้อมทดสอบ
- 4 Dashboard & Reporting ส่วนหน้าเว็บ React + Ant Design แสดงผล Real-time, หน้าต่างค่าเฉลี่ย/แนวโน้ม (Trend), ค้นหา/กรองข้อมูล, ดาต้าไหลรายงาน
- 5 Alert/Notification ตั้งค่าเงื่อนไขเตือน (Threshold/Status) และการแจ้งเตือนเบื้องต้น
- 6 DIW Export ส่งออกไฟล์ตามรูปแบบที่ต้องใช้ยื่นรายงาน (โครงสร้างฟิลด์/หน่วย/เวลา)

2.3.2 ขอบเขตที่อยู่ในการดำเนินงาน (In-scope Functions)

- 1 พัฒนาโปรแกรม DAS/CEMS แบบ Web Application ที่มีความสามารถ เทียบเท่าโปรแกรมในตลาดในฟังก์ชันหลัก
- 2 รองรับอุปกรณ์/ซีมิเตอร์อย่างน้อย 1 แหล่งข้อมูล ต่อหลายจุดวัด (multi-point) ผ่าน Modbus TCP/IP
- 3 คำนวณค่าพื้นฐานที่ใช้จริง (Scaling, Averaging, O₂ Corrected, หน่วย/ทศนิยม) และตรวจสอบข้อมูลผิดปกติ (missing/negative/filter ขั้นต้น)
- 4 จัดเก็บข้อมูลแบบ time-series และเรียกดูย้อนหลังได้ตามช่วงเวลา
- 5 แดชบอร์ดแบบเรียลไทม์ (การ์ดสถานะ, กราฟแนวเวลา, ตาราง) พร้อม Export CSV/Excel และ DIW
- 6 จัดทำเอกสาร API Spec, Modbus Mapping/Tag List, Install/Operation Manual, Test Plan & Test Report
- 7 เตรียม สภาพแวดล้อม Dev/Staging สำหรับสาธิตการทำงานและทดสอบใช้งาน

2.4 ประโยชน์ที่คาดว่าจะได้รับ

2.4.1 ได้ระบบที่ ตอบโจทย์การใช้งานจริง ครอบคลุมฟังก์ชันหลักของตลาด แต่ปรับแต่งให้ตรงบริบท หน่วยงานตัวเอง

2.4.2 องค์กรมี ความเป็นเจ้าของข้อมูล/ซอร์สโค้ด ลดการพึ่งพาผู้ขาย

2.4.3 ตรวจพบเหตุผิดปกติได้ คุณภาพข้อมูลดีและปฏิบัติตามข้อกำหนดได้

2.4.4 TCO ลดลง จากการไม่มีค่าลิขสิทธิ์ และปรับฟังก์ชันเองได้โดยไม่ต้องซื้อโมดูลเพิ่ม

แปลเป็นภาษาไทยดีกว่า

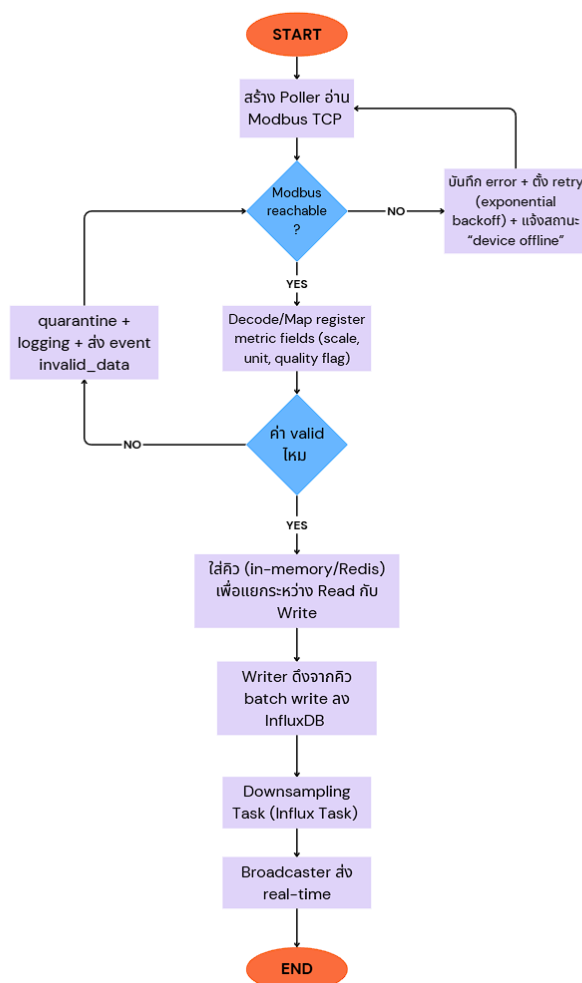
2.5 ขั้นตอนและวิธีการดำเนินงาน

2.5.1 ศึกษาบริบทและเก็บความต้องการ (Requirement Elicitation & Context Study)

- 1 ศึกษา Modbus TCP/IP (Function 03/04/16, Address/Word/Scale/Error/Timeout) และสเปกการรายงาน DIW
- 2 สัมภาษณ์ผู้ใช้งาน/ผู้ดูแลระบบ และ ดูงานโปรแกรมเชิงพาณิชย์ ที่องค์กรใช้อยู่เพื่อทำ Benchmark (ฟังก์ชัน จุดเด่น/ข้อจำกัด)
- 3 สรุป Pain Points/Use cases และกำหนดขอบเขตพีเจอรืรุ่นแรก

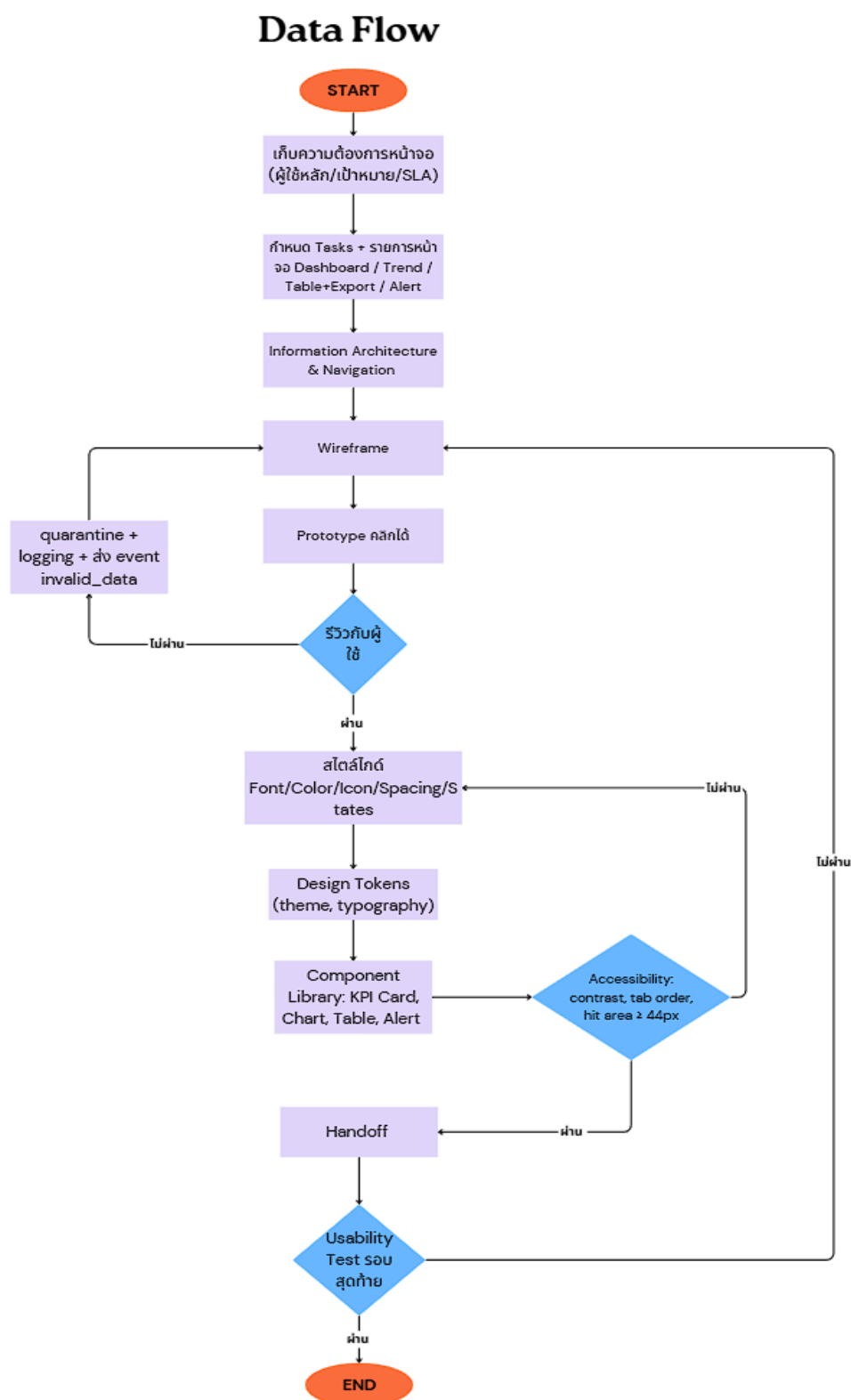
2.5.2 ออกแบบสถาปัตยกรรมและการไหลของข้อมูล (Architecture & Data Flow)

Data Flow



ภาพที่ 2 ไดอะแกรมขั้นตอนและวิธีดำเนินการ

2.5.3 ออกแบบส่วนติดต่อผู้ใช้ (UI/UX Design)



ภาพที่ 3 ไตอะแกรมออกแบบส่วนติดต่อผู้ใช้

2.5.4 ออกแบบข้อมูลและฐานข้อมูล (Data Modeling)

- 1 นิยาม Measurement/Tag/Field, หน่วย/ทศนิยม/การแปลง และกติกาลเวลา (UTC/Local)
- 2 ตั้งค่า InfluxDB Bucket, Retention, Index/Shard, Example Queries

2.5.5 พัฒนาระบบกลางและโมดูลสื่อสาร (Backend Development)

- 1 พัฒนา Modbus Client (Pymodbus) การเชื่อมต่อ, Polling หลายจุดวัด, ตรวจสอบ Error/Retry และ Scaling
- 2 พัฒนา FastAPI Endpoint/Service/Worker, การคำนวณ (เช่น O_2 Corrected, Averaging), เขียนข้อมูลลง InfluxDB, Logging

2.5.6 พัฒนาส่วนหน้า (Frontend Development)

- 1 พัฒนา React + Ant Design เชื่อมต่อแบบ WebSocket/HTTP แสดง Real-time Cards, กราฟแนวเวลา (Trend), ตารางค้นหา/กรอง และปุ่ม Export (CSV/DIW)
- 2 เพิ่มการตั้งค่า Alert/Threshold และหน้าจอจัดการเบื้องต้น

2.5.7 บูรณาการและทดสอบ (Integration & Testing)

- 1 ทดสอบ Unit/Integration/System โดยใช้ Modbus Simulator/อุปกรณ์จริง
- 2 ยืนยันเกณฑ์ยอมรับ: ความถูกต้องการอ่าน $\geq 99\%$, ความหน่วงแดชบอร์ด ≤ 10 วินาที, ทดสอบ 24 ชม. 0 data loss, ไฟล์ DIW Export ผ่านตรวจสอบ

2.5.8 เอกสารและส่งมอบ/สาธิต (Documentation & Handover)

- 1 จัดทำ Install/Operation Manual, User Guide, Modbus Mapping, API Spec และ วีดีโอ/สไลด์สาธิต
- 2 สรุปข้อเสนอแนะ/แผนต่อยอด (Role-based access, Multi-site, โปรโตคอลอื่น เช่น OPC-UA)

2.6 อุปกรณ์และเครื่องมือที่ใช้

2.6.1 Visual Studio Code (VS Code)

Visual Studio Code (VS Code) เป็นโปรแกรมแก้ไขโค้ดสมัยใหม่ของ Microsoft ที่ ฟรีและ ข้ามแพลตฟอร์ม (Windows, macOS, Linux) จุดเด่นคือ เบา เปิดเร็ว แต่ให้ความสามารถระดับ ไกล่เคียง IDE

หัวใจของ VS Code คือ Language Server Protocol (LSP) ทำให้ได้ฟีเจอร์ช่วยเขียนโค้ด ครอบคลุมกับหลายภาษา ได้แก่ IntelliSense/Auto-complete, การไฮไลต์ไวยากรณ์, การตรวจสอบ ข้อผิดพลาด (Lint/Diagnostics), คำสั่ง Go to Definition/References, Refactor/Rename และ

Formatting รองรับ ภาษา อย่าง Python, JavaScript/TypeScript, C/C++, Go, Java, HTML/CSS/JSON/YAML เป็นต้น

ตัวโปรแกรมมี Integrated Terminal และ Debugger ในตัว ตั้งค่า *launch* เพื่อรัน/ไลค์โค้ดได้ทันที พร้อมแผง Problems/Output ช่วยติดตามข้อความแจ้งเตือน นอกจากนี้ยังผนวก Git มาให้—ทำ stage/commit, ดู diff, จัดการ branches/merge ได้จากหน้าต่างเดียว

ความยืดหยุ่นของ VS Code มาจาก Extensions Marketplace ที่มีส่วนขยายหลากหลาย เช่น Python/Pylance, ESLint/Prettier, Docker/Dev Containers, Remote-SSH/WSL, Jupyter/Notebook, REST Client, Markdown, ตลอดจนจิมและไอคอน ผู้ใช้สามารถปรับแต่งได้ลึกผ่าน Settings/settings.json, Keybindings, Snippets, และซิงก์การตั้งค่าด้วย Settings Sync รองรับ Multi-root workspaces เพื่อเปิดหลายโปรเจกต์ในหน้าต่างเดียว



