ระบบตรวจจับคุณภาพอากาศ

เสนอ

อาจารย์ชาญวิทย์ มุสิกะ

จัดทำโดย

นายอภิวิชญ์ ยิ่งใหญ่ธนศักดิ์ 65502100109-5

นายระพีพล คำผา 65502100103-8

นายปกป้อง ปัจจังหรีด 65502100122-8

รายงานนี้เป็นส่วนหนึ่งของการศึกษารายวิชา

Selected Topics in Computer Programming

สาขาวิชา วิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยเทคโนโลยีราชมงคลกรุงเทพ ภาคการศึกษาที่ 1 ปีการศึกษา 2567

หัวข้อโปรเจค ระบบตรวจจับคุณภาพอากาศ

สมาชิก นายอภิวิชญ์ ยิ่งใหญ่ธนศักดิ์ 65502100109-5

นายระพีพล คำผา 65502100103-8

นายปกป้อง ปัจจังหรีด 65502100122-8

นักศึกษาชั้นปีที่ 3

ครูที่ปรึกษา ชาญวิทย์ มุสิกะ

ปีการศึกษา 2567

บทคัดย่อ

โครงการนี้มีวัตถุประสงค์เพื่อพัฒนาระบบตรวจจับคุณภาพอากาศที่สามารถตรวจวัดฝุ่นละออง PM2.5, PM10, แก๊สจากเซ็นเซอร์ MQ, และอุณหภูมิในสิ่งแวดล้อม ระบบใช้เซ็นเซอร์ PMS3003 สำหรับตรวจวัดฝุ่นละอองขนาดเล็ก, เซ็นเซอร์ MQ สำหรับตรวจจับแก๊สที่เป็นอันตรายเช่น แก๊สไวไฟหรือแก๊สคาร์บอนไดออกไซด์ (CO2), และเซ็นเซอร์วัดอุณหภูมิและความชื้นสำหรับตรวจวัดสภาพอากาศ ระบบทำงานร่วมกับไมโครคอนโทรลเลอร์ในการเก็บข้อมูลและส่งต่อข้อมูลไปยังแพลตฟอร์ม IoT เพื่อแสดงผลและวิเคราะห์แบบเรียลไทม์ผ่านแอพพลิเคชันหรือเว็บไซต์

คำสำคัญ: คุณภาพอากาศ, PM2.5, เซ็นเซอร์, อุณหภูมิ, การตรวจจับ

กิตติกรรมประกาศ

โปรเจคเรื่องระบบตรวจจับคุณภาพอากาศฉบับนี้ สำเร็จได้อย่างดี โดยได้รับความอนุเคราะห์จาก อาจารย์ ชาญวิทย์ มุสิกะ อาจารย์ประจำสาขา วิทยาการคอมพิวเตอร์ มหาวิทยาลัยราชมงคลกรุงเทพ อาจารย์ผู้สอนวิชา select topic ที่ให้ความกรุณาเป็นอาจารย์ที่ปรึกษา ประสิทธิ์ประสาทความรู้และทักษะด้านการเขียนรายงาน อีกทั้ง ได้สละเวลาให้คำแนะนำ ข้อเสนอแนะ ตรวจและแก้ไขข้อบกพร่องต่างๆ ด้วยความเอาใจใส่อย่างดียิ่งทาง

คณะผู้จัดทำรู้สึกซาบซึ้งในความอนุเคราะห์จากอาจารย์และขอขอบพระคุณ เป็นอย่างสูงที่ได้ประสิทธิ์ประสาทวิชาให้ตลอดจนการเอื้อเฟื้อสถานที่ และ ช่วยแก้ปัญหาต่างๆ เกี่ยวกับการ ออกแบบและประดิษฐ์อุปกรณ์ สุดท้ายนี้คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ ที่เป็นกำลังใจ และให้การสนับสนุนในทุกเรื่องๆ ทำให้คณะผู้จัดทำสามารถทำงานชิ้นนี้สำเร็จลุล่วงด้วยดีคุณค่า และคุณประโยชน์อันพึงมาจากโครงงานชิ้นนี้ คณะผู้จัดทำโครงงานขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

10 ตุลาคม 2567

คำนำ

ปัจจุบันปัญหามลพิษทางอากาศได้ทวีความรุนแรงมากขึ้น ส่งผลกระทบต่อสุขภาพของประชาชนและสิ่งแวดล้อมอย่างหลีกเลี่ยงไม่ได้ ระบบตรวจจับคุณภาพอากาศจึงเป็นเทคโนโลยีที่มีความสำคัญอย่างยิ่งในการตรวจสอบและเฝ้าระวังคุณภาพอากาศอย่างต่อเนื่อง ซึ่งระบบนี้สามารถวัดปริมาณฝุ่นละออง อุณหภูมิ ความชื้น และก๊าซที่เป็นอันตรายในอากาศได้ โดยข้อมูลที่ได้รับจากการตรวจวัดสามารถนำไปใช้ในการแจ้งเตือนและป้องกันผลกระทบที่อาจเกิดขึ้นต่อสุขภาพมนุษย์ และยังเป็นเครื่องมือที่ช่วยในด้านการกำหนดนโยบายและมาตรการแก้ไขปัญหามลพิษในระดับท้องถิ่นและระดับประเทศ

รายงานฉบับนี้จัดทำขึ้นเพื่อศึกษาหลักการทำงาน องค์ประกอบ และประโยชน์ของระบบตรวจจับคุณภาพอากาศ รวมถึงผลที่เกิดขึ้นจากการนำระบบนี้มาใช้งาน หวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะเป็นประโยชน์แก่ผู้ที่สนใจในด้านการตรวจสอบคุณภาพอากาศและการพัฒนาสิ่งแวดล้อมให้ยั่งยืนต่อไป

สารบัญ

บทที่ 1 บทนำ

1.1 ความเป็นมาของปัญหา

ในปัจจุบัน ปัญหามลพิษทางอากาศได้กลายเป็นประเด็นสำคัญในระดับโลก เนื่องจากมีผลกระทบโดยตรงต่อสุขภาพของประชาชน สิ่งแวดล้อม และคุณภาพชีวิตโดยรวม การพัฒนาอย่างรวดเร็วของเมืองใหญ่ การขยายตัวของอุตสาหกรรม การเผาไหม้เชื้อเพลิงฟอสซิลในภาคการคมนาคม และการใช้พลังงานในกิจกรรมต่างๆ ได้ทำให้ปริมาณมลพิษทางอากาศเพิ่มขึ้นอย่างต่อเนื่อง มลพิษเหล่านี้ประกอบไปด้วยฝุ่นละอองขนาดเล็ก ก๊าซพิษ และสารเคมีที่เป็นอันตราย เช่น ก๊าซคาร์บอนไดออกไซด์ (CO_2) ไนโตรเจนไดออกไซด์ (NO_2) ซัลเฟอร์ไดออกไซด์ (SO_2) และฝุ่นละอองขนาดเล็กกว่า 2.5 ไมครอน ($PM_{2.5}$) ซึ่งเป็นหนึ่งในปัจจัยหลักที่ก่อให้เกิดปัญหาสุขภาพในระยะยาว

องค์การอนามัยโลก (WHO) ระบุว่า มลพิษทางอากาศเป็นสาเหตุสำคัญของการเสียชีวิตก่อนวัยอันควรในหลายประเทศ และส่งผลให้เกิดโรคต่างๆ เช่น โรคระบบทางเดินหายใจ โรคหัวใจ โรคปอด และมะเร็งปอด นอกจากนี้ยังมีผลกระทบต่อสิ่งแวดล้อมอย่างมาก โดยเฉพาะอย่างยิ่งการเปลี่ยนแปลงสภาพภูมิอากาศ (Climate Change) และการทำลายความสมดุลของระบบนิเวศ

ในประเทศไทย ปัญหามลพิษทางอากาศเป็นที่พูดถึงอย่างมากในช่วงหลายปีที่ผ่านมา โดยเฉพาะปัญหาฝุ่นละอองขนาดเล็ก $PM_{2.5}$ ที่มักจะเกิดขึ้นในช่วงฤดูหนาวในเขตภาคเหนือและกรุงเทพมหานคร ซึ่งมาจากการเผาไหม้ทางการเกษตร การจราจรที่หนาแน่น และการก่อสร้าง โดยในช่วงเวลาที่มีค่าฝุ่นละอองสูง ประชาชนจะประสบปัญหาด้านสุขภาพอย่างชัดเจน ทั้งปัญหาเกี่ยวกับทางเดินหายใจ การแพ้ภูมิ และโรคที่เกี่ยวข้องกับระบบหัวใจและหลอดเลือด

ดังนั้น การพัฒนาระบบตรวจจับคุณภาพอากาศจึงกลายเป็นเครื่องมือสำคัญที่ได้รับความสนใจในยุคปัจจุบัน ระบบนี้สามารถวัดระดับมลพิษในอากาศได้อย่างแม่นยำและรวดเร็ว ทั้งในระดับท้องถิ่นและระดับประเทศ ระบบจะประกอบด้วยเซ็นเซอร์ตรวจจับสารมลพิษต่างๆ เช่น ฝุ่นละออง ก๊าซพิษ และอนุภาคขนาดเล็ก โดยข้อมูลที่เก็บรวบรวมได้จากระบบตรวจจับคุณภาพอากาศจะถูกส่งต่อไปเพื่อทำการประมวลผล และวิเคราะห์ข้อมูลเพื่อใช้ในการแจ้งเตือนประชาชนและหน่วยงานที่เกี่ยวข้อง

