การพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะโดยใช้ IoT และ GIS

Development of an GPS Tracking System for Efficient Monitoring and Route Analysis of Garbage Collection Vehicles using IoT and GIS-Base

ปรัชญากุล มณีฉาย

วิทยานิพนธ์ระดับปริญญาตรี เสนอภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม

คณะเกษตรศาสตร์ ทรัพยากรธรรมชาติและสิ่งแวดล้อม มหาวิทยาลัยนเรศวร

เพื่อเป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิทยาศาสตรบัณฑิต

สาขาวิชาภูมิศาสตร์

ตุลาคม 2568

อาจารย์ที่ปรึกษา ประธานหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาภูมิศาสตร์และหัวหน้าภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม คณะเกษตรศาสตร์ทรัพยากรธรรมชาติและสิ่งแวดล้อมได้พิจารณาวิทยานิพนธ์ระดับปริญญาตรีเรื่อง "การพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะโดยใช้ IoT และ GIS (Development of an GPS Tracking System for Efficient Monitoring and Route Analysis of Garbage Collection Vehicles using IoT and GIS-Base) ของ ปรัชญากุล มณีฉาย เห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาภูมิศาสตร์ ของมหาวิทยาลัยนเรศวร

.....

(รองศาสตราจารย์ ดร.สิทธิชัย ชูสำโรง)

อาจารย์ที่ปรึกษาวิทยานิพนธ์

.....

(อาจารย์ ธัญญาลัคน์ จันทร์สมบัติ)

ประธานบริหารหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาภูมิศาสตร์

.....

(ผู้ช่วยศาสตราจารย์ ว่าที่ ร.ต.ดร. รังสรรค์ เกตุอ๊อด)

หัวหน้าภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม

กิตติกรรมประกาศ

วิทยานิพนธ์ระดับปริญญาตรี เรื่องการพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะโดยใช้ IoT และ GIS Development of an GPS Tracking System for Efficient Monitoring and Route Analysis of Garbage Collection Vehicles using IoT and GIS-Base) ฉบับนี้สำเร็จลุล่วงได้ด้วยดี เนื่องจากได้ความกรุณาและให้ความอนุเคราะห์ ช่วยในการ ดำเนินงานจัดทำวิทยานิพนธ์ในครั้งนี้

ขอขอบพระคุณ รองศาสตราจารย์ ดร.สิทธิชัย ชูสำโรง อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้ข้อเสนอแนะ แก้ไข สนับสนุนและมอบแนวคิดต่าง ๆ ที่มีประโยชน์ และยังสละเวลาในการตรวจทานและแก้ไขข้อบกพร่องของงานวิจัย พร้อมทั้งชี้แนวทางการดำเนินงานตลอดระยะเวลาในการทำวิทยานิพนธ์ฉบับนี้ ท่านมีบทบาทสำคัญในการช่วยให้วิทยานิพนธ์ฉบับนี้ก้าวหน้าอย่างมีคุณภาพและครบถ้วนตามเป้าหมายที่กำหนด ทั้งนี้ยังคอยตรวจสอบและแก้ไขข้อบกพร่องและติดตามผลการศึกษาอย่างสม่ำเสมอ และช่วยแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการดำเนินงานอันมีประโยชน์อย่างยิ่ง จนทำให้วิทยานิพนธ์ฉบับนี้เสร็จลุล่วงได้อย่างสมบูรณ์และขอขอบพระคุณคณาจารย์สาขาภูมิศาสตร์ทุกท่านที่ได้ ถ่ายทอดความรู้ต่างๆ กับผู้วิจัย ให้สามารถนำความรู้ที่เรียนมาใช้ให้เกิดประโยชน์สูงสุด และให้คำแนะนำ เพิ่มเติมจนทำให้วิทยานิพนธ์ฉบับนี้สมบูรณ์ยิ่งขึ้น

สุดท้ายนี้ขอกราบขอบพระคุณ บิดา มารดา ที่คอยเป็นกำลังใจและคอยช่วยเหลือมาโดยตลอด เกี่ยวกับ กำลังทรัพย์ตลอดจนสำเร็จการศึกษา รวมถึงเพื่อนๆ พี่ๆ และน้องๆ ทุกท่านที่เป็นผู้สนับสนุนให้คำปรึกษา อย่างสม่ำเสมอเสมอและคอยให้กำลังใจตลอดจนงานวิจัยฉบับนี้เสร็จลุล่วงไปด้วยดีด้วยความเคารพและขอบพระคุณอย่างสูง

ปรัชญากุล มณีฉาย

ชื่อเรื่องวิทยานิพนธ์ การพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะโดยใช้ IoT และ GIS

ผู้วิจัย ปรัชญากุล มณีฉาย

ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.สิทธิชัย ชูสำโรง

ประเภทสารนิพนธ์ วิทยานิพนธ์ วท.บ. สาขาภูมิศาสตร์, มหาวิทยาลัยนเรศวร, 2568

คำสำคัญ ระบบติดตาม, Geo-IoT, WebGIS

บทคัดย่อ

การวิจัยนี้มีวัตถุประสงค์เพื่อ ออกแบบและพัฒนาระบบติดตามรถเก็บขยะแบบเรียลไทม์ โดยประยุกต์ใช้เทคโนโลยี Geo-IoT ซึ่งผสานการทำงานของระบบระบุตำแหน่งด้วยดาวเทียม (GPS) ร่วมกับการสื่อสารผ่านอินเทอร์เน็ตและการแสดงผลบนแผนที่ออนไลน์ผ่านระบบ WebGIS

ระบบดังกล่าวช่วยให้เจ้าหน้าที่เทศบาลสามารถ ติดตามและตรวจสอบตำแหน่ง, เส้นทางการเดินรถ, ความเร็ว, รวมถึงทราบ จุดที่มีปริมาณขยะมากหรือน้อยในแต่ละเส้นทาง โดยอาศัยการคำนวณระยะเวลาการจอดเก็บขยะแต่ละจุด เพื่อใช้ในการวางแผนและจัดการเส้นทางการเก็บขยะในแต่ละรอบวันได้อย่างมีประสิทธิภาพ

ระบบสามารถแสดงข้อมูลเวลาการปฏิบัติงานของรถเก็บขยะได้อย่างต่อเนื่อง พร้อมทั้งมี ระบบแจ้งเตือนเมื่อรถเก็บขยะขับเร็วเกินกำหนด ซึ่งอาจส่งผลให้เกิดการตกหล่นของขยะ ระบบที่พัฒนาขึ้นช่วย เพิ่มประสิทธิภาพในการบริหารจัดการปริมาณขยะในแต่ละพื้นที่ , สนับสนุนการวางแผนเส้นทางการจัดเก็บให้เหมาะสม , ลดภาระการทำงานของเจ้าหน้าที่ , และ เพิ่มความโปร่งใสในการให้บริการของเทศบาล โดยสามารถตรวจสอบข้อมูลย้อนหลังได้อย่างชัดเจน อีกทั้งยังเป็นข้อมูลพื้นฐานสำคัญสำหรับการวางแผนพัฒนาระบบจัดการขยะในอนาคต เพื่อพัฒนาและยกระดับคุณภาพการให้บริการประชาชนให้มีความทันสมัยและมีประสิทธิภาพมากยิ่งขึ้น

Title	Development of an GPS Tracking System for Efficient Monitoring and Route Analysis of Garbage Collection Vehicles using IoT and GIS-Base
Author	Pruchayakul Maneechai
Advisor	Associate Professor Dr. Sittichai Choosumrong
Academic Paper	Thesis B.S.in Geography, Naresuan University 2025
Keywords	Tracking system, Geo-IoT, WebGIS

ABSTRACT

This research aims to design and develop a real-time waste collection vehicle tracking system by applying Geo-IoT technology, which integrates the Global Positioning System (GPS) with internet communication and visualization through an online mapping platform using WebGIS.The developed system enables municipal officers to monitor and verify the location, route, and speed of garbage trucks, as well as identify areas with high or low waste volumes along each route by calculating the duration of time spent stopping at each collection point. This information can then be used to plan and manage waste collection routes more efficiently on a daily basis.The system can continuously display the operating time of garbage trucks and includes an alert function when a truck exceeds the speed limit , which may cause waste spillage, thereby helping to address the problem of dropped waste. The developed system enhances the efficiency of waste management in each area, supports route optimization, reduces the workload of municipal officers, and increases transparency in municipal services. Moreover, it allows for clear historical data review and provides fundamental data for future waste management system planning. Ultimately, the system contributes to improving and modernizing the quality and efficiency of public services.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

บทที่ 1

บทนำ

1.1. ความเป็นมาและความสำคัญของปัญหา

การจัดเก็บขยะในเขตเทศบาลยังขาดระบบติดตามการทำงานของรถเก็บขยะในแบบเรียลไทม์ ส่งผลให้เจ้าหน้าที่ไม่สามารถตรวจสอบตำแหน่งหรือเส้นทางการเดินรถได้อย่างแม่นยำ อาจเกิดความล่าช้าในการเก็บขยะ ไม่ครอบคลุมทุกพื้นที่ และส่งผลให้ประชาชนร้องเรียนเรื่องขยะตกค้างหรือการขับรถเร็วเกินกำหนดทำให้ขยะตกหล่นในพื้นที่ ซึ่งล้วนเป็นภาระของเจ้าหน้าที่เทศบาลในการติดตามประสานงานและแก้ปัญหา

ปัจจุบันเทศบาลตำบลท่าทองมีการจัดเก็บขยะโดยใช้รถเก็บขยะจำนวนจำกัด ซึ่งจะวิ่งตามเส้นทางที่กำหนดไว้ล่วงหน้า ยังไม่มีระบบติดตามแบบเรียลไทม์หรือระบบรายงานสถานะของรถขยะ ทำให้เกิดปัญหาในการตรวจสอบเส้นทาง ความล่าช้า หรือการไม่สามารถเก็บขยะตามตารางเวลาที่กำหนด ส่งผลต่อความพึงพอใจของประชาชนในพื้นที่ รวมถึงเพิ่มภาระงานแก่เจ้าหน้าที่เทศบาลในการประสานงานและจัดการร้องเรียน

หลักการสำคัญการนำระบบติดตามยานพาหนะด้วยเทคโนโลยี **Geo-IoT** โดยใช้ GPS (Global Positioning System) เข้ามาใช้ร่วมกับการสื่อสารผ่านอินเทอร์เน็ตและการแสดงผลบนแผนที่แบบออนไลน์ผ่านระบบ WebGIS เพื่อให้เจ้าหน้าที่เทศบาลสามารถติดตามและตรวจสอบตำแหน่งของรถเก็บขยะได้แบบเรียลไทม์จะช่วยเพิ่มประสิทธิภาพในการจัดการขยะ และการวางแผนเส้นทางในการจัดการปริมาณขยะ และสามารถตรวจสอบข้อมูลย้อนหลังได้แบบโปร่งใส อีกทั้งยังเป็นข้อมูลพื้นฐานสำคัญในการวางแผนพัฒนาระบบจัดการขยะในอนาคต

ดังนั้น วิจัยนี้จึงมุ่งเน้นการพัฒนาและประยุกต์ใช้เทคโนโลยีระบบติดตามยานพาหนะในงานบริการสาธารณะ โดยเฉพาะการติดตามและวิเคราะห์ประสิทธิภาพของรถเก็บขยะในเขตตำบลท่าทอง อำเภอเมืองจังหวัดพิษณุโลก เพื่อเป็นแนวทางในการยกระดับคุณภาพการให้บริการประชาชนอย่างเป็นระบบและทันสมัย

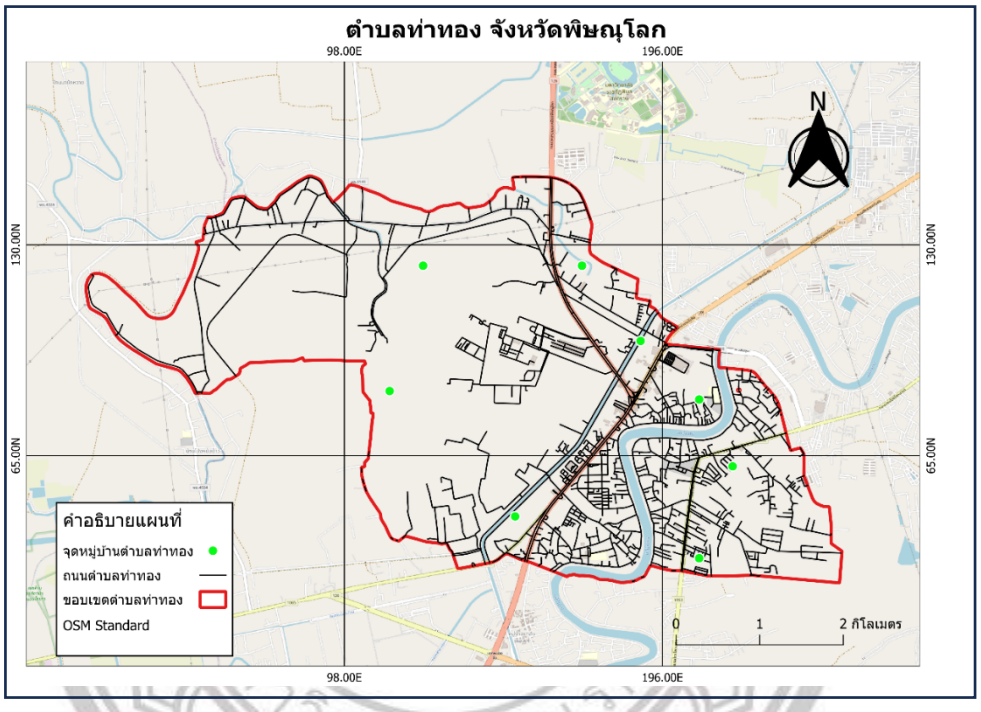
1.2. วัตถุประสงค์

1. เพื่อออกแบบและพัฒนาระบบติดตามรถเก็บขยะด้วยเทคโนโลยี GPS Tracking เพื่อช่วยแก้ไขปัญหาขยะตกหล่นในพื้นที่รับผิดชอบของเทศบาล

2. พัฒนาและออกระบบติดตามเส้นทางการเก็บขยะให้สามารถบันทึกตำแหน่งพิกัดของรถเก็บขยะได้แบบเรียลไทม์ (Real-time) และคำนวณระยะเวลาการจอดเก็บขยะแต่ละจุด เพื่อใช้ในการวิเคราะห์ปริมาณขยะในแต่ละพื้นที่

1.3. ขอบเขตพื้นที่การศึกษา

การศึกษางานวิจัย ได้ทำการพัฒนาระบบ GPS tracking เพื่อทำการพัฒนาระบบติดตามรถเก็บขยะในระดับตำบลให้มีประสิทธิภาพและ การใช้งานง่ายของผู้ที่อยู่ในพื้นที่สามารถดูรอบวันในการเก็บขยะแต่ละช่วงเวลาเพื่อดูตำแหน่งรถเก็บขยะแบบเรียลไทม์มีผลทำให้สามารถรับรู้การมาของรถเก็บขยะได้



ภาพที่ 1 แผนที่ตำบลท่าทอง

1.4. ความสำคัญของงานวิจัย

ระบบติดตามรถเก็บขยะที่พัฒนาขึ้นสามารถช่วยให้เทศบาล บริหารจัดการปริมาณขยะในแต่ละพื้นที่ และวางแผนเส้นทางการจัดเก็บขยะได้อย่างมีประสิทธิภาพ หรือไม่

1.5. นิยามศัพท์เฉพาะ

เทคโนโลยี Geo-IoT

Geo-IoT หมายถึง Geospatial Internet of Things หรือ Internet of Things ที่เกี่ยวข้องกับพื้นที่ ซึ่งเป็นแนวคิดที่รวมระบบ Internet of Things (IoT) กับเทคโนโลยีเชิงพื้นที่ (Geospatial Technology) โดยอุปกรณ์ IoT จะเก็บข้อมูลจากเซนเซอร์ เช่น พิกัด, ความเร็ว, ทิศทาง, จุดที่หยุด ข้อมูลเหล่านี้จะถูกส่งผ่านอินเทอร์เน็ตมายังระบบสารสนเทศภูมิศาสตร์ (GIS) เพื่อวิเคราะห์เชิงพื้นที่และแสดงผลแบบเรียลไทม์ ทำให้สามารถติดตามตำแหน่ง การเคลื่อนที่ และสถานะการทำงานของอุปกรณ์หรือยานพาหนะ รวมถึงใช้วางแผนการปฏิบัติงานหรือเส้นทางได้อย่างมีประสิทธิภาพ

GPS (Global Positioning System)

ระบบ GPS คือระบบดาวเทียมนำทางที่สามารถระบุพิกัดตำแหน่งบนพื้นโลกแบบเรียลไทม์ ประกอบด้วยดาวเทียมอย่างน้อย 24 ดวง ทำให้สามารถคำนวณพิกัด ละติจูด (Latitude), ลองจิจูด (Longitude), ความสูง (Altitude) รวมถึง ความเร็วและทิศทางการเคลื่อนที่ ข้อมูลจากระบบ GPS เป็นพื้นฐานสำคัญสำหรับการติดตามรถเก็บขยะ การวิเคราะห์เส้นทางการเดินรถ และการประเมินเวลาที่ใช้ในการเก็บขยะ พร้อมทั้งให้ข้อมูลเชิงพื้นที่อื่น ๆ ที่ช่วยในการวางแผนเส้นทางและเพิ่มประสิทธิภาพการจัดการขยะ ระบบ GPS มีบทบาทสำคัญใน การพัฒนาระบบติดตามรถเก็บขยะโดยใช้ IoT และ GIS เพื่อให้เจ้าหน้าที่เทศบาลสามารถติดตามตำแหน่งรถแบบเรียลไทม์ ตรวจสอบเส้นทาง และปรับปรุงการบริหารจัดการการเก็บขยะได้อย่างมีประสิทธิภาพ

WebGIS

ระบบ WebGIS คือระบบสารสนเทศภูมิศาสตร์ที่สามารถเข้าถึงและแสดงผลข้อมูลเชิงพื้นที่ผ่านเว็บเบราว์เซอร์ ทำให้ผู้ใช้งานสามารถตรวจสอบ ติดตาม วิเคราะห์ และบริหารจัดการข้อมูลแผนที่โดยไม่จำเป็นต้องติดตั้งซอฟต์แวร์ GIS แบบเดิมบนเครื่องคอมพิวเตอร์ โดยประกอบด้วยฐานข้อมูลเชิงพื้นที่สำหรับจัดเก็บตำแหน่ง รถเก็บขยะ จุดรถวิ่ง เส้นทาง และจุดรวบรวมขยะ เซิร์ฟเวอร์ WebGIS ที่ทำหน้าที่ดึงข้อมูลจากฐานข้อมูล และให้บริการข้อมูลแผนที่แก่ผู้ใช้งาน ส่วนแสดงผลผ่านเว็บเบราว์เซอร์ ใช้เทคโนโลยีเช่น Leaflet, OpenLayers ในการแสดงแผนที่และวิเคราะห์ข้อมูล พร้อมฟังก์ชันวิเคราะห์เชิงพื้นที่ เช่น เส้นทางรถวิ่งย้อนหลัง การติดตามตำแหน่งเรียลไทม์ผ่าน WebGIS และการวิเคราะห์ความหนาแน่นของจุดเก็บขยะ ซึ่งสามารถนำไปประยุกต์ใช้ในระบบติดตามรถเก็บขยะเพื่อแสดงตำแหน่งรถแบบเรียลไทม์และย้อนหลัง วิเคราะห์เวลาจอดเก็บขยะและปริมาณขยะเบื้องต้น ประเมินเส้นทางที่เหมาะสม แจ้งเตือนเหตุการณ์ เช่น รถขับเร็วเกินกำหนด หรือจุดที่มีขยะสะสมสูง และช่วยเพิ่มความโปร่งใสและประสิทธิภาพในการบริหารจัดการบริการขยะของเทศบาล

PostGIS

PostGIS คือส่วนขยายของระบบจัดการฐานข้อมูล PostgreSQL ที่เพิ่มความสามารถในการจัดเก็บ จัดการ และวิเคราะห์ข้อมูลเชิงพื้นที่ ทำให้สามารถบันทึกตำแหน่ง GPS ของรถเก็บขยะ จุด เส้นทาง และจุดเก็บขยะในรูปแบบเชิงพื้นที่ได้อย่างแม่นยำ พร้อมฟังก์ชันในการวิเคราะห์เส้นทาง จากจุดที่รับมาจาก GPS และตรวจสอบความสัมพันธ์เชิงพื้นที่ระหว่างจุดจอดกับปริมาณขยะ ข้อมูลเหล่านี้สามารถนำไปใช้ร่วมกับระบบ WebGIS เพื่อแสดงตำแหน่งรถเก็บขยะแบบเรียลไทม์ วิเคราะห์เวลาจอดเก็บขยะ ประเมินเส้นทางที่เหมาะสม และสนับสนุนการวางแผนจัดการปริมาณขยะในแต่ละพื้นที่อย่างมีประสิทธิภาพ เพิ่มความโปร่งใสและประสิทธิผลในการบริหารจัดการบริการขยะของเทศบาล

ระบบแจ้งเตือนผ่าน Telegram

ระบบแจ้งเตือนผ่าน Telegram คือระบบสื่อสารอัตโนมัติที่เชื่อมต่อกับแอปพลิเคชัน Telegram เพื่อส่งข้อความแจ้งเตือนให้แก่เจ้าหน้าที่เทศบาลทันทีเมื่อเกิดเหตุการณ์สำคัญ เช่น รถเก็บขยะขับเร็วเกินกำหนดตามกฎหมายที่ระบุไว้ ระบบนี้ช่วยให้เจ้าหน้าที่สามารถทราบถึงจุดที่รถเก็บขยะขับเร็วเกินกำหนด ตอบสนองต่อปัญหา รวมทั้งสนับสนุนการวางแผนเส้นทางและบริหารจัดการรถเก็บขยะอย่างมีประสิทธิภาพ โดยข้อมูลแจ้งเตือนสามารถนำไปวิเคราะห์ถึงจุดที่โดนร้องเรียนเรื่องขยะตกหล่นจากการขับรถเร็วเกินกำหนด

ระบบติดตามแบบเรียลไทม์ (Real-Time Tracking System)

ระบบติดตามแบบเรียลไทม์ (Real-Time Tracking System) คือระบบที่สามารถติดตามและแสดงตำแหน่ง ความเร็ว และสถานะการทำงานของรถเก็บขยะได้ทันทีที่เกิดขึ้นบนแผนที่ออนไลน์ โดยอาศัยการเชื่อมต่อข้อมูลจากเซนเซอร์ GPS ผ่านเทคโนโลยี IoT และจัดเก็บในฐานข้อมูลเชิงพื้นที่เพื่อการประมวลผล ระบบนี้ช่วยให้เจ้าหน้าที่เทศบาลสามารถตรวจสอบเส้นทางการวิ่งของรถเก็บขยะ วิเคราะห์เวลาจอดเก็บขยะ ประเมินจุดที่มีปริมาณขยะสูงหรือต่ำ และวางแผนเส้นทางการจัดเก็บขยะได้อย่างมีประสิทธิภาพ ลดปัญหาขยะตกหล่นจากการขับรถเร็วเกินกำหนด และสนับสนุนการบริหารจัดการปริมาณขยะในแต่ละพื้นที่อย่างโปร่งใส