ภาพที่ 4 วิว สติวโอ โค้ด

2.6.2 Git

Git คือระบบควบคุมเวอร์ชันแบบกระจาย (Distributed Version Control System: DVCS) ที่ใช้บันทึกประวัติการแก้ไขซอร์สโค้ดเป็นสแนปชอตต่อเนื่อง โดยแต่ละเครื่องของนักพัฒนามีคลังเก็บ (repository) ฉบับเต็มของตนเองและซิงก์กับรีโมตอย่าง GitHub, GitLab หรือ Bitbucket เพื่อทำงานร่วมกันอย่างปลอดภัย เวอร์กโฟลว์หลักเริ่มจากแก้ไขไฟล์ใน working directory → คัดเลือกไฟล์เข้าสู่ staging area (git add) → บันทึกเป็น commit พร้อมข้อความอธิบาย จากนั้นจึงผลักขึ้นรีโมต (git push) หรือดึงงานเพื่อนร่วมทีม (git pull) การแยกสาขา (branch) ทำให้พัฒนา/ทดลองฟีเจอร์ไปพร้อมกันได้หลายเส้นทางและคงความมั่นคงของสาขาหลัก เมื่อเสร็จงานสามารถรวมประวัติด้วย merge หรือจัดระเบียบด้วย rebase พร้อมแก้ไขความขัดแย้ง (conflict) อย่างมีระบบ นอกจากนี้ Git รองรับแท็ก (tag) สำหรับมาร์กจุดออกเวอร์ชัน, ไฟล์ .gitignore เพื่อกันไฟล์ที่ไม่ต้อง

ติดตาม, ฮุก (hooks) สำหรับสคริปต์อัตโนมัติ และการรีวิวโค้ดผ่าน Pull/Merge Request ซึ่งเชื่อมกับเครื่องมือ CI/CD ได้โดยตรง ผลลัพธ์คือทีมสามารถย้อนเวอร์ชันได้ทันที, ติดตามว่าใครเปลี่ยนอะไรเมื่อใด (traceability), รักษาคุณภาพโค้ด, จัดการออกรุ่นซอฟต์แวร์ได้เป็นระเบียบ และลดความเสี่ยงไฟล์ทับกันเมื่อทำงานร่วมกันจำนวนมาก

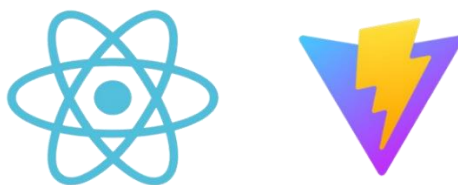


ภาพที่ 5 กิต

2.6.3 React + Vite

React (React.js) เป็นไลบรารี JavaScript แบบโอเพ่นซอร์ส สำหรับพัฒนา ส่วนติดต่อผู้ใช้ (UI) ที่ยึดแนวคิด Declarative และ Component-based โดยแบ่งหน้าจออกเป็นคอมโพเนนต์ย่อยที่นำกลับมาใช้ซ้ำได้ อัปเดตเฉพาะส่วนที่เปลี่ยนด้วยกลไก Virtual DOM และอัลกอริทึม Reconciliation แนวคิดหลักประกอบด้วย JSX (เขียนโครงสร้าง UI ในไฟล์ JS/TS), Props/State, One-way Data Flow, และ Hooks (useState, useEffect, useMemo, useRef, useContext) รวมถึงเครื่องมือเพิ่มประสิทธิภาพอย่าง React.memo และ useCallback ระบบนิเวศรองรับการทำงานร่วมกับ React Router (SPAs), ตัวจัดการสถานะ (Context/Redux/Zustand), ชุดคอมโพเนนต์ Ant Design/Material UI, การทำงานแบบเรียลไทม์ผ่าน WebSocket, การทดสอบด้วย Jest/React Testing Library, และการรองรับ TypeScript อย่างครบถ้วน

Vite เป็นเครื่องมือ Dev Server + Build Tool สำหรับเฟรมเวิร์กเว็บสมัยใหม่ที่อาศัย ES Modules จุดเด่นคือเริ่มเซิร์ฟเวอร์ได้รวดเร็วและมี HMR/Fast Refresh อัปเดตเฉพาะโมดูลที่เปลี่ยนระหว่างบิลด์โปรดักชันใช้ Rollup ใต้ฝา รองรับ code-splitting, tree-shaking, และ dynamic import เพื่อให้ไฟล์ขนาดเล็กและโหลดเร็ว การตั้งค่าอยู่ในไฟล์ vite.config.js(ts) รองรับปลั๊กอินมาตรฐานอย่าง @vitejs/plugin-react (JSX/TSX และ Fast Refresh), การใช้งาน TypeScript, การจัดการไฟล์ CSS/Assets, การกำหนด alias, และตัวแปรสภาพแวดล้อมผ่าน import.meta.env พร้อมคำสั่ง vite สำหรับพัฒนา (vite dev) และบิลด์โปรดักชัน (vite build)



React + Vite

ภาพที่ 6 รีแอกต์ + ไวท์

2.6.4 Tailwind CSS

Tailwind CSS เป็น Utility-First CSS Framework สำหรับการพัฒนาเว็บที่เน้นความยืดหยุ่นและความรวดเร็ว นักพัฒนาสามารถประกอบหน้าตา UI ได้โดยตรงผ่าน class ที่กำหนดคุณสมบัติของสไตล์ เช่น สี ขนาด ระยะห่าง การจัดวาง โดยไม่จำเป็นต้องเขียน CSS แยกเอง ทำให้ปรับแต่งได้ละเอียดระดับองค์ประกอบ (atomic level) และสร้างดีไซน์ได้อิสระมากกว่าการใช้ component library สำเร็จรูป

โครงสร้างของ Tailwind ครอบคลุมยูนิตีลิตี้สำหรับโครงร่าง (Layout) เช่น Flexbox, Grid, Container, Spacing และ Positioning การจัดการข้อความและสี (Typography & Colors) เช่น ฟอนต์ ขนาดตัวอักษร น้ำหนัก สี ขอบ และเงา องค์ประกอบโต้ตอบ (Interactive States) เช่น Hover, Focus, Active และ Dark Mode และการตอบสนองหน้าจอ (Responsive Design) ผ่าน breakpoint utilities เช่น sm:, md:, lg:, xl:

แกนของ Tailwind รองรับ Design Token และ Theming ผ่านไฟล์ `tailwind.config.js` ผู้พัฒนาสามารถปรับแต่ง palette สีหลัก ระยะห่าง รัศมีมุม ฟอนต์ และกำหนด plugin เสริมได้จากศูนย์กลางเดียว เพื่อให้สไตล์สอดคล้องทั้งระบบ และยังรองรับ Dark Mode อย่างเป็นระบบ

ด้านประสิทธิภาพ Tailwind ทำงานร่วมกับ PostCSS และ PurgeCSS (หรือในเวอร์ชันใหม่ใช้ JIT Compiler) เพื่อตัดคลาสที่ไม่ถูกใช้ออก ทำให้ไฟล์ CSS มีขนาดเล็กและโหลดเร็ว อีกทั้งยังรองรับ

การใช้งานร่วมกับ React, Vite, Next.js และเฟรมเวิร์กสมัยใหม่อื่น ๆ ได้สิ้นไหล นักพัฒนาจึงสามารถพัฒนา UI ที่ทั้งรวดเร็วและยืดหยุ่นโดยไม่ต้องผูกติดกับชุดคอมโพเนนต์สำเร็จรูปใด ๆ



ภาพที่ 7 เทลวินด์

2.6.5 WebSocket

WebSocket คือโปรโตคอลสื่อสารแบบเชื่อมต่อคงอยู่และสองทิศทาง (persistent, full-duplex) ระหว่างไคลเอนต์กับเซิร์ฟเวอร์บนการเชื่อมต่อ TCP เพียงเส้นเดียว โดยเริ่มจาก HTTP/1.1 แล้วทำ Upgrade handshake (สถานะ 101 Switching Protocols) จากนั้นทั้งสองฝั่งจะแลกเปลี่ยนข้อมูลเป็นเฟรมข้อความที่มีค่า overhead ต่ำ รองรับทั้งข้อความตัวอักษร (UTF-8) และไบนารี มีเฟรม ping/pong สำหรับตรวจสอบสุขภาพการเชื่อมต่อ และสามารถกำหนด subprotocol เพื่อระบุรูปแบบข้อความร่วมกันได้ ความสำเร็จของ WebSocket คือความหน่วงต่ำและส่ง-รับได้ทันที ต่างจาก polling/long-polling ที่ต้องยิงคำขอซ้ำ ๆ หรือ SSE ที่ส่งข้อมูลได้ทางเดียว เซิร์ฟเวอร์ที่เปิดใช้งานจริงควรใช้ wss:// เพื่อเข้ารหัสด้วย TLS และตรวจหัว Origin เพื่อป้องกันการเชื่อมต่อจากโดเมนที่ไม่พึงประสงค์

ในบริบทโครงการนี้ WebSocket เหมาะอย่างยิ่งกับ แดชบอร์ดแบบเรียลไทม์ และระบบแจ้งเตือน เพราะแบ็กเอนด์ (เช่น FastAPI/Starlette) สามารถผลักข้อมูลค่ามอนิเตอร์ไปยังหน้า React ได้ทันทีโดยไม่ต้องรีเฟรชหน้า ทั้งยังใช้ช่องทางเดียวกันรับคำสั่ง/พารามิเตอร์จากผู้ใช้กลับไปยังเซิร์ฟเวอร์ได้ แนวปฏิบัติที่ควรยึดคือส่งข้อมูลเป็น JSON ที่มีสคีมชัดเจน จัดการการตัดการเชื่อมต่อด้วยกลยุทธ์ reconnect (exponential backoff) และบันทึกเหตุการณ์การเชื่อมต่อ/ข้อผิดพลาดไว้สำหรับตรวจสอบภายหลัง โดยสรุป WebSocket ช่วยให้การอัปเดตค่าเซ็นเซอร์ การแสดงกราฟ และการแจ้งเตือนทำได้สิ้นไหลและสอดคล้องกับความต้องการแบบเรียลไทม์ของระบบ DAS/CEMS



ภาพที่ 8 เว็บซ็อกเก็ต

2.6.6 REST API

REST API คือสถาปัตยกรรมการออกแบบบริการเว็บตามแนวคิด *Representational State Transfer* ที่มุ่งให้การเข้าถึง “ทรัพยากร” (resources) ผ่าน URI ชัดเจน และใช้วิธีการของ HTTP เป็นสัญญาในการทำงาน เช่น GET (อ่าน), POST (สร้าง), PUT/PATCH (แก้ไข), DELETE (ลบ) โดยมีลักษณะสำคัญคือ stateless แต่ละคำขอมีข้อมูลพอเพียงในตัว ไม่ขึ้นกับสถานะเซิร์ฟเวอร์ก่อนหน้า การแลกเปลี่ยนข้อมูลมักใช้ JSON พร้อมกำหนด HTTP status codes สื่อผลลัพธ์ (เช่น 200/201/400/401/404/500), ส่วนหัวสำหรับ caching/ETag/Cache-Control, และแนวปฏิบัติด้าน idempotency (เช่น GET/PUT/DELETE ควรให้ผลลัพธ์คงที่เมื่อเรียกซ้ำ) รวมถึงการจัดหน้า/กรอง/เรียงข้อมูลผ่านพารามิเตอร์ เช่น ?page=...&limit=...&sort=... มิติด้านความปลอดภัยนิยมบังคับ HTTPS, ใช้การยืนยันตัวตนแบบ Bearer/OAuth2-JWT, ควบคุม CORS สำหรับเบราว์เซอร์ และวางแผน versioning ของสัญญา API (เช่น /api/v1/...) เพื่อความเข้ากันได้ในระยะยาว

ในบริบทโครงการนี้ REST API ทำหน้าที่เป็นชั้นสื่อสารระหว่าง Frontend (React/Vite) กับ Backend (FastAPI) เช่น เส้นทางสำหรับอ่านค่าจุดวัด, ดึงกราฟแนวโน้ม/สถิติ, บันทึกการตั้งค่า threshold/tag/unit, และ ส่งออกข้อมูล (CSV/DIW) โดยกำหนดแบบจำลองข้อมูลผ่าน Pydantic เพื่อให้มีการตรวจสอบชนิด/รูปแบบอัตโนมัติ สร้างเอกสาร Open API/Swagger จากโค้ดได้ทันที และตอบกลับด้วยรหัสสถานะและข้อความผิดพลาดที่สม่ำเสมอ ทำให้ส่วนหน้าเรียกใช้งานได้อย่างเชื่อถือได้ ทดสอบอัตโนมัติง่าย และขยายฟังก์ชันในอนาคตโดยไม่กระทบผู้ใช้เดิม



ภาพที่ 9 เรสท์ เอพีไอ

2.6.7 Python

Python เป็นภาษาโปรแกรมระดับสูงแบบอินเทอร์พรีเตอร์และโอเพ่นซอร์สที่โดดเด่นด้วยไวยากรณ์อ่านง่าย (ใช้การเยื้องบรรทัด/indentation) และสนับสนุนหลายแนวทางทั้งเชิงวัตถุ

ฟังก์ชัน และสคริปต์อัตโนมัติ จุดแข็งสำคัญคือมีไลบรารีมาตรฐานครบถ้วนและระบบนิเวศแพ็คเกจขนาดใหญ่บน PyPI พร้อมเครื่องมือจัดการสภาพแวดล้อมอย่าง venv/pip (รวมถึงไฟล์กำหนดค่าทันสมัย เช่น pyproject.toml) ช่วยให้ติดตั้ง ดูแล และทำซ้ำสภาพแวดล้อมได้ง่าย นับตั้งแต่ Python 3.7+ ยังรองรับ type hints และโครงสร้างข้อมูลอย่าง dataclasses ทำให้โค้ดชัดเจนและบำรุงรักษาง่ายขึ้น ทั้งยังมี async/await บน asyncio สำหรับงาน I/O หนักที่ต้องรับ-ส่งข้อมูลต่อเนื่องได้อย่างมีประสิทธิภาพ นอกจากนี้ยังมีเครื่องมือคุณภาพโค้ด เช่น pytest สำหรับทดสอบอัตโนมัติ, black/isort/flake8 สำหรับจัดรูปแบบและตรวจมาตรฐานโค้ด, ระบบล็อก (logging) และอ่านค่าคอนฟิกจากไฟล์ .env ได้สะดวก เมื่อนำไปใช้กับโครงการนี้ Python ทำหน้าที่เป็นแกนของฝั่ง Backend โดยใช้ FastAPI สร้าง REST API และเอกสารอัตโนมัติ (OpenAPI/Swagger), ใช้ Pymodbus ติดต่อ Modbus TCP/IP เพื่ออ่านค่ารีจิสเตอร์จากอุปกรณ์, ใช้ไคลเอนต์ InfluxDB จัดเก็บข้อมูลแบบ time-series พร้อมสคริปต์ ETL/คำนวณ (เช่น scaling/ค่า O₂ Corrected), รองรับงานเบื้องหลัง/กำหนดเวลา (background tasks) และทำงานร่วมกับ Docker เพื่อแพ็คเกจและดีพลอยอย่างสม่ำเสมอ ข้ามแพลตฟอร์มได้ทั้ง Windows/Linux/macOS จึงตอบโจทย์การพัฒนา-ทดสอบ-ส่งมอบระบบ DAS/CEMS ที่ต้องการความคล่องตัว ความน่าเชื่อถือ และการต่อยอดในอนาคต