1.2 วัตถุประสงค์

1. เพื่อพัฒนาระบบตรวจจับคุณภาพอากาศที่สามารถตรวจวัดระดับมลพิษต่างๆ เช่น ฝุ่นละออง (PM2.5, PM10) และ ตรวจจับควันแก๊สมีเทน ได้อย่างแม่นยำและทันเวลา
2. เพื่อสร้างฐานข้อมูลคุณภาพอากาศที่สามารถนำมาใช้ในการศึกษาวิจัยและพัฒนามาตรการควบคุมมลพิษในระยะยาว
3. เพื่อให้สามารถรับทราบข้อมูลคุณภาพอากาศในพื้นที่แบบเรียลไทม์ผ่านจอแสดงผลหรือเว็บไซต์

1.3 ขอบเขตงาน

1. ระบบตรวจจับคุณภาพอากาศจะถูกออกแบบให้สามารถตรวจวัดระดับมลพิษในอากาศ เช่น ฝุ่นละอองขนาดเล็ก (PM2.5, PM10) อุณหภูมิและความชื้นในอากาศ ตรวจจับควันแก๊สมีเทน ในพื้นที่ที่กำหนด
2. ระบบจะประกอบด้วยเซ็นเซอร์สำหรับตรวจจับข้อมูลคุณภาพอากาศ และสามารถส่งข้อมูลไปยังฐานข้อมูลเพื่อการประมวลผลและการวิเคราะห์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถตรวจวัดและตรวจสอบค่าของฝุ่นละอองในอากาศและบันทึกเป็นสถิติข้อมูลได้
2. สนับสนุนการจัดการมลพิษข้อมูลที่ได้จากการตรวจวัดคุณภาพอากาศสามารถนำไปใช้ในการวางแผนและจัดการปัญหามลพิษ
3. ทราบแนวโน้มการเกิดฝุ่นเพื่อหาวิธีการป้องกัน

บทที่ 2 ทฤษฎีที่เกี่ยวข้องและอุปกรณ์ที่ใช้

บทนี้กล่าวถึงทฤษฎีหลักการที่เกี่ยวข้องกับรายงานนี้ ในส่วนแรกจะเป็นเนื้อหาทฤษฎี เกี่ยวกับ IoT (Internet of Things) ฝุ่น PM2.5 จากนั้นจะกล่าวถึงฮาร์ดแวร์สำหรับการทำครั้งนี้

2.1 Internet of Things (IoT)

อินเทอร์เน็ตของทุกสิ่ง หมายถึง เครือข่ายของวัตถุ อุปกรณ์ พาหนะ สิ่งปลูกสร้าง หรือสิ่งของที่มีองค์ประกอบของฮาร์ดแวร์ ซอฟต์แวร์ เซนเซอร์(Sensor) สรรพสิ่งสามารถสื่อสารเชื่อมต่อกันได้ผ่านโพรโทคอลสำหรับการสื่อสาร สรรพสิ่งต่างๆ มี วิธีการระบุตัวตน รับรู้บริบทของสภาพแวดล้อมได้ มีปฏิสัมพันธ์ทำงานร่วมกันได้ สรรพสิ่งต่างๆ ได้แก่ เครื่องจักรในโรงงาน เครื่องจักรกลการเกษตร อุปกรณ์เครื่องใช้ภายในอาคารสำนักงาน อุปกรณ์เครื่องใช้ใน บ้านพักอาศัย อุปกรณ์ทางการแพทย์ ความสามารถในการสื่อสารของสรรพสิ่งจะนำไปสู่การสร้างนวัตกรรม และบริการใหม่ เช่น เซนเซอร์ตรวจจับการเคลื่อนไหวและวัดสัญญาณของผู้ป่วยแล้วส่งสัญญาณไปยังบุคลากร ทางการแพทย์ เป็นต้น

ประโยชน์ของ Internet of Things

ประโยชน์ของการประยุกต์ใช้IoT ทำให้เกิดผลิตภัณฑ์และบริการรูปแบบใหม่มีการใช้ระบบคลาวด์ บนอินเทอร์เน็ต เทคโนโลยี IoT สามารถใช้งานในภาคอุตสาหกรรม การแพทย์ การเกษตร การจัดการพลังงาน การขนส่งโลจิสติกส์และระบบจัดการอาคารเป็นต้น การพัฒนาผลิตภัณฑ์หรือบริการที่ใช้เทคโนโลยี IoT ต้อง มีการเชื่อมต่อสื่อสารระหว่างอุปกรณ์หรือระบบต่างๆ เพื่อดูแลตรวจสอบ ควบคุม แล้วนำข้อมูลที่ได้รับมา วิเคราะห์ ทำให้เพิ่มประสิทธิภาพการทำงานของอุปกรณ์ ระบบ หรือกระบวนการ ให้ดียิ่งขึ้นได้ ด้านการศึกษา เช่น การมาประยุกต์ใช้ QR code, RFID (Radio-frequency identification), NFC (Near field communication) หรือ ใช้เทคโนโลยี AR (Augmented Reality) เพื่อช่วยสร้างประสบการณ์ใน การเรียนรู้ ด้านวิทยาศาสตร์สุขภาพ ด้านการแพทย์ เช่น เซนเซอร์ตรวจวัดร่างกายมนุษย์แล้วส่งข้อมูลเข้าระบบ ประมวลผลเพื่อแจ้งเตือน เพื่อการวิเคราะห์สำหรับบุคลกรทางการแพทย์ ด้านการเกษตรสำหรับ Smart Farming เช่น การติดตั้งเซ็นเซอร์ตรวจสภาพแวดล้อมของอากาศ สภาพแวดล้อมของดิน แล้วส่งข้อมูลเข้าสู่ระบบ เพื่อควบคุมสภาพแวดล้อมที่เหมาะกับการเพาะปลูกพืช -5 - ด้านการรักษาความปลอดภัย เช่น การตรวจจับตำแหน่งบุคคลโดยใช้ ระบบ Smart CCTV การใช้ บัตร RFID สำหรับบุคคลเข้าอาคาร ด้านอุตสาหกรรมการผลิต เช่น อุปกรณ์อัตโนมัติสำหรับเครื่องจักรเพื่อวิเคราะห์ข้อมูลสำหรับการเพิ่ม ผลผลิต ลดต้นทุนการผลิต ด้านโลจิสติกส์ เช่น ใช้ IoT ในการเก็บข้อมูลเพื่อรายงานการส่งสินค้า สถานะของสินค้าในตู้จัดเก็บ สถานะของยานพาหนะที่ใช้ขนส่ง ด้าน Smart City เป็นการบูร

ณาการของระบบต่างๆ เช่น การจัดการไฟฟ้า Smart Grid ร่วมกับ Smart Meter การใช้งานร่วมกับ Smart CCTV เพื่อวิเคราะห์ใบหน้าแบบ Face Recognition สำหรับระบบ รักษาความปลอดภัย การจัดการขนส่งสาธารณะ ด้าน Smart Home เป็นการใช้เทคโนโลยีควบคุมอุปกรณ์ต่างๆ ภายในบ้านให้ทำงานร่วมกัน เช่น การควบคุมอุปกรณ์ไฟฟ้า การตรวจจับสภาพแวดล้อมในบ้านเพื่อควบคุมเครื่องปรับอากาศ การตรวจจับความเคลื่อนไหวเพื่อแจ้งเตือนไปยัง Smart Device สถาปัตยกรรม Internet of Things ความหลากหลายของระบบ IoT ทำให้ปัจจุบันมีการนำเสนอสถาปัตยกรรม Internet of Things หรือ สถาปัตยกรรม IoT ออกเป็นหลายสถาปัตยกรรม [1] เช่น IEEE P2413, oneM2M และ IoT World Forum (IoTWF) Standardize Architecture สำหรับสถาปัตยกรรม IoT แบบ IoTWF ที่กล่าวถึงเป็นสถาปัตยกรรม ที่เกิดจากความร่วมมือของ Cisco, IBM, Gartner, SAP, Samsung และบริษัทอื่น ๆ ในปี ค.ศ. 2014