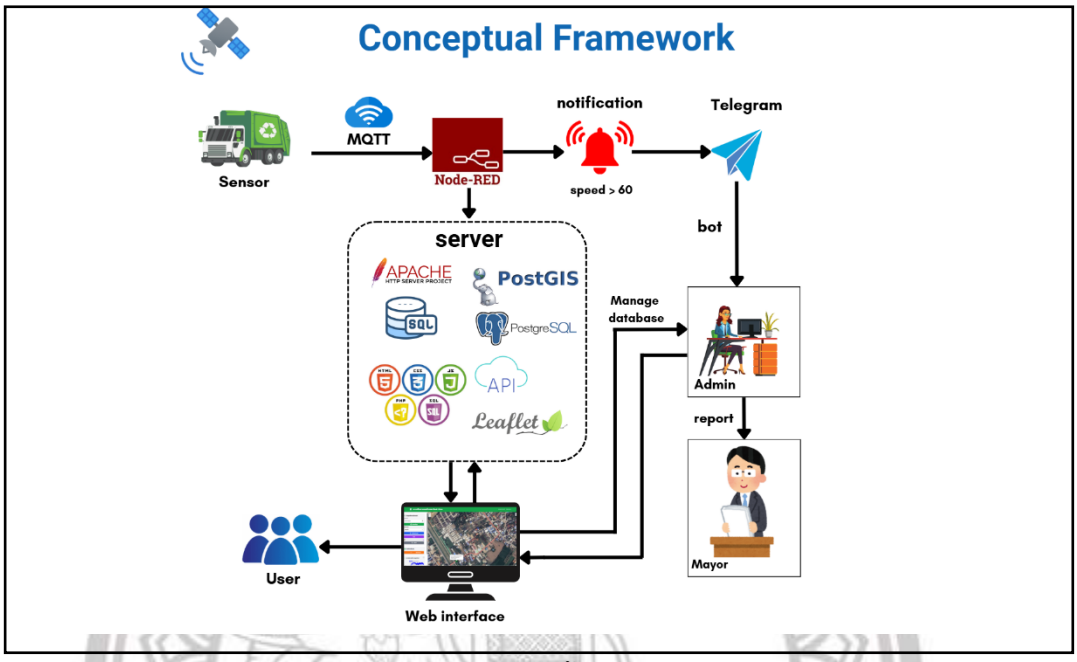
ระบบสารสนเทศภูมิศาสตร์ (Geographic Information System: GIS)

คือระบบที่ใช้ในการเก็บรวบรวม จัดเก็บ วิเคราะห์ ประมวลผล และแสดงผลข้อมูลเชิงพื้นที่และข้อมูลภูมิศาสตร์ เช่น ตำแหน่งถนน จุดเก็บขยะ เส้นทางการวิ่งของรถเก็บขยะ เขตชุมชน ระบบ GIS สามารถผสานข้อมูลจากเซนเซอร์ GPS ของรถเก็บขยะ ข้อมูล IoT และฐานข้อมูลเชิงพื้นที่ เช่น PostGIS เพื่อสร้างแผนที่และวิเคราะห์เชิงพื้นที่ได้อย่างแม่นยำ ทำให้สามารถติดตามเส้นทางรถเก็บขยะแบบเรียลไทม์ วิเคราะห์เวลาจอด ประเมินปริมาณขยะในแต่ละจุด วางแผนเส้นทางการจัดเก็บอย่างมีประสิทธิภาพ และตรวจสอบความสัมพันธ์ระหว่างปริมาณขยะกับพื้นที่ใช้สอย นอกจากนี้ GIS ยังช่วยในการสร้าง Heatmap ของจุดที่มีการเก็บขยะมากหรือน้อย วางแผนข้อมูลและการบริหารจัดการขยะของเทศบาลให้โปร่งใส ทันสมัย และตอบสนองต่อปัญหา

1.6. สมมติฐานของงานวิจัย

เวลาที่รถเก็บขยะจอด ณ จุดใด ๆ มีความสัมพันธ์กับปริมาณขยะ โดยรถจอดนานหมายถึงขยะมาก และรถจอดสั้นหมายถึงขยะน้อย การติดตามเวลาจอดและความเร็วของรถเก็บขยะสามารถช่วยให้เทศบาลวางแผนเส้นทางการเก็บขยะและจัดรถเก็บขยะตามน้ำหนักได้อย่างมีประสิทธิภาพ รวมถึงช่วยลดปัญหาขยะตกหล่นจากการขับรถเร็วเกินกำหนด

1.7. กรอบแนวคิดงานวิจัย



ภาพที่ 2 กรอบแนวคิดของขั้นตอนการดำเนินงาน

1.8. ขั้นตอนการดำเนินงาน

1.8.1. การออกแบบระบบ

ดำเนินการออกแบบระบบ การออกแบบชุดอุปกรณ์ GPS Tracking การออกแบบฐานข้อมูลเชิงพื้นที่สำหรับจัดเก็บข้อมูลตำแหน่งและความเร็ว การออกแบบการส่งข้อมูลด้วย Node-RED และการออกแบบ WebGIS

1.8.2. การพัฒนาฮาร์ดแวร์และเซนเซอร์

ประกอบและพัฒนาชุดอุปกรณ์ GPS Tracking ที่ทำหน้าทีรับค่าพิกัด ละติดจูด-ลองจิจูดและความเร็วรวมถึงความแม่นยำและทิศทาง ทำการส่งข้อมูลไปยังเซิร์ฟเวอร์โดยทดสอบการทำงานของเซนเซอร์สามารถอ่านค่าแม่นยำหรือเสถียรมากน้อยเท่าใดรวมถึงการส่งและรับข้อมูล

1.8.3. การพัฒนาเซิร์ฟเวอร์และฐานข้อมูล

พัฒนาระบบเซิร์ฟเวอร์โดยใช้ Node-RED เพื่อทำหน้าที่รับข้อมูล,ส่งข้อมูล และกรองข้อมูลที่ส่งมาจากเซนเซอร์ ต่อมานำข้อมูลที่เข้ามาไปจัดเก็บลงยังฐานข้อมูล PostgreSQL ที่ประยุกต์ใช้ PostGIS เพื่อรองรับข้อมูลเชิงพื้นที่ geometry รวมถึงการสร้าง API เพื่อส่งข้อมูลไปแสดงผลบนหน้าเว็บไซต์แบบเรียลไทม์

1.8.4. พัฒนา WebGIS และการแจ้งเตือน

พัฒนาเว็บไซต์แสดงผล WebGIS ใช้ไลบรารี Leaflet เพื่อแสดงตำแหน่งปัจจุบันของรถเก็บขยะบนแผนที่ และพัฒนาฟังก์ชันเพิ่มเติม คือ การเล่นเส้นทางย้อนหลัง แสดงสถานะของรถเก็บขยะ เช่น จอดเก็บขยะ,รถวิ่งปกติ แสดงแผนที่ Heatmap เพื่อดูจุดที่มีการเก็บขยะมากหรือน้อย นอกจากนี้ยังมีการพัฒนาระบบแจ้งเตือนผ่าน Telegram เมื่อตรวจพบการขับรถเร็วเกินกำหนด

1.8.5. ทดสอบระบบ GPS Tracking

ติดตั้งชุดอุปกรณ์ GPS Tracking บนรถเก็บขยะในพื้นที่ศึกษา และดำเนินการทดสอบในการวิ่งเก็บขยะจริงเพื่อตรวจสอบความถูกต้องของข้อมูล ตำแหน่ง ความเร็ว จุดเก็บขยะ การทำงานของระบบและประสิทธิภาพ

1.8.6. ประเมินผลและสรุปผล

รวบรวมข้อมูลจากการทดสอบการติดตั้ง GPS Tracking และทำการวิเคราะห์ผลที่ได้เพื่อตอบคำถามงานวิจัยและยืนยันสมมติฐาน

1.9. ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากการพัฒนาระบบ GPS Tracking โดยใช้เทคโนโลยี IoT และ GIS ได้แก่ การเพิ่มประสิทธิภาพในการติดตามตำแหน่ง ความเร็ว และเวลาการจอดของรถเก็บขยะแบบเรียลไทม์ ทำให้เจ้าหน้าที่เทศบาลสามารถวางแผนเส้นทางการเก็บขยะได้อย่างเหมาะสม ลดปัญหาการเก็บขยะล่าช้าและไม่ครอบคลุมพื้นที่ สนับสนุนการจัดการปริมาณขยะในแต่ละพื้นที่อย่างเหมาะสมผ่านการวิเคราะห์จุดที่มีขยะสูงหรือต่ำ ลดปัญหาการขับรถเร็วเกินกำหนดและขยะตกหล่นด้วยระบบแจ้งเตือนผ่าน Telegram เพิ่มความโปร่งใสและความรับผิดชอบในการให้บริการด้วยข้อมูลย้อนหลังและ Heatmap ของเส้นทางการเก็บขยะ รวมถึงเป็นข้อมูลพื้นฐานสำคัญสำหรับการวางแผนและพัฒนาระบบจัดการขยะในอนาคต พร้อมทั้งสนับสนุนการพัฒนางานบริการสาธารณะให้ทันสมัย มีประสิทธิภาพ และสามารถนำไปประยุกต์ใช้กับงานบริการสาธารณะอื่น ๆ ได้อย่างต่อเนื่อง

บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

การพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะโดยใช้ IoT และ GIS ผู้วิจัยได้ศึกษาแนวคิดในการพัฒนาระบบเพื่อแก้ไขปัญหาขยะในแต่ละเส้นทางเพื่อวางแผนจัดการ มีรายละเอียดต่อไปนี้

2.1. เครื่องมือที่ช่วยในการพัฒนาระบบ

2.1.1. อุปกรณ์ที่ใช้ในการพัฒนา Hardware: NodeMCU ESP8266, GPS Shield, Arduino UNO, ANN-MB1-00

2.1.2. ภาษาที่ใช้สำหรับบอร์ดไมโครคอนโทรลเลอร์: C/C++

2.1.3. โปรแกรมที่ใช้เขียนโค้ดลงเซนเซอร์: Arduino IDE

2.1.4. โปรแกรมในการส่งข้อมูล: MQTT Broker, Node-RED

2.1.5. โปรแกรมจัดการฐานข้อมูล: PostgreSQL และ PostGIS

2.1.6. ภาษาที่ใช้ในการเขียนโปรแกรม: HTML, CSS, JavaScript, PHP, SQL

2.1.7. ไลบรารีที่ใช้ในการแสดงผลแผนที่: Leaflet.js

2.1.8. โปรแกรมที่ใช้สร้างหน้าเว็บ: Visual Studio Code

2.1.9. โปรแกรมแจ้งเตือน: Telegram

2.1.10. ซอฟต์แวร์อื่น ๆ: MS4W,

2.1.11. Backend และ Frontend ของระบบ

2.2. เอกสารงานวิจัยที่เกี่ยวข้อง

2.1. เครื่องมือที่ช่วยในการพัฒนาระบบ

2.1.1. อุปกรณ์ที่ใช้ในการพัฒนา Hardware

NodeMCU ESP8266 เป็นไมโครคอนโทรลเลอร์ที่มีโมดูล Wi-Fi ในตัว ซึ่งได้รับความนิยมอย่างแพร่หลายในงานพัฒนา ระบบ Internet of Things (IoT) เนื่องจากมีขนาดเล็ก ราคาประหยัด และสามารถเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตได้โดยตรง โดยไม่ต้องใช้โมดูลเสริมเพิ่มเติม ตัวบอร์ดสามารถเขียนโปรแกรมได้ด้วยภาษา C/C++ ผ่าน Arduino IDE

GPS Shield (GlobalSat EB-5365RE SIRF IV), Arduino UNO R3 เป็นโมดูลรับสัญญาณจากระบบดาวเทียมนำทาง (Global Positioning System: GPS) ซึ่งสามารถระบุตำแหน่งพิกัดบนพื้นโลกได้อย่างแม่นยำ ทั้งค่าละติจูด (Latitude), ลองจิจูด (Longitude), ความเร็ว (Speed) และเวลา (Time) โดยโมดูลนี้จะทำหน้าที่รับสัญญาณจากดาวเทียมอย่างน้อย 3–4 ดวงขึ้นไป เพื่อคำนวณตำแหน่งปัจจุบันของรถเก็บขยะในแต่ละช่วงเวลาในการพัฒนาระบบนี้

เสารับสัญญาณ ANN-MB1-00 เป็นโมดูลรับสัญญาณดาวเทียมเช่นเดียวกับ GPS Shield แต่มีความสามารถในการปรับปรุงความแม่นยำของพิกัดตำแหน่งและเสถียรภาพของสัญญาณมากขึ้น โมดูลนี้รองรับการติดตามหลายระบบดาวเทียม เช่น GPS, GLONASS, BeiDou หรือ Galileo

2.1.2. . ภาษาที่ใช้สำหรับบอร์ดไมโครคอนโทรลเลอร์

ภาษา C/C++

ภาษาซี (C) และซีพลัสพลัส (C++) เป็นภาษาการเขียนโปรแกรมเชิงโครงสร้างและเชิงวัตถุที่ใช้กันอย่างแพร่หลายในงานพัฒนาไมโครคอนโทรลเลอร์ โดยเฉพาะในแพลตฟอร์ม Arduino IDE (Integrated Development Environment) ซึ่งออกแบบมาเพื่อเขียน แก้ไข และอัปโหลดโปรแกรมไปยังบอร์ดไมโครคอนโทรลเลอร์ เช่น Node MCU ESP8266 หรือ Arduino UNO ภาษาซี/ซีพลัสพลัสถูกใช้ในการเขียนโค้ดสำหรับควบคุมการทำงานของอุปกรณ์ IoT เช่น การรับค่าพิกัดจากโมดูล GPS การคำนวณตำแหน่ง ความเร็ว และเวลาจากข้อมูลดาวเทียม โดยอาศัยไลบรารีสำเร็จรูป เช่น TinyGPS++, PubSubClient, และ WiFiClient เพื่ออำนวยความสะดวกในการพัฒนาและลดความซับซ้อนของโค้ด ภาษาซี/ซีพลัสพลัสสำหรับ Arduino IDE เป็นส่วนสำคัญในการควบคุมและสื่อสารในระบบติดตามรถเก็บขยะแบบเรียลไทม์ เนื่องจากช่วยให้สามารถเขียนโปรแกรมควบคุมอุปกรณ์ฮาร์ดแวร์ได้อย่างยืดหยุ่น มีประสิทธิภาพสูง และทำงานร่วมกับเทคโนโลยี IoT ได้อย่างสมบูรณ์แบบ

2.1.3. โปรแกรมที่ใช้เขียนโค้ดลงเซนเซอร์: Arduino IDE

โปรแกรม Arduino IDE (Integrated Development Environment) สำหรับการเขียนโค้ดและอัปโหลดโปรแกรมลงในไมโครคอนโทรลเลอร์หรือบอร์ด Arduino เช่น Arduino UNO, ESP8266, และอุปกรณ์อื่น ๆ ที่รองรับ โดยสามารถเขียนโปรแกรมด้วยภาษาที่คล้ายกับ C/C++ ได้ง่าย ๆ มีไลบรารีสำเร็จรูปสำหรับจัดการเซนเซอร์ อุปกรณ์เชื่อมต่อ และการสื่อสารข้อมูล เช่น GPS, Wi-Fi, และ MQTT รวมทั้งสามารถคอมไพล์และส่งโค้ดไปยังบอร์ดเพื่อให้ทำงานตามฟังก์ชันที่กำหนดได้ทันที ทำให้ Arduino IDE เป็นเครื่องมือหลักสำหรับพัฒนาระบบ IoT

นอกจากนี้ Arduino IDE ยังมี คุณสมบัติและฟังก์ชันเสริมที่ช่วยอำนวยความสะดวกในการพัฒนาระบบ IoT และ Embedded System อย่างครบวงจร เช่น การจัดการ ไลบรารี (Library Manager) ซึ่งช่วยให้ผู้พัฒนาสามารถติดตั้งและเรียกใช้งานไลบรารีที่สนับสนุนอุปกรณ์และเซนเซอร์ประเภทต่าง ๆ ได้อย่างรวดเร็ว โดยไม่ต้องเขียนโค้ดจากศูนย์ เช่น ไลบรารีสำหรับ เซนเซอร์วัดอุณหภูมิและความชื้น (DHT, BMP280), เซนเซอร์ความชื้นดิน, โมดูล GPS, Wi-Fi, และโปรโตคอล MQTT เป็นต้น

Arduino IDE ยังมี เครื่องมือ Serial Monitor ซึ่งทำหน้าที่แสดงข้อมูลที่บอร์ดส่งออกมาแบบเรียลไทม์ ช่วยให้นักพัฒนาสามารถ ตรวจสอบค่าเซนเซอร์, ค่าการสื่อสาร, หรือสถานะของอุปกรณ์ ได้ทันที ทำให้สามารถ Debug และปรับปรุงโค้ดได้สะดวก นอกจากนี้ยังสามารถใช้ Serial Plotter เพื่อแสดงค่าข้อมูลเป็นกราฟ ทำให้เห็นแนวโน้มของข้อมูลหรือสัญญาณเซนเซอร์แบบเรียลไทม์

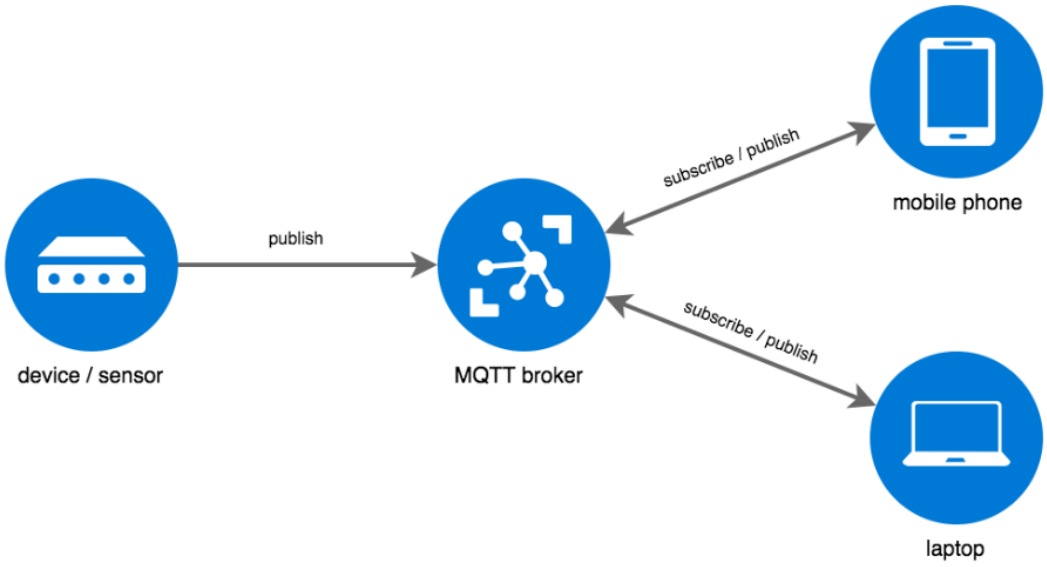
อีกหนึ่งความสามารถสำคัญคือ Arduino IDE สามารถ ทำงานร่วมกับบอร์ดหลายประเภทและหลายแพลตฟอร์ม ผ่าน Board Manager เช่น ESP8266, ESP32, Arduino UNO, Mega, Nano และบอร์ดอื่น ๆ ที่รองรับ ทำให้ผู้พัฒนาสามารถพัฒนาโปรเจกต์ที่ซับซ้อนและหลากหลายได้โดยใช้ IDE เดียว

ด้วยคุณสมบัติเหล่านี้ Arduino IDE จึงเหมาะสมอย่างยิ่งสำหรับการพัฒนาระบบ IoT (Internet of Things) ที่ต้องการให้ไมโครคอนโทรลเลอร์สามารถ อ่านค่าจากเซนเซอร์, ประมวลผลข้อมูล, และส่งข้อมูลไปยังเซิร์ฟเวอร์หรือ Cloud ได้อย่างต่อเนื่องและมีประสิทธิภาพ โดยสามารถนำข้อมูลที่ได้ไปใช้งานต่อ เช่น การแสดงผลผ่านหน้าเว็บ, การจัดเก็บลงฐานข้อมูล PostgreSQL, หรือการนำไปวิเคราะห์เชิงสถิติและ Machine Learning

2.1.4. โปรแกรมในการส่งข้อมูล: MQTT Broker, Node-RED

MQTT คือโปรโตคอลสื่อสารแบบเบา (Lightweight Protocol) ที่ออกแบบมาสำหรับอุปกรณ์ที่มีข้อจำกัดด้านพลังงานและแบนด์วิดท์ต่ำ โปรโตคอลนี้ใช้โมเดลการสื่อสารแบบ Publish/Subscribe โดยมี MQTT Broker ทำหน้าที่เป็นเซิร์ฟเวอร์กลางในการจัดการข้อความ. อุปกรณ์ IoT เช่น ESP8266 ที่ติดตั้งบนรถเก็บขยะ จะทำหน้าที่เป็น Publisher โดยส่งข้อมูลตำแหน่งพิกัดจาก GPS, ความเร็ว, และข้อมูลสถานะการทำงานไปยัง Broker ในรูปแบบของข้อความ (Message) ผ่านหัวข้อ (Topic) ที่กำหนดไว้ล่วงหน้า. การใช้ MQTT ช่วยให้การรับ–ส่งข้อมูลระหว่างอุปกรณ์ปลายทางมีความเสถียรและลดภาระการสื่อสารของเครือข่าย. โปรโตคอลนี้จึงเหมาะสมอย่างยิ่งกับการประยุกต์ใช้ในระบบติดตามรถเก็บขยะที่ต้องการการอัปเดตข้อมูลแบบเรียลไทม์และมีการทำงานตลอดเวลา

การส่งข้อมูลระหว่างเซนเซอร์หรืออุปกรณ์ IoT ใช้ โปรโตคอล MQTT ที่ออกแบบมาสำหรับอุปกรณ์ที่มีข้อจำกัดด้านพลังงาน โดยมี MQTT Broker ทำหน้าที่เป็นเซิร์ฟเวอร์กลางสำหรับจัดการข้อความแบบ Publish/Subscribe ซึ่งช่วยให้การรับ–ส่งข้อมูลระหว่างอุปกรณ์ปลายทาง (Client) มีความเสถียรและลดภาระการสื่อสารของเครือข่าย อุปกรณ์ IoT เช่น NodeMCU ESP8266 จะส่งข้อมูลตำแหน่งพิกัดจาก GPS, ความเร็ว และข้อมูลสถานะการทำงานไปยัง Broker ในรูปแบบของข้อความ (Message) ผ่านหัวข้อ (Topic) ที่กำหนดไว้ล่วงหน้า



ภาพที่ 3 กระบวนการทำงาน MQTT

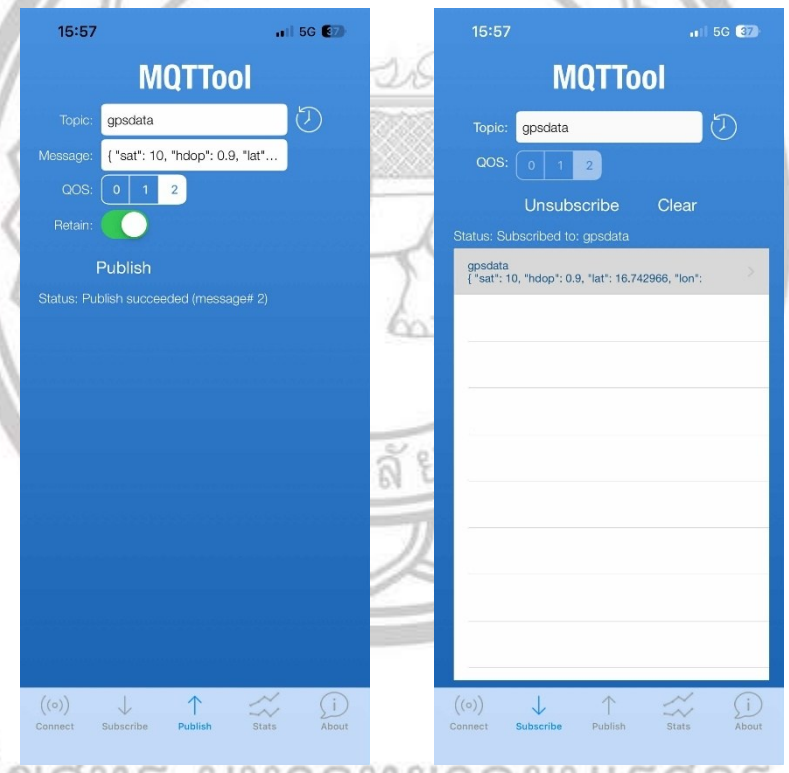
ที่มา https://leeluchowdaryravi.wordpress.com/2018/02/18/local-mqtt-broker-setup/

Publish อุปกรณ์หรือโปรแกรมที่ ส่งข้อมูล ขึ้นไปยังระบบ การทำงานร่วมกับ Node-RED ตัวอย่างเช่น บอร์ด ESP8266 ส่งค่าพิกัด GPS ไปยัง MQTT ที่เป็นตัวกลางแล้วส่งต่อไปยัง Node-RED

Subscribe โปรแกรมหรือระบบที่ รับข้อมูล ที่ถูกส่งมา โดยจะต้องทำการ Subscribe กับหัวข้อ Topic ที่ต้องการรับข้อมูล

Topic คือ ชื่อหัวข้อ ที่ใช้ในการจัดหมวดหมู่ข้อมูลระหว่างผู้ส่งและผู้รับ เพื่อให้แต่ละฝ่ายรู้ว่าข้อมูลที่ต้องการอยู่ในหัวข้อใด ตัวอย่างที่ใช้งาน ‘gpsdata’