ภาพที่ 10 ไพธอน

2.6.8 Fast API

Fast API คือเฟรมเวิร์กภาษา Python สำหรับพัฒนา Web API/บริการไมโครเซอร์วิส ที่เน้นความเร็วและความง่ายในการเขียน โดยทำงานบนสถาปัตยกรรม ASGI (รองรับเซิร์ฟเวอร์เช่น Uvicorn/Hypercorn) พื้นฐานของเฟรมเวิร์กประกอบด้วย

- 1 Starlette จัดการ routing, middleware, background tasks, WebSocket
- 2 Pydantic ตรวจสอบและแปลงชนิดข้อมูลอัตโนมัติจาก type hints ของ Python

คุณลักษณะเด่น

- 1 ประสิทธิภาพสูง ออกแบบให้ทำงานแบบ async/await ได้เต็มรูปแบบ เหมาะกับงาน I/O หนัก เช่น รับ-ส่งข้อมูลเรียลไทม์

- 2 ตรวจสอบและจัดรูปข้อมูลอัตโนมัติ ใช้ pydantic/type hints เพื่อ validate และ serialize ข้อมูลเข้า-ออก พร้อมข้อความผิดพลาดที่ชัดเจน
- 3 เอกสาร API อัตโนมัติ สร้าง Open API/JSON Schema และหน้า Swagger UI / ReDoc ให้ทันทีจากโค้ด
- 4 Dependency Injection จัดการการพึ่งพา (เช่น การเชื่อมฐานข้อมูล, การตรวจสอบสิทธิ์) อย่างเป็นระบบและทดสอบง่าย
- 5 ระบบความปลอดภัย มียูทิลิตี้สำหรับ OAuth2/JWT, API Key, Basic Auth และ CORS Middleware
- 6 พีเจอร์เว็บ/งานเบื้องหลัง รองรับ Background Task, WebSocket, การส่ง Response แบบกำหนดเอง, จัดการข้อยกเว้น (Exception handlers)
- 7 ทดสอบสะดวก มี Test Client (อิง Starlette) ใช้เขียน Unit/Integration Test ได้ง่าย

ระบบนิเวศและการใช้งานทั่วไป

เชื่อมต่อได้ทั้ง SQL/NoSQL/Time-series DB, ทำงานร่วมกับ ORM/ODM (เช่น SQLAlchemy, Tortoise ORM, Beanie/ODMantic), รวมถึงการใช้งานร่วมกับคิวงาน/เมสเสจบัส และการดีพลอยผ่าน Docker/Containers ได้อย่างแพร่หลาย



ภาพที่ 11 ฟาสต์ เอ พี ไอ

2.6.9 Modbus

Modbus คือโพรโตคอลสื่อสารอุตสาหกรรมอายุยืน ใช้แลกเปลี่ยนข้อมูลระหว่างอุปกรณ์วัด/ควบคุม (PLC, Analyzer, Transmitter) กับคอมพิวเตอร์หรือระบบ SCADA/DAS จุดเด่นคือ เรียบง่าย มาตรฐานเปิด และรองรับอุปกรณ์หลากหลายแบรนด์ โดยมี “รสชาติ” หลัก 3 แบบ

- 1 Modbus RTU (RS-485/RS-232): กรอบข้อมูลแบบไบนารี + CRC ตรวจสอบความถูกต้อง, จังหวะเวลาเงียบ 3.5 ตัวอักษรคั่นเฟรม
- 2 Modbus ASCII (RS-485/RS-232): ส่งเป็นตัวอักษร ASCII + LRC, ใช้ง่ายแต่ช้ากว่า RTU

3 Modbus TCP/IP (เครือข่าย IP พอร์ต 502) ห่อด้วยหัว MBAP (Transaction ID, Protocol ID, Length, Unit ID) บน TCP, ไม่ใช่ CRC (ตรวจโดย TCP)

โมเดลข้อมูล (Data Model) Modbusกำหนดพื้นที่ข้อมูลมาตรฐาน 4 กลุ่ม ซึ่งแต่ละกลุ่มคือที่อยู่คนละกอง

- 1 Coils บิตอ่าน/เขียน (0/1)
- 2 Discrete Inputs บิตอ่านอย่างเดียว
- 3 Input Registers (3xxxx) รีจิสเตอร์ 16 บิตอ่านอย่างเดียว
- 4 Holding Registers (4xxxx) รีจิสเตอร์ 16 บิตอ่าน/เขียน

เคล็ดลับ เอกสารมักเขียน 40001, 30001 เป็นต้น แต่วิธีส่งจริงใช้ ออฟเซต 0-based (เช่น 40001 มักเท่ากับ address 0)

คำสั่งสำคัญ (Function Codes FC)

- 1 อ่านบิต 01 (Read Coils), 02 (Read Discrete Inputs)
- 2 อ่านรีจิสเตอร์ 03 (Read Holding), 04 (Read Input)
- 3 เขียนเดี่ยว 05 (Write Single Coil), 06 (Write Single Register)
- 4 เขียนหลายตัว 15/0F (Write Multiple Coils), 16/10 (Write Multiple Registers)
- 5 วินิจฉัย/อื่น ๆ 08 (Diagnostics) + ซับฟังก์ชันข้อจำกัดทั่วไปของกรอบคุณค่าขอ (ต่อหนึ่งคำสั่ง) อ่านรีจิสเตอร์ได้ ~125 ตัวต่อครั้ง, เขียนหลายรีจิสเตอร์ ~123; อ่านบิตได้สูงสุดราว 2000 บิต (ค่าจริงขึ้นกับอุปกรณ์)

ชนิดข้อมูลและ Endianness

รีจิสเตอร์มาตรฐานกว้าง 16 บิต (big-endian); ค่าที่กว้างกว่า (เช่น float32, int32, uint32, float64) จะกินหลายรีจิสเตอร์และอาจมีรูปแบบ Word/Byte order ต่างกันตามผู้ผลิต (เช่น AB-CD, BA-DC ฯลฯ) จึงต้องอาศัย Tag/Address Map ของอุปกรณ์เสมอ และมักต้องมี สเกล/สเกล (scaling) เพื่อแปลงเป็นหน่วยจริง

บทบาทอุปกรณ์และการสื่อสาร

ศัพท์ใหม่ Modbus ใช้ Client/Server (แทน Master/Slave เดิม)

- 1 Client ส่งคำขอ (poll/เขียน), Server ตอบสนอง
- 2 ใน TCP มี Unit ID เพื่อระบุอุปกรณ์ปลายทางเมื่อใช้เกตเวย์ไปสายอนุกรมหลายตัว
- 3 รูปแบบใช้งานทั่วไปคือ Polling เป็นช่วงเวลา + ตั้ง timeout/retry + ตรวจสอบ exception codes (เช่น 01 Illegal Function, 02 Illegal Data Address, 03 Illegal Data Value, 04 Slave Device Failure)

ประสิทธิภาพและข้อจำกัด

- 1 PDU สูงสุด ~253 ไบต์ → ทำให้มีเพดานจำนวนรีจิสเตอร์/บิตต่อคำขอ
- 2 RTU ต้องรักษา silent interval TCP ไม่มีเงื่อนไขเวลาแบบนั้นแต่ควบคุมอัตรา poll เพื่อไม่รบกวนอุปกรณ์
- 3 ควรอ่านแบบ “กวาดช่วงต่อเนื่อง (contiguous block)” เพื่อลดจำนวนคำขอ และ แยกคำขอเมื่อข้ามช่วงที่อยู่/ชนิดข้อมูล
- 4 จำกัดจำนวนการเชื่อมต่อพร้อมกันตามสเปกอุปกรณ์ (บางรุ่นรับ TCP sessions ได้ไม่กี่เส้น)

ความปลอดภัยและการติดตั้ง

Modbus ดั้งเดิม ไม่มีการยืนยันตัวตน/เข้ารหัส แนวทางปฏิบัติ: แยกเครือข่าย (VLAN/ไฟร์วอลล์), จำกัดไอพี/พอร์ต, ใช้ VPN/TLS Gateway หากต้องข้ามเครือข่ายสาธารณะ, บันทึก Log และตั้ง read-only กับจุดวิกฤต

แนวทางปฏิบัติที่ดี (Best Practices)

- 1 ใช้ Tag/Address Map ที่ยืนยันกับผู้ผลิต/เอกสารล่าสุด (รวม scaling/หน่วย/คำสั่ง)
- 2 ตั้ง timeout และ retry with backoff; บันทึก exception และสถิติการเชื่อมต่อ
- 3 รวมคำขอ รีจิสเตอร์ที่อยู่ติดกัน, แยกตามชนิด (3xxxx/4xxxx) และอย่าเกินขีดสูงสุด
- 4 รองรับ คำสลับ word/byte สำหรับค่าสองรีจิสเตอร์ขึ้นไป
- 5 ตรวจสอบคุณภาพข้อมูล (missing/negative/unreasonable) ก่อนนำไปคำนวณ/แสดงผล



ภาพที่ 12 มอดบัส

2.6.10 Influx DB

Influx DB คือฐานข้อมูลแบบ Time-Series ที่ออกแบบมาสำหรับข้อมูลที่มีตราประทับเวลา (timestamp) เช่น ค่าจากเซนเซอร์, เมตริกระบบ, IoT และเหตุการณ์ต่าง ๆ โครงสร้างข้อมูลยึดแนวคิด measurement–tags–fields–time โดยที่ tags (เช่น device_id, site, point) เป็นคีย์-ค่า

แบบจัดทำดัชนีเพื่อค้นหาเร็ว ส่วน *fields* (เช่น *value*, *status*) คือค่าที่เก็บจริงและไม่ทำดัชนี โดยเวลาเป็นคอลัมน์บังคับที่ละเอียดระดับมิลลิวินาที-นาโนวินาที การเขียนข้อมูลนิยมใช้ Line Protocol ผ่าน HTTP/Client SDK และจัดการอายุข้อมูลด้วย Retention Policy (ใน Influx DB 2.x เรียกว่า Bucket) เพื่อกำหนดว่าจะเก็บไว้นานเท่าใด พร้อมรองรับงาน downsampling/rollup (คำนวณค่าเฉลี่ย 1 นาที/15 นาที/1 ชั่วโมง) ผ่าน Tasks/Continuous Queries ลดภาระสโตร์และเร่งความเร็วการอ่าน สำหรับการสืบค้น Influx DB รองรับทั้ง Flux (ภาษา query รุ่นใหม่ที่ยืดหยุ่นสำหรับ join/transform) และ InfluxQL (สไตล์ SQL ที่คุ้นเคยในรุ่นก่อน) ขึ้นกับเวอร์ชันที่ใช้งานจริง นอกจากนี้เอนจินสโตร์/ดัชนีถูกออกแบบให้เขียนเร็ว อ่านเร็ว และบีบอัดข้อมูลได้ดี เหมาะกับข้อมูลจำนวนมากที่ไหลเข้าต่อเนื่องตลอดเวลา

ในโครงการ DAS/CEMS สามารถกำหนดโครงสร้างเช่น *measurement* = "cems_measurement" โดยมี *tags* ได้แก่ *device_id*, *point*, *unit*, *site* และ *fields* ได้แก่ *value:float*, *Statesian* แล้วตั้ง Bucket/Retention (เช่น 90 วัน) ควบคู่กับ Tasks สำหรับทำค่าเฉลี่ย 1m/15m/1h เพื่อแสดงผลบนแดชบอร์ดได้รวดเร็ว ฝั่งบริการ Fast API เขียนข้อมูลแบบเป็นชุด (batch write) และกำหนด precision/เวลาเป็น UTC เพื่อหลีกเลี่ยงปัญหาไทม์โซน ขณะ query ใช้เงื่อนไขเวลา (range) และตัวดำเนินการสรุปผล (mean, max, min) เพื่อดึง “ค่าล่าสุด”, กราฟแนวเวลา, หรือสถิติช่วงรายงาน นอกจากนี้ควรออกแบบ *tags* ให้มี cardinality ต่ำ-ปานกลาง (หลีกเลี่ยงการสร้างแท็กใหม่ไม่สิ้นสุด เช่น ใส่เวลาเป็นแท็ก) และใช้การ batching/retry เมื่อเขียนข้อมูลเพื่อความเร็ว เมื่อจัดรูปแบบข้อมูลตามแนวทางนี้ Influx DB จะรองรับการแสดงผลแบบเรียลไทม์ การตรวจสอบย้อนหลัง และการส่งออกข้อมูลไปยังรายงานที่ต้องใช้ (เช่นไฟล์ DIW) ได้อย่างมีประสิทธิภาพ



ภาพที่ 13 อินฟลัก ดี บี

2.6.11 Docker

Docker คือแพลตฟอร์ม คอนเทนเนอร์ไรซ์ (containerization) สำหรับแพ็คเกจพร้อมไลบรารี/การตั้งค่าที่จำเป็นให้เป็น อิมเมจ (image) แล้วรันเป็น คอนเทนเนอร์ (container) ที่แยกสภาพแวดล้อมจากกันบนเคอร์เนลเดียวกัน (อาศัย namespaces/cgroups) ทำให้ติดตั้งเร็ว พกพา

ง่าย และได้ผลลัพธ์สม่ำเสมอระหว่างเครื่องนักพัฒนา-เซิร์ฟเวอร์จริง อิมเมจสร้างจากไฟล์ Docker file แบบเป็นชั้น (layers) จึงแคชและบิลด์ซ้ำได้ไว เผยแพร่/ดึงอิมเมจจาก Registry (เช่น Docker Hub) ได้สะดวก สำหรับระบบที่มีหลายบริการใช้ร่วมกันสามารถประกอบด้วย Docker Compose เพื่อกำหนดบริการหลายตัว เครือข่ายภายใน (networks), โวลุ่มถาวร (volumes) และตัวแปรสภาพแวดล้อม (.env) ได้ในไฟล์เดียว รวมถึงตั้ง healthcheck, restart policy, และจำกัดทรัพยากร (CPU/RAM) ของแต่ละคอนเทนเนอร์