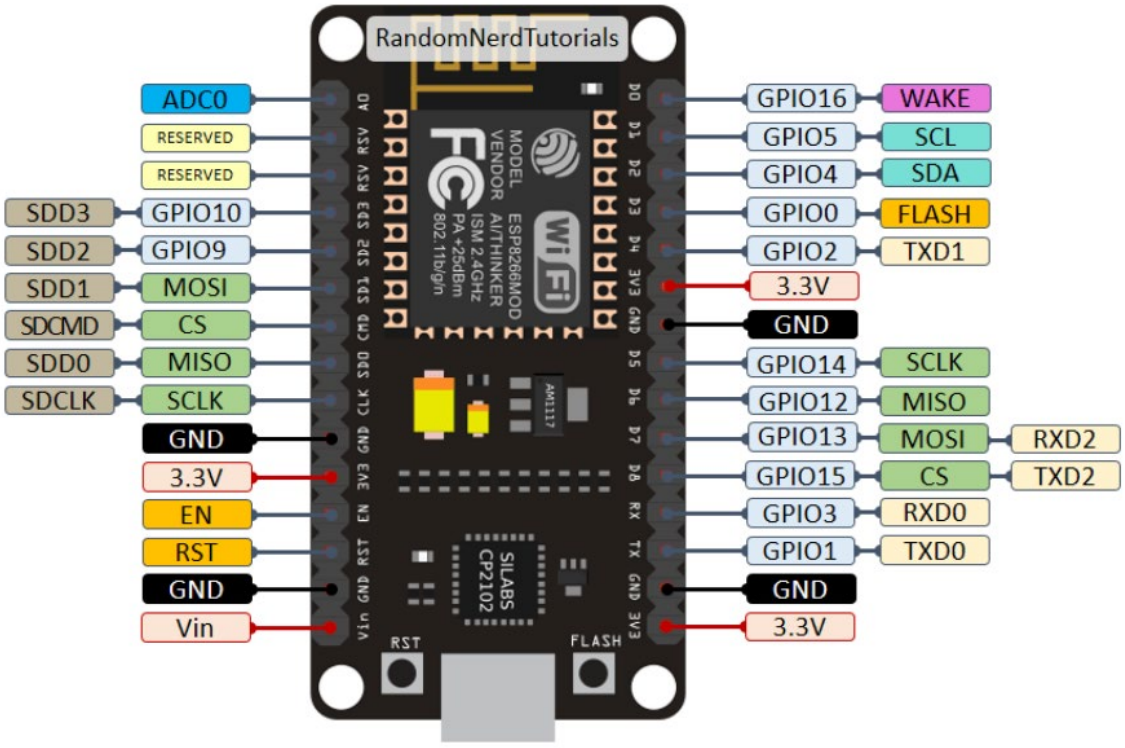
2.2 ฝุ่น PM2.5

ฝุ่น PM2.5 ย่อมาจากคำว่า“Particulate matter with diameter of less than 2.5micron” หน่วยงาน ป้องกันสิ่งแวดล้อมประเทศสหรัฐอเมริกา US. EPA (United State Environmental Protection Agency) ได้ทำ การกำหนดค่ามาตรฐานของฝุ่นละอองขนาดเล็กในอากาศที่เป็นอันตรายต่อสุขภาพมนุษย์เอาไว้ โดยใช้ค่า PM (Particulate Matters) เป็นเกณฑ์ในการตรวจวัดคุณภาพอากาศ ประกอบด้วย ฝุ่น PM 10 หรือที่โดยทั่วไป 5 (Research article) Journal of Engineering Technology Access ISSN: 2774 – 0889 Vol. 2 No. 2 เรียกว่า “ฝุ่นหยาบ” (Course Particles) คือ อนุภาคฝุ่นละอองในอากาศที่มีเส้นผ่าศูนย์กลางขนาด 2.5 - 10 ไมครอน ฝุ่นประเภทนี้เมื่อรวมกันเป็นจำนวนมากแล้วมักจะสังเกตเห็นได้ง่าย เช่น ฝุ่นที่เกาะอยู่ตามข้าวของเครื่องใช้, เกสรดอกไม้ หรือฝุ่นละอองจากงานก่อสร้าง เป็นต้น ฝุ่น PM2.5 หรือที่เรียกว่า “ฝุ่นละเอียด” (Final Particles) คือ อนุภาคฝุ่นละอองในอากาศที่มีขนาดเส้นผ่าศูนย์กลางอยู่ที่ 2.5 ไมครอน และฝุ่นละอองแขวนลอย (Suspended Particulate Matter) มีขนาดระหว่าง 0.001 ถึง 100 ไมครอนฝุ่นละอองที่กำหนดในมาตรฐานคุณภาพอากาศในบรรยากาศทั่วไปประกอบด้วยฝุ่นละอองขนาดใหญ่ หรือที่เรียกว่าฝุ่นรวม (Total Suspended Particulate – TSP) ฝุ่นละอองขนาดไม่เกิน 10 ไมครอน (PM10) และฝุ่นละอองขนาดไม่เกิน 2.5 ไมครอน (PM2.5) ทั้งนี้ฝุ่นรวมและฝุ่นละอองขนาดไม่เกิน 10 ไมครอน (PM10) จัดเป็นฝุ่นหยาบเกิดจากกระบวนการทางฟิสิกส์ ได้แก่ ลมพัดฝุ่นดิน เถ้าภูเขาไฟ ละอองน้ำทะเล การบดย่อย ขัดสีสึกกร่อนและการฟุ้งกระจายของวัสดุอุตสาหกรรมและหิน ดิน ทราย ส่วนฝุ่นขนาดไม่เกิน 2.5 ไมครอนจัดเป็นฝุ่นละเอียดเกิดจากการเผาไหม้เชื้อเพลิงเกิดเป็นฝุ่นควันโดยตรง (เรียกว่า ฝุ่นปฐมภูมิ - primary particle) หรือเกิดเป็นก๊าซซึ่งอาจกลั่นตัวเป็นเม็ดฝุ่นเริ่มต้นและรวมตัวกันเป็นเม็ดฝุ่นขนาดใหญ่ขึ้น (เรียกว่า ฝุ่นทุติยภูมิ–secondary particle) ตัวอย่างเช่น แอมโมเนียมไนเตรต แอมโมเนียมซัลเฟต

2.3 ฮาร์ดแวร์

Node MCU ESP8266

Node MCU ESP8266 บนเพตฟอร์ม Arduino สามารถเชื่อมต่อกับ Wi-Fi ได้, สามารถเขียน โปรแกรมด้วย Arduino IDE ได้เช่นเดียวกับ Arduino และบอร์ดก็มีราคาถูกมาก ๆ เหมาะแก่ผู้ที่คิดจะเริ่มต้นศึกษา หรือทดลองใช้งานเกี่ยวกับ Arduino, IoT, อิเล็กทรอนิกส์ หรือแม้แต่การ นำไปใช้จริงในโปรเจคต่าง ๆ ก็ตาม เพราะราคาไม่แพง ภายในบอร์ดของ Node MCU ประกอบด้วย ESP8266 (ไมโครคอนโทรลเลอร์ ที่ สามารถเชื่อมต่อ Wi-Fi ได้) พร้อมอุปกรณ์อำนวยความสะดวกต่าง ๆ เช่น พอร์ต micro USB สำหรับจ่ายไฟ/อัพโหลดโปรแกรม, ชิพสำหรับอัพโหลดโปรแกรมผ่านสาย USB, ชิพแปลง แรงดันไฟฟ้า และขาสำหรับเชื่อมต่ออุปกรณ์ภายนอก ขาของ Node MCU



ภาพที่ 1 *บอร์ด ESP8266 12-E NodeMCU kit 1*

เซนเซอร์วัดอุณหภูมิและความชื้น

เซนเซอร์วัดอุณหภูมิ (Temperature) และความชื้นสัมพัทธ์(RH: Relative Humidity) ในอากาศที่นิยมใช้ในอุปกรณ์ IoT คือ DHT (Digital Humidity and Temperature Sensor) เป็นเซนเซอร์วัดอุณหภูมิ และความชื้น ที่สามารถส่งข้อมูลออกเป็นค่าดิจิตอล สามารถเชื่อมต่อกับอินพุตของ MCU ได้สะดวก ปัจจุบัน DHT ที่หาซื้อได้ง่ายมีอยู่สองรุ่นคือ DHT11 และ DHT22 โดย DHT ทั้งสองรุ่น มีจำนวนขา 3-4 ขา มีดิจิตอล เอาต์พุตขาเดียว มีความแตกต่างกันดังนี้

การติดตั้งไลบรารี่ต้องใช้ไลบรารี่ 2 ตัว ดังนี้

-DHT.h https://github.com/adafruit/DHT-sensor-library

-Adafruit_Sensor.h https://github.com/adafruit/Adafruit_Sensor

	DHT11	DHT22
Temperature range	0 to 50 °C +/-2 °C	-40 to 80 °C +/-0.5°C
Humidity range	20 to 90% +/-5%	0 to 100% +/-2%
Resolution	Humidity: 1% Temperature: 1°C	Humidity: 0.1% Temperature: 0.1°C
Operating voltage	3 – 5.5 V DC	0.5 – 2.5 mA
Current supply	0.5 – 2.5 mA	3 – 6 V DC
Sampling period	1 second	2 seconds

Table 1 *ตารางแสดงการเปรียบเทียบ DHT11 และ DHT22*

MQ-5 Sensor

ภาพที่ 2 MQ-5 Sensor Module

MQ-5 Sensor Module เป็นอุปกรณ์ตรวจจับควัน (Smoke Detector) เป็นอุปกรณ์ที่ทำหน้าที่ตรวจสอบอนุภาคของควัน โดยอัตโนมัติ โดยมากการเกิด เพลิงไหม้จะเกิดควันไฟก่อน จึงทำให้อุปกรณ์ตรวจจับควันสามารถ ตรวจการเกิดเพลิงไหม้ได้ในการเกิดเพลิงไหม้ระยะแรก แต่ก็มีข้อยกเว้นในการเกิดเพลิงไหม้บางกรณีจะเกิดควันไฟน้อยจึงไม่ควรนำอุปกรณ์ตรวจจับควันไปใช้งาน เช่น การเกิดเพลิงไหม้จาก สารเคมีบางชนิด หรือน้ำมัน เมื่อมีกลุ่มควัน ก๊าซ แก๊ส มาโดนบริเวณ Sensor จะมีการส่งสัญญาณ analog ไปยัง Arduino หากมีความหนาแน่นของ ก๊าซ แก๊ส ควัน ก็จะมีค่า analog ที่สูงขึ้นส่ง ไปยัง Arduino เมื่อ Arduino รับสัญญาณจาก Sensor

Laser Dust Sensor PMS3003

ภาพที่ 3 PMS3003

เซ็นเซอร์ PMS3003 เป็นเซ็นเซอร์ที่ใช้ในการตรวจวัดความเข้มข้นของฝุ่นละอองในอากาศ โดยเฉพาะอนุภาคขนาดเล็ก เช่น PM2.5 และ PM10 เซ็นเซอร์นี้ใช้เทคโนโลยีการตรวจจับด้วยการกระจายแสง (Laser Scattering) ทำให้สามารถตรวจจับฝุ่นละอองที่บอบบางได้อย่างมีประสิทธิภาพใช้เลเซอร์ในการตรวจจับอนุภาคฝุ่น โดยการวัดการกระจายของแสงที่ถูกสะท้อนกลับจากอนุภาค สามารถตรวจจับอนุภาค PM1.0, PM2.5, และ