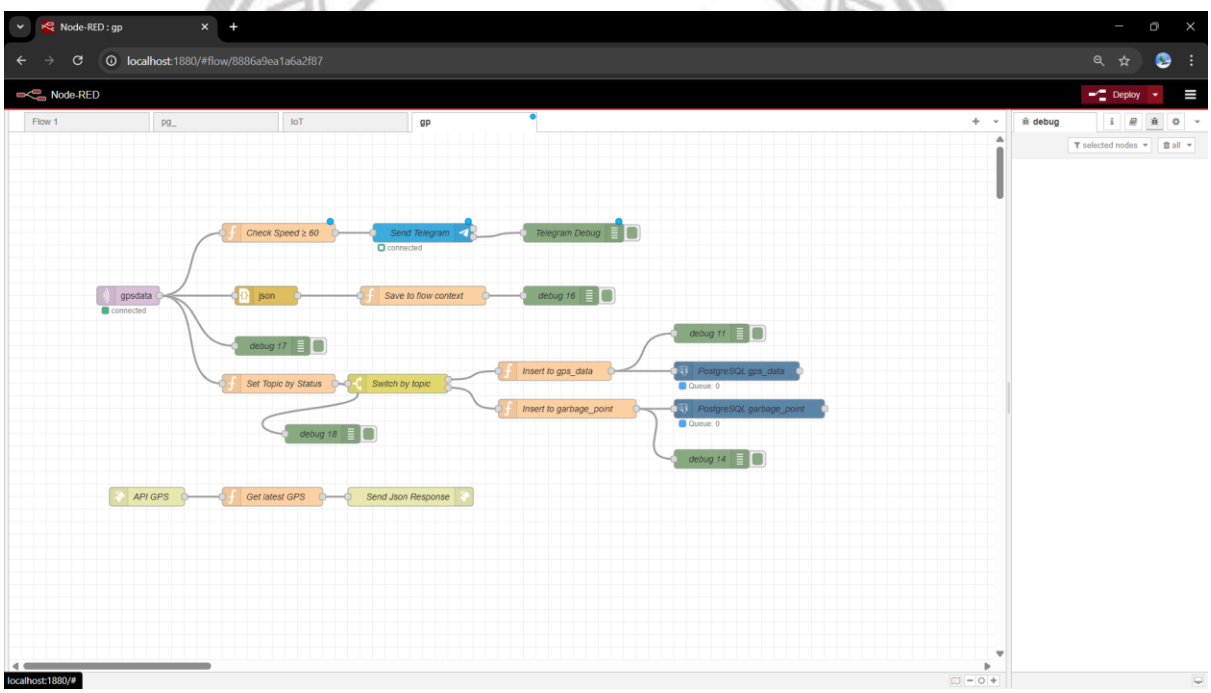
Broker คือตัวกลางที่จะรับข้อมูลทั้งหมดมาจาก publish ไม่ว่าจะ topic อะไรก็ตามแล้วทำการจัดส่งข้อมูลไปยัง Subscribe เพื่อรับค่าไปใช้งานต่อ



ภาพที่ 4 ตัวอย่างการทำงาน Publish/Subscribe

Node-RED เป็นเครื่องมือพัฒนาแบบ Flow-based Programming Tool ที่ถูกออกแบบมาเพื่อการเชื่อมต่อและประมวลผลข้อมูลจากอุปกรณ์ IoT และบริการบนเว็บ Node-RED จะทำหน้าที่เป็น Subscribe โดยเชื่อมต่อกับ MQTT Broker เพื่อดึงข้อมูลแบบเรียลไทม์จากรถเก็บขยะมาประมวลผล การประมวลผลนี้รวมถึงการกรอง (Filtering), การแปลงรูปแบบข้อมูลให้อยู่ในรูปแบบ JSON หรือรูปแบบอื่นที่เหมาะสมกับการจัดเก็บในฐานข้อมูล เช่น PostgreSQL/PostGIS นอกจากนี้ Node-RED ยังทำหน้าที่ส่งข้อมูลไปยังระบบ WebGIS เพื่อแสดงผลตำแหน่งและเส้นทางการเคลื่อนที่ของรถเก็บขยะบนแผนที่แบบเรียลไทม์ การทำงานส่งข้อมูลไปเก็บยังฐานข้อมูล, ส่งแจ้งเตือนไปยัง Telegram

สามารถใช้งานง่าย ระหว่างการลากเส้น ดัดแปลง แก้ไข เพิ่มเติมตามที่ต้องการ สามารถเปลี่ยนโปรโตคอลปลายทางได้ เช่น ต้นทาง MQTT ปลายทาง HTTP



ภาพที่ 5 ตัวอย่างการทำงาน Node-RED

เริ่มต้นที่ Topic ที่รับค่ามากจาก MQTT ที่ส่งมาจาก sensor ส่งต่อไปกรองข้อมูลที่ Set Topic Status จากนั้นแยกข้อมูลโดยมีเงื่อนไขที่ Switch by topic แล้วส่งแยกข้อมูลไปเก็บตามที่กำหนดไว้ ตัวอย่างในรูปคือการแยกไปเก็บข้อมูลที่ฐานข้อมูล gps_data คือข้อมูลรถวิ่งปกติและ garbage_point คือจุดเก็บขยะ

การทำงานร่วมกันของ MQTT Broker และ Node-RED ช่วยให้กระบวนการส่งข้อมูลจากอุปกรณ์ IoT ไปยังระบบฐานข้อมูลและเว็บแอปพลิเคชันมีความต่อเนื่อง รวดเร็ว และมีประสิทธิภาพมากยิ่งขึ้น อีกทั้งยังสามารถรองรับการขยายระบบในอนาคตได้ง่าย เหมาะสมกับการประยุกต์ใช้ในระบบติดตามรถเก็บขยะที่ต้องการการอัปเดตข้อมูลแบบเรียลไทม์และมีการทำงานตลอดเวลา

2.1.5. โปรแกรมจัดการฐานข้อมูล: PostgreSQL และ PostGIS

1. ประวัติและความเป็นมา

PostgreSQL เป็น ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ที่มีต้นกำเนิดจากโครงการ Ingres ของมหาวิทยาลัยแคลิฟอร์เนีย เบิร์กลีย์ ในช่วงปี 1980 และเริ่มพัฒนาภายใต้ชื่อ Postgres เพื่อรองรับฟีเจอร์ใหม่ เช่น การเก็บข้อมูลแบบ Object-Relational ต่อมาในปี 1996 ได้เปลี่ยนชื่อเป็น PostgreSQL เพื่อสะท้อนความสามารถในการรองรับ SQL มาตรฐาน ได้เต็มรูปแบบ

ความโดดเด่นของ PostgreSQL คือเป็น โอเพนซอร์ส ภายใต้สัญญาอนุญาตแบบ PostgreSQL License ที่อนุญาตให้ใช้งาน แก้ไข และเผยแพร่ได้อย่างเสรี ทำให้มีชุมชนพัฒนาขนาดใหญ่และมีการอัปเดตอย่างต่อเนื่อง

2. การรองรับระบบปฏิบัติการ

PostgreSQL สามารถติดตั้งและใช้งานได้บนหลายระบบปฏิบัติการ เช่น

- **Linux** (Ubuntu, CentOS, Debian, Red Hat)
- **Windows** (Windows Server และ Windows Desktop)
- **macOS**
- **Unix-based OS** อื่น ๆ เช่น FreeBSD, Solaris

ความสามารถข้ามแพลตฟอร์มนี้ช่วยให้ระบบสามารถออกแบบและปรับใช้ได้ตามความต้องการขององค์กรและโครงสร้างพื้นฐานไอที

3. ความสามารถหลักของ PostgreSQL

3.1 การจัดเก็บข้อมูล

- รองรับ **ข้อมูลเชิงโครงสร้าง (Structured Data)** เช่น ตาราง SQL
- รองรับ **ข้อมูลกึ่งโครงสร้าง (Semi-structured Data)** เช่น JSON, XML, Hstore
- รองรับ **ข้อมูลเชิงภูมิศาสตร์ (Geospatial Data)** ผ่านส่วนขยาย ฟ
- รองรับ **Array, JSONB, UUID** และชนิดข้อมูลอื่น ๆ สำหรับการพัฒนาระบบที่ซับซ้อน

3.2 การทำงานแบบหลายผู้ใช้ (Multi-user)

PostgreSQL ออกแบบให้สามารถรองรับผู้ใช้งานจำนวนมากพร้อมกันได้อย่างมีประสิทธิภาพ

- **MVCC (Multi-Version Concurrency Control):** ช่วยให้หลายผู้ใช้สามารถอ่านและเขียนข้อมูลพร้อมกันโดยไม่เกิด deadlock
- การล็อกระดับแถว (Row-level Locking) ช่วยให้ประสิทธิภาพสูงแม้ฐานข้อมูลใหญ่

3.3 การรักษาความถูกต้องของข้อมูล (Data Integrity)

- **Constraints:** เช่น PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK
- **Triggers & Rules:** สามารถกำหนดกฎและเงื่อนไขในการอัปเดตข้อมูลอัตโนมัติ
- **Foreign Data Wrapper (FDW):** เชื่อมต่อข้อมูลจากฐานข้อมูลอื่น เช่น MySQL, MongoDB, Oracle

PostgreSQL เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System: RDBMS) ที่ได้รับความนิยมอย่างแพร่หลาย ด้วยความสามารถในการจัดเก็บข้อมูลเชิงโครงสร้าง และข้อมูลกึ่งโครงสร้าง อย่างมีประสิทธิภาพ PostgreSQL เป็นซอฟต์แวร์แบบโอเพนซอร์ส (Open Source) ที่รองรับการใช้งานทั้งบนระบบปฏิบัติการ Windows, Linux และ macOS ทำให้สามารถติดตั้งและใช้งานได้อย่างยืดหยุ่นตามความต้องการของระบบ นอกจากนี้ยังมีความสามารถด้านการทำงานแบบหลายผู้ใช้ (Multi-user) การควบคุมธุรกรรมข้อมูล (Transaction Control) และการรักษาความถูกต้องของข้อมูล (Data Integrity) ตามมาตรฐาน ACID (Atomicity, Consistency, Isolation, Durability) ซึ่งเหมาะสมกับระบบที่ต้องการความน่าเชื่อถือสูง

PostGIS เป็นส่วนขยาย (Extension) ที่เพิ่มศักยภาพของ PostgreSQL ให้สามารถรองรับข้อมูลเชิงพื้นที่ (Spatial Data) และข้อมูลภูมิสารสนเทศ (Geographic Information System: GIS) ได้โดยตรงภายในฐานข้อมูล โดยรองรับรูปแบบข้อมูลทางภูมิศาสตร์ตามมาตรฐานของ Open Geospatial Consortium (OGC) เช่น จุด (Point), เส้น (LineString), พื้นที่ (Polygon) รวมถึงฟังก์ชันทางคณิตศาสตร์และภูมิศาสตร์กว่า 300 ฟังก์ชัน เช่น ST_Distance() สำหรับคำนวณระยะทางระหว่างจุด, ST_Intersects() สำหรับตรวจสอบการตัดกันของวัตถุเชิงพื้นที่, ST_Buffer() สำหรับสร้างเขตพื้นที่โดยรอบ เป็นต้น

ในงานวิจัยนี้ ระบบติดตามรถเก็บขยะได้นำ PostgreSQL มาใช้เป็นฐานข้อมูลกลาง (Central Database) เพื่อจัดเก็บข้อมูลจากอุปกรณ์ IoT ได้แก่ ข้อมูลพิกัดตำแหน่ง GPS, ความเร็วของรถ, เวลา, ทิศทางการเคลื่อนที่

และสถานะของรถในแต่ละช่วงเวลา ข้อมูลทั้งหมดจะถูกส่งเข้าสู่ฐานข้อมูลผ่านการประมวลผลของ Node-RED ที่เชื่อมต่อกับ MQTT Broker เพื่อให้สามารถจัดเก็บได้อย่างเป็นระบบและปลอดภัย ขณะที่ PostGIS จะช่วยในการจัดเก็บข้อมูลพิกัดในรูปแบบเชิงพื้นที่ (Geometry Type) ทำให้สามารถเรียกใช้งานข้อมูลร่วมกับโปรแกรมแผนที่ เช่น Leaflet หรือ QGIS เพื่อแสดงผลในรูปแบบ WebGIS ได้โดยตรง

นอกจากนี้ การใช้ PostGIS ยังสามารถทำการวิเคราะห์เชิงพื้นที่ (Spatial Analysis) ภายในฐานข้อมูลได้ เช่น

- การคำนวณระยะทางการเดินทางของรถเก็บขยะในแต่ละวัน
- การสร้างเส้นทาง (Route) จากข้อมูลพิกัดจุดที่เก็บได้เพื่อวิเคราะห์ความเหมาะสมของเส้นทาง
- การเปรียบเทียบระยะทางและเวลาเดินรถระหว่างวัน เพื่อหาความแตกต่างของประสิทธิภาพ
- คำนวณระยเวลาจากจุดแรกที่เริ่มทำงานและหลังจบงาน
- คำนวณจุดที่หยุดเก็บขยะ ระยะเวลาในการจอด

การผสานการทำงานระหว่าง PostgreSQL และ PostGIS ช่วยให้ระบบสามารถจัดเก็บและวิเคราะห์ข้อมูลเชิงพื้นที่ได้อย่างครบวงจร ตั้งแต่การรับข้อมูลจากอุปกรณ์ต้นทาง ไปจนถึงการประมวลผลและแสดงผลในระบบ WebGIS แบบเรียลไทม์ ซึ่งไม่เพียงช่วยให้เจ้าหน้าที่สามารถติดตามตำแหน่งของรถเก็บขยะได้ทันที แต่ยังสามารถนำข้อมูลไปใช้วางแผนปรับปรุงเส้นทางการเก็บขยะ เพิ่มประสิทธิภาพการใช้ทรัพยากร และสนับสนุนการตัดสินใจในการบริหารจัดการงานเก็บขยะขององค์กรปกครองส่วนท้องถิ่น

2.1.6. ภาษาที่ใช้ในการเขียนโปรแกรม

ภาษา HTML

ภาษา HTML (HyperText Markup Language) เป็นภาษามาร์กอัป (Markup Language) ที่ใช้สำหรับสร้างและกำหนดโครงสร้างของเอกสารบนเว็บเพจ (Web Page) โดยไม่ได้เป็นภาษาสำหรับเขียนโปรแกรม (Programming Language) โดยตรง แต่เป็นภาษาที่ใช้กำหนดรูปแบบการจัดวางเนื้อหา เช่น ข้อความ ภาพ ตาราง ลิงก์ หรือแบบฟอร์ม เพื่อให้เว็บเบราว์เซอร์ (Web Browser) สามารถแสดงผลเนื้อหาได้อย่างถูกต้องและเป็นระเบียบ HTML ทำงานโดยใช้ “แท็ก” (Tag) ซึ่งเป็นคำสั่งที่อยู่ในเครื่องหมาย < > เพื่อบอกให้เบราว์เซอร์เข้าใจว่าเนื้อหานั้นคืออะไร เช่น <p> ใช้สำหรับย่อหน้า, <h1> ถึง <h6> สำหรับหัวข้อ, <img> สำหรับแทรกรูปภาพ, และ <a> สำหรับลิงก์เชื่อมโยง ภาษา HTML ได้รับการพัฒนาโดย “Tim Berners-Lee” ในปี ค.ศ. 1991 เวอร์ชันที่นิยมใช้ในปัจจุบันคือ HTML5 ซึ่งเพิ่มความสามารถรองรับการทำงานร่วมกับ JavaScript และ CSS ได้อย่างสมบูรณ์ นอกจากนี้ HTML ยังเป็นพื้นฐานสำคัญในการพัฒนาเว็บไซต์ทุกประเภท โดยทำงานร่วมกับ CSS (Cascading Style Sheets) เพื่อจัดรูปแบบการแสดงผล และ

JavaScript เพื่อเพิ่มความสามารถเชิงโต้ตอบทำให้ HTML เป็นองค์ประกอบหลักของเทคโนโลยีเว็บที่เรียกว่า “Frontend” หรือส่วนที่ผู้ใช้มองเห็นและโต้ตอบกับระบบบนหน้าเว็บได้โดยตรง

ภาษา CSS

CSS (Cascading Style Sheets) เป็นภาษาที่ถูกออกแบบมาเพื่อกำหนดลักษณะการแสดงผลขององค์ประกอบต่าง ๆ บนเว็บเพจโดยแยกออกจากโครงสร้างของ HTML ทำให้สามารถควบคุมสี ตัวอักษร ขนาด การจัดวาง ตลอดจนเอฟเฟกต์ต่าง ๆ ของหน้าเว็บได้อย่างยืดหยุ่นและเป็นระบบ โดยสามารถนำไปประยุกต์ใช้ทำให้ผู้พัฒนาสามารถปรับแต่งรูปแบบของเว็บเพจได้โดยไม่กระทบต่อโครงสร้างเนื้อหา อีกทั้งยังรองรับการออกแบบให้เหมาะสมกับอุปกรณ์และหน้าจอหลากหลายขนาด (Responsive Design) และช่วยให้การจัดการสไตล์ของเว็บไซต์มีความสอดคล้องและง่ายต่อการบำรุงรักษาในระยะยาว

การแยกส่วนโครงสร้างและการนำเสนอ (Separation of Concerns): CSS ช่วยให้การกำหนดรูปแบบ (Style) แยกออกจาก โครงสร้าง (Structure) ที่กำหนดโดย HTML ซึ่งทำให้โค้ด HTML สะอาดขึ้น (Semantic HTML) และการจัดการสไตล์ของเว็บไซต์มีความสอดคล้องและง่ายต่อการบำรุงรักษาในระยะยาว

การออกแบบให้รองรับหลายอุปกรณ์ (Responsive Design): CSS รองรับการออกแบบที่ปรับเปลี่ยนตามอุปกรณ์และหน้าจอหลากหลายขนาด เทคนิคนี้ใช้ Media Queries ใน CSS เพื่อกำหนดชุดกฎเกณฑ์ (Rules) ที่แตกต่างกันสำหรับอุปกรณ์ที่มีความกว้างหน้าจอ (Viewport) ต่างกัน ทำให้หน้าเว็บแสดงผลได้อย่างเหมาะสมทั้งบนคอมพิวเตอร์, แท็บเล็ต, และโทรศัพท์มือถือ

การจัดการกับลำดับความสำคัญ (Cascading and Specificity): คำว่า "Cascading" ในชื่อ CSS หมายถึง ลำดับความสำคัญ ที่ถูกกำหนดให้กับกฎเกณฑ์ต่าง ๆ กล่าวคือ หากมีกฎ (Rule) การกำหนดรูปแบบที่ขัดแย้งกัน ระบบจะใช้หลักการเฉพาะเจาะจง (Specificity) และลำดับในการประกาศ เพื่อตัดสินว่าสไตล์ใดควรถูกนำมาใช้กับองค์ประกอบ HTML นั้น ๆ ทำให้การจัดการสไตล์มีความเป็นระบบและคาดการณ์ผลลัพธ์ไ

ความสามารถด้านภาพเคลื่อนไหวและมัลติมีเดีย (Animations and Multimedia): CSS สมัยใหม่ (CSS3) มีคุณสมบัติในการสร้างภาพเคลื่อนไหว (Animations) และการเปลี่ยนผ่าน (Transitions) ที่ทำงานบนฝั่งไคลเอ็นต์ (Client-side) ซึ่งช่วยเพิ่มความน่าสนใจให้กับเว็บไซต์โดยไม่ต้องพึ่งพา JavaScript มากนัก

การกำหนดตำแหน่ง (Layout Techniques): CSS มีเครื่องมือที่มีประสิทธิภาพสำหรับการจัดวางองค์ประกอบบนหน้าเว็บ เช่น Flexbox (Flexible Box Layout Module) สำหรับการจัดเรียงองค์ประกอบในหนึ่งมิติ (แถวหรือคอลัมน์) และ CSS Grid Layout สำหรับการจัดวางองค์ประกอบในสองมิติ (แถวและคอลัมน์พร้อมกัน) ซึ่งเป็นเครื่องมือสำคัญในการสร้างโครงสร้างเว็บไซต์ที่ซับซ้อนและมีการจัดวางที่ซับซ้อน

ภาษา JavaScript

JavaScript เป็นภาษาโปรแกรมที่นักพัฒนาใช้ในการสร้างหน้าเว็บแบบอินเทอร์แอคทีฟ ตั้งแต่การรีเฟรชฟีดสื่อโซเชียลไปจนถึงการแสดงภาพเคลื่อนไหวและแผนที่แบบอินเทอร์แอคทีฟ ฟังก์ชันของ JavaScript สามารถปรับปรุงประสบการณ์ที่ผู้ใช้จะได้รับจากการใช้งานเว็บไซต์ และในฐานะที่เป็นภาษาในการเขียนสคริปต์ฝั่งไคลเอ็นต์ (Client) จึงเป็นหนึ่งในเทคโนโลยีหลักของ World Wide Web ยกตัวอย่างเช่น เมื่อคุณท่องเว็บแล้วเห็นภาพสไลด์ เมนูดร็อปดาวน์แบบคลิกให้แสดงผล หรือสีองค์ประกอบที่เปลี่ยนบนหน้าเว็บ นั่นคือคุณเห็นเอฟเฟกต์ของ JavaScript

ภาษาโปรแกรมทั้งหมดทำงานด้วยการแปลไวยากรณ์ที่คล้ายภาษาอังกฤษเป็นโค้ดสำหรับเครื่อง จากนั้นระบบปฏิบัติการจะเรียกใช้โค้ดนั้น JavaScript ได้รับการจัดประเภทอย่างกว้าง ๆ ว่าเป็นภาษาเขียนสคริปต์ หรือภาษาที่แปลผลแล้ว โค้ด JavaScript ได้รับการแปลผล—นั่นคือ แปลโดยตรงเป็นโค้ดภาษาสำหรับเครื่อง ด้วยกลไกล JavaScript ในขณะที่ในภาษาโปรแกรมอื่น ๆ คอมไพเลอร์จะคอมไพล์โค้ดทั้งหมดเป็นโค้ดสำหรับเครื่องในขั้นตอนที่แยกต่างหาก ดังนั้น ภาษาเขียนสคริปต์ทั้งหมดจึงเป็นภาษาโปรแกรม แต่ไม่ใช่ว่าภาษาโปรแกรมทั้งหมดจะเป็นภาษาเขียนสคริปต์เสมอไป

ภาษา PHP

ภาษา PHP (Hypertext Preprocessor) เป็นภาษาสคริปต์ที่ทำงานฝั่งเซิร์ฟเวอร์ (Server-side) ถูกออกแบบมาเพื่อใช้พัฒนาเว็บแอปพลิเคชันแบบไดนามิก (Dynamic Web Application) หรือเว็บไซต์ที่มีการตอบสนองต่อผู้ใช้ได้อย่างยืดหยุ่น โดยภาษา PHP ถูกสร้างขึ้นครั้งแรกโดย Rasmus Lerdorf ในปี ค.ศ. 1994 และได้กลายเป็นหนึ่งในภาษาที่นิยมใช้มากที่สุดสำหรับการพัฒนาเว็บไซต์ เนื่องจากมีโครงสร้างภาษาที่เข้าใจง่าย ทำงานร่วมกับฐานข้อมูลได้หลากหลาย เช่น MySQL, PostgreSQL, SQLite และสามารถฝังรวมเข้ากับภาษา HTML ได้โดยตรง

PHP ทำงานโดยฝั่งเซิร์ฟเวอร์จะประมวลผลโค้ด PHP ก่อน แล้วส่งผลลัพธ์ที่ได้ในรูปแบบ HTML กลับไปยังเว็บเบราว์เซอร์ของผู้ใช้ ทำให้เหมาะสำหรับเว็บไซต์ที่ต้องมีการรับ–ส่งข้อมูล เช่น ระบบล็อกอิน, ระบบจัดการฐานข้อมูล หรือเว็บที่ต้องแสดงผลตามเงื่อนไขของผู้ใช้งาน ปัจจุบัน PHP ยังคงเป็นภาษาหลักของเว็บไซต์จำนวนมากทั่วโลก