ในโครงการนี้ Docker ช่วยให้สปีนระบบ Fast API (Backend), Influx DB (Time-series DB), และบริการเสริม (เช่น Modbus Simulator หรือพรีอ็อกซีเซิร์ฟเวอร์) ขึ้นมาทดสอบ/สาธิตได้ทันทีผ่าน docker-compose up โดยแม่ปอร์ตสื่อสาร (เช่น 8000/8080/8086/502) และผูก volume ให้ Influx DB เพื่อคงข้อมูลไม่หายเมื่อคอนเทนเนอร์หยุด นอกจากนี้ยังทำ multi-stage build เพื่อลดขนาดอิมเมจ (เช่น บิลด์ React ด้วย Node แล้วเสิร์ฟด้วย Nginx เบสเล็ก), ใช้ผู้ใช้ non-root, ใส่ไฟล์ .docker ignore กันไฟล์ไม่จำเป็น, จัดเก็บค่าลับผ่าน environment/secret, และแยกเครือข่ายภายในเพื่อความปลอดภัย แนวทางนี้ทำให้การพัฒนา-ทดสอบ-ดีพลอยระบบ DAS/CEMS มีความสม่ำเสมอ เชื่อถือได้ และขยายต่อในสภาพแวดล้อมจริงได้ง่าย



ภาพที่ 14 ด็อกเกอร์

2.6.12 Electron Builder

electron-builder คือเครื่องมือสำหรับ “แพ็ก เซ็น กระจาย” แอป Electron แบบข้ามแพลตฟอร์มให้เป็นไฟล์ติดตั้งจริง (installer) ของ Windows / macOS / Linux ในขั้นตอนเดียว โดยกำหนดค่าจาก package.json (ฟิลด์ build) หรือไฟล์ electron-builder.yml/json จากนั้นสั่ง electron-builder เพื่อสร้างอาร์ติแฟกต์ตามระบบปฏิบัติการเป้าหมายได้อัตโนมัติ จุดเด่นคือรองรับอัปเดตอัตโนมัติ (ผ่านไลบรารีคู่มือ electron-updater) จากแหล่งปล่อยเช่น GitHub Releases, S3 หรือเซิร์ฟเวอร์ทั่วไป, จัดการ code signing (Windows Authenticode, macOS codesign + Notarization) รวมทั้งตั้งค่า App ID, ไอคอน, เวอร์ชัน, ชื่อไฟล์, สิทธิ์/entitlements และสคริปต์ after/before pack ได้ครบถ้วน เหมาะมากเมื่อแอปเว็บถูกห่อเป็นแอปเดสก์ท็อป (offline/ติดตั้งในเครื่องลูกค้า) หรือเมื่อองค์กรต้องการตัวติดตั้งมาตรฐาน/อัปเดตเงียบ

รูปแบบไฟล์ติดตั้งที่รองรับ (ตัวอย่าง)

- 1 Windows NSIS (ค่าเริ่มต้น), nsis-web, portable, appx (สำหรับ Store บางกรณี)
- 2 macOS dmg, zip, pkg, mas (Mac App Store) พร้อม notarization
- 3 Linux AppImage, deb, snap, rpm, pacman เป็นต้น

การทำงานร่วมกับ CI/CD: ผูกกับ GitHub Actions/GitLab CI ได้ง่ายเพื่อ build หลายแพลตฟอร์ม, สร้าง Release, อัปเดตอาร์ติแฟกต์ และตั้ง “ช่องอัปเดต” (stable/beta) ได้ตามกระบวนการปล่อยของทีม



electron-builder

ภาพที่ 15 อิเล็กตรอน บิลด์เดอร์

2.6.13 Modbus Simulator

Modbus Simulator คือเครื่องมือสำหรับจำลองอุปกรณ์ในระบบ Modbus ให้ทำหน้าที่เหมือนปลายทางแบบ Server/Slave (ได้ทั้ง RTU/ASCII ผ่านพอร์ตอนุกรม และ TCP/IP พอร์ต 502) เพื่อใช้พัฒนาและทดสอบโดยไม่ต้องมีเครื่องจริง ตัวซอฟต์แวร์/ฮาร์ดแวร์จำลองจะให้เรากำหนด Unit ID, จำนวนการเชื่อมต่อ, และสร้าง “หลายอุปกรณ์เสมือน” ได้ พร้อมกำหนดช่วงที่อยู่ของ โมเดล ข้อมูลทั้ง 4 กลุ่ม (Coils, Discrete Inputs, Input Registers 3xxxx, Holding Registers 4xxxx) รวมถึงชนิดข้อมูลที่แมพกับรีจิสเตอร์ 16 บิต เช่น int16, uint16, และแบบหลายรีจิสเตอร์อย่าง int32/uint32/float32/float64 โดยเลือก word/byte order และ scaling/หน่วย ได้ตามสเปกผู้ผลิต นอกจากนี้ยังตั้งค่า “รูปแบบสัญญาณ” ให้สมจริงได้ (ค่าคงที่, สุ่ม, ramp/step, sine หรือ replay จาก CSV) เพื่อดูพฤติกรรมระบบเมื่อค่าเปลี่ยนต่อเนื่อง สามารถ จำลองเหตุขัดข้อง เช่น หน่วงเวลา, timeout, ตอบ exception codes (01/02/03/04) หรือหลุดการเชื่อมต่อ เพื่อทดสอบความทนทานของฝั่งไคลเอนต์ ระหว่างทดสอบมักมี log คำขอ-คำตอบ และตัวเลือกบันทึกแพ็กเก็ตสำหรับวิเคราะห์ภายหลัง จุดใช้งานหลักคือยืนยัน Tag/Address Map ให้ถูก (รวมถึงความจริงที่ว่าเลขเอกสารอย่าง 40001 มักแปลเป็น address 0 บนสาย), ตรวจสอบ scaling/หน่วย/endianness, ปรับ poll interval/timeout/retry ของไคลเอนต์ (เช่น Pymodbus), และทดสอบบูรณาการปลายทางแบบ end-to-end (อ่านค่า คำนวณบน FastAPI เขียน InfluxDB แสดงบนแดชบอร์ดแบบเรียลไทม์/

ส่งออกไฟล์) ก่อนขึ้นอุปกรณ์จริง ช่วยลดความเสี่ยงหน้างาน ทำซ้ำสถานการณ์เดิมได้ และวัดสมรรถนะระบบภายใต้โหลดที่ควบคุมได้; ขณะใช้งานควรแยกเครือข่ายทดสอบ/จำกัดพอร์ต 502 ให้เฉพาะเครื่องเกี่ยวข้องเพื่อความปลอดภัย



ภาพที่ 16 มอดบัส สเลฟ

2.6.14 VM Were

VMware เป็นแพลตฟอร์ม Virtualization Software สำหรับสร้างและรันเครื่องเสมือน (Virtual Machine VM) หลายเครื่องบนคอมพิวเตอร์เครื่องเดียว ผู้ใช้สามารถกำหนดสเปกของ VM ได้เอง เช่น จำนวนคอร์/CPU, หน่วยความจำ (RAM), ขนาดดิสก์, การ์ดเครือข่าย (vNIC) และไฟล์บูต (ISO) นอกจากนี้ยังมีเครื่องมือสำคัญ เช่น

- Snapshot / Clone สำหรับบันทึกสถานะหรือทำสำเนา VM
- โหมดเครือข่าย (NAT, Bridged, Host-only) สำหรับจำลองการเชื่อมต่อเครือข่ายคุณสมบัติเหล่านี้ช่วยให้สามารถติดตั้งและทดสอบระบบได้อย่างปลอดภัย และสามารถทำซ้ำการทดสอบได้ง่าย

การใช้งานในโครงการนี้ในโครงการปัจจุบัน ผู้พัฒนาเลือกใช้ VMware เพื่อจำลองสภาพแวดล้อมเซิร์ฟเวอร์ สำหรับติดตั้งและทดสอบสแต็ก FastAPI, InfluxDB, React (Vite) รวมถึงเครื่องมือเสริมอย่าง Docker และ Modbus Simulator โดยมีการปรับสเปก VM ให้เหมาะสม จากนั้นทำการ ทดลองโหลด (Load/Stress Test) เพื่อวัดค่าการใช้งาน CPU, RAM, Disk, Network รวมถึง Latency และ Throughput ของเส้นทางข้อมูลแบบ end-to-end เพื่อกำหนดสเปกขั้นต่ำที่รองรับงาน ก่อนนำระบบขึ้นใช้งานจริงนอกจากนี้ยังใช้ Snapshot เพื่อทดสอบกรณีเกิดความผิดพลาด เช่น ดิสก์เต็มหรือการเชื่อมต่อขัดข้อง แล้วสามารถย้อนกลับได้อย่างรวดเร็ว ช่วยลดความเสี่ยงและประหยัดเวลาระหว่างการพัฒนาและสาธิตระบบ



ภาพที่ 17 วีเอ็ม แวร์

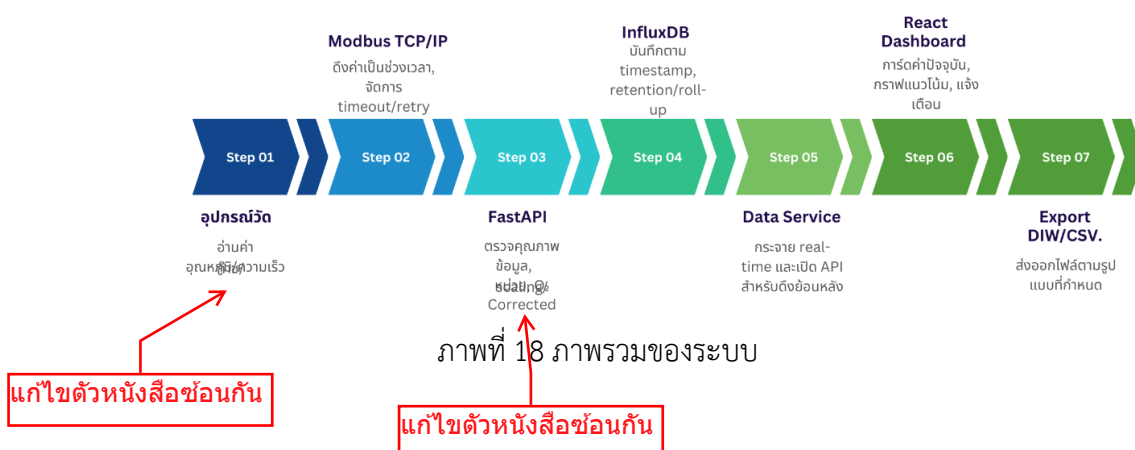
2.7 รายละเอียดของโครงการ

2.7.1 ภาพรวมระบบ

โครงการนี้มีวัตถุประสงค์เพื่อพัฒนาโปรแกรม Continuous Emission Monitoring System / Data Acquisition System (CEMS/DAS) ในรูปแบบโปรแกรมเดสก์ท็อป (Desktop Application) ที่สามารถติดตั้งและใช้งานได้บนระบบปฏิบัติการ Windows (.exe) โดยใช้สถาปัตยกรรมที่รวมส่วนประมวลผลและส่วนติดต่อผู้ใช้ไว้ในแอปพลิเคชันเดียวกันเป็น Dashboard Real-time พร้อม Alert/Export DIW

โปรแกรมสามารถรองรับการอ่านค่าจากอุปกรณ์ตรวจวัดก๊าซและเซนเซอร์ผ่านโปรโตคอล Modbus TCP/IP จากนั้นนำข้อมูลที่ได้อ่านมาประมวลผลด้วยบริการกลาง (Backend) ที่พัฒนาด้วยภาษา Python และเฟรมเวิร์ก FastAPI เพื่อทำการคำนวณที่จำเป็น เช่น Scaling, Averaging และ O₂ Corrected ก่อนจะจัดเก็บไว้ในฐานข้อมูลเชิงเวลา (Time-series Database) ได้แก่ InfluxDB

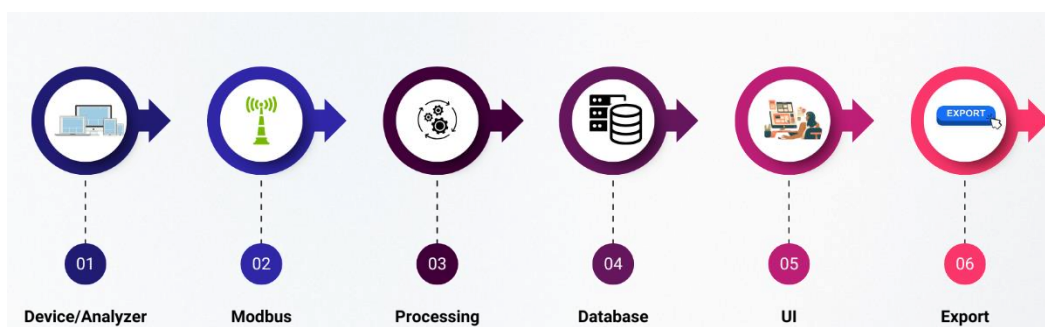
ข้อมูลที่ถูกจัดเก็บสามารถแสดงผลได้ผ่านส่วนติดต่อผู้ใช้ (Frontend) ซึ่งพัฒนาด้วย React และบรรจุรวมในแอปพลิเคชัน ด้วยเทคโนโลยี Electron ทำให้ผู้ใช้งานสามารถเปิดใช้งานเป็นโปรแกรม .exe ได้โดยตรง ผู้ใช้งานสามารถตรวจสอบค่าปัจจุบัน ดูกราฟแนวโน้มย้อนหลัง ค้นหาข้อมูลตามช่วงเวลา รวมถึงดาวน์โหลดรายงานในรูปแบบ CSV, PDF และ DIW ตามที่หน่วยงานกำกับดูแลกำหนด นอกจากนี้ยังมีฟังก์ชันการแจ้งเตือนเมื่อค่าที่วัดได้เกินกว่าค่ามาตรฐาน เพื่อให้ผู้ใช้งานสามารถตอบสนองได้อย่างทันท่วงที



2.7.2 สถาปัตยกรรมและการไหลของข้อมูล (Data Flow)

ระบบ CEMS/DAS มีการไหลของข้อมูลเป็นลำดับดังนี้

- 1 Device/Analyzer อุปกรณ์ตรวจวัด เช่น Gas Analyzer หรือ Sensor จะอ่านค่าพารามิเตอร์ เช่น SO₂, NO_x, O₂, CO และค่ากายภาพอื่น ๆ
- 2 Modbus TCP/IP ส่งข้อมูลจากอุปกรณ์มายังโปรแกรม ผ่านโปรโตคอลมาตรฐาน Modbus TCP/IP โดยสามารถกำหนดรอบเวลา (Polling Interval), Timeout และ Retry ได้
- 3 Processing (Python/FastAPI) ทำหน้าที่ ตรวจสอบความถูกต้องของข้อมูล (Validation), การ Scaling/Unit Conversion และคำนวณค่า O₂ Corrected หรือค่าเฉลี่ย
- 4 Database (InfluxDB) เก็บข้อมูลแบบ Time-series พร้อม Timestamp, Tags และ Fields เพื่อรองรับการเรียกดูย้อนหลังและการสรุปผลในช่วงเวลา
- 5 User Interface (Electron .exe) ผู้ใช้งานสามารถตรวจสอบค่าปัจจุบัน ดูกราฟแนวโน้มย้อนหลัง และตั้งค่าการแจ้งเตือนผ่าน UI ที่พัฒนาในรูปแบบโปรแกรมเดสก์ท็อป
- 6 Export ระบบสามารถส่งออกรายงานในรูปแบบ CSV/Excel และไฟล์ DIW ตามรูปแบบที่หน่วยงานกำกับกำหนด