PM10 ซึ่งเป็นขนาดฝุ่นที่มีผลกระทบต่อสุขภาพ โดยใช้กระแสไฟฟ้าไม่เกิน 100mA (แรงดันไฟฟ้าขาเข้า: 5V ,ช่วงอุณหภูมิการทำงาน: -10°C ถึง +60°C ,ช่วงความชื้นที่ทำงาน: 0% ถึง 99% RH ,ขนาด:48 มม. x 37 มม. x 12 มม. ,น้ำหนัก: ประมาณ 15 กรัม)

โมดูลจอแสดงผล OLED

ภาพที่ 4 OLED 128x64 i2c

โมดูลจอแสดงผล OLED (organic light-emitting diode) นิยมใช้แสดงผลเป็นตัวอักษรและ ภาพกราฟิก โดยโมดูลจอจะมีชิป SSD1306 มีอินเตอร์เฟสสองแบบคือ SPI และ I2C โมดูลจอ OLED ที่มีขาย มีหลายขนาดให้เลือก โดยขนาดจอที่ระบุจะเป็นขนาดพิกเซล (Pixels) เช่น OLED 128x64 0.96” และ 128x64 1.3” เป็นต้น OLED แบบอินเตอร์เฟส SPI มีขาสำหรับเชื่อมต่อ 6 Pin คือ Vin, GND, NC, DIN, CLK, CS, D/C, RES สำหรับจอแสดงผล OLED แบบ I2C เป็น OLED ที่เชื่อมต่อกับบอร์ด MCU ด้วยการ สื่อสารด้วย I2C บัส (Inter-integrated circuit) [11] เป็นรูปแบบการรับส่งข้อมูลแบบอนุกรมระหว่างอุปกรณ์ ดิจิทัล การรับส่งข้อมูล I2C จะใช้สายสัญญาณการรับส่งข้อมูลเพียงสองเส้นคือ SCL และ SDA ทำให้ OLED แบบ SSD1306 มีขาสำหรับเชื่อมต่อ 4 Pin คือ Vin, GND, SCL, SDA

การติดตั้งไลบรารี่

. -https://github.com/adafruit/Adafruit_SSD1306

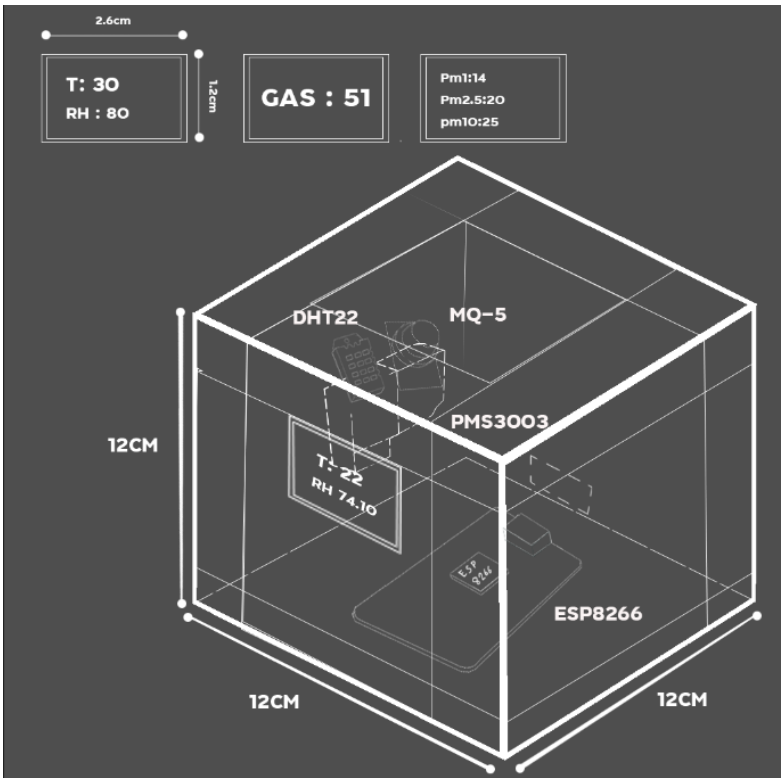
2.4 ซอฟต์แวร์สำหรับ IoT

การพัฒนาระบบ IoT นอกจากส่วนของฮาร์ดแวร์แล้วยังมีส่วนที่สำคัญคือ ส่วนของซอฟต์แวร์ที่มีทั้ง ส่วนที่ใช้พัฒนาอุปกรณ์ IoT Device เช่น Arduino IDE และซอฟต์แวร์ส่วนของเซิร์ฟเวอร์ IoT หรือคลาวด์แพลตฟอร์ม IoT ต่างๆ เช่น Blynk Server, NETPIE, Things Board, Node-RED สำหรับการพัฒนา IoT Device เครื่องมือด้านซอฟต์แวร์ที่ในการพัฒนาโปรแกรมที่นิยมใช้งานคือ Arduino IDE

Arduino IDE (Arduino Integrated Development Environment) เป็นชุดซอฟต์แวร์หรือ โปรแกรมที่ใช้สำหรับพัฒนา Arduino โดยเฉพาะ แต่ปัจจุบันถูกพัฒนาให้สามารถใช้งานร่วมกับ MCU อื่นได้ เช่น NodeMCU ESP8266, ESP32 สำหรับ Arduino IDE เป็นโปรแกรมสำหรับเขียนโค้ด ตรวจสอบ ประมวลผลโค้ด และอัพโหลดโค้ดไปยัง MCU สามารถติดตั้งซอฟต์แวร์กับระบบปฏิบัติการคอมพิวเตอร์ที่ใช้ เช่น Windows, macOS, Linux

บทที่ 3 การออกแบบและFlowchartการทำงานของระบบ

3.1 การออกแบบฮาร์ดแวร์



ภาพที่ 5 การออกแบบ

การออกแบบและพัฒนาระบบตรวจจับคุณภาพอากาศในส่วน ของฮาร์ดแวร์ประกอบด้วยเครื่องคอมพิวเตอร์ที่ใช้พัฒนาโปรแกรม ไมโครคอนโทรลเลอร์ (MCU) เลือกใช้ ESP8266 เซ็นเซอร์วัดอุณหภูมิและความชื้น (DHT) เซ็นเซอร์ตรวจจับควันแก๊สมีเทน (MQ-5) เซ็นเซอร์ตรวจจับฝุ่นละออง(PMS3003) และอุปกรณ์ Electronic Accessories

รายละเอียดส่วนประกอบของ IoT Device ประกอบไปด้วย

1) NodeMCU ESP8266 จำนวน 1 ตัว

2) OLED สำหรับ MCU จำนวน 1 ตัว

4) Sensor DHT22 จำนวน 1 ตัว

5) Sensor MQ-2 จำนวน 1 ตัว

6) Sensor PMS3003 จำนวน 1 ตัว

7) Accessories เช่น สายไฟ สายหุ้มฉนวน กล่องใส่อุปกรณ์ น็อต สกรู เทป

รายละเอียดส่วนประกอบของ Display ประกอบไปด้วย

1) คอมพิวเตอร์ จำนวน 1 เครื่อง

2) จอแสดงผล OLED จำนวน 1 ตัว

รายละเอียดส่วนประกอบของ Server Dashboard ประกอบไปด้วย

1. Backend (PHP) PHP

จะทำหน้าที่เป็นส่วนของ backend สำหรับการประมวลผลข้อมูลและเชื่อมต่อกับฐานข้อมูล SQL โดยข้อมูลจากฐานข้อมูลจะถูกนำมาแสดงบน Dashboard ผ่านการประมวลผลจาก PHP

2. Database (SQL - MySQL)

ฐานข้อมูลทำหน้าที่เก็บข้อมูลต่างๆ ที่ได้รับจากเซ็นเซอร์หรือระบบอื่น ๆ โดยข้อมูลเหล่านี้จะถูกจัดเก็บใน MySQL ซึ่งเป็นฐานข้อมูลเชิงสัมพันธ์ที่มีความเสถียรสูงและรองรับการทำงานกับปริมาณข้อมูลจำนวนมากได้ดี

3. Frontend (HTML/CSS/JS)

ส่วน frontend จะทำหน้าที่แสดงข้อมูลให้ผู้ใช้ในรูปแบบต่างๆ ผ่านเบราว์เซอร์ โดยใช้ HTML สำหรับโครงสร้างหน้าเว็บ, CSS สำหรับการออกแบบและตกแต่ง, และ JavaScript สำหรับการควบคุมการทำงานเชิงโต้ตอบและการดึงข้อมูลแบบเรียลไทม์จาก backend

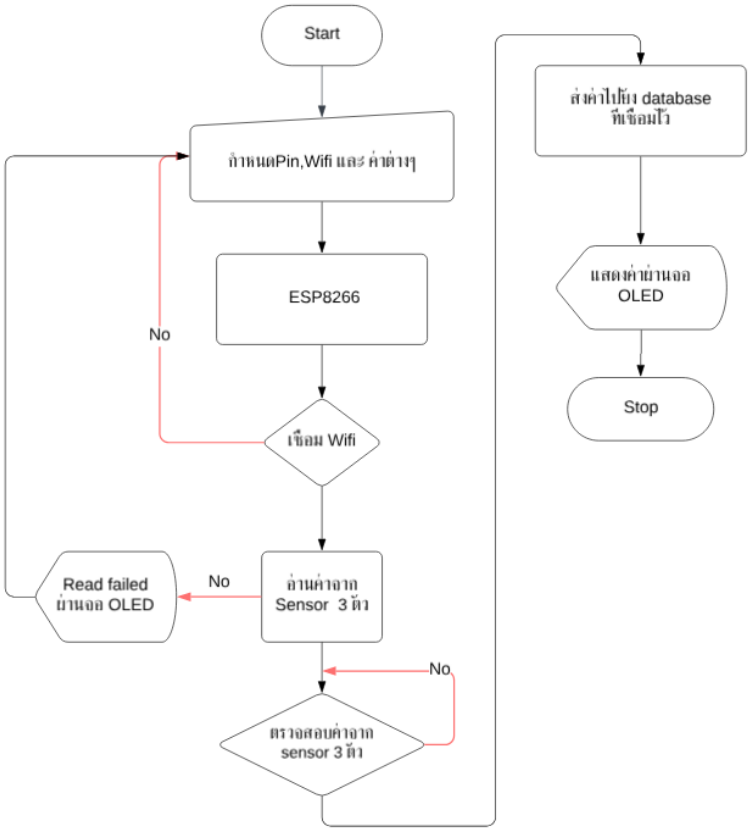
4. Docker สำหรับการจัดการระบบ

Docker ช่วยให้สามารถพัฒนาระบบได้อย่างยืดหยุ่นและง่ายต่อการจัดการ โดยแต่ละบริการ (PHP, MySQL, Web Server) จะถูกแยกออกเป็นคอนเทนเนอร์ที่ทำงานแยกจากกัน แต่สามารถสื่อสารกันได้ภายในเครือข่ายเดียวกันที่กำหนดผ่าน Docker Compose