ภาษา SQL

Structured Query Language (SQL) เป็นภาษาโปรแกรมสำหรับจัดเก็บและประมวลผลข้อมูลในฐานข้อมูลแบบเชิงสัมพันธ์ ฐานข้อมูลแบบเชิงสัมพันธ์เก็บข้อมูลในรูปแบบตารางที่มีแถวและคอลัมน์ที่เป็นตัวแทนของหมวดข้อมูลที่แตกต่างกันและความสัมพันธ์ต่างๆ ระหว่างค่าข้อมูล สามารถใช้คำสั่ง SQL ในการ

จัดเก็บ ปรับปรุง ลบ ค้นหา และดึงข้อมูลจากฐานข้อมูล นอกจากนี้ยังสามารถใช้ SQL ในการรักษาและเพิ่มประสิทธิภาพการทำงานของฐานข้อมูล ระบบการจัดการฐานข้อมูลแบบเชิงสัมพันธ์ใช้ภาษาแบบสอบถามที่มีโครงสร้าง (SQL) ในการจัดเก็บและจัดการข้อมูล ระบบจัดเก็บตารางฐานข้อมูลหลายเชื่อมต่อกันและกัน MS SQL Server MySQL หรือ MS Access เป็นตัวอย่างของระบบการจัดการฐานข้อมูลแบบเชิงสัมพันธ์ ต่อไปนี้เป็นส่วนประกอบของระบบดังกล่าว

2.1.7. ไลบรารีที่ใช้ในการแสดงผลแผนที่: Leaflet.js

Leaflet.js เป็นไลบรารีภาษา JavaScript แบบโอเพนซอร์สที่ใช้สำหรับสร้างและแสดงผลแผนที่ออนไลน์บนเว็บเบราว์เซอร์ โดยรองรับการแสดงแผนที่แบบโต้ตอบ (Interactive Map) เช่น การซูม เลื่อน และคลิกเลือกตำแหน่งต่าง ๆ บนแผนที่ Leaflet.js สามารถรวมข้อมูลจากแหล่งต่าง ๆ เช่น OpenStreetMap, Mapbox หรือ WMS และสามารถแสดงข้อมูลเชิงพื้นที่ เช่น จุด (Marker), เส้นทาง (Polyline), พื้นที่ (Polygon) และแผนที่ความหนาแน่น (Heatmap) ได้อย่างง่ายดาย รวมถึงรองรับการเพิ่มฟังก์ชันเสริมและปลั๊กอินสำหรับการวิเคราะห์เชิงพื้นที่หรือการแสดงข้อมูลแบบเรียลไทม์ ทำให้ Leaflet.js เป็นเครื่องมือสำคัญในการสร้างระบบ WebGIS และแผนที่ออนไลน์สำหรับการติดตาม วิเคราะห์ และแสดงผลข้อมูลเชิงพื้นที่บนเว็บ

2.1.8. โปรแกรมที่ใช้สร้างหน้าเว็บ: Visual Studio Code

Visual Studio Code เป็นโปรแกรมแก้ไขโค้ด (Code Editor) แบบโอเพนซอร์สที่สามารถใช้งานได้บนหลายระบบปฏิบัติการ เช่น Windows, macOS และ Linux ซึ่งรองรับการเขียนโปรแกรมได้หลายภาษา เช่น HTML, CSS, JavaScript, PHP, SQL, Python, C/C++ และภาษาอื่น ๆ VS Code มีคุณสมบัติเด่นหลายประการ เช่น การเน้นสีไวยากรณ์ ที่ช่วยให้อ่านโค้ดได้ง่ายขึ้น การเติมโค้ดอัตโนมัติ (IntelliSense) ที่ช่วยให้เขียนโค้ดได้รวดเร็วและลดข้อผิดพลาด การดีบักโค้ด (Debugging) ที่ช่วยตรวจสอบและแก้ไขข้อผิดพลาดได้ทันที รวมถึงการติดตั้งส่วนขยาย (Extensions) เพื่อเพิ่มฟังก์ชันใหม่ ๆ ตามความต้องการของผู้พัฒนา นอกจากนี้ VS Code ยังสามารถทำงานร่วมกับระบบควบคุมเวอร์ชัน Git, สามารถเชื่อมต่อกับเซิร์ฟเวอร์สำหรับพัฒนาเว็บ, และรองรับการสร้างโปรเจกต์แบบหลายไฟล์ได้อย่างสะดวก ทำให้ VS Code เป็นเครื่องมือสำคัญในการพัฒนาเว็บแอปพลิเคชัน ระบบ WebGIS หรือระบบที่เกี่ยวข้องกับ IoT อย่างเช่นระบบติดตามรถเก็บขยะแบบเรียลไทม์ เพราะช่วยให้ผู้พัฒนาสามารถเขียน แก้ไข และทดสอบโค้ดได้อย่างรวดเร็ว แม่นยำ และมีประสิทธิภาพ

2.1.9. โปรแกรมแจ้งเตือน: Telegram

ภาพที่ 6 Telegram

Telegram เป็นแอปพลิเคชันสื่อสารแบบข้อความ (Messaging Application) ที่สามารถใช้งานได้ทั้งบนคอมพิวเตอร์และอุปกรณ์มือถือ มีความปลอดภัยสูงด้วยการเข้ารหัสข้อมูล (End-to-End Encryption) และรองรับการส่งข้อความ, ไฟล์, รูปภาพ, วิดีโอ รวมถึงสติกเกอร์และลิงก์ต่าง ๆ Telegram ยังสามารถสร้าง Bot ซึ่งเป็นโปรแกรมอัตโนมัติที่สามารถส่งข้อความแจ้งเตือน (Notification) หรือทำงานตามคำสั่งที่กำหนดล่วงหน้าได้ โดย Bot จะเชื่อมต่อกับระบบหรือเซิร์ฟเวอร์ผ่าน API ทำให้สามารถส่งข้อมูลแบบเรียลไทม์จากอุปกรณ์ IoT หรือระบบ WebGIS ไปยังเจ้าหน้าที่ได้ทันที การใช้ Telegram Bot ช่วยให้เจ้าหน้าที่เทศบาลได้รับการแจ้งเตือนเมื่อเกิดเหตุการณ์สำคัญ เช่น รถเก็บขยะขับเร็วเกินกำหนด ระบบนี้ช่วยเพิ่มประสิทธิภาพในการตรวจสอบและบริหาร

2.1.10. ซอฟต์แวร์อื่น ๆ: MS4W

MS4W (MapServer for Windows) เป็นแพ็กเกจซอฟต์แวร์ที่จัดเตรียมเครื่องมือสำหรับติดตั้งและใช้งาน MapServer บนระบบปฏิบัติการ Windows อย่างครบวงจร โดย MapServer เป็น Web Map Server ที่ใช้สำหรับสร้าง แสดงผล และให้บริการแผนที่เชิงพื้นที่ผ่านเว็บ MS4W รวมเอาเครื่องมือที่จำเป็นหลายอย่างไว้พร้อมใช้งาน เช่น Apache Web Server, MapServer, PHP และเครื่องมืออื่น ๆ ทำให้ผู้พัฒนาสามารถเริ่มพัฒนาเว็บ GIS หรือ WebGIS ได้ทันทีโดยไม่ต้องติดตั้งแต่ละโปรแกรมแยกกัน

MS4W รองรับการอ่านข้อมูลจากแหล่งข้อมูลเชิงพื้นที่ต่าง ๆ เช่น ไฟล์ Shapefile หรือ ฐานข้อมูลเชิงพื้นที่ (เช่น PostGIS) และสามารถส่งออกข้อมูลเป็นภาพแผนที่ (Map Images) หรือบริการข้อมูลแผนที่แบบเว็บ (Web Map Service) ทำให้ผู้ใช้สามารถสร้างแผนที่แบบโต้ตอบ แก้ไขการแสดงผลของชั้นข้อมูล (Layers) สัญลักษณ์ สี และสไตล์ของแผนที่ได้ตามต้องการ นอกจากนี้ยังสามารถใช้สคริปต์ PHP เพื่อสร้างแผนที่ที่ตอบสนองต่อผู้ใช้

2.2. เอกสารและงานวิจัยที่เกี่ยวข้อง

สกรณ์ บุษบง, ณัฐพล จั้นพลแสน, ศุภกร ชาติชัย (2019) **“การพัฒนาต้นแบบระบบพร้อมชุดอุปกรณ์ติดตามรถขนส่งพัสดุแบบ เรียลไทม์โดยใช้เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง”** การวิจัยเรื่องการพัฒนาต้นแบบระบบพร้อมชุดอุปกรณ์ติดตามรถขนส่งพัสดุแบบเรียลไทม์โดยใช้ อินเทอร์เน็ตของสรรพสิ่ง มีวัตถุประสงค์ 2 ส่วน คือ 1) เพื่อพัฒนาชุดอุปกรณ์ติดตามพัสดุแบบเรียลไทม์ผ่าน เว็บแอพพลิเคชั่น 2) เพื่อประเมินประสิทธิภาพการทำงานของระบบจากผู้เชี่ยวชาญและผู้ใช้งาน โดยระบบ แบ่งออกเป็นสองส่วนคือ ส่วนของชุดอุปกรณ์และส่วนของเว็บแอพพลิเคชั่น ส่วนของอุปกรณ์ทำหน้าที่ส่ง ข้อมูลตำแหน่งละติจูด ลองจิจูด ความเร็วการเคลื่อนที่และข้อมูลพัสดุไปยัง Firebase Real-time Database และ NETPIE Cloud Platform ใช้เว็บแอพพลิเคชั่นในการจัดการข้อมูล Firebase Real-time Database การพัฒนาต้นแบบระบบพร้อมชุด อุปกรณ์ติดตามรถขนส่งพัสดุแบบเรียลไทม์โดยใช้ เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่งครั้งนี้ พบว่าเป็นไปตามวัตถุประสงค์ที่ตั้งไว้โดยระบบสามารถ ทำการระบุตำแหน่งและแสดงตำแหน่งบนเว็บ แอพพลิเคชั่นได้ถูกต้อง รวมทั้งสามารถเพิ่มพัสดุใน การติดตามตำแหน่ง ซึ่งสามารถนำไปประยุกต์ใช้กับ การขนส่งต่าง ๆ ได้ ไม่ว่าจะเป็นการส่งพัสดุ การส่ง สินค้าต่าง ๆ ที่ต้องมีการตรวจสอบสถานะของสินค้า รวมถึงทางการแพทย์ เช่นการส่งเลือด หรืออวัยวะ ปลูกถ่าย ที่จำเป็นจะต้องระบุเวลาที่พัสดุดังกล่าว จะเดินทางมาถึงสถานพยาบาล ซึ่งอาจเพิ่มเซนเซอร์ ในการวัดอุณหภูมิในการแจ้งอุณหภูมิปัจจุบันของ พัสดุ การทราบตำแหน่งปัจจุบันทำให้สามารถ คาดการณ์เวลาที่พัสดุจะเดินทางมาถึงและช่วยให้ สถานพยาบาลสามารถเตรียมเครื่องมือทางการแพทย์ พร้อมที่จะปฏิบัติหน้าที่ได้ทันที

Bakhtiar Ali, Muhammad Awais Javed, & Alharbi, Abeer A. K. (n.d.). (2024) **“Internet of Things-Assisted Vehicle Route Optimization for Municipal Solid Waste Collection.”** งานวิจัยนี้ได้นำเสนอ กลไกการเก็บขยะที่เป็นนวัตกรรม โดยใช้เทคโนโลยี Internet of Things (IoT) และอัลกอริทึมการวิเคราะห์การตัดสินใจแบบหลายเกณฑ์ที่เรียกว่า TOPSIS เพื่อแก้ไขปัญหาความท้าทายที่สำคัญในการเก็บขยะมูลฝอยชุมชนในเมืองอัจฉริยะ (Smart City) โดยมีวัตถุประสงค์เพื่อพัฒนาแนวทางการจัดการเส้นทางการเก็บขยะมูลฝอยของเทศบาลให้มีประสิทธิภาพมากขึ้น ผ่านการประยุกต์ใช้เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง Internet of Things (IoT) ร่วมกับอัลกอริทึมการตัดสินใจแบบหลายเกณฑ์ ชื่อว่า TOPSIS ในการวิเคราะห์และเลือกเส้นทางที่เหมาะสมที่สุดสำหรับรถเก็บขยะ ระบบที่พัฒนาขึ้นสามารถเชื่อมต่อกับเซนเซอร์ตรวจจับปริมาณขยะในถังและข้อมูลจากยานพาหนะได้แบบเรียลไทม์ ทำให้สามารถวิเคราะห์และปรับเปลี่ยนเส้นทางการเก็บขยะให้มีความยืดหยุ่นตามสภาวะจริงได้ ผลการทดลองพบว่าการใช้ระบบ IoT ร่วมกับอัลกอริทึม TOPSIS สามารถช่วยลดระยะทางการเดินรถ ลดการใช้พลังงานเชื้อเพลิง และเพิ่มประสิทธิภาพของการบริหารจัดการขยะในเมืองอัจฉริยะได้อย่างมีนัยสำคัญ

Mohammadhossein Ghahramani, MengChu Zhou, Anna Molter, & Francesco Pilla (2022)“ **IoT-Based Route Recommendation for Intelligent Waste Management System**”มีจุดประสงค์เพื่อพัฒนาระบบแนะนำเส้นทางการเก็บขยะที่ชาญฉลาด โดยอาศัยเทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง Internet of Things (IoT) และระบบประมวลผลข้อมูลขนาดใหญ่เพื่อเพิ่มประสิทธิภาพของกระบวนการจัดการขยะในเขตเมือง งานวิจัยนี้ได้นำเสนอโมเดลอัจฉริยะที่สามารถรวบรวมข้อมูลจากเซนเซอร์ตรวจจับปริมาณขยะที่ติดตั้งในถังขยะและข้อมูลการจราจรแบบเรียลไทม์ จากนั้นนำข้อมูลดังกล่าวมาประมวลผลด้วยอัลกอริทึมการเรียนรู้ของเครื่อง (Machine Learning) เพื่อวิเคราะห์และคาดการณ์ปริมาณขยะในแต่ละพื้นที่ รวมถึงแนะนำเส้นทางที่เหมาะสมที่สุดสำหรับรถเก็บขยะในช่วงเวลานั้น ๆ

ระบบดังกล่าวสามารถปรับเปลี่ยนเส้นทางแบบไดนามิก ตามสถานการณ์จริง เช่น เมื่อมีถังขยะที่เต็มเร็วกว่าปกติ หรือเมื่อพบการจราจรติดขัดในบางเส้นทาง เพื่อให้รถเก็บขยะสามารถจัดลำดับการเก็บได้อย่างมีประสิทธิภาพและตรงตามความต้องการจริงในพื้นที่ ทั้งนี้ระบบยังช่วยลดระยะทางการขับขี่ ลดเวลาในการปฏิบัติงาน และลดการใช้เชื้อเพลิง ซึ่งส่งผลให้ลดการปล่อยก๊าซคาร์บอนไดออกไซด์และมลพิษทางอากาศ

ผลการทดสอบระบบต้นแบบในพื้นที่ทดลองแสดงให้เห็นว่าการประยุกต์ใช้ IoT และ Big Data Analytics สามารถเพิ่มความแม่นยำในการวิเคราะห์เส้นทางได้มากขึ้นกว่าวิธีดั้งเดิม โดยเฉพาะอย่างยิ่งในสภาพแวดล้อมเมืองที่มีการเปลี่ยนแปลงของข้อมูลแบบต่อเนื่อง ระบบสามารถลดระยะทางเฉลี่ยของการเก็บขยะได้มากกว่า 20% และลดเวลาการปฏิบัติงานโดยรวมได้กว่า 15% เมื่อเทียบกับระบบเก็บขยะรูปแบบเดิม แสดงให้เห็นถึงศักยภาพของการใช้เทคโนโลยี IoT ในการบริหารจัดการทรัพยากรและการดำเนินงานของเมืองอัจฉริยะ

Ibraheem Kasim Ibraheem, Salam Wisam Hadi (2018) **“Design and Implementation of a Low-Cost Secure Vehicle Tracking System”** งานวิจัยนี้มีจุดประสงค์เพื่อพัฒนาต้นแบบระบบติดตามยานพาหนะ (Vehicle Tracking System: VTS) ที่มีต้นทุนต่ำและมีความปลอดภัยสูง โดยใช้เทคโนโลยีไร้สาย XBee ร่วมกับระบบ GPS ระบบถูกแบ่งออกเป็นสองส่วนหลักคือ ชุดอุปกรณ์ในยานพาหนะ (Vehicle unit) ที่ประกอบด้วยไมโครคอนโทรลเลอร์ Arduino, โมดูล XBee และ GPS receiver (gy-gps6mv2) เพื่อรับและส่งข้อมูลพิกัดแบบเรียลไทม์ และ สถานีตรวจสอบ (Monitoring station) ที่ใช้โมดูล XBee อีกตัวเชื่อมต่อกับคอมพิวเตอร์เพื่อรับข้อมูลพิกัดและแสดงผลบน Google Earth การเลือกใช้ XBee แทน GSM/GPRS ช่วยลดค่าใช้จ่ายในการรับส่งข้อมูลได้อย่างมีนัยสำคัญ ผลการทดสอบแสดงให้เห็นว่าระบบสามารถติดตามตำแหน่งยานพาหนะได้อย่างต่อเนื่องและแม่นยำ แม้ในพื้นที่ที่มีสิ่งปลูกสร้างหนาแน่นและมีสัญญาณรบกวน นอกจากนี้ยังมีการใช้ซอฟต์แวร์ XCTU เพื่อตรวจสอบการเชื่อมต่อระหว่างโมดูล XBee และมีการปรับตั้งค่าใน Google Earth เพื่อแสดงผลพิกัดแบบเรียลไทม์

Idriss Moumen, Najat Rafalia, Jaafar Abouchabaka and Marouane Aoufi1 (2023)

“Real-time GPS Tracking System for IoT-Enabled Connected Vehicles” บทความนี้ได้นำเสนอระบบติดตาม GPS แบบเรียลไทม์สำหรับเครือข่ายยานยนต์ที่เชื่อมต่อกัน (Connected Vehicles) โดยผสานรวมเทคโนโลยี Internet of Things (IoT) และการสื่อสารระหว่างยานพาหนะ (V2X) ระบบนี้ใช้ฮาร์ดแวร์หลักอย่าง Arduino Uno R3 เป็นไมโครคอนโทรลเลอร์ และโมดูล SIM800L GSM และ NEO6M GPS ในการรับข้อมูลตำแหน่ง ข้อมูลที่ได้จะถูกส่งไปยัง Firebase Realtime Database เพื่อจัดเก็บและประมวลผล ผู้ใช้งานสามารถตรวจสอบข้อมูลตำแหน่งของยานพาหนะผ่านอินเทอร์เฟซบนเว็บที่พัฒนาด้วย Node.js และ Socket.IO จุดเด่นของระบบนี้คือสามารถนำไปประยุกต์ใช้เพื่อการจัดการการจราจร, การกำหนดเส้นทางเพื่อประหยัดพลังงาน, และการลดมลพิษได้ในอนาคต

OrvenE.Llantos, OlgaJoyL (2024) **“Real-Time Tracker for Moving Objects with Enhanced Signal and Customized User Interface through Broadcasting Global Positioning Systems Data”** งานวิจัยนี้ได้นำเสนอการพัฒนาอุปกรณ์ติดตาม GPS แบบเรียลไทม์สำหรับวัตถุเคลื่อนที่ขนาดเล็ก โดยอุปกรณ์ซึ่งมีชื่อว่า "Prototype-1" ถูกออกแบบให้มีขนาดกะทัดรัดและน้ำหนักเบาเพียง 19 กรัม ทำให้เหมาะสมสำหรับการติดตั้งบนสัตว์ขนาดเล็ก เช่น นก หรืออุปกรณ์เคลื่อนที่อื่น ๆ โดยอุปกรณ์ดังกล่าวสามารถรับสัญญาณตำแหน่งจากดาวเทียม GPS และส่งข้อมูลไปยังเซิร์ฟเวอร์คลาวด์ผ่านเครือข่าย GSM ซึ่งผู้ใช้งานสามารถเข้าถึงข้อมูลตำแหน่งได้แบบเรียลไทม์ผ่านเว็บแอปพลิเคชันและมือถือ นอกจากนี้ Prototype-1 ยังมีอินเทอร์เฟซผู้ใช้ที่ออกแบบมาให้สามารถแสดงเส้นทางการเคลื่อนที่ ข้อมูลความเร็ว และเวลาการเคลื่อนที่ของวัตถุอย่างชัดเจน พร้อมทั้งสามารถตั้งค่าความถี่ในการอัปเดตตำแหน่งได้ตามความต้องการของผู้ใช้งาน การทดลองใช้งานกับนกแสดงให้เห็นว่าอุปกรณ์มีความแม่นยำสูงและสามารถอัปเดตตำแหน่งได้รวดเร็วกว่าอุปกรณ์เชิงพาณิชย์ทั่วไป โดยค่า Mean Absolute Error (MAE), Root Mean Square Error (RMSE) และ Standard Deviation (SD) มีค่าน้อยกว่าอุปกรณ์เชิงพาณิชย์ ซึ่งสะท้อนถึงความแม่นยำและความสม่ำเสมอของข้อมูลที่ได้รับ ทำให้ Prototype-1 มีศักยภาพในการนำไปประยุกต์ใช้งานในหลายด้าน เช่น การติดตามสัตว์ป่า การตรวจสอบเส้นทางการเคลื่อนที่ของยานพาหนะ หรือการติดตามอุปกรณ์เคลื่อนที่ที่ต้องการการเก็บข้อมูลแบบเรียลไทม์ โดยอุปกรณ์ดังกล่าวช่วยเพิ่มประสิทธิภาพและความสะดวกในการเก็บข้อมูลเชิงพื้นที่และเคลื่อนที่ได้อย่างมีประสิทธิผล

Praveen Kumar, Raj Mohan Singh, Sailesh N. Behera (2022) **“A GIS-based Route Optimization Approach for Municipal Solid Waste Management”** ได้เน้นถึงความสำคัญของการผสานข้อมูลหลายแหล่งเข้าด้วยกันในระบบ GIS เช่น ข้อมูลจำนวนประชากร พื้นที่ให้บริการของเทศบาล ข้อมูลความจุของถังขยะ และตารางเวลาการเก็บขยะ เพื่อสร้างแบบจำลองการจัดการขยะที่มีประสิทธิภาพสูงสุด นอกจากนี้ ผู้วิจัยยังได้ใช้เทคนิคการจำลอง (Simulation) เพื่อเปรียบเทียบผลลัพธ์ของเส้นทางที่ได้รับการเพิ่มประสิทธิภาพกับเส้นทางแบบดั้งเดิม

ผลการทดลองแสดงให้เห็นว่า เส้นทางที่ได้รับการปรับปรุงสามารถลดระยะทางรวมได้เฉลี่ย 15–25% และลดการใช้เชื้อเพลิงได้มากกว่า 20% เมื่อเทียบกับระบบเก็บขยะเดิม อีกทั้งยังลดการปล่อยก๊าซเรือนกระจกจากยานพาหนะ ซึ่งเป็นประโยชน์ต่อสิ่งแวดล้อมโดยตรงงานวิจัยยังเสนอแนวทางการประยุกต์ใช้ระบบดังกล่าวในอนาคต เช่น การเชื่อมต่อข้อมูลแบบเรียลไทม์จากเซนเซอร์ในถังขยะ (Smart Bin) และการใช้ข้อมูลจากระบบ GPS เพื่อปรับปรุงเส้นทางแบบไดนามิก (Dynamic Routing) ซึ่งจะช่วยให้ระบบสามารถตอบสนองต่อสถานการณ์จริงได้อย่างรวดเร็วและยืดหยุ่นมากขึ้น โดยมีเป้าหมายสุดท้ายคือการพัฒนา “ระบบจัดการขยะอัจฉริยะ” (Smart Waste Management System) ที่สามารถลดต้นทุน เพิ่มประสิทธิภาพ และสนับสนุนแนวคิดเมืองอัจฉริยะ (Smart City) อย่างยั่งยืน

บทที่ 3

วิธีการดำเนินงานวิจัย

ในการศึกษาวิจัยการพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะโดยใช้ IoT และ GIS มีวิธีการดำการประกอบด้วย 2 ส่วนดังนี้

1. เครื่องมือที่ช่วยในการพัฒนาระบบ

2. วิธีการดำเนินงาน

3.1. เครื่องมือที่ช่วยในการพัฒนาระบบ

3.1.1. อุปกรณ์ที่ใช้ในการพัฒนา Hardware: NodeMCU ESP8266, GPS Shield, Arduino UNO, ANN-MB1-00