ภาพที่ 19 สถาปัตยกรรมการไหลของข้อมูล

2.7.3 ฟังก์ชันหลัก (Functional Requirements)

1 Data Acquisition

- รองรับการอ่านค่าผ่านโปรโตคอล Modbus TCP/IP
- กำหนดรอบเวลา (Polling Interval), Timeout และ Retry
- รองรับการเชื่อมต่อหลายจุดวัด (Multi-point) ในระบบเดียว

2 Data Processing

- ตรวจสอบความถูกต้องของข้อมูล (Validation)
- Scaling/Unit Conversion ตามที่กำหนด
- การคำนวณค่าเฉลี่ย (1 นาที, 15 นาที, 1 ชั่วโมง)
- การคำนวณ O₂ Corrected เพื่อปรับค่าตามข้อกำหนดของ DIW
- กรองค่าที่ผิดปกติ เช่น Missing/Negative Value

3 Data Storage

- จัดเก็บข้อมูลในฐานข้อมูล InfluxDB แบบ Time-series
- เก็บข้อมูลพร้อม Timestamp, Tags, Fields
- มีการกำหนด Retention Policy และ Roll-up เพื่อใช้พื้นที่อย่างมีประสิทธิภาพ
- รองรับการเรียกดูย้อนหลังตามช่วงเวลา

4 User Interface (Desktop Application)

- แสดงค่าปัจจุบันในรูปแบบการ์ดสถานะ (Realtime KPI Cards)
- กราฟแนวโน้ม (Trend Graph) แบบเลือกช่วงเวลาและซูม/แพนได้
- ตารางบันทึกข้อมูลย้อนหลัง (Data Log) พร้อมฟังก์ชันค้นหา/กรอง
- ระบบแจ้งเตือน (Alert/Notification) เมื่อค่าพารามิเตอร์เกินเกณฑ์ที่กำหนด
- เมนูตั้งค่าเบื้องต้น เช่น Threshold, Tag List, Unit

5 Data Export & Reporting

- รองรับการส่งออกข้อมูลในรูปแบบ CSV/PDF
- รองรับการสร้างไฟล์ DIW ตามโครงสร้างที่หน่วยงานกำกับกำหนด
- ตรวจสอบความครบถ้วนของข้อมูลก่อนสร้างรายงาน

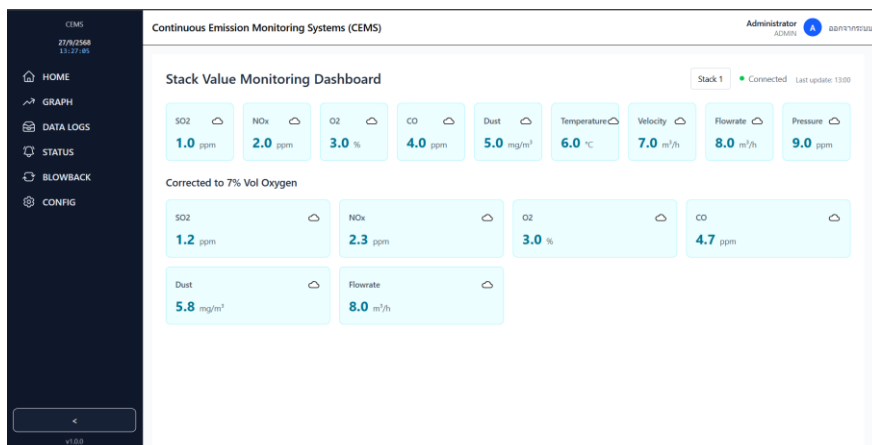
6 System Administration

- กำหนดบทบาทผู้ใช้งาน (Role-based Access Control: Admin, Operator, Viewer)
- จัดการการเชื่อมต่ออุปกรณ์และตั้งค่าพารามิเตอร์
- จัดเก็บค่าการตั้งค่าในไฟล์ Config แยกจากโค้ดหลัก

โปรแกรม CEMS/DAS ที่พัฒนาขึ้นในรูปแบบ Desktop Application ถูกออกแบบให้มี 6 หน้าหลักในการทำงาน ได้แก่ HOME, GRAPH, DATA LOGS, STATUS, BLOWBACK และ CONFIG โดยแต่ละหน้ามีหน้าที่และฟังก์ชันแตกต่างกันดังนี้

1 หน้า Home

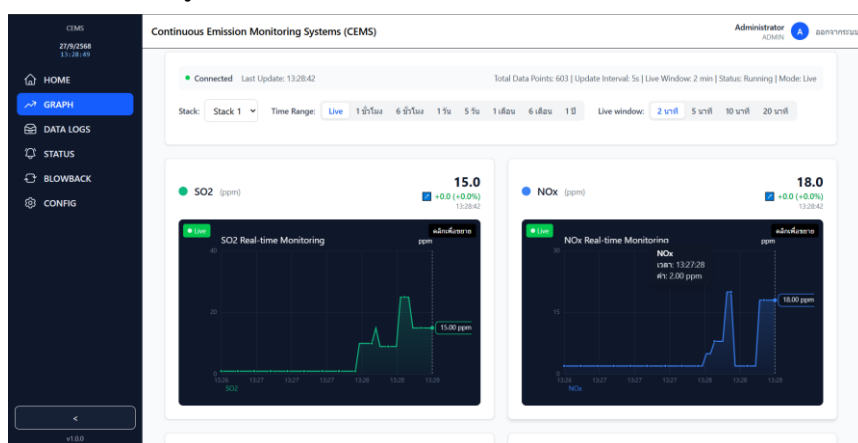
ใช้สำหรับแสดงค่าพารามิเตอร์มลพิษแบบเรียลไทม์ในรูปแบบการ์ด (Realtime Cards) โดยการ์ดแต่ละใบแสดงค่าปัจจุบัน พร้อมการเปลี่ยนสีตามระดับสถานะ (ปกติ / เตือน / อันตราย) และมีการแสดงสถานะการเชื่อมต่อกับอุปกรณ์ รวมถึงเวลาที่อัปเดตล่าสุด นอกจากนี้ยังมีส่วนแสดงค่าที่ปรับแก้ตามมาตรฐาน $O_2 = 7\%$ (Corrected) เพื่อให้สอดคล้องกับข้อกำหนดของหน่วยงาน



ภาพที่ 20 หน้าหลัก

2 หน้า Graph

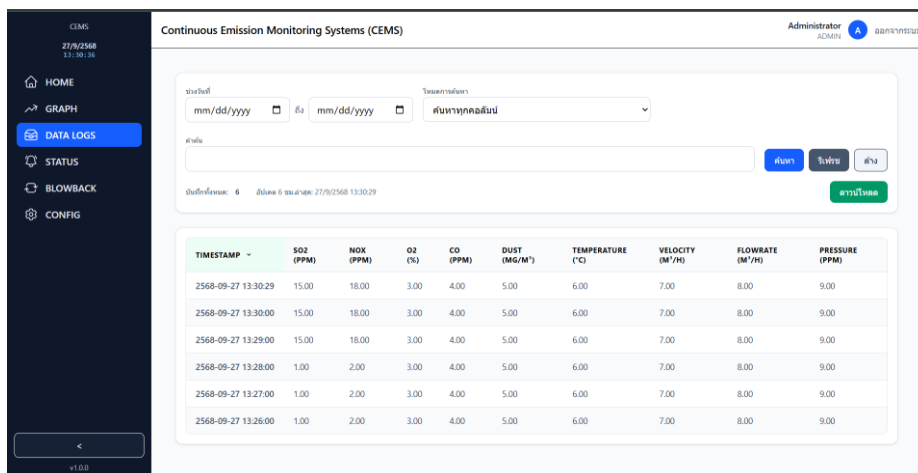
ใช้สำหรับดูแนวโน้มของพารามิเตอร์แต่ละตัวเป็นรายกราฟ โดยรองรับทั้งโหมด Live (รับข้อมูลผ่าน WebSocket และเลื่อนแบบเรียลไทม์) และโหมด Historical (ดึงข้อมูลย้อนหลังตามช่วงเวลาที่ต้องการ) ผู้ใช้สามารถเลือก Stack/ช่วงเวลา ปรับขนาดหน้าต่าง Live และกด Pause/Resume ได้ เมื่อเลื่อนเมาส์จะมี Tooltip แสดงค่า ณ เวลานั้น และสามารถคลิกเพื่อขยายกราฟดูรายละเอียดเพิ่มเติม



ภาพที่ 21 หน้ากราฟ

3 หน้า Data logs

ใช้สำหรับดูข้อมูลย้อนหลังในรูปแบบตาราง ผู้ใช้สามารถกำหนดช่วงวันที่ ค้นหาแบบทุกคอลัมน์หรือเฉพาะคอลัมน์ และเรียงลำดับตามหัวตารางได้ นอกจากนี้ยังรองรับการดาวน์โหลดข้อมูลเป็นไฟล์ CSV/PDF โดยเลือกช่วงเวลาและคอลัมน์ที่ต้องการได้ เพื่อใช้ในการจัดทำรายงาน



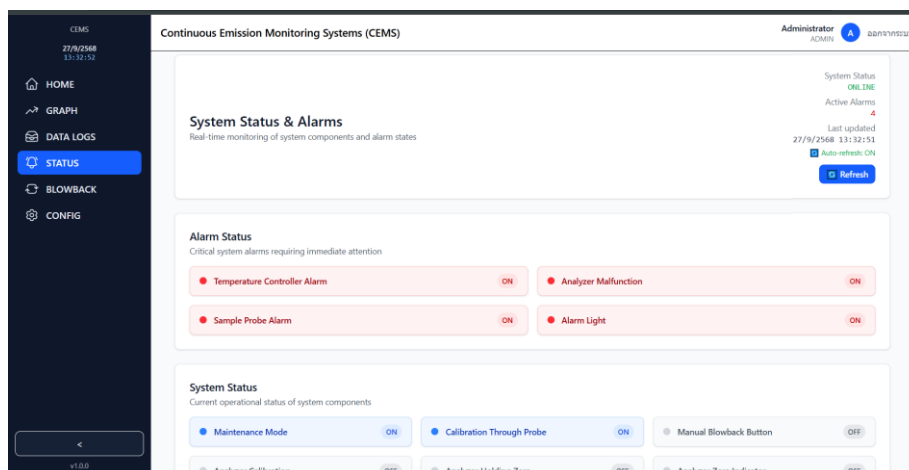
The screenshot shows the 'Continuous Emission Monitoring Systems (CEMS)' interface. The left sidebar contains navigation links: HOME, GRAPH, DATA LOGS (selected), STATUS, BLOWBACK, and CONFIG. The main area displays a data log table with columns: TIMESTAMP, SO2 (PPM), NOX (PPM), O2 (%), CO (PPM), DUST (MG/M³), TEMPERATURE (°C), VELOCITY (M/H), FLOWRATE (M³/H), and PRESSURE (PPM). The table contains 7 rows of data for the date 27/9/2568.

TIMESTAMP	SO2 (PPM)	NOX (PPM)	O2 (%)	CO (PPM)	DUST (MG/M ³)	TEMPERATURE (°C)	VELOCITY (M/H)	FLOWRATE (M ³ /H)	PRESSURE (PPM)
2568-09-27 13:30:29	15.00	18.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00
2568-09-27 13:30:00	15.00	18.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00
2568-09-27 13:29:00	15.00	18.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00
2568-09-27 13:28:00	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00
2568-09-27 13:27:00	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00
2568-09-27 13:26:00	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00

ภาพที่ 22 หน้าดูข้อมูลย้อนหลัง

4 หน้า Status

ใช้สำหรับตรวจสอบสถานะการทำงานของระบบและสัญญาณเตือนแบบเรียลไทม์ โดยรับข้อมูลผ่าน WebSocket และแสดงผลเป็นการ์ดสถานะ (ON/OFF) และการ์ดเตือน (Alarm) ผู้ใช้สามารถกด Refresh เพื่อดึงข้อมูลล่าสุดแบบแมนนวล และสามารถยืนยันรับทราบเหตุเตือนได้ผ่าน API Acknowledge ซึ่งผลจะอัปเดตกลับมาบนหน้าจอโดยอัตโนมัติ

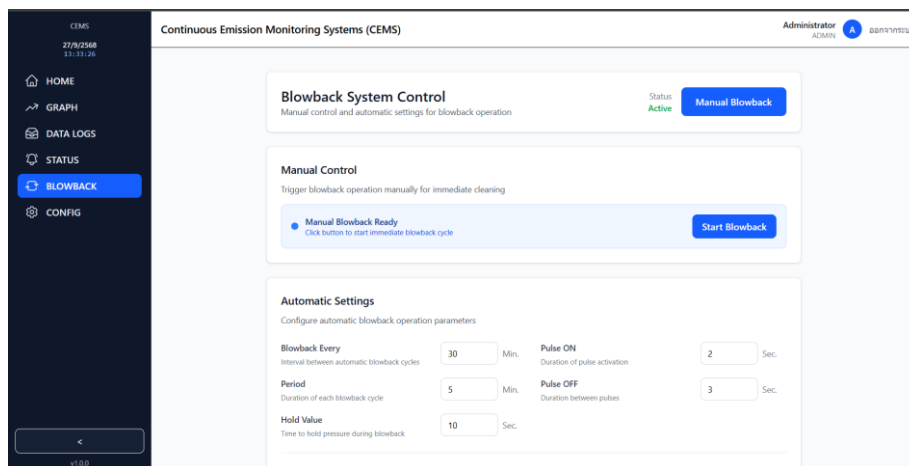


The screenshot shows the 'Continuous Emission Monitoring Systems (CEMS)' interface. The left sidebar contains navigation links: HOME, GRAPH, DATA LOGS, STATUS (selected), BLOWBACK, and CONFIG. The main area displays the 'System Status & Alarms' section. It includes a 'System Status' card showing 'System Status: ONLINE', 'Active Alarms: 4', and 'Last updated: 27/9/2568 13:32:51'. Below this is an 'Alarm Status' card showing four active alarms: 'Temperature Controller Alarm', 'Analyzer Malfunction', 'Sample Probe Alarm', and 'Alarm Light'. At the bottom is a 'System Status' card showing various operational status components: 'Maintenance Mode' (ON), 'Calibration Through Probe' (ON), 'Manual Blowback Button' (OFF), 'Analyzer Calibration' (OFF), 'Analyzer Holding Zero' (OFF), and 'Analyzer Zero Indicator' (OFF).