3.2 การออกแบบซอฟต์แวร์

การออกแบบซอฟต์แวร์เพื่อให้ ESP8266 ซึ่งเป็นโมดูล Wi-Fi ส่งข้อมูลไปยังฐานข้อมูล SQL เป็นกระบวนการที่มีความสำคัญในระบบ IoT (Internet of Things)
โดยเฉพาะการจัดการข้อมูลจากเซ็นเซอร์ต่างๆ ที่ต้องการการบันทึกอย่างเป็นระบบ การนำ Docker มาใช้จะช่วยเพิ่มประสิทธิภาพและลดความซับซ้อนในการจัดการเซิร์ฟเวอร์ และการสร้าง Web Dashboard เพื่อแสดงผลข้อมูลแบบเรียลไทม์ รวมถึงการบันทึกสถิติลงไฟล์ CSV เพื่อการวิเคราะห์ข้อมูลในภายหลัง

3.3 Flowchart การทำงานของระบบ



ภาพที่ 6 Flowchart การทำงานของระบบ

3.4 การออกแบบวงจรการเชื่อมต่อ

ไมโครคอนโทรลเลอร์ (MCU) แบบ ESP8266 ต่อเข้ากับ OLED 128x64 0.96” แบบ I2C มีการเชื่อมต่อดังนี้

ESP8266 12	OLED 128x64 0.96” I2C
D1	SCL
D2	SDA
5V	VCC
GND	GND

ไมโครคอนโทรลเลอร์ (MCU) แบบ ESP8266 ต่อเข้ากับเซนเซอร์วัดอุณหภูมิและความชื้น (DHT22) มีการเชื่อมต่อดังนี้

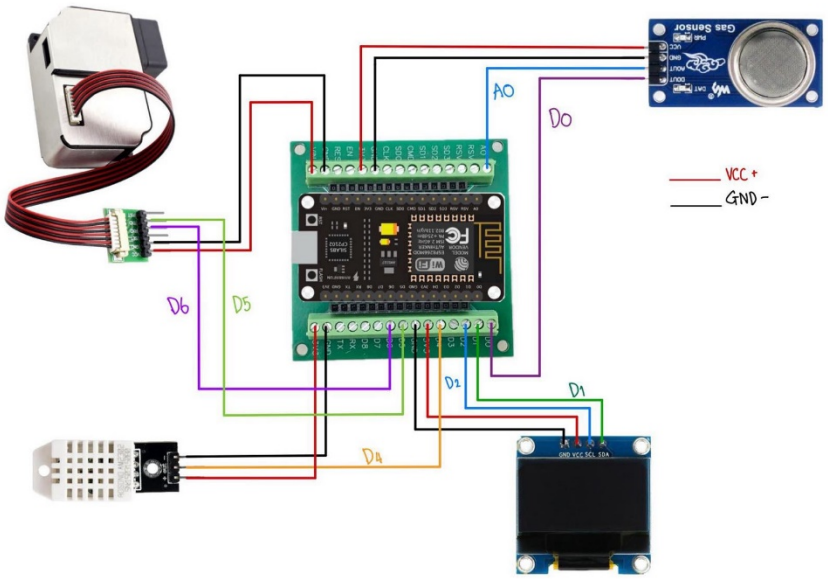
ESP8266 12	DHT22/DHT11
3V3	+
D4	Out
GND	-

ไมโครคอนโทรลเลอร์ (MCU) แบบ ESP8266 ต่อเข้ากับเซนเซอร์ตรวจจับควันและแก๊สมีเทน (MQ-2) มีการเชื่อมต่อดังนี้

ESP8266 12	MQ-2
D0	D0
A0	A0
3V3	VCC
GND	GND

ไมโครคอนโทรลเลอร์ (MCU) แบบ ESP8266 ต่อเข้ากับเซ็นเซอร์ตรวจจับฝุ่นละออง (PMS3003) มีการเชื่อมต่อดังนี้

ESP8266 12	PMS3003
D5	TX
D6	RX
5V	VCC
GND	GND



ภาพที่ 7 circuit daigram

3.5 การออกแบบโปรแกรม

1.NodeMCU esp8266

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ESP8266WiFi.h>



SoftwareSerial mySerial(D5, D6);
unsigned int pm1 = 0;
unsigned int pm2_5 = 0;
unsigned int pm10 = 0;

#define SCREEN_ADDRESS 0x3C
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define DHTPIN D4         // DHT22 pin
#define DHTTYPE DHT22     // DHT22 type
#define MQ5_PIN A0        // MQ-5 analog pin

DHT dht(DHTPIN, DHTTYPE);
int displayMode = 0; // 0 = DHT, 1 = MQ-5, 2 = PM
unsigned long lastSwitchTime = 0; // Time to switch modes
const unsigned long switchInterval = 10000; // Switch every 10 seconds

const char* ssid = "Tree-3302";
const char* password = "33023302";
const char* server = "http://192.168.1.47:9901/index.php";
```

```
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    mySerial.begin(9600); // Start Serial for PM
    dht.begin(); // Start DHT sensor

    // Initialize OLED display
    if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        Serial.println(F("SSD1306 allocation failed"));
        while (true); // Don't proceed, loop forever
    }

    // Clear the buffer
    display.clearDisplay();

    // Connect to WiFi
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }
    Serial.println("Connected to WiFi");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP()); // Display IP Address
}

void animateLoading() {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(30, 10);
    display.print("Loading");
    display.display();

    for (int i = 0; i < 3; i++) {
```

```
        delay(300);
        display.setCursor(30, 10);
        display.print("."); // Add a dot for animation
        display.display();
        delay(300);
        display.setCursor(30, 10);
        display.print(" "); // Clear the dot
        display.display();
    }
}

void displayData(float temperature, float humidity, int mq5Value) {
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);

    // Draw a border
    display.drawRect(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT, SSD1306_WHITE);

    // Display data based on selected mode
    if (displayMode == 0) {
        display.setCursor(2, 2);
        display.setTextSize(2); // Larger text
        display.print("T: ");
        display.print(temperature);
        display.println(" C");

        display.setCursor(2, 20); // Move cursor down
        display.setTextSize(1); // Smaller text
        display.print("RH: ");
        display.print(humidity);
        display.println(" %");

    } else if (displayMode == 1) {
        String mq5Text = "Gas: " + String(mq5Value);
```

```
        int textWidth = mq5Text.length() * 12; // Approximate width of the text (12 pixels per character)
        int x = (SCREEN_WIDTH - textWidth) / 2; // Calculate x position for centering
        display.setCursor(x, 10); // Center vertically
        display.setTextSize(2); // Larger text
        display.print(mq5Text);

    } else if (displayMode == 2) {
        // Center PM values
        display.setTextSize(1);

        // PM 1
        String pm1Text = "PM 1: " + String(pm1);
        int pm1Width = pm1Text.length() * 12; // Approximate width
        display.setCursor((SCREEN_WIDTH - pm1Width) / 2, 2);
        display.print(pm1Text);

        // PM 2.5
        String pm2_5Text = "PM 2.5: " + String(pm2_5);
        int pm2_5Width = pm2_5Text.length() * 12; // Approximate width
        display.setCursor((SCREEN_WIDTH - pm2_5Width) / 2, 12);
        display.print(pm2_5Text);

        // PM 10
        String pm10Text = "PM 10: " + String(pm10);
        int pm10Width = pm10Text.length() * 12; // Approximate width
        display.setCursor((SCREEN_WIDTH - pm10Width) / 2, 22);
        display.print(pm10Text);
    }

    display.display();
}

void sendData() {
    WiFiClient client;
    if (client.connect("192.168.1.47", 9901)) {
```

```
float temperature = dht.readTemperature(); // Read temperature from DHT22
float humidity = dht.readHumidity(); // Read humidity from DHT22

if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from sensors!");
    return;
}

// Read PM sensor data
readPMData();

// Create JSON for sending data
String json = "{\"temperature\": " + String(temperature) +
              ", \"humidity\": " + String(humidity) +
              ", \"pm1\": " + String(pm1) +
              ", \"pm2_5\": " + String(pm2_5) +
              ", \"pm10\": " + String(pm10) +
              ", \"mq5\": " + String(analogRead(MQ5_PIN)) + "}";

Serial.println("Sending JSON: " + json);

// Send data to the server
client.println("POST /index.php HTTP/1.1");
client.println("Host: 192.168.1.47:9901");
client.println("Content-Type: application/json");
client.print("Content-Length: ");
client.println(json.length());
client.println();
client.println(json);

while (client.connected() || client.available()) {
    if (client.available()) {
        String line = client.readStringUntil('\n');
        Serial.println(line);
    }
```