3.1.2. ภาษาที่ใช้สำหรับเซนเซอร์: C/C++

3.1.3. โปรแกรมที่ใช้เขียนโค้ดลงเซนเซอร์: Arduino IDE

3.1.4. โปรแกรมในการส่งข้อมูล: MQTT Broker, Node-RED

3.1.5. โปรแกรมจัดการฐานข้อมูล: PostgreSQL และ PostGIS

3.1.6. ภาษาที่ใช้ในการเขียนโปรแกรม: HTML, CSS, JavaScript, PHP, SQL

3.1.7. ไลบรารีที่ใช้ในการแสดงผลแผนที่: Leaflet.js

3.1.8. โปรแกรมที่ใช้สร้างหน้าเว็บ: Visual Studio Code

3.1.9. โปรแกรมแจ้งเตือน: Telegram

3.1.10. ซอฟต์แวร์อื่น ๆ: MS4W,

3.2. วิธีการดำเนินงาน

3.2.1 การออกแบบระบบ

ระบบติดตามรถเก็บขยะนี้ถูกออกแบบตามแนวคิด Geo-IoT (Geospatial Internet of Things) เพื่อออกแบบระบบ Real-time ที่มีประสิทธิภาพ โดยเริ่มต้นจาก ชุดอุปกรณ์ (Vehicle Unit) ที่ติดตั้ง GPS Tracking บนรถเก็บขยะ ซึ่งทำหน้าที่เก็บข้อมูลตำแหน่งและความเร็วและส่งข้อมูลไปยัง

เซิร์ฟเวอร์โดยใช้โปรโตคอล MQTT (Message Queuing Telemetry Transport) ที่มีความเบาและรวดเร็ว จากนั้นข้อมูลจะถูกส่งเข้าสู่ Node-RED Server ซึ่งทำหน้าที่เป็น Middleware ในการรับ, กรอง, และประมวลผลข้อมูลดิบ เช่น การตรวจสอบความถูกต้องของ GPS การสร้าง Flow เพื่อแจ้งเตือนไปยัง Telegram ทันทีหากตรวจพบเงื่อนไข (เช่น ขับเร็วเกิน 60 กม./ชม.) และจัดรูปแบบข้อมูลเพื่อนำเข้าสู่ Database Server สำหรับจัดเก็บข้อมูลทั้งหมดของระบบ ในขณะที่ Web Server & API จะทำหน้าที่เป็นส่วนเชื่อมต่อ โดยใช้ API ในการดึงข้อมูลตำแหน่งปัจจุบันและข้อมูลย้อนหลังมาแสดงผลแบบเรียลไทม์บน WebGIS Interface ซึ่งเป็นส่วนติดต่อผู้ใช้ที่แสดงผลข้อมูลทั้งหมดบนแผนที่ออนไลน์ได้ในทันที

3.2.2. การพัฒนาฮาร์ดแวร์และเซนเซอร์

NodeMCU ESP8266

ภาพที่ 7 บอร์ด NodeMCU ESP8266

ที่มา https://www.robotsiam.com/product/106/nodemcu-v2-cp2102-lua-wifi-esp8266-esp-12e

NodeMCU ESP8266 เป็นไมโครคอนโทรลเลอร์ที่มีโมดูล Wi-Fi ในตัว ซึ่งได้รับความนิยมอย่างแพร่หลายในงานพัฒนา ระบบ Internet of Things (IoT) เนื่องจากมีขนาดเล็ก ราคาประหยัด และสามารถเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตได้โดยตรง โดยไม่ต้องใช้โมดูลเสริมเพิ่มเติม ตัวบอร์ดสามารถเขียนโปรแกรมได้ด้วยภาษา C/C++ ผ่าน Arduino IDE และรองรับการเชื่อมต่อกับอุปกรณ์เซนเซอร์ต่าง ๆ เช่น GPS Shield เพื่อรับค่าพิกัดตำแหน่งแบบเรียลไทม์ รวมทั้งสามารถส่งข้อมูลตำแหน่ง ความเร็ว และเวลาการทำงานของรถเก็บขยะไปยัง MQTT Broker ผ่านเครือข่าย Wi-Fi ได้อย่างต่อเนื่อง

GPS Shield (GlobalSat EB-5365RE SIRF IV), Arduino UNO R3

ภาพที่ 8 บอร์ด **GPS Shield, Arduino UNO** R3

ที่มา https://forum.arduino.cc/t/tinygps-shows-only-chars-rx-increasing/190539

GPS Shield เป็นโมดูลรับสัญญาณจากระบบดาวเทียมนำทาง (Global Positioning System: GPS) ซึ่งสามารถระบุตำแหน่งพิกัดบนพื้นโลกได้อย่างแม่นยำ ทั้งค่าละติจูด (Latitude), ลองจิจูด (Longitude), ความเร็ว (Speed) และเวลา (Time) โดยโมดูลนี้จะทำหน้าที่รับสัญญาณจากดาวเทียมอย่างน้อย 3–4 ดวงขึ้นไป เพื่อคำนวณตำแหน่งปัจจุบันของรถเก็บขยะในแต่ละช่วงเวลาในการพัฒนาระบบนี้ GPS Shield ถูกติดตั้งและเชื่อมต่อเข้ากับ Arduino UNO ซึ่งทำหน้าที่เป็นบอร์ดควบคุมกลางที่ใช้รับข้อมูลจาก GPS Shield ผ่านการสื่อสารแบบอนุกรม (Serial Communication) จากนั้น Arduino UNO จะประมวลผลเบื้องต้น เช่น การแปลงข้อมูลจากสัญญาณ NMEA (National Marine Electronics Association) ให้อยู่ในรูปแบบตัวเลขพิกัดที่พร้อมนำไปใช้งาน

เสารับสัญญาณ ANN-MB1-00

ภาพที่ 9 ANN-MB1-00 GLOBALSAT

ที่มา https://www.digikey.co.th/th/products/detail/u-blox/ANN-MB1-00/14835875

เป็นโมดูลรับสัญญาณดาวเทียมเช่นเดียวกับ GPS Shield แต่มีความสามารถในการปรับปรุงความแม่นยำของพิกัดตำแหน่งและเสถียรภาพของสัญญาณมากขึ้น โมดูลนี้รองรับการติดตามหลายระบบดาวเทียม เช่น GPS, GLONASS, BeiDou หรือ Galileo ทำให้สามารถรับสัญญาณได้ต่อเนื่อง แม้ในสภาพแวดล้อมที่มีสิ่งกีดขวาง เช่น อาคารสูงหรือพื้นที่มีต้นไม้หนาแน่น ในการพัฒนาระบบติดตามรถเก็บขยะ ANN-MB1-00 ถูกเชื่อมต่อกับ Arduino UNO เพื่อเก็บข้อมูลพิกัดและอ่านค่าแสดงผ่าน Serial

OLED 128x64 V2.0 แบบ I2C 1.3 นิ้ว

ภาพที่ 10 OLED 128x64 V2.0 แบบ I2C 1.3 นิ้ว

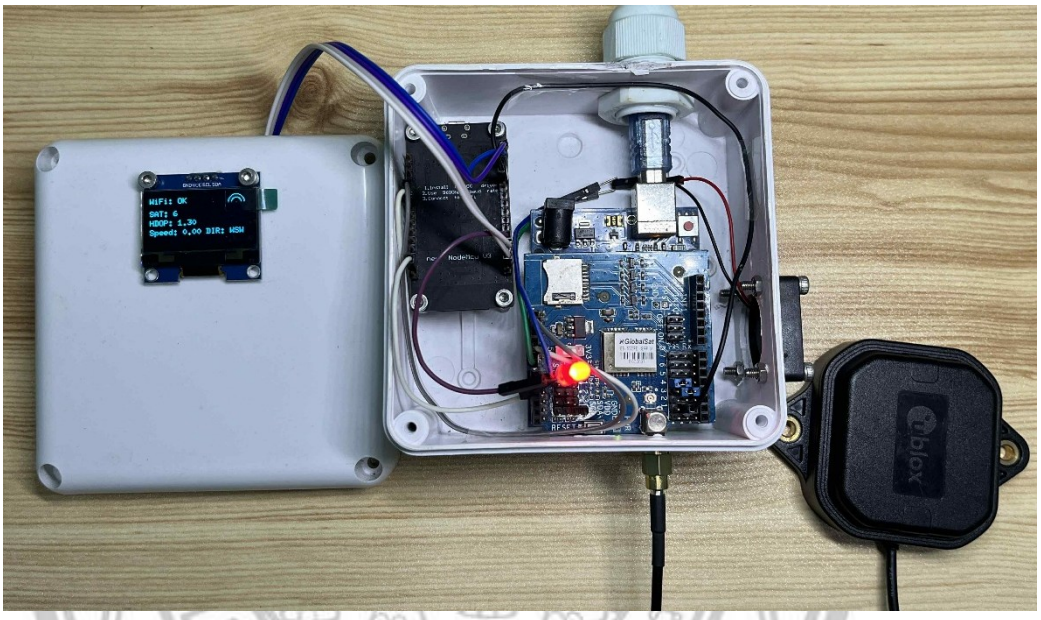
ที่มา https://docs.cirkitdesigner.com/component/a983a292-04ed-464a-8e70-77365ceef41b/oled-13

จอ OLED (Organic Light-Emitting Diode) ขนาด 1.3 นิ้ว มักมีความละเอียดที่ 128 x 64 พิกเซล และใช้ชิปควบคุม เช่น SH1106 เป็นโมดูลหน้าจอที่เหมาะสำหรับโครงการขนาดเล็กที่ต้องการการแสดงผลข้อมูลที่ชัดเจนและประหยัดพลังงาน ลักษณะเด่นที่ส่งผลต่อการแสดงผล

1. ความคมชัดสูง (High Contrast Ratio): เทคโนโลยี OLED ทำให้แต่ละพิกเซลเปล่งแสงได้ด้วยตัวเอง (Self-Illumination) และพิกเซลที่ไม่ทำงานจะดับสนิทเป็นสีดำจริง ทำให้ตัวอักษรหรือกราฟิกที่แสดงผลมีความคมชัด แม้ในสภาพแสงน้อย
2. มุมมองกว้าง (Wide Viewing Angle): หน้าจอมีมุมมองกว้างกว่า 160 องศา ทำให้ข้อมูลยังคงมองเห็นได้ชัดเจนแม้ผู้ใช้จะมองจากด้านข้าง ไม่จำเป็นต้องมองตรงหน้าจอ

3. การใช้พลังงานต่ำ: เนื่องจากไม่ต้องใช้ไฟพื้นหลัง (Backlight) เหมือนจอ LCD พลังงานจะถูกใช้เฉพาะพิกเซลที่สว่างเท่านั้น (เช่น การแสดงผลปกติใช้เพียง 0.08W) ทำให้เหมาะอย่างยิ่งสำหรับการใช้งานกับอุปกรณ์ที่ใช้แบตเตอรี่หรือชุดอุปกรณ์ IoT ที่มีการจำกัดพลังงาน

รูปแบบการต่อวงจรเซนเซอร์



ภาพที่ 11 รูปแบบการต่อเซนเซอร์

ชุดคำสั่งข้อมูล

ขั้นตอนแรกในการจัดการกับการทำงานของเซนเซอร์คือโปรแกรม Arduino IDE ซึ่งเป็นโปรแกรมสำหรับเขียนภาษา C เพื่อกำหนดคำสั่งเงื่อนไขในการทำงานให้กับเซนเซอร์ด้วยการอัพโหลดคำสั่งที่เขียนไปยังตัวเซนเซอร์ที่เชื่อมต่อกับคอมพิวเตอร์โดยใช้โค้ดแยกกันระหว่างบอร์ดNodeMCU esp8266 ที่ใช้เชื่อมต่อเครือข่ายส่งข้อมูลและจอแสดงผล และบอร์ด GPS Shield, Arduino UNO R3 ใช้อ่านค่า GPS ไลบรารีที่ใช้

เชื่อมต่อเครือข่าย & ส่งข้อมูล: **ESP8266WiFi.h, WiFiManager.h, PubSubClient.h**

อ่าน GPS & ประมวลผลข้อมูล: **TinyGPS++.h, SoftwareSerial.h**

แสดงผลบนจอ OLED: **Wire.h, U8g2lib.h**

แปลงข้อมูลเป็น JSON: **ArduinoJson.h**

โค้ดฝั่งบอร์ด NodeMCU esp8266 ใช้สำหรับการอ่านค่า serial จาก GPS Shield, Arduino UNO R3 และทำการแสดงผลบนหน้าจอ Oled 1.3 นิ้ว มีไลบารีเชื่อมต่อ WIFI และส่งข้อมูลผ่าน MQTT ไปยัง Node-RED

```
#include <Wire.h>
#include <U8g2lib.h>
#include <ESP8266WiFi.h>
#include <WiFiManager.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

// ===== OLED Config =====
#define OLED_SDA D2
#define OLED_SCL D1
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);

// ===== MQTT Config =====
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;
const char* mqtt_topic = "gpsdata";

// ===== Pin Config =====
#define LED_PIN D3 // กระพริบเมื่อส่ง MQTT

WiFiClient espClient;
PubSubClient client(espClient);

// ===== ตัวแปรเก็บค่าล่าสุด =====
int sat = 0;
float hdop = 0.0;
float speed = 0.0;
String dir = "-";

// ===== ตัวแปรหน้าจอ =====
int displayPage = 0;        // 0=เริ่มต้น, 1=WiFi Info, 2=GPS
unsigned long pageStart = 0;
```

ภาพที่ 12 ชุดคำสั่งเซนเซอร์

โค้ดฝั่ง จาก GPS Shield, Arduino UNO R3 สามารถรับข้อมูลจากโมดูล GPS ได้ โดย SoftwareSerial สร้างพอร์ตอนุกรมเสมือนสำหรับรับข้อความจาก GPS ส่วน TinyGPS++ ทำหน้าที่ถอดรหัสข้อมูล NMEA ให้เป็นข้อมูลเชิงตัวเลข เช่น พิกัด ความเร็ว และเวลา ซึ่งสามารถนำไปประมวลผลหรือส่งต่อไปยังระบบอื่น

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

static const int RXPin = 4, TXPin = 3;     // กำหนดขา RX, TX ของ SoftwareSerial
static const uint32_t GPSBaud = 9600;

TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);

double lastLat = 0.0;
double lastLon = 0.0;
const double movementThreshold = 5.0;

unsigned long stopStartTime = 0;
bool stopFlag = false;

void setup() {
  Serial.begin(9600);
  ss.begin(GPSBaud);
  Serial.println(F("TinyGPS++ JSON every 5s if moved > 5m or stopped > 15 sec"));
}

void loop() {
  // อ่านข้อมูล GPS จาก SoftwareSerial
  while (ss.available()) {
    gps.encode(ss.read());
  }
```

ภาพที่ 13 ชุดคำสั่งเซนเซอร์

3.2.3. การพัฒนาเซิร์ฟเวอร์และฐานข้อมูล

Node-RED ถูกนำมาใช้ในการจัดการการไหลของข้อมูล (Data Flow) เนื่องจากความสามารถในการจัดการโปรโตคอล IoT และการประมวลผลแบบ Visual Programming โดยมีขั้นตอนการทำงานหลัก 3 ส่วนดังนี้:

1. การรับและประมวลผลข้อมูลขาเข้า (Data Ingestion and Processing)

- รับค่า MQTT: ข้อมูลตำแหน่งและความเร็วจากชุดอุปกรณ์ GPS Tracking (รถเก็บขยะ) จะถูกรับเข้ามาที่โหนด gpsdata ซึ่งทำหน้าที่เป็น MQTT Input Node ข้อมูลที่รับเข้ามาจะเป็นรูปแบบ Raw Data
- แปลงข้อมูล: ข้อมูลจะถูกส่งต่อไปยังโหนด json เพื่อแปลงข้อมูลให้อยู่ในรูปแบบ JSON ที่สามารถนำไปประมวลผลต่อในระบบได้
- จัดเก็บ Context: ข้อมูลที่แปลงแล้วจะถูกส่งไปยังโหนด Save to flow context เพื่อจัดเก็บข้อมูลไว้ในหน่วยความจำชั่วคราวของ Flow ซึ่งช่วยให้ข้อมูลสามารถถูกเรียกใช้โดยส่วนอื่น ๆ ของระบบได้พร้อมกัน

2. การสร้างระบบแจ้งเตือน (Notification System)

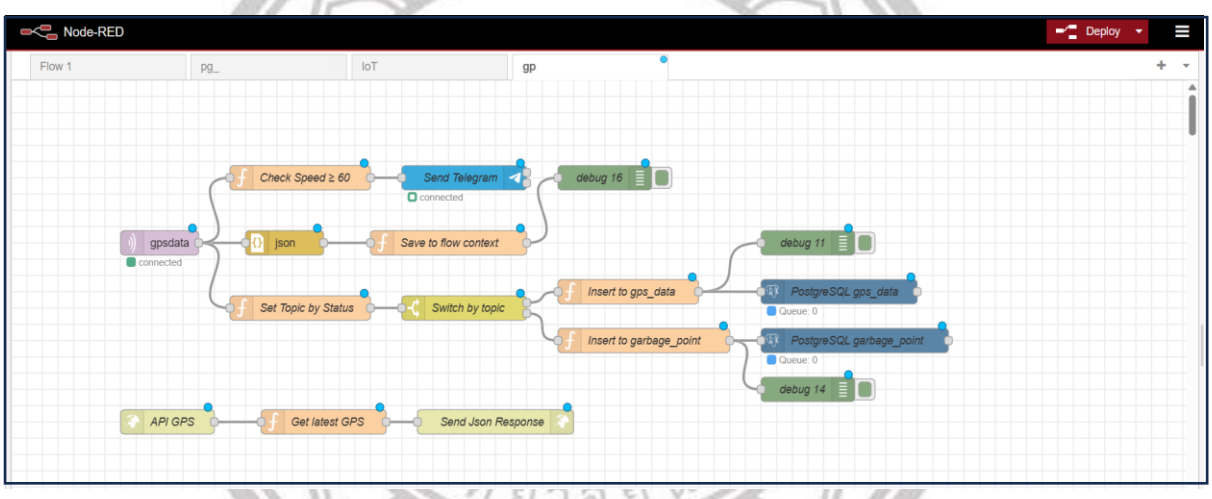
ตรวจสอบความเร็ว: ข้อมูลที่แปลงเป็น JSON แล้วจะถูกส่งเข้าสู่โหนด Check Speed > 60 โหนดนี้จะทำหน้าที่ตรวจสอบเงื่อนไขตามสมมติฐานของงานวิจัย : หากความเร็วเกิน 60 กม./ชม. ข้อมูลจะถูกส่งต่อไปยังโหนด Send Telegram ทันที ซึ่งทำหน้าที่ส่งข้อความแจ้งเตือนความเร็วเกินกำหนดไปยังเจ้าหน้าที่เทศบาลผ่านแอปพลิเคชัน Telegram

3. การจัดเก็บข้อมูลไปยังฐานข้อมูล (Database Storage)

- การเตรียมข้อมูล: ข้อมูลจะถูกส่งผ่านสองเส้นทางพร้อมกันเพื่อเข้าสู่ฐานข้อมูล:
- เส้นทางที่ 1: จัดเก็บข้อมูลเรียลไทม์ (gps_data): ข้อมูลจะถูกส่งไปยังโหนด Insert to gps_data ก่อนเข้าสู่ฐานข้อมูล PostgreSQL ในตาราง PostgreSQL_gps_data ซึ่งเป็นการจัดเก็บข้อมูลพิกัดและความเร็วทั้งหมด
- เส้นทางที่ 2: จัดเก็บข้อมูลวิเคราะห์ (garbage_point): ข้อมูลจะถูกส่งไปยังโหนด Set Topic by Status และโหนด Switch by topic เพื่อวิเคราะห์ว่าข้อมูลนั้นเข้าข่ายเป็น จุดจอดเก็บขยะ หรือไม่ หากเข้าข่าย ข้อมูลจะถูกส่งต่อไปยังโหนด Insert to garbage_point ก่อนเข้าสู่ตาราง PostgreSQL_garbage_point

Node-RED ยังถูกใช้ในการสร้าง **API Endpoint** เพื่อให้หน้าเว็บ WebGIS สามารถเรียกดูข้อมูลแบบเรียลไทม์ได้

- โหนด API: โหนด API GPS ทำหน้าที่เป็น HTTP In Node ซึ่งเป็นจุดเข้าถึง API ที่หน้าเว็บจะเรียกใช้
- ดึงข้อมูลล่าสุด: เมื่อมีการเรียกใช้ API ข้อมูลจะถูกส่งต่อไปยังโหนด Get latest GPS ซึ่งทำหน้าที่ดึงข้อมูลตำแหน่งล่าสุดจากฐานข้อมูลหรือ Context
- ส่งข้อมูลออก: ข้อมูลที่ดึงมาได้จะถูกส่งผ่านโหนด Send Json Response ซึ่งทำหน้าที่เป็น HTTP Response Node เพื่อส่งข้อมูลพิกัดล่าสุดในรูปแบบ **JSON** กลับไปยังหน้าเว็บ WebGIS ให้แสดงผลตำแหน่งรถแบบเรียลไทม์



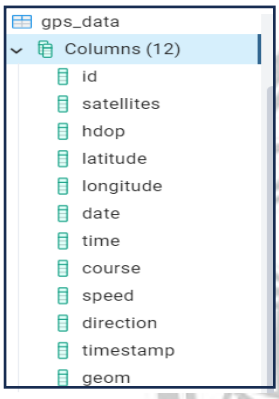
ภาพที่ 14 Flow การทำงานของ Node-RED

ฐานข้อมูลของระบบนี้ถูกออกแบบให้ใช้ PostgreSQL ร่วมกับส่วนขยาย PostGIS เพื่อให้สามารถจัดเก็บ, จัดการ, และวิเคราะห์ข้อมูลพิกัดทางภูมิศาสตร์ได้อย่างมีประสิทธิภาพ

โครงสร้างตารางข้อมูล gps_data

- id : รหัสอ้างอิงที่ไม่ซ้ำกัน
- satellites : จำนวนดาวเทียมที่รับสัญญาณ
- hdop : ค่าความคลาดเคลื่อนในแนวราบใช้ประเมินความแม่นยำของพิกัด
- latitude : ค่าละติจูดของตำแหน่ง
- longitude : ค่าลองจิจูดของตำแหน่ง
- date : วันที่บันทึกข้อมูล
- time : เวลาที่บันทึกข้อมูล

- course : องศาของทิศทางการเคลื่อนที่
- speed : ความเร็วในการเคลื่อนที่ของรถ ในหน่วย กิโลเมตรต่อชั่วโมง (km/hr)
- direction : ทิศทางการเคลื่อนที่ (เช่น N, NE, S)
- timestamp : วันที่และเวลาที่ข้อมูลถูกบันทึกเข้าสู่ฐานข้อมูลโดยอัตโนมัติ
- geom : ฟิลด์เชิงพื้นที่ ที่เก็บพิกัด latitude และ longitude ในรูปแบบ Point ตามระบบพิกัด EPSG:4326 (WGS 84



	integer	satellites integer	hdop real	latitude double precision	longitude double precision	date character varying (10)	time character varying (8)	course real	speed real	direction character vary
1988	1990	9	1	16.744375	100.20438	31/07/2025	15:22:54	18.38	23.19	NNE
1989	1991	9	1	16.744427	100.2044	31/07/2025	15:22:55	15.24	23.02	NNE
1990	1992	9	1	16.744473	100.20441	31/07/2025	15:22:56	16.96	20.48	NNE
1991	1993	9	1	16.744547	100.20444	31/07/2025	15:22:58	18.51	14.78	NNE
1992	1994	9	1	16.74464	100.20448	31/07/2025	15:23:00	19.02	19.76	NNE
1993	1995	9	1	16.744694	100.2045	31/07/2025	15:23:01	19.43	21.96	NNE
1994	1996	9	1	16.744747	100.20451	31/07/2025	15:23:02	18.87	22.34	NNE
1995	1997	9	1	16.744799	100.20454	31/07/2025	15:23:03	18.47	22.06	NNE
1996	1998	9	1	16.744856	100.20455	31/07/2025	15:23:04	19.03	22.35	NNE
1997	1999	9	1	16.744917	100.20457	31/07/2025	15:23:05	17.88	23.28	NNE
1998	2000	9	1	16.74498	100.20459	31/07/2025	15:23:06	17.54	24.37	NNE
1999	2001	9	1	16.745041	100.20461	31/07/2025	15:23:07	15.3	25.06	NNE
2000	2002	9	1	16.745098	100.20462	31/07/2025	15:23:08	14.65	23.91	NNE

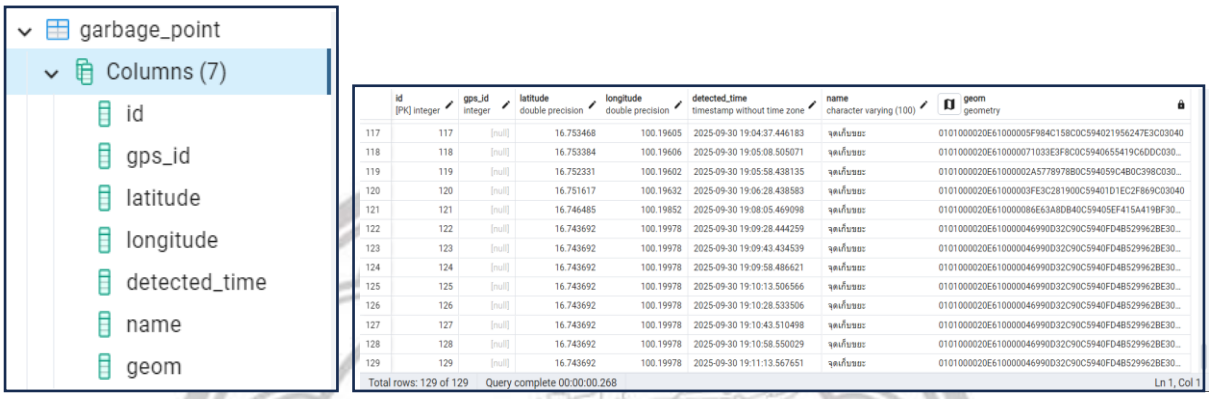
ภาพที่ 15 ฐานข้อมูล gps_data

นอกเหนือจากการจัดเก็บข้อมูลติดตามแบบเรียลไทม์ในตาราง gps_data แล้ว ระบบยังใช้ตาราง garbage_point เพื่อจัดเก็บข้อมูลจุดที่รถหยุดนิ่งและคาดว่าเป็นจุดที่มีการปฏิบัติงานเก็บขยะ ตารางนี้เป็นหัวใจสำคัญในการวิเคราะห์ความสัมพันธ์ระหว่าง เวลาจอด กับ ปริมาณขยะ ซึ่งเป็นไปตามสมมติฐานของงานวิจัย

โครงสร้างตารางข้อมูล garbage_point

- Id : รหัสอ้างอิงที่ไม่ซ้ำกันสำหรับแต่ละจุดเก็บขยะที่ถูกตรวจจับ
- gps_id : รายการข้อมูลพิกัด (ID) ในตาราง gps_data ที่เป็นจุดเริ่มต้นของการจอดเก็บขยะ โดยมีการกำหนดให้ ลบข้อมูลนี้ออกด้วยหากรายการใน gps_data ถูกลบ (ON DELETE CASCADE)
- latitude : ค่าละติจูดของจุดเก็บขยะ
- longitude : ค่าลองจิจูดของจุดเก็บขยะที่ตรวจจับได้
- detected_time : วันที่และเวลาที่ระบบตรวจพบจุดจอด/จุดเก็บขยะ

- name : ชื่อหรือคำอธิบายของจุด (ค่าเริ่มต้นคือ 'จุดเก็บขยะ')
- geom : ฟิลด์เชิงพื้นที่ ที่เก็บพิกัดในรูปแบบ Point ตามระบบพิกัด EPSG:4326 (WGS 84) เพื่อให้ PostGIS สามารถวิเคราะห์และแสดงผลบนแผนที่ได้อย่างมีประสิทธิภาพ



ภาพที่ 16 ฐานข้อมูล garbage_point

การออกแบบโครงสร้างตาราในลักษณะนี้ ช่วยให้สามารถจัดเก็บข้อมูลที่ส่งมายังฐานข้อมูลได้อย่างมีระเบียบและสามารถติดตามความคืบหน้าของการแก้ไขปัญหาได้ การจัดการข้อมูลเช่นนี้ยังช่วยให้การแจ้งนำไปใช้งานโดยเรียกจากฐานข้อมูลได้โดยตรง