ภาพที่ 23 หน้าสแตตัส

5 หน้า Blowback

ใช้สำหรับควบคุมการเป่าเขม่าของระบบดักฝุ่น โดยผู้ใช้สามารถกดสั่ง Manual เพื่อเป่าทันที หรือกำหนดค่าการทำงานอัตโนมัติ เช่น ช่วงห่างของรอบ ความยาวรอบ เวลากดแรงดัน และพัลส์ ON/OFF จากนั้นบันทึกค่าด้วยปุ่ม Save Settings ระบบจะแสดงสถานะสรุป เช่น รอบล่าสุดและรอบถัดไปเพื่อให้วางแผนบำรุงรักษาได้สะดวก

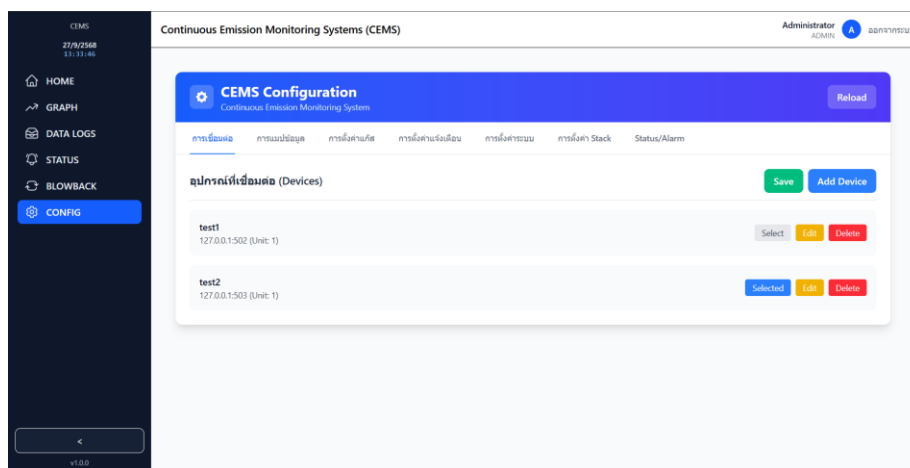


ภาพที่ 24 หน้าโบลวแบ็ค

6 หน้า Config

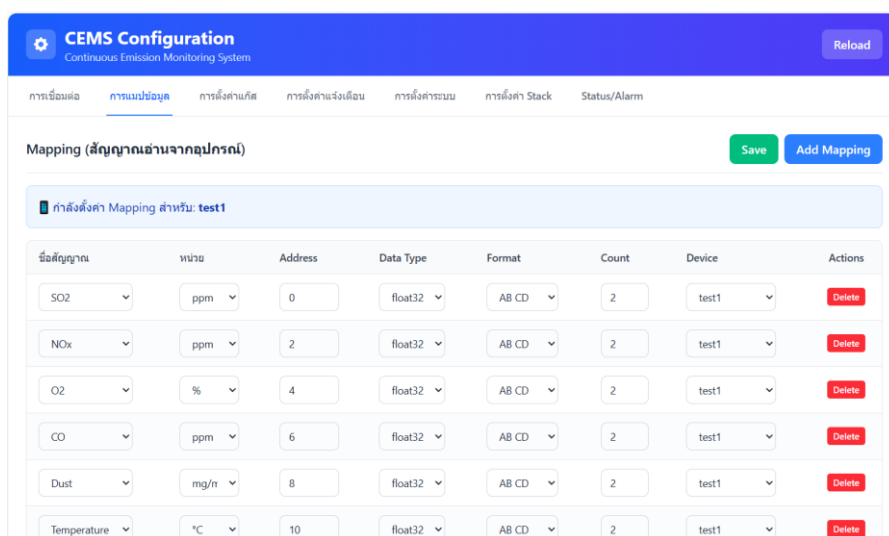
หน้า Config คือศูนย์รวมการตั้งค่าระบบ CEMS ทั้งหมด ตั้งแต่การเชื่อมต่ออุปกรณ์ การแมปสัญญาณ การกำหนดว่าจะโซ่แก๊สอะไรและหน่วยอะไร เกณฑ์แจ้งเตือน พารามิเตอร์ระบบ ขนาดปล่อง (Stack) ไปจนถึงการแมปสถานะและสัญญาณเตือนสำหรับหน้าตรวจสอบแบบเรียลไทม์ หน้านี้มีปุ่ม Reload เพื่อดึงค่าจากแบ็กเอนด์ใหม่ และทุกครั้งที่เกิด Save ค่าจะถูกบันทึกกลับไปยังเซิร์ฟเวอร์ ส่วนการเลือกอุปกรณ์ (Select) ในแท็บ Devices จะมีผลให้แท็บที่เกี่ยวข้อง เช่น Mapping และ Status/Alarm แสดงเฉพาะข้อมูลของอุปกรณ์นั้น ทำให้จัดการง่ายและลดความสับสน

ในแท็บ Devices ใช้สำหรับเพิ่ม แก๊ส และลบอุปกรณ์ Modbus ที่จะอ่านค่าจริง เช่น กำหนดโหมด TCP/RTU, IP/Port และ Unit ID จุดสำคัญคือเมื่อแก้ไขชื่ออุปกรณ์ ระบบจะอัปเดตชื่อเดียวกันใน Mapping ที่เกี่ยวข้องให้อัตโนมัติ และเมื่อลบอุปกรณ์ ระบบจะเตือนว่ามี Mapping ที่เกี่ยวข้องและจะลบตามไปด้วย ผู้ใช้สามารถกด Select เพื่อเลือกอุปกรณ์เป้าหมายก่อนย้ายไปตั้งค่าต่อในแท็บอื่น



ภาพที่ 25 แท็บการเชื่อมต่ออุปกรณ์

ในแท็บ Mapping ใช้กำหนดว่าสัญญาณแต่ละตัว (เช่น SO₂, NO_x, O₂, CO, Dust ฯลฯ) อยู่ที่ Address/Data Type/Format/Count ไตของอุปกรณ์ไหน มีตัวช่วยอัตโนมัติหลายจุด เช่น เมื่อเลือกชื่อสัญญาณ ระบบจะใส่หน่วยเริ่มต้นที่เหมาะสมให้อัตโนมัติ และเมื่อเปลี่ยน Data Type จะปรับ Count เริ่มต้นให้เหมาะสมทันที ผู้ใช้ควรเลือก Device ให้เรียบร้อยก่อนเพิ่ม Mapping ใหม่ เพื่อลดโอกาสสับสนในภายหลัง



ภาพที่ 26 กำหนดสัญญาณ

ในแท็บ Gas Settings เป็นที่ตั้งค่าว่าหน้าจอส่วนต่าง ๆ จะ “โชว์แก๊สอะไร” และ “โชว์แบบใด” รวมถึงกำหนดหน่วยและค่าเกณฑ์เตือนเบื้องต้น ผู้ใช้สามารถเปิด/ปิดการแสดงผลของแต่ละแก๊ส เปิดการแสดงผลค่าแบบ Corrected และปรับหน่วยให้ตรงกับงานจริงได้ เมื่อเลือกชื่อแก๊ส ระบบจะปรับค่า key ให้สัมพันธ์กันเพื่อให้ดึงข้อมูลมาแสดงได้ถูกต้อง การกด Save จะบันทึกชุดการตั้งค่าทั้งหมดไปยังแบ็กเอนด์

Enable	Show Corrected	ชื่อแก๊ส	Unit	Warning/Alarm	Action
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	SO2	ppm	80	Delete
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NOx	ppm	80	Delete
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	O2	%	80	Delete
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CO	ppm	80	Delete
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Dust	mg/m³	80	Delete
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Temperature	°C	80	Delete
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Velocity	m³/h	80	Delete

ภาพที่ 27 แท็บกำหนดการ์ดที่จะแสดงในหน้าโฮม

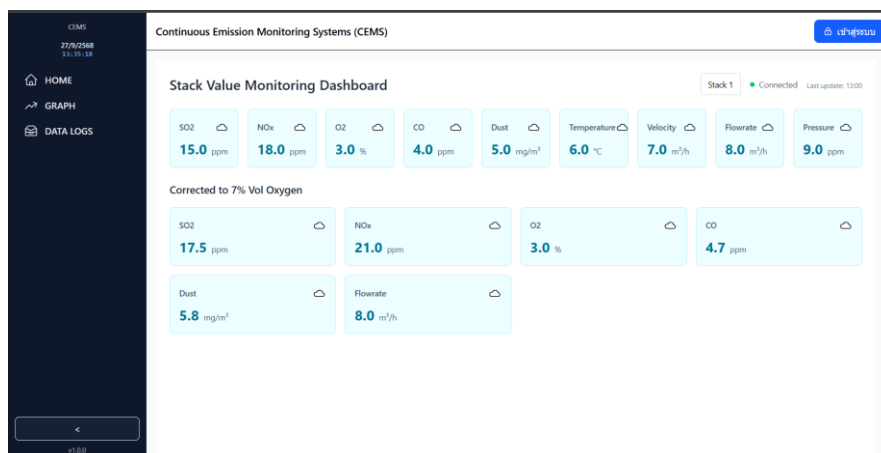
แท็บ Status/Alarm ใช้แมปชื่อสถานะและสัญญาณเตือนกับ Address และ Device เพื่อให้หน้า Status & Alarms โชว์แบบเรียลไทม์ได้ถูกต้อง ผู้ใช้เลือกประเภทเป็น Status หรือ Alarm จากนั้นเลือกชื่อจากรายการสำเร็จรูป ซึ่งระบบจะป้องกันชื่อซ้ำโดยอัตโนมัติ สามารถกำหนด Address และผูกกับอุปกรณ์ที่ใช้จริง รวมทั้งเปิด/ปิดรายการนั้นได้ การเปลี่ยนประเภทจะรีเซ็ตตัวเลือกชื่อให้ตรงชุด และถ้าในแท็บ Devices เลือกอุปกรณ์ไว้ รายการในหน้านี้จะแสดงเฉพาะของอุปกรณ์ที่เลือก ช่วยให้จัดระเบียบง่ายและลดโอกาสผิดพลาด เมื่อทำครบให้กด Save เพื่อบันทึกการแมปทั้งหมด

Name	Type	Address	Device	Enabled	Actions
Temperature Controller Alarm	Alarm	0	test2	<input checked="" type="checkbox"/>	Delete
Analyzer Malfunction	Alarm	1	test2	<input checked="" type="checkbox"/>	Delete
Sample Probe Alarm	Alarm	2	test2	<input checked="" type="checkbox"/>	Delete
Alarm Light	Alarm	3	test2	<input checked="" type="checkbox"/>	Delete
Maintenance Mode	Status	4	test2	<input checked="" type="checkbox"/>	Delete
Calibration Through Probe	Status	5	test2	<input checked="" type="checkbox"/>	Delete

ภาพที่ 28 แท็บเชื่อมต่อสัญญาณสเตตัส/อะลาม

7 หน้าหลัก ยังไม่มีการ Login Role

จะแสดงหน้าทั้ง 3 หน้าได้แก่ HOME, GRAPH และ DATA LOGS



ภาพที่ 29 หน้าแรกที่ยังไม่มีการล็อกอิน

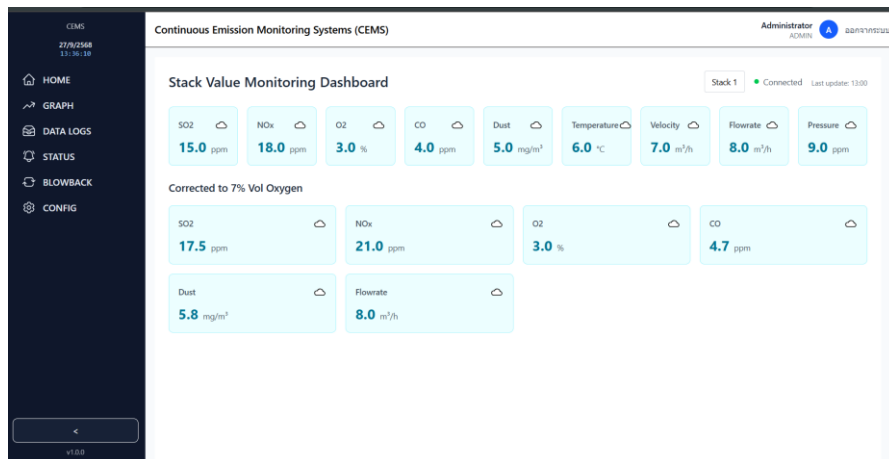
8 การเลือก Role User ใช้งาน

เลือกได้สองมุมมองได้แก่ Admin และ User

ภาพที่ 30 หน้าเลือกล็อกอิน

9 การล็อกอินแบบ Admin

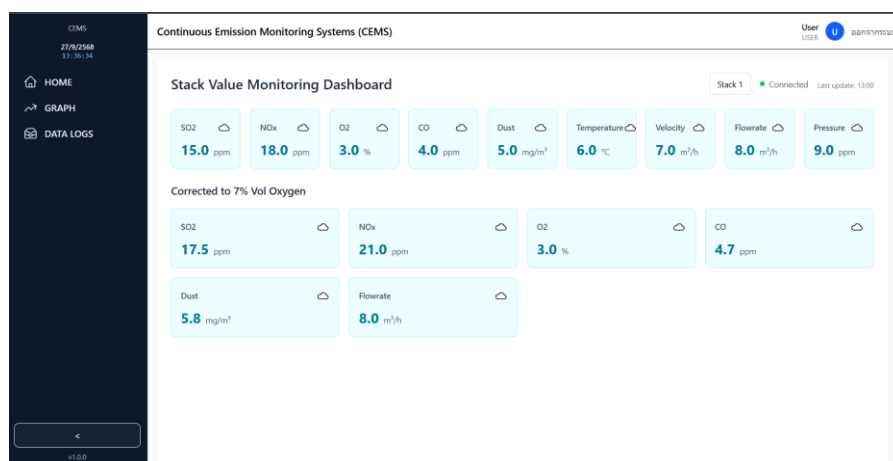
สำหรับการล็อกอินแบบ Admin จะสามารถเข้าถึงการตั้งค่าแต่หน้าทั้งหมดได้สำหรับผู้ที่มีสิทธิ์ที่จะเข้าถึงระบบหรือการตั้งค่าระบบได้