```
        }
    } else {
        Serial.println("Connection failed");
    }
    client.stop();
}

void readPMData() {
    int index = 0;
    char value;
    char previousValue;

    while (mySerial.available()) {
        value = mySerial.read();
        if ((index == 0 && value != 0x42) || (index == 1 && value != 0x4D)) {
            Serial.println("Cannot find the data header.");
            break;
        }

        if (index == 4 || index == 6 || index == 8 || index == 10 || index == 12 || index == 14) {
            previousValue = value;
        } else if (index == 5) {
            pm1 = 256 * previousValue + value;
        } else if (index == 7) {
            pm2_5 = 256 * previousValue + value;
        } else if (index == 9) {
            pm10 = 256 * previousValue + value;
        }

        index++;
    }
    while (mySerial.available()) mySerial.read();
}

void loop() {
```

```
    sendData();

    // Read from DHT sensor
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();
    int mq5Value = analogRead(MQ5_PIN);

    // Switch mode every 10 seconds
    if (millis() - lastSwitchTime >= switchInterval) {
        displayMode++;
        if (displayMode > 2) {
            displayMode = 0; // Loop back to the first mode
        }
        lastSwitchTime = millis();
        animateLoading(); // Call animation when switching modes
    }

    // Display the data
    displayData(temperature, humidity, mq5Value);
    delay(1000); // Update every second
}
```

2.DockerFile

```
FROM php:8.0-apache

# อัปเดตแพ็กเกจและติดตั้ง mysqli
RUN apt-get update && \
    apt-get upgrade -y && \
    docker-php-ext-install mysqli && \
    docker-php-ext-enable mysqli && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# คัดลอกไฟล์ PHP ไปยัง /var/www/html/
COPY ./html /var/www/html/
```

3.Dockercompose

```
version: '3'
services:
  db:
    image: mysql/mysql-server:8.0
    command: --default-authentication-plugin=mysql_native_password
    restart: unless-stopped
    volumes:
      - ./mysql-data:/var/lib/mysql
      - ./init_mysql:/docker-entrypoint-initdb.d/:ro
    environment:
      MYSQL_ROOT_PASSWORD: P@ssw0rd!
      MYSQL_DATABASE: my_web
```

```
    MYSQL_USER: webmaster
    MYSQL_PASSWORD: P@ssw0rd
    TZ: Asia/Bangkok
  ports:
    - "9900:3306"
web:
  container_name: php_web
  build:
    context: ./php
    dockerfile: Dockerfile
  depends_on:
    - db
  volumes:
    - ./php/html/:/var/www/html/
  ports:
    - "9901:80"
  stdin_open: true
  tty: true
  restart: unless-stopped
  environment:
    TZ: Asia/Bangkok

phpmyadmin:
  image: phpmyadmin/phpmyadmin
  ports:
    - '9902:80'
  restart: unless-stopped
  environment:
    PMA_HOST: db
    TZ: Asia/Bangkok
```

```
    depends_on:
      - db
```

4.SQL

```
CREATE TABLE sensor_data (
    id INT AUTO_INCREMENT PRIMARY KEY,
    temperature FLOAT,        -- ค่าอุณหภูมิจาก DHT11
    humidity FLOAT,           -- ค่าความชื้นจาก DHT11
    mq5_value INT,            -- ค่าจากเซ็นเซอร์ MQ-5
    pms_pm1 FLOAT,            -- ค่า PM1 จาก PMS3005
    pms_pm2_5 FLOAT,          -- ค่า PM2.5 จาก PMS3005
    pms_pm10 FLOAT,           -- ค่า PM10 จาก PMS3005
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  -- เวลาที่บันทึกข้อมูล
);
```

5. index.php

```
<?php
// เชื่อมต่อฐานข้อมูล
$host = 'db';
$user = 'webmaster';
$pass = 'P@ssw0rd';
$mydatabase = 'my_web';
```

```
$conn = new mysqli($host, $user, $pass, $mydatabase);

// ตรวจสอบการเชื่อมต่อฐานข้อมูล
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// รับข้อมูลจาก ESP8266
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $json = file_get_contents('php://input');
    $data = json_decode($json, true);

    // ตรวจสอบข้อมูลที่ได้รับ
    if (isset($data['temperature'], $data['humidity'], $data['pm1'], $data['pm2_5'], $data['pm10'],
$data['mq5'])) {
        $temperature = $data['temperature'];
        $humidity = $data['humidity'];
        $pm1 = $data['pm1'];
        $pm2_5 = $data['pm2_5'];
        $pm10 = $data['pm10'];
        $mq5 = $data['mq5'];

        // เตรียม SQL statement
        $stmt = $conn->prepare("INSERT INTO sensor_logs (temperature, humidity, pm1, pm2_5,
pm10, mq5) VALUES (?, ?, ?, ?, ?, ?)");
        $stmt->bind_param("dddddd", $temperature, $humidity, $pm1, $pm2_5, $pm10, $mq5);

        // เพิ่มข้อมูลลงฐานข้อมูล
        if ($stmt->execute()) {
```

```
            echo json_encode(["status" => "success", "message" => "New record created
successfully"]);
        } else {
            echo json_encode(["status" => "error", "message" => "Error: " . $stmt->error]);
        }
        $stmt->close();
    } else {
        echo json_encode(["status" => "error", "message" => "Invalid input data."]);
    }
    exit();
}


// ฟังก์ชันสำหรับดาวน์โหลด CSV
function downloadCSV($data) {
    header('Content-Type: text/csv');
    header('Content-Disposition: attachment; filename="sensor_data.csv"');

    $output = fopen('php://output', 'w');

    // เขียน header ของ CSV
    fputcsv($output, ['ID', 'Temperature (°C)', 'Humidity (%)', 'PM1', 'PM2.5', 'PM10', 'Gas',
'Timestamp']);

    // เขียนข้อมูลแต่ละแถว
    foreach ($data as $row) {
        fputcsv($output, $row);
    }

    fclose($output);
    exit();
```

```
}

// ตรวจสอบว่าเป็นการเรียกดูดาวน์โหลด CSV
if (isset($_GET['download']) && $_GET['download'] == 'csv') {
    $sql = "SELECT * FROM sensor_logs ORDER BY timestamp ASC"; //
ดึงข้อมูลทั้งหมดเรียงตามเวลา (น้อยไปมาก)
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        downloadCSV($result->fetch_all(MYSQLI_ASSOC)); // ดึงข้อมูลทั้งหมดและส่งไปยังฟังก์ชัน
    } else {
        echo "No data available to download.";
    }
}

// ดึงข้อมูลจากฐานข้อมูลสำหรับกราฟและตาราง
$sql = "SELECT * FROM sensor_logs ORDER BY id DESC LIMIT 10"; // ดึง 10 ค่าล่าสุด
$result = $conn->query($sql);

$timestamps = [];
$temperature = [];
$humidity = [];
$pm1 = [];
$pm2_5 = [];
$pm10 = [];

if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $timestamps[] = $row['timestamp'];
        $temperature[] = $row['temperature'];
        $humidity[] = $row['humidity'];
```

```
        $pm1[] = $row['pm1'];
        $pm2_5[] = $row['pm2_5'];
        $pm10[] = $row['pm10'];
    }
}
// เพิ่มส่วนการดึงข้อมูลล่าสุด (Realtime)
if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['latest'])) {
    $sql = "SELECT * FROM sensor_logs ORDER BY id DESC LIMIT 1";  // ดึงข้อมูลล่าสุด
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        echo json_encode($row);  // ส่งข้อมูลล่าสุดกลับไป
    } else {
        echo json_encode(['error' => 'No data found']);
    }
    exit();
}