```sql
CREATE TABLE gps_data (
  id SERIAL PRIMARY KEY,
  satellites INT,
  hdop REAL,
  latitude DOUBLE PRECISION,
  longitude DOUBLE PRECISION,
  date VARCHAR(10),
  time VARCHAR(8),
  course REAL,
  speed REAL,
  direction VARCHAR(3),
  timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  geom geometry(Point, 4326)
);
```

```sql
CREATE TABLE garbage_point (
  id SERIAL PRIMARY KEY,
  gps_id INT REFERENCES gps_data(id) ON DELETE CASCADE,
  latitude DOUBLE PRECISION,
  longitude DOUBLE PRECISION,
  detected_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  name VARCHAR(100) DEFAULT 'จุดเก็บขยะ',
  geom geometry(Point, 4326)
);
```

ภาพที่ 17 สร้างฐานข้อมูล gps และฐานข้อมูลจุดเก็บขยะ

3.2.4. พัฒนา WebGIS

ขั้นตอนนี้เป็นการพัฒนาเว็บไซต์แสดงผลระบบสารสนเทศภูมิศาสตร์บนเว็บ (WebGIS) เพื่อเป็นส่วนติดต่อผู้ใช้หลัก (User Interface) ให้แก่เจ้าหน้าที่เทศบาลในการติดตาม ประเมินผล และวิเคราะห์ข้อมูลเชิงพื้นที่ของรถเก็บขยะแบบเรียลไทม์ โดยใช้ภาษา HTML, CSS, JavaScript และ PHP ในการพัฒนาระบบ และใช้ไลบรารี Leaflet.js สำหรับการแสดงผลแผนที่

โค้ดในส่วนของ get_gps_data.php รับวันที่จากผู้ใช้แล้วเชื่อมต่อฐานข้อมูล GPS เพื่อเตรียมดึงข้อมูลตำแหน่ง (ในรูปแบบ GeoJSON) ถ้าไม่มีวันที่ส่งมา ระบบจะส่งข้อมูลเปล่ากลับแทน

```php
<?php
header('Content-Type: application/json');

$host = "host=localhost";
$port = "port=5432";
$dbname = "dbname=IoT_gps";
$credentials = "user=postgres password=postgres";

$db = pg_connect("$host $port $dbname $credentials");

$date = $_GET['date'] ?? null;
$date = trim($date);

if (!$date) {
    echo json_encode([
        'line' => ['type' => 'FeatureCollection', 'features' => []],
        'points' => ['type' => 'FeatureCollection', 'features' => []]
    ]);
    exit;
}
```

ภาพที่ 18 โค้ดในส่วนของ เชื่อมฐานข้อมูล get_gps_data.php

โค้ดในส่วนนี้ดึงข้อมูลตามวันที่เลือกจากฐานข้อมูล ส่วนแรก line ดึงเส้นทางรถที่ไปมาเรียงลำดับตามเวลา ส่วนต่อมา points ดึงข้อมูลจุดที่ gps เก็บลงฐานข้อมูล และต่อมา garbage ดึงจุดตำแหน่งขยะ

```php
$sql_line = "SELECT ST_AsGeoJSON(ST_Transform(ST_MakeLine(geom ORDER BY \"timestamp\"), 4326)) AS geojson
             FROM gps_data
             WHERE DATE(\"timestamp\") = '$date_escaped'";

$sql_points = "SELECT *, ST_AsGeoJSON(ST_Transform(geom,4326)) as geojson
               FROM gps_data
               WHERE DATE(\"timestamp\") = '$date_escaped'";

$sql_garbage = "SELECT id, gps_id, latitude, longitude, detected_time, name,
                       ST_AsGeoJSON(ST_Transform(geom,4326)) as geojson
                FROM garbage_point
                WHERE DATE(detected_time) = '$date_escaped'";

$result_line = pg_query($db, $sql_line);
$result_points = pg_query($db, $sql_points);
$result_garbage = pg_query($db, $sql_garbage);
```

ภาพที่ 19 โค้ดในส่วนของการดึงข้อมูล get_gps_data.php

โค้ดในส่วนของ “ตัวแปลงผลลัพธ์จากฐานข้อมูล GPS และ Garbage เป็น GeoJSON” ที่พร้อมส่งออก ให้หน้าเว็บหรือแผนที่ใช้แสดงเส้นทางรถ จุดรถ และจุดขยะได้ทันที

```php
$result_line = pg_query($db, $sql_line);
$result_points = pg_query($db, $sql_points);
$result_garbage = pg_query($db, $sql_garbage);

$line_geojson = ['type' => 'FeatureCollection', 'features' => []];
$points_geojson = ['type' => 'FeatureCollection', 'features' => []];
$garbage_geojson = ['type' => 'FeatureCollection', 'features' => []];

if ($result_line && $row_line = pg_fetch_assoc($result_line)) {
    if ($row_line['geojson']) {
        $line_geojson['features'][] = [
            'type' => 'Feature',
            'geometry' => json_decode($row_line['geojson'], true),
            'properties' => ['description' => "เส้นทางรถเก็บขยะวันที่ $date"]
        ];
    }
}

if ($result_points) {
    while ($row = pg_fetch_assoc($result_points)) {

        $timeFormatted = date('Y-m-d H:i:s', strtotime($row['timestamp']));
        $points_geojson['features'][] = [
            'type' => 'Feature',
            'geometry' => json_decode($row['geojson'], true),
            'properties' => [
                'time' => $timeFormatted,
                'lat' => $row['latitude'] ?? null,
                'lon' => $row['longitude'] ?? null,
                'speed' => $row['speed'] ?? null,
                'satellites' => $row['satellites'] ?? null,
                'course' => $row['course'] ?? 0
            ]
        ];
    }
}

if ($result_garbage) {
    while ($row = pg_fetch_assoc($result_garbage)) {
    $geometry = json_decode($row['geojson'], true);
    $coordinates = $geometry['coordinates'];

    $garbage_geojson['features'][] = [
        'type' => 'Feature',
        'geometry' => $geometry,
        'properties' => [
            'id' => $row['id'],
            'gps_id' => $row['gps_id'],
            'latitude' => $coordinates[1],
            'longitude' => $coordinates[0],
            'detected_time' => $row['detected_time'],
            'name' => $row['name']
            ]
        ];
    }
}
```


ภาพที่ 20 โค้ดในส่วนของการแปลงผลลัพธ์จากฐานข้อมูล get_gps_data.php

ส่วนของสรุปข้อมูล PHP สำหรับดึงข้อมูล GPS และขยะจากฐานข้อมูล PostgreSQL/PostGIS แล้วรวมเป็นสรุปสถิติ ของแต่ละวัน พร้อมส่งออกเป็น JSON

```php
// 1. ระยะทางรวม (หน่วย km)
$sql_distance = "SELECT
    ST_Length(ST_Transform(ST_MakeLine(geom ORDER BY timestamp), 32647)) / 1000 AS distance_km
    FROM gps_data
    WHERE DATE(timestamp) = '$date_escaped'";
$result_distance = pg_query($db, $sql_distance);
$row_distance = pg_fetch_assoc($result_distance);
$distance_km = round($row_distance['distance_km'] ?? 0, 2);

// 2. เวลาเดินรถรวม
$sql_duration = "SELECT
    to_char(MAX(timestamp) - MIN(timestamp), 'HH24:MI:SS') AS total_duration
    FROM gps_data
    WHERE DATE(timestamp) = '$date_escaped'";
$result_duration = pg_query($db, $sql_duration);
$row_duration = pg_fetch_assoc($result_duration);
$total_duration = $row_duration['total_duration'] ?? "00:00:00";

// 3. Overspeed %
$sql_overspeed = "SELECT
    (COUNT(*) FILTER (WHERE speed > 60) * 100.0 / COUNT(*)) AS overspeed_percent
    FROM gps_data
    WHERE DATE(timestamp) = '$date_escaped'";
$result_overspeed = pg_query($db, $sql_overspeed);
$row_overspeed = pg_fetch_assoc($result_overspeed);
$overspeed_percent = round($row_overspeed['overspeed_percent'] ?? 0, 1);

// 4. ความเร็วเฉลี่ย
$sql_avg_speed = "SELECT AVG(speed) AS avg_speed
    FROM gps_data
    WHERE DATE(timestamp) = '$date_escaped'";
$result_avg_speed = pg_query($db, $sql_avg_speed);
$row_avg_speed = pg_fetch_assoc($result_avg_speed);
$avg_speed = round($row_avg_speed['avg_speed'] ?? 0, 1);

// 5. จำนวนจุดเก็บขยะ
$sql_garbage_count = "SELECT COUNT(*) AS garbage_count
    FROM garbage_point
    WHERE DATE(detected_time) = '$date_escaped'";
$result_garbage_count = pg_query($db, $sql_garbage_count);
$row_garbage_count = pg_fetch_assoc($result_garbage_count);
$garbage_count = (int) ($row_garbage_count['garbage_count'] ?? 0);
```

ภาพที่ 21 โค้ดในส่วนของการสรุปข้อมูลของ get_gps_data.php

โค้ดในส่วนของ **Script.js** ส่วนที่สื่อสารระหว่างข้อมูลจาก PHP/PostgreSQL กับ Leaflet Map ส่วนนี้เป็นการสร้าง layer สำหรับแผนที่ Leaflet และสำหรับจุด GPS

```
var map = L.map("map").setView([16.7476, 100.1937], 15);

var osm = L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
  maxZoom: 20,
  attribution: "© OpenStreetMap contributors",
});

var googleSat = L.tileLayer(
  "http://{s}.google.com/vt?lyrs=s&x={x}&y={y}&z={z}",
  {
    maxZoom: 20,
    subdomains: ["mt0", "mt1", "mt2", "mt3"],
  }
);

googleSat.addTo(map);

var pointLayer = L.geoJSON(null, {
  onEachFeature: function (feature, layer) {
    let p = feature.properties;
    let popup = `
      <b>เวลา:</b> ${p.date} ${p.time}<br/>
      <b>ดาวเทียม:</b> ${p.satellites}<br/>
      <b>ความเร็ว:</b> ${p.speed} km/h<br/>
      <b>พิกัด:</b> ${p.lat}, ${p.lon}`;
    layer.bindPopup(popup);
  },
  pointToLayer: function (feature, latlng) {
    return L.circleMarker(latlng, {
      radius: 5,
      color: "blue",
      fillOpacity: 0.7,
    });
  },
});
```

ภาพที่ 22 การสร้าง layer มาแสดงบนแผนที่

ส่วนนี้สร้าง layer เส้นทางและจุดเก็บขยะมาแสดงข้อมูล

```
var lineLayer = L.geoJSON(null, {
  style: {
    color: "red",
    weight: 4,
    opacity: 0.8,
  },
  onEachFeature: function (feature, layer) {
    let desc = feature.properties.description || "เส้นทาง";
    layer.bindPopup(`<b>${desc}</b>`);
  },
});


var garbageLayer = L.geoJSON(null, {
  pointToLayer: function (feature, latlng) {
    return L.marker(latlng, {
      icon: L.icon({
        iconUrl: "trashbin.png",
        iconSize: [25, 25],
        iconAnchor: [12, 25],
      }),
    });
  },
  onEachFeature: function (feature, layer) {
    const p = feature.properties;
```

ภาพที่ 23 การสร้าง layer มาแสดงบนแผนที่

โค้ดในส่วนนี้ตรวจสอบว่าจุดขยะพิกัดซ้ำกันหรือไม่ ถ้ามีคำนวณ เวลาจอดรวม จากจุดแรกถึงจุดท้าย และแสดงข้อมูลทั้งหมดใน popup ของ Marker

```
let durationText = "";
if (samePoints.length > 1) {

  samePoints.sort(
    (a, b) =>
      parseTimeToMs(a.properties.detected_time) -
      parseTimeToMs(b.properties.detected_time)
  );

  const start = parseTimeToMs(samePoints[0].properties.detected_time);
  const end = parseTimeToMs(
    samePoints[samePoints.length - 1].properties.detected_time
  );
  const duration = end - start;

  durationText = `<br/><b>เวลาจอดรวม:</b> ${formatDuration(duration)}`;
} else {
  // ถ้าเป็นจุดเดียว → แสดงว่า "0 วินาที" (หรือจอดน้อยกว่า 1 วินาที)
  durationText = `<br/><b>เวลาจอด:</b> 0 วินาที`;
}

let timeStr = p.detected_time.split(".")[0];

layer.bindPopup(`
  <b>ชื่อ:</b> ${p.name}<br/>
  <b>เวลา:</b> ${timeStr}<br/>
  <b>พิกัด:</b> ${p.latitude}, ${p.longitude}
  <b>จำนวนครั้งที่จอด:</b> ${stopCount}
  ${durationText}
`);
```

ภาพที่ 24 ตรวจสอบจุดซ้ำและคำนวณเป็นเวลาจอด

ส่วนนี้โหลดไฟล์สถานที่ต่างๆสำหรับตำบลท่าทองโดยดึงจาก thathong.geojson และแสดงไอคอนพร้อม popup ชื่อสถานที่ เมื่อคลิกที่ Marker สีแดงจะแสดงข้อมูลสถานที่บน popup

```
var thathongLayer = L.geoJSON(null);
var iconLocation = L.icon({
  iconUrl: "Redmark.png",
  iconSize: [30, 30],
  iconAnchor: [15, 30],
  popupAnchor: [0, -30],
});


fetch("thathong.geojson")
  .then((res) => res.json())
  .then((data) => {
    thathongLayer.addData(data);
    thathongLayer.eachLayer(function (layer) {
      var name = layer.feature?.properties?.name || "ไม่มีชื่อ";
      layer.bindPopup("<b>สถานที่:</b> " + name);
      if (layer.setIcon) {
        layer.setIcon(iconLocation);
      }
    });
    map.addLayer(thathongLayer);
```

ภาพที่ 25 เรียกข้อมูลสถานที่ตำบลท่าทอง

ส่วนนี้เป็นการติดตามแบบเรียลไทม์ของรถเก็บขยะ โดยจะแสดงข้อมูลของรถเก็บขยะ พร้อมแปลงเวลาใช้กับประเทศไทยโดยดึงข้อมูล Api จาก Node-RED ส่งมาในส่วนของ 127.0.0.1.1880/api/gps/data

```
// Live Tracking
let liveTrackingInterval;
let isLiveTrackingActive = false;
let vehicleMarker;

// ฟังก์ชันแปลงเวลา UTC -> เวลาไทย (UTC+7)
function convertToThaiTime(dateStr, timeStr) {
  const [day, month, year] = dateStr.split("/").map(Number);
  const [hh, mm, ss] = timeStr.split(":").map(Number);

  let localDate = new Date(year, month - 1, day, hh, mm, ss);

  localDate.setHours(localDate.getHours() + 7);

  const hh2 = String(localDate.getHours()).padStart(2, "0");
  const mm2 = String(localDate.getMinutes()).padStart(2, "0");
  const ss2 = String(localDate.getSeconds()).padStart(2, "0");

  return `${hh2}:${mm2}:${ss2}`;
}
function toggleLiveTracking() {
  const button = document.getElementById("liveTrackingBtn");

  if (isLiveTrackingActive) {
    clearInterval(liveTrackingInterval);
    isLiveTrackingActive = false;

    if (vehicleMarker) {
      map.removeLayer(vehicleMarker);
      vehicleMarker = null;
    }

    button.innerText = "เริ่มติดตามสด";
    button.style.backgroundColor = "■#ffdd57";
    button.style.color = "■#4b3678";
    showMessageBox("หยุดการติดตามสดแล้ว");
  } else {
    // --- เคลียร์เลเยอร์ทั้งหมดก่อนเริ่มติดตามสด ---
    pointLayer.clearLayers();
    lineLayer.clearLayers();
    garbageLayer.clearLayers();

    liveTrackingInterval = setInterval(() => {
      fetch("http://127.0.0.1:1880/api/gpsdata")
        .then((res) => res.json())
        .then((data) => {
          console.log("Latest GPS data:", data);
```

ภาพที่ 26 โค้ดสำหรับติดตามสด

แสดงสถานะกำลังเก็บขยะหรือรถขับปกติ สีแดง หมายถึง รถจอดเก็บขยะ สีเขียว หมายถึงรถวิ่งปกติ

```
    const lat = parseFloat(data.lat);
    const lon = parseFloat(data.lon);
    const course = parseFloat(data.course) || 0;

    // เลือกสีตาม status
    const statusColor = data.status === "จุดเก็บขยะ" ? "red" : "green";
    const thaiTime = convertToThaiTime(data.date, data.time);
    const popupContent = `
    <b>สถานะ:</b>
    <span style="
      display:inline-block;
      width:12px;
      height:12px;
      border-radius:50%;
      background-color:${statusColor};
      margin-right:5px;
    "></span>
```


ภาพที่ 27 โค้ดสำหรับแสดงสถานะของรถเก็บขยะ

ในส่วนนี้เป็นการปรับใช้ไอคอนให้แสดงรถหันหัวไปตามทิศทางจริง อัปเดททิศทางเพื่อดูติดตามรถแบบเรียลไทม์

```
if (!vehicleMarker) {
  const vehicleIcon = L.icon({
    iconUrl: "truck_topview.png", // ใช้ไอคอน top view
    iconSize: [50, 50],
    iconAnchor: [25, 25], // หมุนรอบกลาง icon
  });

  vehicleMarker = L.marker([lat, lon], {
    icon: vehicleIcon,
    rotationAngle: course, // หมุนตาม course
    rotationOrigin: "center center",
  })
    .addTo(map)
    .bindPopup(popupContent)
    .openPopup();
} else {
  vehicleMarker.setLatLng([lat, lon]);
  vehicleMarker.setRotationAngle(course); // อัปเดตทิศทาง
  vehicleMarker.setPopupContent(popupContent);
}

map.panTo([lat, lon]);
```

ภาพที่ 28 โค้ดไอคอนรถแสดงหันตามทิศทาง

ส่วนของ การเล่นเส้นทางย้อนหลัง โดยสามารถตรวจสอบพิกัดจากข้อมูลจริง แล้วอัปเดตตำแหน่งรถทีละจุดทุก 1 วินาทีเพื่อให้ผู้ใช้เห็นการเคลื่อนที่ของรถบนแผนที่แบบย้อนหลัง

```
playbackData = pointLayer.toGeoJSON().features;
currentPlaybackIndex = 0;

playbackInterval = setInterval(() => {
  if (currentPlaybackIndex < playbackData.length) {
    const point = playbackData[currentPlaybackIndex];
    const coords = point.geometry.coordinates;
    const latlng = [coords[1], coords[0]];
    const course = parseFloat(point.properties.course) || 0;

    if (!playbackMarker) {
      playbackMarker = L.marker(latlng, {
        icon: L.icon({
          iconUrl: "truck_topview.png", // ใช้ top view icon
          iconSize: [50, 50],
          iconAnchor: [25, 25],
        }),
        rotationAngle: course, // หมุนตามค่า course
        rotationOrigin: "center center",
      })
        .addTo(map)
        .bindPopup(
          `เวลา: ${point.properties.time}<br>
       ความเร็ว: ${point.properties.speed || 0} km/h<br>
       Course: ${course}°`
        )
        .openPopup();

    } else {
      playbackMarker
        .setLatLng(latlng)
        .setRotationAngle(course)
        .bindPopup(
          `เวลา: ${point.properties.time}<br>
           ความเร็ว: ${point.properties.speed || 0} km/h<br>
           Course: ${course}°`
        );
    }
```

ภาพที่ 29 โค้ดสำหรับดูการเล่นเส้นทางย้อนหลัง

ในส่วนของการสร้างแผนที่ความหนาแน่นของจุดเก็บขยะโดยดึงมาจากฐานข้อมูลผ่าน API โหลดข้อมูล กรองพิกัด สร้าง heatmap และ marker แสดงบนแผนที่ leaflet

```
let heatLayer = null;
let garbageMarkers = []; // เก็บ marker

function renderGarbageHeatmap() {
  const date = document.getElementById("dateInput").value;
  if (!date) {
    showMessageBox("กรุณาเลือกวันที่");
    return;
  }

  function clearHeatmap(showMsg = true) {
    if (heatLayer) {
      map.removeLayer(heatLayer);
      heatLayer = null;
      document.getElementById("totalPoints").innerText = 0;

      if (showMsg) {
        showMessageBox("เคลียร์ Heatmap เรียบร้อยแล้ว");
      }
    }
  }

  fetch(`http://localhost/GPS_Tracking/get_gps_data.php?date=${date}`)
    .then((res) => res.json())
    .then((data) => {
      const garbageData = data.garbage_point;
      if (!garbageData?.features?.length) {
        showMessageBox("ไม่พบข้อมูลจุดเก็บขยะ");
        return;
      }

      const validFeatures = garbageData.features.filter(
        (f) => f.geometry?.coordinates
      );
      if (!validFeatures.length) {
        showMessageBox("ไม่พบจุดเก็บขยะที่มีพิกัด");
        return;
      }