ภาพที่ 31 การเข้าถึงแบบ แอดมิน

10 การ Login แบบ User หรือ Viewer

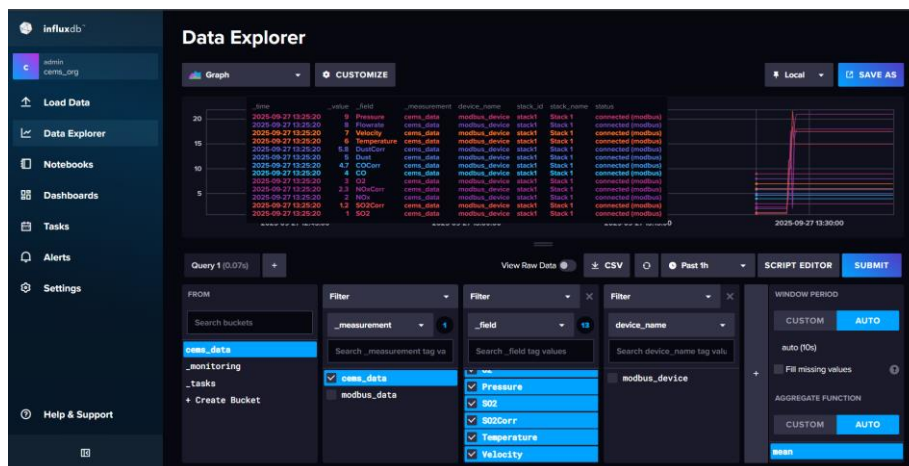
จะเป็นการเข้าถึงได้ 3 หน้าเหมือนหน้าหลักตอนที่ยังไม่ได้ login สำหรับผู้ที่คอยตรวจสอบแค่ค่าข้อมูล การดูการแจ้งเตือนหรือการทำงานของเครื่องมือวัด แต่ไม่สามารถที่จะเข้าถึงการตั้งค่าการเชื่อมต่อของระบบหลัก



ภาพที่ 32 การเข้าถึงแบบผู้ใช้หรือผู้เข้าชม

11 หน้าฐานข้อมูล

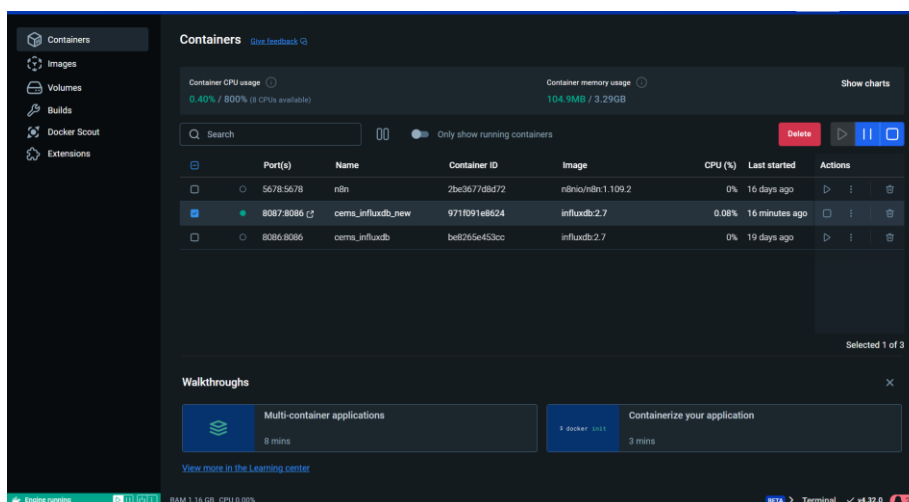
InfluxDB Data Explorer เป็นเครื่องมือสำหรับสำรวจและแสดงผลข้อมูลในฐานข้อมูลแบบ Time Series Database ของ InfluxDB โดยถูกใช้เพื่อจัดเก็บและวิเคราะห์ข้อมูลที่เปลี่ยนแปลงตามเวลา เช่น ข้อมูลจากระบบ CEMS (Continuous Emission Monitoring System) ไม่ว่าจะเป็นค่า SO₂, NO_x, O₂, Temperature หรือ Pressure ผู้ใช้งานสามารถเลือก Bucket, Measurement, Field และ Tag ที่ต้องการ พร้อมทั้งกำหนดช่วงเวลาเพื่อดูข้อมูลทั้งในรูปแบบตารางและกราฟ รวมถึงสามารถส่งออกเป็นไฟล์ CSV ได้ ทำให้ Data Explorer เป็นส่วนสำคัญที่ช่วยตรวจสอบและติดตามข้อมูลได้อย่างสะดวก รวดเร็ว และมีประสิทธิภาพ



ภาพที่ 33 หน้าฐานข้อมูล

12 Docker

Docker Desktop เป็นเครื่องมือที่ใช้ในการจัดการและรัน Container ซึ่งช่วยให้สามารถสร้างสภาพแวดล้อมการทำงานที่แยกออกจากระบบหลักได้อย่างสะดวกและมีประสิทธิภาพ โดยในโครงการนี้ได้ใช้ Docker สำหรับรันฐานข้อมูล InfluxDB ในรูปแบบ Container ทำให้การติดตั้งและการใช้งานฐานข้อมูลง่ายต่อการควบคุม ไม่ว่าจะเป็นการเริ่มต้น (start), หยุด (stop), รีสตาร์ท หรือการลบ Container อีกทั้งยังสามารถตรวจสอบการใช้ทรัพยากรของระบบ เช่น CPU และหน่วยความจำได้แบบเรียลไทม์ จึงทำให้การจัดการฐานข้อมูลมีความยืดหยุ่น ปลอดภัย และเหมาะสมกับงานด้านการเก็บและวิเคราะห์ข้อมูลจากระบบ CEMS



ภาพที่ 34 การรัน ด็อกเกอร์ ของ ฐานข้อมูล

2.7.4 คุณลักษณะเชิงคุณภาพ (Non-Functional Requirements)

- 1 Accuracy การอ่าน/คำนวณ $\geq 99\%$ (เทียบข้อมูลอ้างอิง/สูตร)

- 2 Latency จากรับค่าถึงแสดงผลบน Dashboard ≤ 1 s ที่โหลดปกติ
- 3 Reliability ทดสอบต่อเนื่อง 24 ชม. 0 data loss ในสภาพแวดล้อมทดสอบ
- 4 Scalability รองรับ ≥ 1 แหล่งข้อมูล หลายจุดวัด และเพิ่มจุดวัดได้โดยไม่แก้โค้ดหลัก (ใช้ Tag list)
- 5 Maintainability โครงสร้างโค้ดแยกชั้น service/repository, มีเอกสาร API/Mapping ครบ

2.7.5 ส่วนติดต่อผู้ใช้และเส้นทางการใช้งาน (UI/Flows)

Login Dashboard (การตั้งค่าปัจจุบัน + กราฟรวม) Trend (เลือกจุดวัด/ช่วงเวลา/ชุม) Alerts (ประวัติ/สถานะ)
Reports/Export (CSV/DIW) Settings (Threshold/Tag/Unit ขั้นพื้นฐาน)

2.7.6 โครงสร้างข้อมูลและฐานข้อมูล (InfluxDB)

- 1 Measurement cems_measurement
- 2 Tags device_id, point, unit, site
- 3 Fields value:float, status:int, quality:int
- 4 Retention ค่าเริ่มต้น 90 วัน (ปรับได้) + Roll-up (1m, 15m, 1h)
- 5 ตัวอย่าง Query อ่านค่า point เฉลี่ย 1 ชม./ช่วงเวลา, ตรวจสอบค่าขาดหาย

2.7.7 การเชื่อมต่อภายนอก/รายงาน DIW

- 1 Modbus TCP/IP รองรับ Function 03/04/16, รองรับคำสั่ง Retry/Back-off
- 2 DIW Export แปลงหน่วย/รูปแบบเวลาให้ตรงสปีค, ตรวจสอบถ้วนของฟิลด์ก่อนสร้างไฟล์

2.7.8 สิทธิผู้ใช้และความปลอดภัยเบื้องต้น

- 1 บทบาท Admin/Operator/Viewer (พื้นฐาน)
- 2 การจัดเก็บค่าเชื่อมต่อ/รหัสผ่านในไฟล์ตั้งค่าเฉพาะสภาพแวดล้อม, ปิด endpoint ที่ไม่จำเป็นในรุ่นสาธิต

2.7.9 แผนทดสอบและเกณฑ์ยอมรับ

- 1 Unit/Integration/System test ด้วย Modbus Simulator/อุปกรณ์จริง
- 2 ตรวจสอบ Accuracy, Latency, Data completeness, Export DIW ผ่านตรวจสอบ
- 3 เอกสาร: Test Plan, Test Case, Test Report

2.7.10 เครื่องมือ/เทคโนโลยี

Python 3.11, FastAPI, Pymodbus, InfluxDB 2.x, React + Ant Design, Git/VS
Code/Postman, Modbus Simulator

บทที่ 3

สรุปผลการปฏิบัติงาน

3.1 สิ่งที่คาดหวัง

- 1 การเข้ามาฝึกงานครั้งนี้เพื่อเรียนรู้การทำงานจริงและสภาพแวดล้อมจริง
- 2 คาดว่าจะได้ลงเขียนโปรแกรมจริงและเรียนรู้จากข้อผิดพลาดที่เกิดขึ้นเพื่อเป็นประสบการณ์ในอนาคต
- 3 คาดหวังว่าจะได้เข้าใจการทำงานในสภาพแวดล้อมจริงของบริษัท

3.2 ประโยชน์ที่ได้รับจากการปฏิบัติงาน

3.2.1 ประโยชน์ต่อตนเอง

- 1 ได้ฝึกเขียนโค้ดแบบจริงจึงได้ลงทำอะไรหลายๆอย่าง~~ลงอะไรหลายๆ~~
- 2 ได้ฝึกระเบียบและความรับผิดชอบ หลักๆเลยคือการมาทำงานให้ตรงเวลา และทำงานที่ได้รับมอบหมาย
- 3 เรียนรู้งานในด้านฮาร์ดแวร์ไม่ว่าจะเป็นตัวเครื่องมือต่างๆที่ไม่รู้จักอุปกรณ์เครื่องมือวัดต่างๆ

3.2.2 ประโยชน์ต่อสถานประกอบการ

- 1 ได้ตัวต้นแบบที่สามารถทำงานได้เทียบเท่าโปรแกรมเชิงพาณิชย์และสามารถเป็นแนวทางในการพัฒนาต่อให้สมบูรณ์และตามท้องถื่นที่ต้องการได้
- 2 จัดทำเอกสารประกอบการใช้งานและ source codeที่ใช้พัฒนา

3.2.3 ประโยชน์ต่อมหาวิทยาลัย

- 1 ทำให้สาขาวิชามี ผลงานเชื่อมโยงกับภาคอุตสาหกรรมจริง
- 2 เพิ่มความน่าเชื่อถือและสร้างโอกาสความร่วมมือระหว่าง มหาวิทยาลัย และ บริษัท

3.3 วิเคราะห์จุดเด่น จุดด้อย โอกาส อุปสรรค (Swot Analysis)

3.3.1 ประสิทธิภาพที่ประทับใจ / ประสิทธิภาพพิเศษ

- 1 ได้รับความรู้ใหม่ๆไม่ว่าจะด้านโปรแกรมที่ใช้ในอุตสาหกรรมจริงๆ
- 2 ได้เรียนรู้เกี่ยวกับเครื่องมือวัด การทำงานของเซ็นเซอร์ที่ใช้ในการวัด
- 3 ได้ฝึกเขียนโค้ดแบบเต็มที่ใช้เครื่องมือ ภาษา AI และองค์ประกอบอื่นๆที่ใช้พัฒนาโปรได้อย่างเต็มที่ไม่จำกัด

3.3.2 จุดแข็ง

- 1 อยากที่จะเรียนรู้สิ่งใหม่ๆเสมอ
- 2 ชอบที่จะได้รับความรู้ใหม่ๆ

3.3.3 จุดอ่อน

- 1 พื้นฐานยังไม่แน่นพอที่จะรับความรู้ใหม่ๆได้หมด
- 2 การเข้าใจในภาษาที่เขียนโปรแกรม การเข้าใจ logic ที่จะต้องมีในโปรแกรม การเข้าใจในเครื่องมือที่ใช้ในการพัฒนายังไม่มากพอ

3.3.4 โอกาสทางสภาพแวดล้อม

- 1 ได้เรียนรู้กับอุปกรณ์จริงๆโปรแกรมจริงที่ใช้ในปัจจุบัน
- 2 ได้เรียนรู้กับคนที่ทำงานจริงไม่ว่าจะเป็นด้านอุปกรณ์ การติดตั้งรวมถึงใช้โปรแกรมที่ทำอยู่นี้เป็นประจำ

3.3.5 อุปสรรคทางสภาพแวดล้อม

- 1 ต้องปรับตัวและหาทางใช้เครื่องมือที่เข้าใจในการทำโปรแกรมเพราะโปรแกรมจริงต้องใช้ VB.NET แต่ไม่ถนัดและไม่เข้าใจจึงต้องพลิกแพลงเพื่อให้ภาษาที่ทำอยู่ ให้สามารถออกมาเป็นโปรแกรมEXEได้
- 2 ความรู้ที่มีไม่มากพอต้องใช้ความพยายามอย่างมากเพื่อที่จะแก้บั๊กในโปรแกรมแต่ละครั้งทำให้เสียเวลาและกำลังใจในการพัฒนาไปอย่างมาก
- 3 เวลาที่จำกัดเนื่องจากการฝึกสหกิจใช้เวลา 4 เดือน หรือ 1 เทอมซึ่งคิดว่าน้อยไปสำหรับการที่จะพัฒนาโปรแกรมที่จะใช้ในการทำงานจริงและไม่ให้มีบั๊กในการใช้งานเยอะ

บทที่ 4

ปัญหาและข้อเสนอแนะ

บทที่ 4
ลองปรับภาษาให้ดูเป็นทางการขึ้น
และสะท้อนผลเชิงบวก
ดังตัวอย่างด้านล่าง

สถานประกอบการ

ปัญหา

- 1 ปัญหาด้านเวลา คือที่ที่ดูแลงานเข้ามาเยอะพอตีในช่วงที่มาฝึกงานทำให้มีเวลาน้อยในการที่จะปรึกษาปัญหา ทั้งด้วยเกรงใจเลยถามเท่าที่ปัญหาที่แก้ไม่ได้จริงๆ
- 2 งานส่วนมากเป็นงานเครื่องมือวัดจึงทำให้เรียนรู้ช้าเพราะไม่รู้จัก