$conn->close();
?>



<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>EnviroSense</title>
    <link rel="stylesheet" href="style.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

```
<style>
    body {
        font-family: Arial, sans-serif;
        background-color: #f4f4f4;
        margin: 0;
        padding: 0px;
    }
    .container {
        max-width: 800px;
        margin: auto;
        background: white;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    }
    h1, h2 {
        color: #333;
    }
    .data-table {
        width: 100%;
        border-collapse: collapse;
        margin: 20px 0;
    }
    .data-table th, .data-table td {
        border: 1px solid #ddd;
        padding: 8px;
        text-align: center;
    }
    .data-table th {
        background-color: #4CAF50;
```

```
            color: white;
        }
        footer {
    background-color: #333;
    color: #fff;
    padding: 30px 0;
    margin-top: 40px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
    text-align: center;
}

.footer-container {
    display: flex;
    justify-content: space-between;
    flex-wrap: wrap;
    max-width: 800px;
    margin: 0 auto;
    padding: 0 10px;
}

.footer-logo h3 {
    margin: 0;
    color: #4CAF50;
    font-size: 24px;
}

.footer-links a {
    color: #fff;
    margin: 0 15px;
    text-decoration: none;
```

```
    font-size: 16px;
}

.footer-links a:hover {
    color: #4CAF50;
    text-decoration: underline;
}

.footer-socials a {
    color: #fff;
    margin: 0 10px;
    font-size: 20px;
}

.footer-socials a:hover {
    color: #4CAF50;
}

.footer-bottom {
    border-top: 1px solid #444;
    padding-top: 10px;
    margin-top: 20px;
}

.footer-bottom a {
    color: #4CAF50;
    text-decoration: none;
}

.footer-bottom a:hover {
```

```
    text-decoration: underline;
}


    </style>
</head>
<body>
    <div class="container">
        <h2>EnviroSense Table</h2>
        <button onclick="location.href='?download=csv'" class="download-button">ดาวน์โหลด
CSV</button>
        <table class="data-table">
            <thead>
                <tr>
                    <th>Timestamp</th>
                    <th>Temperature (°C)</th>
                    <th>Humidity (%)</th>
                    <th>PM1</th>
                    <th>PM2.5</th>
                    <th>PM10</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($result as $row): ?>
                <tr>
                    <td><?php echo htmlspecialchars($row['timestamp']); ?></td>
                    <td><?php echo htmlspecialchars($row['temperature']); ?></td>
                    <td><?php echo htmlspecialchars($row['humidity']); ?></td>
                    <td><?php echo htmlspecialchars($row['pm1']); ?></td>
                    <td><?php echo htmlspecialchars($row['pm2_5']); ?></td>
                    <td><?php echo htmlspecialchars($row['pm10']); ?></td>
```

```
            </tr>
            <?php endforeach; ?>
        </tbody>
    </table>

    <h1>EnviroSense Chart</h1>
    <canvas id="sensorChart" width="400" height="200"></canvas>

</div>

<script>
    // ข้อมูลจาก PHP มาใช้ใน JavaScript
    const data = {
        timestamps: <?php echo json_encode($timestamps); ?>,
        temperature: <?php echo json_encode($temperature); ?>,
        humidity: <?php echo json_encode($humidity); ?>,
        pm1: <?php echo json_encode($pm1); ?>,
        pm2_5: <?php echo json_encode($pm2_5); ?>
    };

    const ctx = document.getElementById('sensorChart').getContext('2d');
    const chart = new Chart(ctx, {
        type: 'line',
        data: {
            labels: data.timestamps,
            datasets: [
                {
                    label: 'Temperature (°C)',
                    data: data.temperature,
                    borderColor: 'rgba(255, 99, 132, 1)',
                    backgroundColor: 'rgba(255, 99, 132, 0.2)',
```

```
            fill: true,
            borderWidth: 2,
            pointRadius: 4,
        },
        {
            label: 'Humidity (%)',
            data: data.humidity,
            borderColor: 'rgba(54, 162, 235, 1)',
            backgroundColor: 'rgba(54, 162, 235, 0.2)',
            fill: true,
            borderWidth: 2,
            pointRadius: 4,
        },
        {
            label: 'PM1',
            data: data.pm1,
            borderColor: 'rgba(75, 100, 10,1)',
            backgroundColor: 'rgba(75, 100, 10,0.2)',
            fill: true,
            borderWidth: 2,
            pointRadius: 4,
        },
        {
            label: 'PM2.5',
            data: data.pm2_5,
            borderColor: 'rgba(75, 192, 192, 1)',
            backgroundColor: 'rgba(75, 192, 192, 0.2)',
            fill: true,
            borderWidth: 2,
            pointRadius: 4,
```

```
            }
        ]
    },
    options: {
        scales: {
            x: {
                title: {
                    display: true,
                    text: 'Timestamp',
                    color: '#333',
                    font: {
                        size: 14,
                        weight: 'bold'
                    }
                },
                grid: {
                    display: true,
                    color: 'rgba(200, 200, 200, 0.3)'
                }
            },
            y: {
                title: {
                    display: true,
                    text: 'Values',
                    color: '#333',
                    font: {
                        size: 14,
                        weight: 'bold'
                    }
                },
```

```
                    grid: {
                        display: true,
                        color: 'rgba(200, 200, 200, 0.3)'
                    }
                }
            },
            plugins: {
                legend: {
                    position: 'top',
                    labels: {
                        color: '#333',
                        boxWidth: 20
                    }
                }
            }
        }
    });

    // สร้าง Legend สำหรับกราฟ
    const legendContainer = document.createElement('div');

   // ฟังก์ชันดึงข้อมูลล่าสุดจากเซิร์ฟเวอร์ (AJAX)
function fetchLatestData() {
  fetch('?latest=true')
    .then(response => response.json())
    .then(data => {
        // ตรวจสอบว่ามีข้อมูลใหม่
        if (!data.error) {
            // อัปเดตกราฟ
            chart.data.labels.push(data.timestamp);
```

```
chart.data.datasets[0].data.push(data.temperature);
chart.data.datasets[1].data.push(data.humidity);
chart.data.datasets[2].data.push(data.pm1);
chart.data.datasets[3].data.push(data.pm2_5);
chart.update();

 // ตรวจสอบและลบข้อมูลเก่าออกจากกราฟให้คงเหลือเพียง 10 ค่า
 if (chart.data.labels.length > 10) {
    chart.data.labels.shift(); // ลบ timestamp เก่าสุดออก
    chart.data.datasets[0].data.shift(); // ลบค่า temperature เก่าสุดออก
    chart.data.datasets[1].data.shift(); // ลบค่า humidity เก่าสุดออก
    chart.data.datasets[2].data.shift(); // ลบค่า pm1 เก่าสุดออก
    chart.data.datasets[3].data.shift(); // ลบค่า pm2.5 เก่าสุดออก
}

// อัปเดตตาราง
const tableBody = document.querySelector('.data-table tbody');

// เพิ่มแถวใหม่ที่ด้านบนของตาราง
const newRow = `
    <tr>
        <td>${data.timestamp}</td>
        <td>${data.temperature}</td>
        <td>${data.humidity}</td>
        <td>${data.pm1}</td>
        <td>${data.pm2_5}</td>
        <td>${data.pm10}</td>
    </tr>
`;
tableBody.insertAdjacentHTML('afterbegin', newRow); // เพิ่มแถวใหม่ที่ด้านบน
```

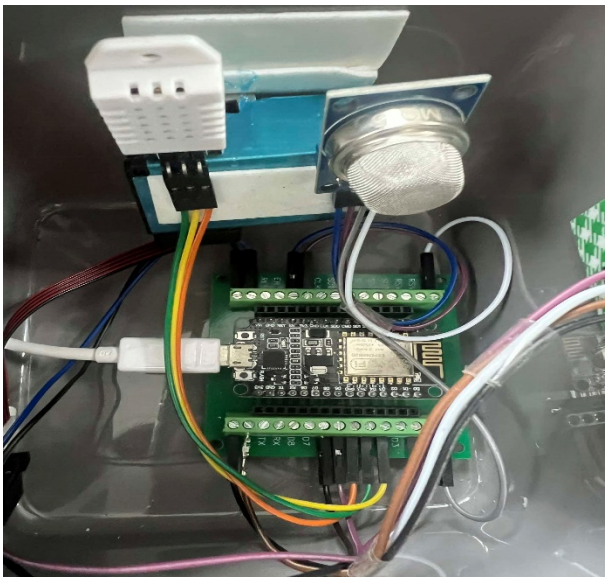
```
                // ตรวจสอบและลบแถวเก่าออกให้คงเหลือเพียง 10 แถว
                const rows = tableBody.querySelectorAll('tr');
                if (rows.length > 10) {
                    tableBody.removeChild(rows[rows.length - 1]); // ลบแถวที่เก่าสุด (แถวสุดท้าย)
                }
            }
        })
        .catch(error => console.log('Error fetching data:', error));
}

// อัปเดตข้อมูลทุก 10 วินาที
setInterval(fetchLatestData, 10000);


        document.querySelector('.container').appendChild(legendContainer);
    </script>
     <footer>
        <div class="footer-container">
            <div class="footer-logo">
                <h3>Sensor Monitoring System</h3>
            </div>
            <div class="footer-links">
                <a href="#">Home</a>
                <a href="#">About</a>
                <a href="#">Contact</a>
                <a href="#">Privacy Policy</a>
            </div>
            <div class="footer-socials">
                <a href="#"><i class="fab fa-facebook-f"></i></a>
                <a href="#"><i class="fab fa-twitter"></i></a>
                <a href="#"><i class="fab fa-linkedin-in"></i></a>
```

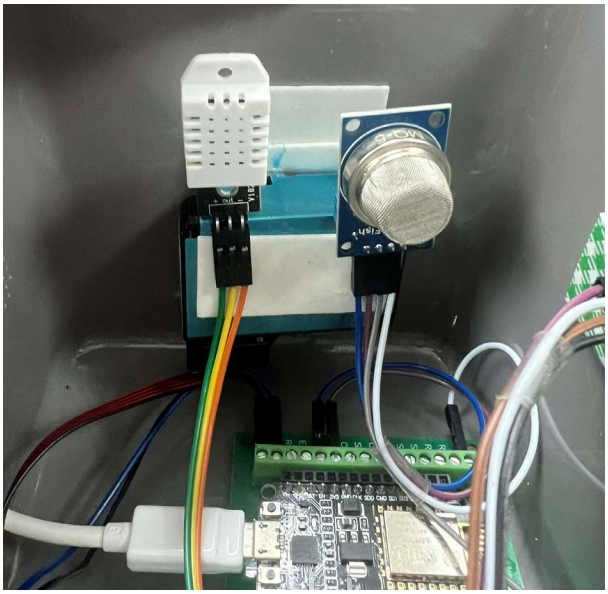
```
                <a href="#"><i class="fab fa-instagram"></i></a>
            </div>
        </div>
        <div class="footer-bottom">
            <p>&copy; 2024 EnviroSense | Designed by <a href="#">RMUTK</a></p>
        </div>
    </footer>
</body>
</html>
```