      // --- เคลียร์ heatmap และ markers เดิม ---
      if (heatLayer) {
        map.removeLayer(heatLayer);
        heatLayer = null;
      }
      markerLayer.clearLayers(); // <<< ลบ marker เดิมทั้งหมด

      // --- Heatmap ---
      heatLayer = L.heatLayer(
        validFeatures.map((f) => [
          f.geometry.coordinates[1],
          f.geometry.coordinates[0],
          1,
        ]),
        { radius: 25, blur: 15, maxZoom: 17 }
      ).addTo(map);
```

ภาพที่ 30 โค้ดสำหรับสร้างแผนที่ความหนาแน่นของจุดเก็บขยะ

ส่วนนี้เป็นการสร้างกราฟ ความเร็ว/วันและเวลา ของจุดที่ gps ส่งมายังฐานข้อมูลและเรียกออกมาแสดงโดยจะมีกราฟที่บอกความเร็วปกติจะเป็นจุดสีเขียวแต่เมื่อเกิน 60 km/h จุดนั้นจะแสดงเป็นสีแดงเพื่อแยกง่ายต่อการหาจุดไหนที่ขับรถเร็วเกินกำหนด

```
let chartData = { labels: [], speeds: [] };
let miniChart = null;
let largeChart = null;

function renderSpeedChart(features) {
  const labels = features.map((f) => f.properties.time);
  const speeds = features.map((f) => parseFloat(f.properties.speed) || 0);

  chartData.labels = labels;
  chartData.speeds = speeds;

  const over60 = speeds.map((s) => (s > 60 ? s : null));

  const ctx = document.getElementById("speedChart").getContext("2d");
  if (miniChart) miniChart.destroy();

  const over60Count = speeds.filter((s) => s > 60).length;

  miniChart = new Chart(ctx, {
    type: "line",
    data: {
      labels,
      datasets: [
        {
          label: "Speed (km/hr)", data: speeds, borderColor: "blue", backgroundColor: "rgba(0,0,255,0.1)", tension: 0.3, pointRadius: 2,
        },
        {
          label: "Speed > 60", data: over60, borderColor: "red", backgroundColor: "rgba(255,0,0,0.1)", tension: 0.3, pointRadius: 3,
        },
      ],
    },
    options: {
      responsive: true,
      plugins: {
        title: {
          display: true,
          text: `Overspeed Points: ${over60Count}`, // ✅ โชว์จำนวน
          color: "red",
          font: { size: 14, weight: "bold" },
        },
```

ภาพที่ 31 โค้ดสำหรับสร้างกราฟความเร็ว/วันและเวลา

ส่วนนี้เป็นโค้ดที่คุณส่งมานี้เป็น PHP สำหรับดึงข้อมูล GPS และขยะจากฐานข้อมูล PostgreSQL/PostGIS แล้วรวมเป็น สรุปสถิติ (Summary) ของแต่ละวัน พร้อมส่งออกเป็น JSON เพื่อใช้ในหน้าเว็บหรือ WebGIS โดยแสดงเป็นกล่องข้อความเมื่อกดปุ่มสามารถเปิด-ปิดเพื่อดูข้อมูลสรุปที่เชื่อมกับ get_gps_data.php แล้วเรียกมาแสดงบนหน้าเว็บ

```
// Summary Data  //
function showDailySummary() {
  console.log("showDailySummary called");
  const date = document.getElementById("dateInput").value;
  console.log("Selected date:", date);
  if (!date) {
    showMessageBox("กรุณาเลือกวันที่ก่อน");
    return;
  }

  const box = document.getElementById("summaryBox");
  if (!box) return;

  // เรียกข้อมูลจาก PHP
  fetch(`http://localhost/GPS_Tracking/get_gps_data.php?date=${date}`)
    .then((res) => res.json())
    .then((data) => {
      console.log("Data received:", data);
      if (!data || !data.summary) {
        showMessageBox("ไม่พบข้อมูลสรุปสำหรับวันที่เลือก");
        return;
      }

      const s = data.summary;

      box.classList.remove("hidden");
      box.innerHTML = `
        <b>🗓️ สรุปผลการเดินรถวันที่ ${s.date}</b><br>
        🚗 <b>ระยะทางรวม:</b> ${s.distance_km.toFixed(2)} km<br>
        ⏱️ <b>เวลาเดินรถ:</b> ${s.total_duration}<br>
        🚨 <b>Overspeed:</b> ${s.overspeed_percent.toFixed(1)}%<br>
        ⚙️ <b>ความเร็วเฉลี่ย:</b> ${s.avg_speed.toFixed(1)} km/h<br>
        🗑️ <b>จำนวนจุดเก็บขยะ:</b> ${s.garbage_count} จุด<br>
        <button onclick="hideDailySummary()" class="mt-2 w-full bg-red-500 hover:bg-red-600 text-white font-bold py-1 px-2 rounded-md">ปิด</button>
      `;
    })
    .catch((err) => {
      console.error(err);
      showMessageBox("เกิดข้อผิดพลาดในการดึงข้อมูลสรุป");
    });
}