มหาวิทยาลัย

ปัญหา

- 1 เอกสารที่เยอะระดับนึงเพราะต้องเตรียมตัวนิเทศถึง2ครั้งและต้องมีเล่มรายงาน สไลด์นำเสนอทำให้ไฟกสหลายจุดไปหน่อยทั้งต้องเตรียมตัวในการพรีเซนต์แต่ละรอบและต้องทำโปรแกรม
- 2 การบันทึกการทำงานในแต่ละวันรวมทั้งต้องให้พี่เลี้ยงเซ็นทุกวัน เนื่องด้วยพี่เลี้ยงติดงานเยอะ และต้องเขียนรายละเอียดการทำงานแต่ละวันยังต้องเขียนรายงานเป็นอาทิตย์อีกซึ่งคิดว่าเป็นการบันทึกซ้ำซ้อน

ข้อเสนอแนะ

- 1 ควรที่จะเอากาการเขียนรายงานในแต่ละวันออกเพราะมีบันทึกเป็นรายอาทิตย์อยู่แล้วและมี การติดตามสทกิจหรือนิเทศถึง2ครั้งก็น่าจะเพียงพอสำหรับการรายงานและติดตามการทำงานแล้ว
- 2 เนื่องจากสิ่งที่เรียนได้ใช้จริงจึงอยากให้อาจารย์สอนละเอียดขึ้นอีกนิดไม่ว่าจะเป็นทั้ง ฮาร์ดแวร์หรือมีโปรเจกที่อาจารย์สอนพาทำแบบจริงจังที่ไม่ใช่ให้นักศึกษาไปหาทำกันเอง
- 3 อยากให้เพิ่มระยะเวลาการฝึกงานขึ้นเพราะจะได้ฝึกประสบการณ์แบบจริงๆเพราะเวลา ปัจจุบันแค่ทำโปรเจกและเตรียมพรีเซนต์โปรเจกที่อาจารย์มาแต่ละครั้งก็กินเวลาไปเยอะ ทำให้การเรียนรู้กับงานจริงๆมันน้อยมาก

นักศึกษา

ปัญหา

- 1 ความรู้ไม่มากพอไม่ว่าจะเป็นด้านภาษาโปรแกรม การใช้เครื่องมือ หรือการเขียนlogicที่ใช้ในโปรแกรมยังอ่อนอยู่มาก
- 2 ปัญหาในการเรียนรู้งานเนื่องจากไม่รู้จักรในด้านเครื่องมือวัดและอุปกรณ์ไฟฟ้า

ข้อเสนอแนะ

- 1 ต้องฝึกฝนและเรียนรู้ให้มากกว่านี้และมีสมาธิกับการทำงานกว่านี้ตั้งใจเรียนรู้หาข้อมูลถึงแม้จะไม่รู้จักก็ต้องหาความรู้ได้

บทที่ 4

สรุปปัญหาและข้อเสนอแนะ

จากการปฏิบัติงานสหกิจศึกษาในโครงการพัฒนาระบบตรวจวัดและติดตามปริมาณมลพิษทางอากาศ (CEMS) ณ บริษัท อะนาไลติกอลซิสเต็ม เอ็นจิเนียริง (ประเทศไทย) จำกัด ตลอดระยะเวลา 4 เดือน ข้าพเจ้าได้ประสบพบเจอกับความท้าทายหลากหลายรูปแบบ ซึ่งล้วนเป็นประสบการณ์ที่นำไปสู่การเรียนรู้และการพัฒนาที่สำคัญ บทนี้จึงเป็นการสรุปปัญหาที่พบเจอ พร้อมทั้งนำเสนอข้อเสนอแนะเชิงสร้างสรรค์ เพื่อเป็นประโยชน์ต่อสถานประกอบการ มหาวิทยาลัย และแนวทางการพัฒนาตนเองของข้าพเจ้าในอนาคต

4.1 ข้อเสนอแนะสำหรับสถานประกอบการ

1. การบริหารเวลาและการให้ค่าปรึกษา

- **ปัญหา:** ในช่วงเวลาที่ข้าพเจ้าเข้าปฏิบัติงาน พนักงานที่ปรึกษามีภาระงานค่อนข้างมาก ทำให้การหาเวลาเพื่อปรึกษาหรือขอคำแนะนำในเชิงลึกเป็นไปได้จำกัด
- **ข้อเสนอแนะ:** เพื่อเพิ่มประสิทธิภาพในการดูแลและให้ค่าปรึกษแก่นักศึกษาฝึกงานในรุ่นต่อไป อาจพิจารณาจัดสรรเวลาสำหรับการประชุมติดตามความคืบหน้าประจำสัปดาห์ที่แน่นอน เช่น 30-60 นาทีต่อสัปดาห์ เพื่อให้ นักศึกษาสามารถรวบรวมคำถามและรับคำแนะนำได้อย่างเป็นระบบและเต็มที่ ซึ่งจะช่วยส่งเสริมการเรียนรู้ของนักศึกษาได้ดียิ่งขึ้นแม้ในช่วงที่พนักงานมีภาระงานสูง

2. การถ่ายทอดความรู้เกี่ยวกับผลิตภัณฑ์และระบบ

- **ปัญหา:** งานส่วนใหญ่ของบริษัทเกี่ยวข้องกับเครื่องมือวัดและอุปกรณ์ในโรงงานอุตสาหกรรม ซึ่งเป็นความรู้เฉพาะทางที่ข้าพเจ้าไม่คุ้นเคย ทำให้ช่วงแรกของการปฏิบัติงานต้องใช้เวลาในการเรียนรู้และทำความเข้าใจค่อนข้างมาก
- **ข้อเสนอแนะ:** อาจพิจารณาจัดทำเอกสารสรุปภาพรวมของผลิตภัณฑ์หลัก (เช่น CEMS, SWAS) เกี่ยวกับหลักการทำงานของเครื่องมือวัดพื้นฐานสำหรับนักศึกษาใหม่ เพื่อให้ นักศึกษาสามารถทำความเข้าใจบริบทของงานได้รวดเร็วยิ่งขึ้น และสามารถเริ่มลงมือปฏิบัติงานในส่วนที่ได้รับมอบหมายได้อย่างมั่นใจ

4.2 ข้อเสนอแนะสำหรับมหาวิทยาลัย

1. การปรับปรุงกระบวนการเอกสารและการประเมินผล

- **ปัญหา:** กระบวนการจัดทำเอกสารของรายวิชาสหกิจศึกษามีความซ้ำซ้อนในบางส่วน เช่น การบันทึกการปฏิบัติงานรายวันควบคู่ไปกับการสรุปรายงานรายสัปดาห์ และการเตรียมการนำเสนอเพื่อนิเทศงานถึงสองครั้ง ซึ่งใช้เวลาค่อนข้างมาก
- **ข้อเสนอแนะ:**

มหาวิทยาลัยอาจพิจารณารวบยอดการบันทึกการทำงานให้เหลือเพียงรายงานสรุปรายสัปดาห์ที่มีรายละเอียดครอบคลุม เพื่อลดความซ้ำซ้อนและให้นักศึกษาสามารถมุ่งเน้นกับการปฏิบัติงานในโครงการหลักได้อย่างเต็มศักยภาพ

2. การบูรณาการเนื้อหาเชิงปฏิบัติเข้ากับหลักสูตร

- **ปัญหา:** การปฏิบัติงานจริงทำให้ตระหนักว่า ความรู้เชิงลึกเกี่ยวกับโปรโตคอลสื่อสารในอุตสาหกรรม (เช่น Modbus) และสถาปัตยกรรมซอฟต์แวร์สมัยใหม่มีความสำคัญอย่างยิ่ง
- **ข้อเสนอแนะ:** อยากเสนอให้อาจารย์ผู้สอนพิจารณาเพิ่มเนื้อหาหรือจัดทำโครงการจำลอง (Workshop/Project-based Learning) ที่เกี่ยวข้องกับเทคโนโลยีที่ใช้จริงในภาคอุตสาหกรรมมากขึ้น เพื่อให้ นักศึกษารุ่นต่อไปมีความพร้อมและสามารถปรับตัวเข้ากับการทำงานจริงได้รวดเร็วยิ่งขึ้น

3. การพิจารณาเรื่องระยะเวลาของโครงการสหกิจศึกษา

- **ปัญหา:** ระยะเวลาการฝึกงาน 4 เดือน หรือ 1 ภาคการศึกษา ถือเป็นเวลาที่ค่อนข้างจำกัดสำหรับการพัฒนาโครงการที่มีความซับซ้อนให้สมบูรณ์และปราศจากข้อผิดพลาด (Bug)
- **ข้อเสนอแนะ:** หากเป็นไปได้ การขยายระยะเวลาการฝึกงานอาจเป็นประโยชน์อย่างยิ่ง จะทำให้นักศึกษามีโอกาสได้เรียนรู้วงจรการพัฒนาซอฟต์แวร์ที่ครบถ้วน ตั้งแต่การออกแบบ พัฒนา ไปจนถึงการทดสอบและปรับปรุงแก้ไข ซึ่งเป็นประสบการณ์ที่ทรงคุณค่าอย่างมาก

4.3 การวิเคราะห์และแนวทางการพัฒนาตนเอง

1. การเรียนรู้และเติมเต็มช่องว่างทางเทคนิค

- **ปัญหา:** ข้าพเจ้าพบว่าความรู้พื้นฐานด้านการเขียนโปรแกรมและตรรกะในการแก้ปัญหาอย่างไม่แข็งแรงพอ โดยเฉพาะเมื่อต้องเผชิญกับโจทย์ที่ซับซ้อน ทำให้การแก้ไขปัญหามักใช้เวลานานเกินความจำเป็น
- **แนวทางการพัฒนา:** การปฏิบัติงานครั้งนี้เป็นเครื่องเตือนใจให้ต้องหมั่นฝึกฝนทักษะการเขียนโค้ดให้มากขึ้น ศึกษาหาความรู้เพิ่มเติมเกี่ยวกับโครงสร้างข้อมูล (Data Structures) และอัลกอริทึม (Algorithms) อย่างสม่ำเสมอ รวมทั้งต้องมีสมาธิและวางแผนการทำงานอย่างเป็นระบบ เพื่อให้สามารถรับมือกับความท้าทายใหม่ๆ ได้อย่างมีประสิทธิภาพ

2. การพัฒนาทักษะการปรับตัวและการแก้ปัญหา

- **ความท้าทาย:** เดิมทีโปรแกรมของบริษัทส่วนใหญ่พัฒนาด้วย VB.NET แต่เนื่องจากข้าพเจ้าไม่มีความถนัด จึงต้องประยุกต์ใช้เทคโนโลยีที่ตนเองคุ้นเคย (Python, React) และหาทางทำให้ผลลัพธ์สุดท้ายสามารถทำงานได้ตามโจทย์ คือเป็นโปรแกรม .exe บน Windows
 - **สิ่งที่ได้เรียนรู้:** ความท้าทายนี้กลายเป็นโอกาสให้ได้เรียนรู้การใช้เครื่องมือใหม่ๆ เช่น Electron Builder เพื่อสร้าง Desktop Application จากเว็บเทคโนโลยี
- ซึ่งถือเป็นการพัฒนาทักษะการแก้ปัญหาเฉพาะหน้าและการปรับตัวให้เข้ากับข้อจำกัดได้อย่างดีเยี่ยม และเป็นประสบการณ์ที่จะเป็นประโยชน์อย่างยิ่งต่อการทำงานในอนาคต

บรรณานุกรม

- [1] InfluxData, “InfluxDB Documentation,” 2025. [Online]. Available: <https://docs.influxdata.com/> Accessed: 28 Sep 2025.
- [2] InfluxData, “Line Protocol,” 2025. [Online]. Available: <https://docs.influxdata.com/influxdb/v2/write-data/> Accessed: 28 Sep 2025.
- [3] InfluxData, “Query data (Flux / InfluxQL),” 2025. [Online]. Available: <https://docs.influxdata.com/influxdb/v2/query-data/> Accessed: 28 Sep 2025.
- [4] Docker Inc., “Docker Documentation,” 2025. [Online]. Available: <https://docs.docker.com/> Accessed: 28 Sep 2025.
- [5] Python Software Foundation, “Python 3 Documentation,” 2025. [Online]. Available: <https://docs.python.org/3/> Accessed: 28 Sep 2025.
- [6] FastAPI, “FastAPI Documentation,” 2025. [Online]. Available: <https://fastapi.tiangolo.com/> Accessed: 28 Sep 2025.
- [7] React, “React Documentation,” 2025. [Online]. Available: <https://react.dev/> Accessed: 28 Sep 2025.
- [8] Vite, “Vite Guide,” 2025. [Online]. Available: <https://vitejs.dev/guide/> Accessed: 28 Sep 2025.
- [9] Ant Design, “Ant Design of React,” 2025. [Online]. Available: <https://ant.design/docs/react/introduce> Accessed: 28 Sep 2025.
- [10] Tailwind Labs, “Tailwind CSS Documentation,” 2025. [Online]. Available: <https://tailwindcss.com/docs> Accessed: 28 Sep 2025.
- [11] Pymodbus, “Pymodbus Documentation,” 2025. [Online]. Available: <https://pymodbus.readthedocs.io/> Accessed: 28 Sep 2025.
- [12] Modbus Organization, “MODBUS Application Protocol Specification,” 2024. [Online]. Available: <https://modbus.org/> Accessed: 28 Sep 2025.
- [13] IETF, “The WebSocket Protocol (RFC 6455),” 2011. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc6455> Accessed: 28 Sep 2025.
- [14] Electron, “Electron Documentation,” 2025. [Online]. Available: <https://www.electronjs.org/docs/latest> Accessed: 28 Sep 2025.

- [15] electron-builder, “electron-builder Documentation,” 2025. [Online]. Available: <https://www.electron.build/> Accessed: 28 Sep 2025.
- [16] VMware, “VMware Workstation/Fusion Documentation,” 2025. [Online]. Available: <https://www.vmware.com/support/pubs/> Accessed: 28 Sep 2025.
- [17] Microsoft, “WebSockets in ASP.NET Core/HTTP,” 2025. [Online]. Available: <https://learn.microsoft.com/aspnet/core/fundamentals/websockets> Accessed: 28 Sep 2025

ภาคผนวก