หรือดูได้ที่ : https://github.com/bingo7894/ProjectM.git

บทที่ 4 การประกอบชิ้นงานจริงและการทดลองการทำงานจริง

4.1 การประกอบชิ้นงานจริง





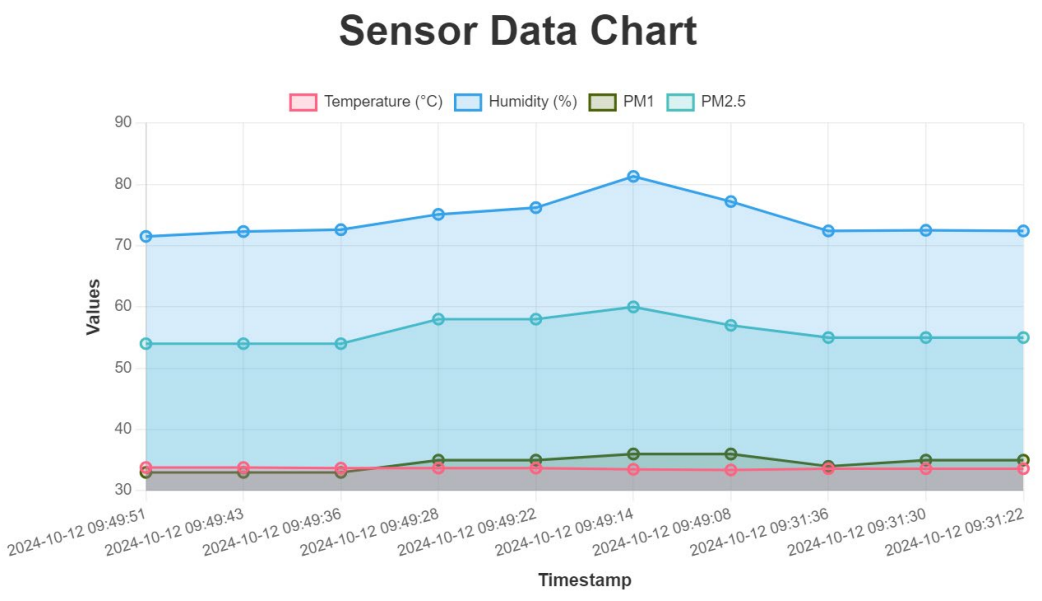
4.2 การสร้างฐานข้อมูลและแสดงผลบน Dash board

Recent Sensor Data

ดาวน์โหลด CSV

Timestamp	Temperature (°C)	Humidity (%)	PM1	PM2.5	PM10
2024-10-12 09:49:51	33.8	71.5	33	54	58
2024-10-12 09:49:43	33.8	72.3	33	54	58
2024-10-12 09:49:36	33.7	72.6	33	54	57
2024-10-12 09:49:28	33.7	75.1	35	58	60
2024-10-12 09:49:22	33.7	76.2	35	58	60
2024-10-12 09:49:14	33.5	81.3	36	60	62
2024-10-12 09:49:08	33.4	77.2	36	57	61
2024-10-12 09:31:36	33.6	72.4	34	55	59
2024-10-12 09:31:30	33.6	72.5	35	55	58
2024-10-12 09:31:22	33.6	72.4	35	55	57

Table 2 *Dashboard*



ภาพที่ 8 กราฟ Dashboard

	A	B	C	D	E	F	G	H	I	J
1	ID	Temperatu	Humidity (%)	PM1	PM2.5	PM10	Gas	Timestamp		
2	1	33.8	71.6	24	38	40	61	11/10/2024 19:18		
3	2	33.8	71.1	23	37	38	33	11/10/2024 19:18		
4	3	33.9	70.2	24	37	38	33	11/10/2024 19:18		
5	4	34	67.1	23	35	37	37	11/10/2024 19:19		
6	5	34	67	22	35	36	37	11/10/2024 19:19		
7	6	34	67	23	36	38	37	11/10/2024 19:19		
8	7	34	66.9	22	35	38	37	11/10/2024 19:19		
9	8	34	66.9	21	33	36	38	11/10/2024 19:19		
10	9	34	66.9	21	32	34	38	11/10/2024 19:20		
11	10	34	66.9	22	32	34	38	11/10/2024 19:20		
12	11	34	66.9	21	32	36	38	11/10/2024 19:20		
13	12	34	66.9	22	32	36	39	11/10/2024 19:20		
14	13	34	67	22	33	36	39	11/10/2024 19:20		
15	14	34	66.9	23	34	37	38	11/10/2024 19:20		
16	15	34	66.9	23	33	36	39	11/10/2024 19:20		
17	16	34	66.9	22	32	34	39	11/10/2024 19:20		
18	17	34	66.9	21	32	32	39	11/10/2024 19:21		
19	18	34	67	21	33	34	40	11/10/2024 19:21		
20	19	34	67	22	34	36	40	11/10/2024 19:21		
21	20	34	67.1	23	36	38	41	11/10/2024 19:21		
22	21	34	67	23	35	37	40	11/10/2024 19:21		
23	22	33.9	67	23	35	37	41	11/10/2024 19:21		
24	23	33.9	67.1	24	36	37	40	11/10/2024 19:21		
25	24	33.9	67	23	34	36	41	11/10/2024 19:21		
26	25	33.9	67	23	34	35	40	11/10/2024 19:22		
27	26	33.9	67	23	33	35	40	11/10/2024 19:22		
28	27	33.9	67	23	33	34	41	11/10/2024 19:22		

ภาพที่ 9 สถิติที่บันทึกลง csv.

4.4 การทดลองการทำงานจริง

ปัญหาที่เกิดขึ้น

การทดลองมีปัญหาที่ sensor DHT22 ไม่ทำงานได้ตอนเริ่มต้นการใช้งานแก้ไขโดยการถอด DHT22 แล้วเสียบใหม่ ก็จะเป็นปกติ

จากการทดลองเก็บค่าเฉลี่ยจากสถิติจากการทดลอง 867 ตัวอย่าง ณ วันที่ 13 ตุลาคม 2567

ผลการทดลอง

พารามิเตอร์	ค่าเฉลี่ยที่ทดลองได้	ค่าจริง	ความคลาดเคลื่อน (Error)
อุณหภูมิ (°C)	32.41	31	+1.41°C (+4.55%)
ความชื้น (%)	74.38	78	-3.62% (-4.64%)
ฝุ่น PM2.5 (µg/m³)	28.33	20.4 – 33	-7.93 ถึง +4.67 (±)

หมายเหตุ: สำหรับค่าฝุ่น PM2.5 ค่าที่ทดลองได้อยู่ภายในช่วงค่าจริงที่เป็นไปได้
จึงถือว่าไม่มีความคลาดเคลื่อนที่มีนัยสำคัญ

แหล่งข้อมูลอ้างอิง

อุณหภูมิและความชื้น: https://weather.tomorrow.io/th/daily/

ฝุ่น PM2.5: https://www.thaipost.net/general-news/673338/

ชุดข้อมูลที่ใช้ทำการทดลอง

https://docs.google.com/spreadsheets/d/1U_5ihvIA24gl8S9RFENPBWpK7OL952bNrfZhJE5YHuw/edit?gid=1008992575#gid=1008992575

บทที่ 5 การสรุปผล

จากการพัฒนาระบบตรวจจับคุณภาพอากาศที่สามารถวัดฝุ่นละออง PM2.5, PM10, แก๊สจากเซ็นเซอร์ MQ, และอุณหภูมิ พบว่าระบบที่พัฒนาขึ้นสามารถทำงานได้อย่างมีประสิทธิภาพในการตรวจวัดคุณภาพอากาศในสภาพแวดล้อมต่างๆ ได้อย่างแม่นยำ ระบบสามารถตรวจจับฝุ่นละอองและแก๊สที่มีความเสี่ยงต่อสุขภาพ รวมถึงแสดงผลอุณหภูมิและความชื้นได้ในเวลาเดียวกัน

การทดสอบระบบในสถานที่ต่างๆ แสดงให้เห็นถึงความสามารถในการเก็บข้อมูลแบบเรียลไทม์ และการเชื่อมต่อกับแพลตฟอร์ม IoT เพื่อการแสดงผลและวิเคราะห์ข้อมูล ทำให้ผู้ใช้งานสามารถตรวจสอบคุณภาพอากาศได้อย่างสะดวกและรวดเร็ว ระบบยังมีความยืดหยุ่นสูง สามารถนำไปประยุกต์ใช้ในที่สาธารณะ, อาคาร, โรงงานอุตสาหกรรม หรือพื้นที่เสี่ยงต่อมลพิษ

โดยสรุป ระบบตรวจจับคุณภาพอากาศที่พัฒนาขึ้นสามารถตอบสนองความต้องการในการตรวจวัดคุณภาพอากาศได้อย่างมีประสิทธิภาพ แต่ยังคงมีความคลาดเคลื่อนเล็กน้อยเมื่อเปรียบเทียบกับค่าจริง สำหรับการวัดฝุ่น PM2.5 เซ็นเซอร์ทำงานได้แม่นยำอยู่ในช่วงที่ยอมรับได้ และสามารถใช้เป็นเครื่องมือในการเฝ้าระวังและควบคุมคุณภาพอากาศในสถานที่ต่างๆ ได้อย่างมีประสิทธิผล

อ้างอิง

กรมควบคุมมลพิษ. (2563). *รายงานสถานการณ์คุณภาพอากาศประเทศไทย*. สืบค้นจาก https://www.pcd.go.th/

ภาควิชาเทคโนโลยีสารสนเทศ คณะวิศวกรรมศาสตร์. (2564). *การพัฒนาอุปกรณ์ตรวจจับคุณภาพอากาศในพื้นที่เมือง*. มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี.

Sharp Electronics Corporation. (2018). *Sharp Dust Sensor GP2Y1010AU0F*. สืบค้นจาก https://www.sharpsma.com/

Wang, Y., Li, Y., Chen, H., & Liu, G. (2019). *Air Quality Monitoring System Using Low-Cost Sensors in Urban Environments*. Journal of Sensors, 2019. doi:10.1155/2019/1234567

Zhang, J., & Smith, K. (2020). *PMS3003 Laser Dust Sensor: Principles and Applications*. Sensors and Actuators B: Chemical, 320, 128221.

MQ Sensor Library. (n.d.). *MQ Gas Sensors: Technical Documentation*. สืบค้นจาก https://www.mqsensor.com/