function hideDailySummary() {
  const box = document.getElementById("summaryBox");
  box.classList.add("hidden");
}
```

ภาพที่ 32 โค้ดสำหรับสรุปข้อมูลเพื่อโชว์บนหน้าเว็บ

3.2.5. ทดสอบระบบ GPS Tracking

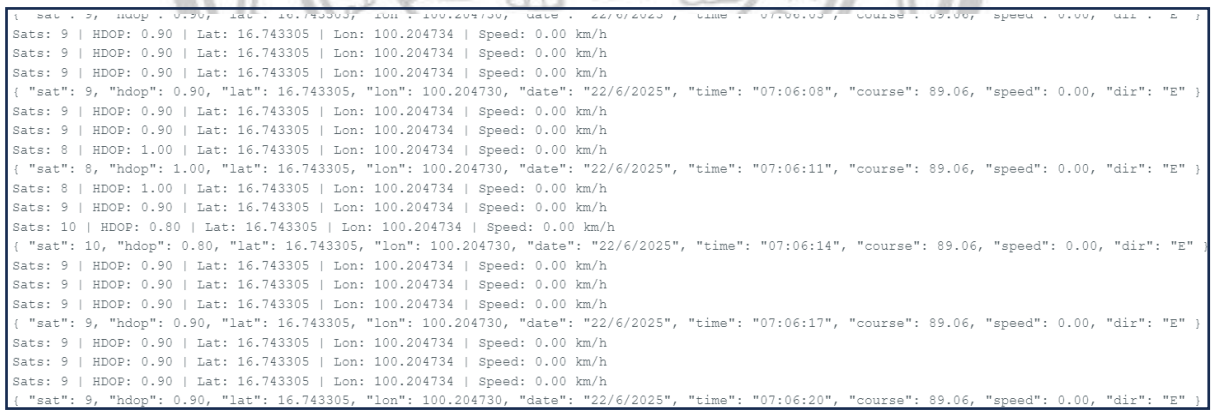
ในการทดสอบระบบ GPS Tracking นี้ มีวัตถุประสงค์เพื่อตรวจสอบการทำงานของอุปกรณ์ GPS และการส่งข้อมูลไปยังระบบ backend รวมถึงการแสดงผลบน WebGIS โดยเน้นการประเมินความถูกต้อง ความครบถ้วน และความเสถียรของข้อมูลทั้งในระดับอุปกรณ์และซอฟต์แวร์

ขั้นตอนการทดสอบระบบ

1. อ่านค่า serial บน Arduino IDE เพื่อดูเซนเซอร์สามารถรับค่าดาวเทียมได้หรือไม่
2. ตรวจสอบการส่งข้อมูลผ่าน MQTT ไปยัง Node-RED เพื่อประมวลผลต่อ
3. ทดสอบการส่งข้อมูลเข้าไปเก็บยังฐานข้อมูลผ่าน Node-RED โดยใช้ PostgeSQL
4. ตรวจสอบการแสดงผลบน WebGIS
5. ทดสอบติดตั้งบนรถยนต์

1. อ่านค่า serial บน Arduino IDE เพื่อดูเซนเซอร์สามารถรับค่าดาวเทียมได้หรือไม่

ผลทดสอบเซนเซอร์สามารถอ่านค่าได้และรับสัญญาณได้ดี

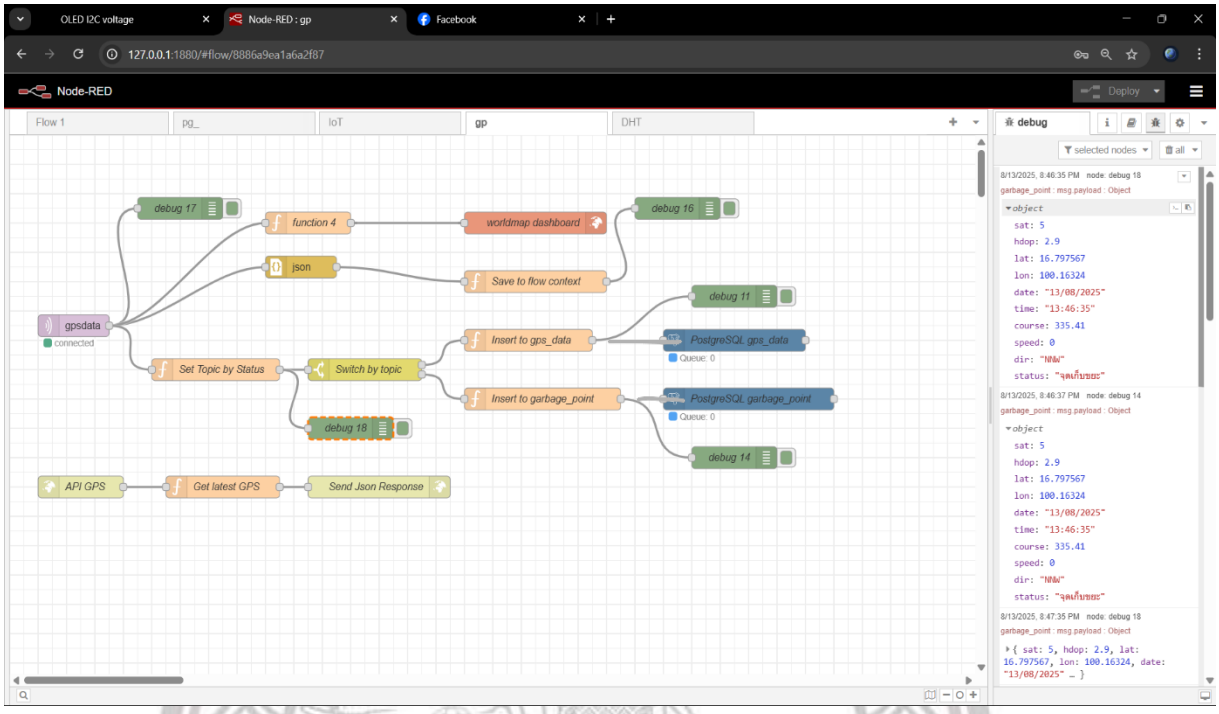


ภาพที่ 33 serial อ่านค่า GPS

2. ตรวจสอบการส่งข้อมูลผ่าน MQTT โดไปยัง Node-RED เพื่อประมวลผลต่อ

การทดสอบระบบ GPS Tracking ขั้นตอนแรกคือการตรวจสอบการส่งข้อมูลจากอุปกรณ์ไปยังเซิร์ฟเวอร์กลาง โดยใช้โปรโตคอล MQTT ซึ่งเหมาะสำหรับอุปกรณ์ IoT เนื่องจากมีน้ำหนักเบาและส่งข้อมูลได้แบบเรียลไทม์

ผลทดสอบ ข้อมูลจากเซนเซอร์สามารถส่งขึ้นมายัง Node-RED ได้สำเร็จด้านขวาของรูปภาพแสดงข้อมูลที่ Node-RED รับมาเรียงตามลำดับที่อ่านค่า



ภาพที่ 34 Node-RED รับข้อมูลได้สำเร็จ

3. ทดสอบการส่งข้อมูลเข้าไปเก็บยังฐานข้อมูลผ่าน Node-RED โดยใช้ PostgeSQL

ผลทดสอบ ข้อมูลส่งเข้ามาเก็บยังฐานข้อมูลตามที่กำหนดไว้ใน Node-RED

	id [PK] integer	gps_id integer	latitude double precision	longitude double precision	detected_time timestamp without time zone	name character varying (100)	geom geometry
12	67	[null]	16.952892	100.12498	2025-08-24 13:19:43.529639	จุดเก็บขยะ	0101000020E6100000DC291I
13	68	[null]	16.952892	100.12498	2025-08-24 13:19:55.235948	จุดเก็บขยะ	0101000020E6100000DC291I
14	69	[null]	16.952892	100.12498	2025-08-24 13:20:10.278854	จุดเก็บขยะ	0101000020E6100000DC291I
15	70	[null]	16.952892	100.12498	2025-08-24 13:20:25.305092	จุดเก็บขยะ	0101000020E6100000DC291I
16	71	[null]	16.952892	100.12498	2025-08-24 13:20:40.326819	จุดเก็บขยะ	0101000020E6100000DC291I
17	72	[null]	17.104132	100.13657	2025-08-24 13:39:33.550267	จุดเก็บขยะ	0101000020E610000058E718
18	73	[null]	17.104132	100.13657	2025-08-24 13:39:45.333885	จุดเก็บขยะ	0101000020E610000058E718
19	74	[null]	17.104132	100.13657	2025-08-24 13:40:00.727368	จุดเก็บขยะ	0101000020E610000058E718
20	75	[null]	17.104132	100.13657	2025-08-24 13:40:16.421275	จุดเก็บขยะ	0101000020E610000058E718
21	76	[null]	17.104132	100.13657	2025-08-24 13:40:32.703704	จุดเก็บขยะ	0101000020E610000058E718
22	77	[null]	17.104132	100.13657	2025-08-24 13:40:45.387596	จุดเก็บขยะ	0101000020E610000058E718

ภาพที่ 35 เก็บข้อมูลลงฐานข้อมูล

4. ตรวจสอบการแสดงผลบน WebGIS

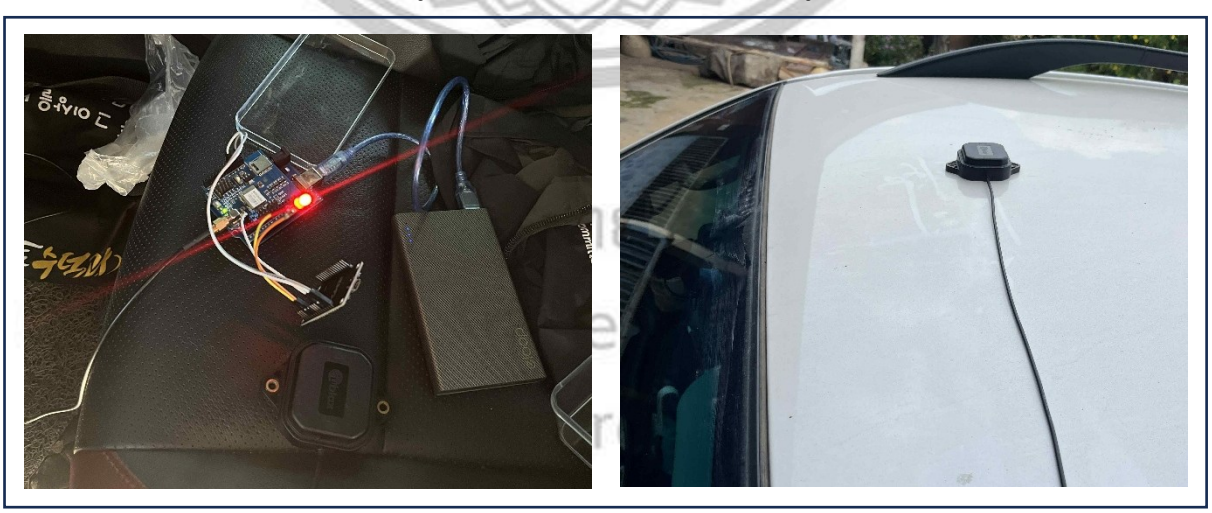
ผลทดสอบ ข้อมูลจากฐานข้อมูลสามารถเรียกมาแสดงบน WebGIS โดยไม่ผิดเพี้ยน

ภาพที่ 36 ทดสอบแสดงผลบน WebGIS

5. ทดสอบติดตั้งบนรถยนต์

ทำการติดตั้งเสารับสัญญาณ ANN-MB1-00 รับค่าดาวเทียมต่อกับบอร์ด GPS Shield, Arduino UNO R3 อ่านค่าGPS และส่งข้อมูลผ่าน MQTT ไปยัง Node-RED ผ่าน Wifi ที่ปล่อยให้บอร์ด NodeMCU esp8266

ผลทดสอบ สามารถอ่านค่าและส่งข้อมูลไปประมวลผลและเก็บลงฐานข้อมูล



ภาพที่ 37 ทดสอบติดตั้งบนรถยนต์

บทที่ 4

ผลการวิจัย

งานวิจัยชิ้นนี้เป็นการพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะโดยใช้ IoT และ GIS ได้มีการพัฒนาระบบติดตามที่ใช้ IoT ที่ทำให้สามารถติดตามรถเก็บขยะได้แบบ Real-Time ร่วมกับแผนที่ออนไลน์ WebGIS เพื่อติดตามและดูข้อมูลเส้นทางและมีการวิเคราะห์ข้อมูลเชิงพื้นที่ สำหรับจุดที่รถจอดและเส้นทางที่รถไปในวันนั้น ๆ โดยผลการพัฒนาระบบติดติดตามรถเก็บขยะ แบ่งออกเป็น

1. ผลการออกแบบและพัฒนาอุปกรณ์เซนเซอร์
2. ผลลัพธ์ WebGIS

4.1. ผลการออกแบบและพัฒนาอุปกรณ์เซนเซอร์

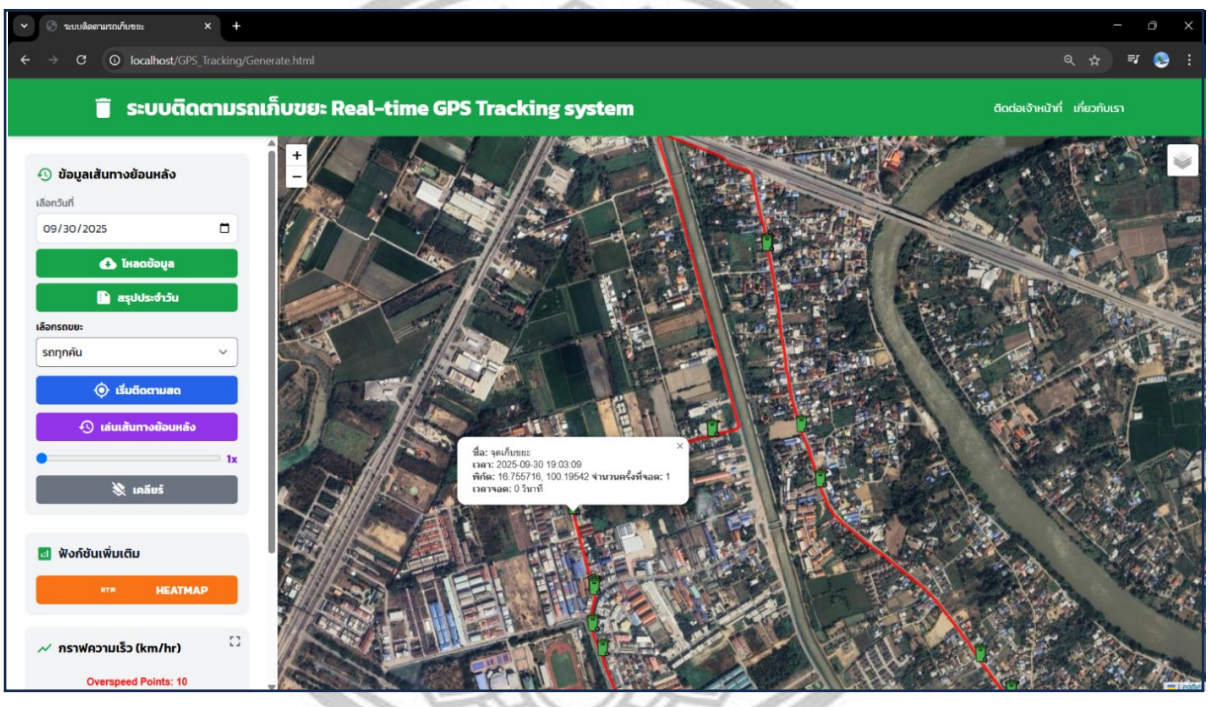
จุดเก็บขยะ (Garbage Point) ระบบสามารถตรวจจับและบันทึก จุดที่รถหยุดนิ่งเกิน 15 วินาที โดยคาดว่าเป็นจุดที่มีการปฏิบัติงานเก็บขยะ ข้อมูลนี้จะถูกเก็บในตาราง garbage_point ซึ่งเป็นหัวใจสำคัญในการวิเคราะห์ปริมาณขยะตามสมมติฐาน เส้นทางที่รถวิ่งในวันนั้น: ข้อมูลพิกัดตำแหน่งและความเร็วจะถูกบันทึกอย่างต่อเนื่องในตาราง gps_data ซึ่งถูกนำมาสร้างเป็น เส้นทางเดินรถย้อนหลัง (Route Playback) บนระบบ WebGIS อุปกรณ์เซนเซอร์สามารถทำงานได้ตามต้องการ และส่งข้อมูลตำแหน่งแบบเรียลไทม์ รวมถึงข้อมูลจุดจอดเก็บขยะ เพื่อให้เจ้าหน้าที่สามารถติดตามและวิเคราะห์เส้นทางได้อย่างมีประสิทธิภาพ

ภาพที่ 38 ผลการออกแบบและพัฒนาอุปกรณ์เซนเซอร์

4.2. ผลลัพธ์ WebGIS

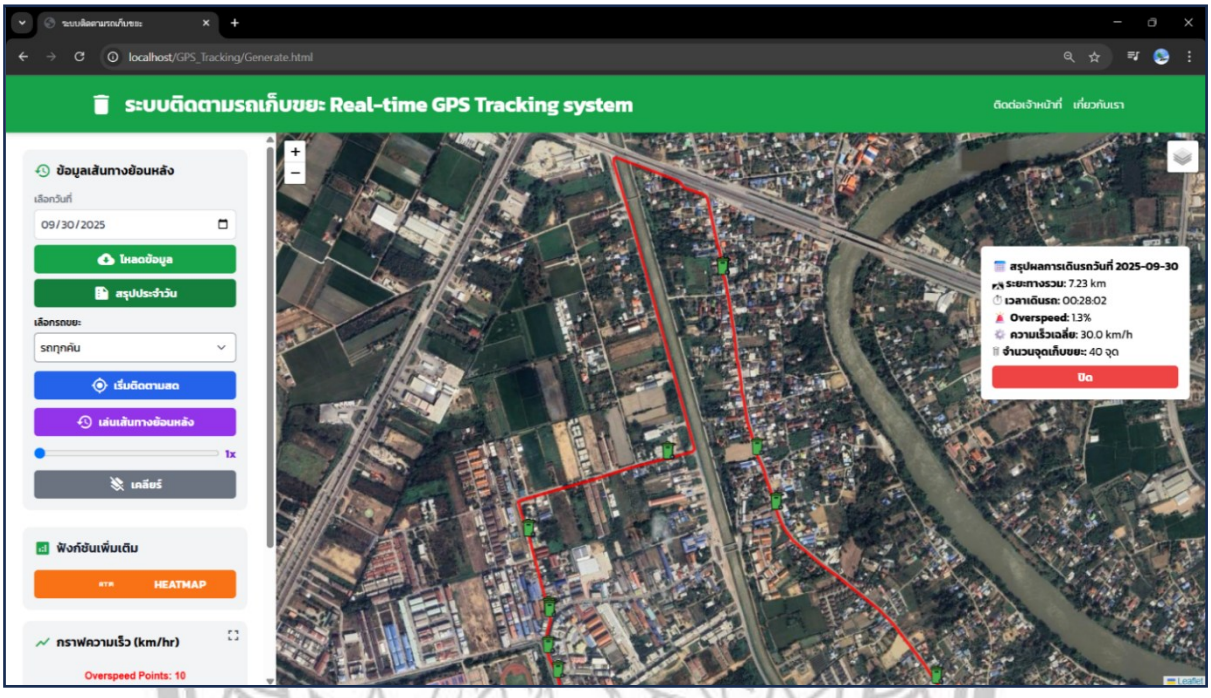
การพัฒนา WebGIS ใช้ภาษา HTML, CSS, JavaScript, PHP เพื่อให้สามารถใช้งานได้ผ่านออนไลน์หรือเว็บเบราว์เซอร์ โครงสร้างหลักของหน้าเว็บได้แก่

4.2.1. โหลดข้อมูลจากวันที่ เพื่อแสดงข้อมูลที่ดึงจากฐานข้อมูลมาแสดงเป็นเส้นทาง จุดรถ จุดเก็บขยะรวมไปถึงกราฟความเร็ว/วันเวลา เพื่อดูจุดไหนขับรถเร็วเกินกำหนด และแสดง popup ของจุดเก็บขยะว่าใช้เวลาจอดเก็บขยะกี่วินาทีกี่ครั้ง ในแถบด้านซ้ายมือ



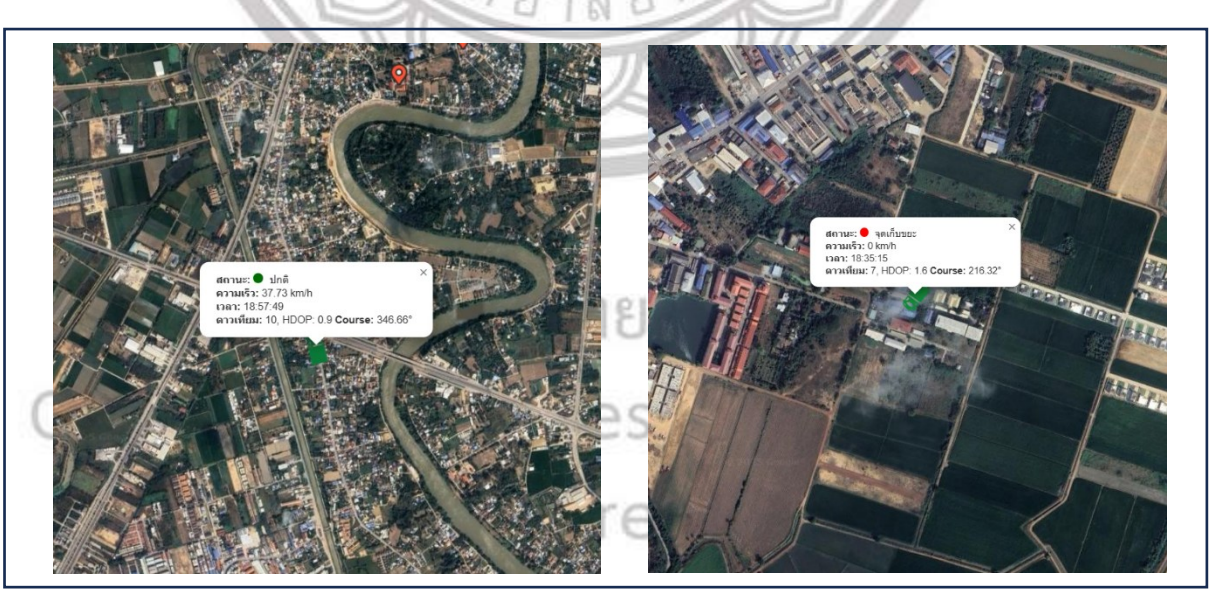
ภาพที่ 39 หน้าเว็บโหลดข้อมูลแสดงตำแหน่งในวันที่เลือก

4.2.2. ข้อมูลสรุปประจำวันโดยจะสรุปเวลาที่ใช้ ระยะทางที่ไปทั้งหมดตั้งแต่เริ่ม จำนวนจุดที่เก็บขยะ ความเร็วเฉลี่ยในรอบนั้นและเปอร์เซ็นต์ ขับรถเร็วเกินกำหนดในแต่ละวัน



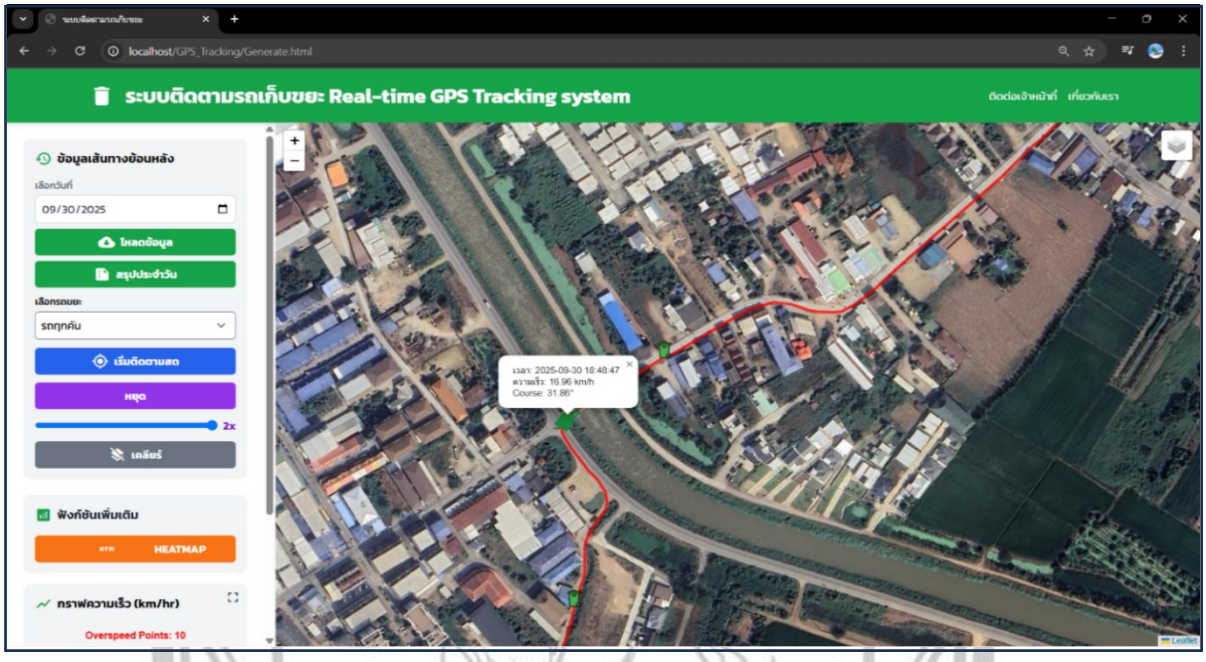
ภาพที่ 40 หน้าเว็บกล่องข้อความสรุปประจำวัน

4.2.3. ระบบติดตามแบบเรียลไทม์ สามาดูตำแหน่งของรถที่ติดเซนเซอร์ว่าอยู่ที่ไหน แสดงสถานะ รถกำลังจอดเก็บขยะ หรือ รถวิ่งปกติ รถแสดงทิศทางตามองศาจริงสามารถเห็นตำแหน่งได้อย่างชัดเจน



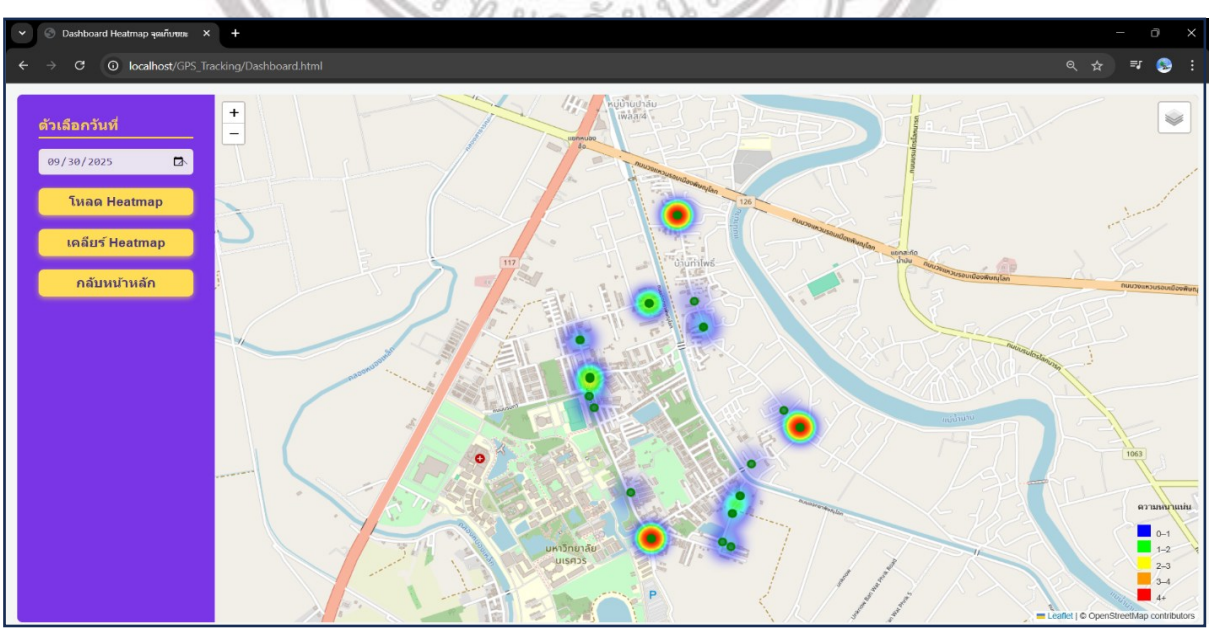
ภาพที่ 41 หน้าเว็บระบบติดตามเรียลไทม์

4.2.4. การเล่นเส้นทางย้อนหลัง โดยสามารถตรวจสอบเส้นทางย้อนหลังได้ว่ารถคันนี้ไปเส้นทางไหนมา จอดเก็บขยะตรงไหนและสามารถปรับความเร็วในการแสดงเส้นทางย้อนหลังได้



ภาพที่ 42 หน้าเว็บระบบเล่นเส้นทางย้อนหลัง

4.2.5. หน้าแสดงจุดความหนาแน่นของจุดเก็บขยะ (Heatmap) โดยต้องเลือกวันที่ก่อนเพื่อดึงข้อมูลจุด garbage_point โดยมีระดับความแน่นบอกตั้ง 1-4 0-1, 1-2, 2-3, 3-4, 4+ โดยตัวเลขจะแสดงถึงจำนวน



ภาพที่ 43 หน้าเว็บแสดงจุดความหนาแน่นจากจุดเก็บขยะ

4.2.6. กราฟแสดงข้อมูล ความเร็ว/วันเวลา สามารถแสดงจุดความเร็วแต่ละเส้นทางที่ไปโดยเมื่อคลิกไปยังจุดบนกราฟสีเขียวจะเชื่อมกับแผนที่บน Leaflet.js จะเด้งไปยังจุดนั้น และมีไฮไลท์จุดสีแดงที่โชว์ถึงความเร็วเกินกำหนดที่เราไว้คือ 60 เมื่อคลิกไปยังจุดสีแดงจะเด้งไปยัง จุดที่ใช้ความเร็วเกินกำหนด

ภาพที่ 44 หน้าเว็บแสดง กราฟแสดงจุดที่ใช้ความเร็ว

4.2.7. การแจ้งเตือนเมื่อความเร็วเกินกำหนดบน Telegram

ภาพที่ 45 การแจ้งเตือนเมื่อความเร็วเกินกำหนดบน Telegram

บทที่ 5

5.1. สรุปผลการวิจัย

การพัฒนาระบบ GPS Tracking เพื่อเพิ่มประสิทธิภาพการติดตามและวิเคราะห์เส้นทางรถเก็บขยะ โดยใช้ IoT และ GIS ได้บรรลุวัตถุประสงค์หลัก การพัฒนาอุปกรณ์และระบบ Geo-IoT Tracking อุปกรณ์ประสบความสำเร็จในการออกแบบและพัฒนา ชุดอุปกรณ์ติดตามรถเก็บขยะ ที่ประกอบด้วยไมโครคอนโทรลเลอร์ (Node MCU ESP8266/Arduino UNO) และโมดูล GPS ซึ่งสามารถติดตั้งบนรถเก็บขยะและทำงานได้อย่างมีประสิทธิภาพ การส่งข้อมูล สามารถรวบรวมข้อมูลตำแหน่งทางภูมิศาสตร์, ความเร็ว, และเวลา ได้แบบ เรียลไทม์ และส่งข้อมูลผ่านโปรโตคอล MQTT ไปยังเซิร์ฟเวอร์ฐานข้อมูล PostgreSQL/PostGIS ได้อย่างแม่นยำการพัฒนาระบบ WebGIS และการวิเคราะห์เส้นทางพัฒนา ระบบ WebGIS โดยใช้ Leaflet.js ซึ่งสามารถ แสดงผลการติดตามรถแบบเรียลไทม์ และแสดง เส้นทางเดินรถย้อนหลัง ในแต่ละวัน ทำให้ง่ายต่อการติดตามและตรวจสอบจุดเก็บขยะ ระบบสามารถประมวลผลและตรวจจับ จุดเก็บขยะ (Garbage Point) โดยอัตโนมัติจากข้อมูลการหยุดนิ่งของรถ (หยุดเกิน 15 วินาที) ทำให้สามารถทราบ ตำแหน่งและจำนวนครั้ง ในการปฏิบัติงานจริงในพื้นที่การเพิ่มประสิทธิภาพการบริหารจัดการ สถิติการดำเนินงาน ระบบแสดง สรุปสถิติการดำเนินงานรายวัน ทั้งในด้านระยะทางรวม, เวลาการทำงานรวม ,และจำนวนจุดเก็บขยะ ซึ่งเป็นประโยชน์ในการประเมินประสิทธิภาพการแจ้งเตือน มีการพัฒนาระบบ แจ้งเตือนความเร็วเกินกำหนด ผ่าน Telegram ทันทีที่รถขับเกิน 60 กม./ชม. เพื่อช่วยควบคุมพฤติกรรมการขับขี่และลดปัญหาอุบัติเหตุ

5.2. อภิปรายผลการวิจัย

ความแม่นยำในการประเมินภาระงาน การใช้ "เวลาการหยุดนิ่งของรถ" เป็นตัวแปรแทน ปริมาณขยะหรือ ภาระงาน ณ จุดเก็บขยะ แสดงให้เห็นถึงแนวทางที่สามารถทำได้จริงและมีประสิทธิภาพในการประเมินความหนาแน่นของขยะ ซึ่งเป็นข้อมูลเชิงพื้นที่ที่สำคัญในการวางแผนเส้นทางเดินรถใหม่ และจัดสรรทรัพยากรให้สอดคล้องกับความต้องการจริงของแต่ละพื้นที่การเพิ่มประสิทธิภาพการกำกับดูแล การทำงานแบบเรียลไทม์ ของระบบ WebGIS และการแจ้งเตือน Telegram ถือเป็นปัจจัยสำคัญที่ช่วยเพิ่ม ความโปร่งใส และประสิทธิภาพในการควบคุมกำกับดูแล เจ้าหน้าที่สามารถตรวจสอบเส้นทางการทำงานและการละเมิดวินัย (เช่น การขับรถเร็ว) ได้ทันที ซึ่งส่งผลให้พนักงานเกิดความตระหนักและปฏิบัติงานตามมาตรฐานมากขึ้นความเหมาะสมของเทคโนโลยีบูรณาการ การบูรณาการเทคโนโลยี IoT สำหรับการเก็บข้อมูลภาคสนาม, MQTT/Node-RED สำหรับการสื่อสารและประมวลผลข้อมูล, และ GIS สำหรับการวิเคราะห์และแสดงผลเชิงพื้นที่ แสดงให้เห็นถึงการทำงานของระบบที่เหมาะสมและมีความเสถียรสำหรับการประยุกต์ใช้ในการบริหารจัดการขององค์กรปกครองส่วนท้องถิ่น

5.3. ข้อเสนอแนะสำหรับพัฒนาระบบในอนาคต

1. พัฒนาระบบสื่อสารสำรอง เช่น การใช้ LoRaWAN ในพื้นที่ที่สัญญาณ 3G/4G อ่อนแอ เพื่อเปรียบเทียบประสิทธิภาพและความคุ้มค่ากับต้นทุน Wi-Fi/Cellular

2. ปรับปรุงการใช้งานเส้นทาง แนะนำให้สามารถทำให้ดูได้ว่า เส้นทางนี้รถจะมาเก็บหรือไม่ แล้วจะมาถึงตอนไหนเพื่อลดปัญหาการกองขยะทิ้งไว้ก่อนถึงวันมาเก็บ

3. การเพิ่มเซนเซอร์ตรวจวัดปริมาณขยะจริง ควรมีการต่อยอดงานวิจัยโดยการเพิ่ม เซนเซอร์วัดน้ำหนัก หรือเซนเซอร์วัดระดับขยะ เพื่อให้ได้ข้อมูล ปริมาณขยะจริง มาใช้ในการวิเคราะห์และจัดสรรทรัพยากร

บรรณานุกรม

สกรณ์ บุษบง, ณัฐพล จั้นพลแสน, ศุภกร ชาติชัย (2019) **การพัฒนาต้นแบบระบบพร้อมชุดอุปกรณ์ติดตามรถขนส่งพัสดุแบบ เรียลไทม์โดยใช้เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง.** วารสารวิชาการเทคโนโลยีและการสื่อสาร, สถบันมหาวิทยาลัยราชภัฏบุรีรัมย์

Bakhtiar Ali, Muhammad Awais Javed, & Alharbi, Abeer A. K. (2023) **Internet of Things-Assisted Vehicle Route Optimization for Municipal Solid Waste Collection.** Applied Sciences, 14(1), 287.

Mohammadhossein Ghahramani, Mengchu Zhou, Anna Molter, & Francesco Pilla (2022) **IoT-Based Route Recommendation for Intelligent Waste Management. .** IEEE Internet of Things Journal, *9*(18), 17978–17989.

Ibraheem Kasim Ibraheem, Salam Wisam Hadi (2018) **Design and Implementation of a Low-Cost Secure Vehicle Tracking System.** In 2018 IEEE International Iraqi Conference Engineering Technology and their Applications (IICETA) (pp. **146–150**).

Idriss Moumen, Najat Rafalia, Jaafar Abouchabaka and Marouane Aoufi1 (2023) **Real-time GPS Tracking System for IoT-Enabled Connected Vehicles.** E3S Web Conf., 412,

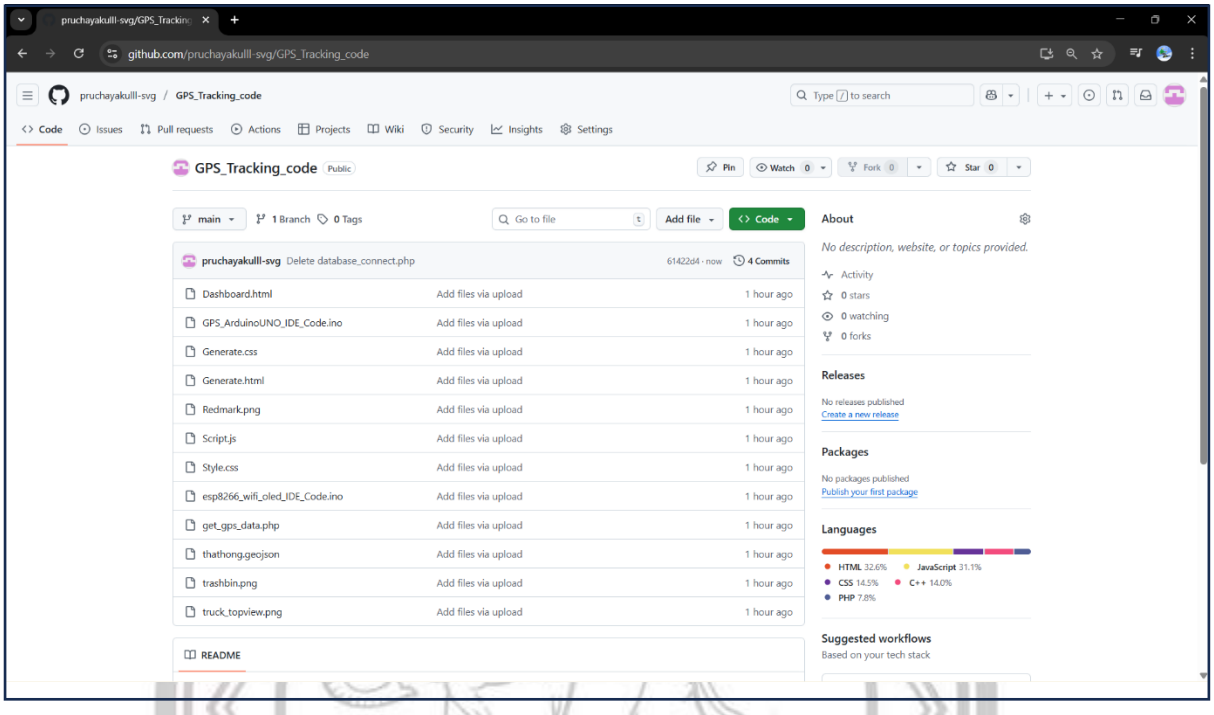
OrvenE.Llantos, OlgaJoyL (2024) “**Real-Time Tracker for Moving Objects with Enhanced Signal and Customized User Interface through Broadcasting Global Positioning Systems Data.** Procedia Computer Science, 251, 132-141

Praveen Kumar, Raj Mohan Singh, Sailesh N. Behera (2022) **A GIS-based Route Optimization Approach for Municipal Solid Waste Management.** International Journal of Environmental Protection, 12(3), 1616–1623.

ภาคผนวก

โค้ดฉบับเต็มสามารถเข้าถึงได้ลิงค์ต่อไปนี้ (github)

URL สำหรับเข้าถึงโค้ดทั้งหมด : https://github.com/pruchayakulll-svg/GPS_Tracking_code



Dashboard.html : หน้าแสดง heatmap

GPS_ArduinoUNO_IDE_Code.ino : โค้ดสำหรับบอร์ด ArduinoUNO

Generate.css : ตกแต่งในส่วนของหน้าเว็บหลัก

Generate.html : หน้าเว็บหลักใช้แสดงข้อมูลต่างๆ

Redmark.png : marker สีแดง

Script.js : รวมทุกฟังก์ชันในการทำงานของหน้าเว็บ

Style.css : ตกแต่งกล่องข้อความในหน้าเว็บและอื่นๆ

Esp8266_wifi_oled_IDE_Code.ino : โค้ดสำหรับบอร์ด NodeMCU esp8266 ที่ใช้ส่งข้อมูลผ่าน MQTT โดยใช้ wifi และการแสดงผลบนหน้าจอ Olde 1.3 นิ้ว

get_gps_data.php : การเชื่อมต่อฐานข้อมูล, ดึงข้อมูลจากฐานข้อมูล PostgreSQL

thathong.geojson : ข้อมูลสถานที่ในตำบลท่าทอง

trashbin.png : รูปภาพถังขยะที่ใช้งานบนเว็บ

truck_topview.png : รูปภาพรถแนวตั้งที่ใช้แสดงบนหน้าเว็บโดยสามารถหัวตามทิศจริงตามค่าที่ได้เรียก

ประวัติผู้วิจัย

ชื่อ – ชื่อสกุล ปรัชญากุล มณีฉาย

วัน เดือน ปี เกิด 20 กันยายน 2546

ที่อยู่ปัจจุบัน 34/1 ม.5 ต.สร่างโศก อ.บ้านหมอ จ.สระบุรี 18130

ประวัติการศึกษา

พ.ศ. 2565 - ปัจจุบัน ระดับอุดมศึกษา วท.บ. สาขาภูมิศาสตร์ มหาวิทยาลัยนเรศวร อำเภอเมืองพิษณุโลก จังหวัดพิษณุโลก 65000 เกรดเฉลี่ย 3.08

พ.ศ. 2562 - 2565 ระดับมัธยมศึกษาตอนปลาย (วิทย์-คอม) โรงเรียนพิบูลวิทยาลัย อ.เมืองลพบุรี จ.ลพบุรี 15000

พ.ศ. 2559 - 2561 ระดับมัธยมศึกษาตอนต้น โรงเรียนวินิตศึกษา ในพระราชูปถัมภ์ฯ อ.เมืองลพบุรี จ.ลพบุรี 15000

กิจกรรมที่เข้าร่วม

1. ร่วมจัดเตรียมโครงการค่ายภูมิศาสตร์ ณ อุทยานภูหินร่องกล้า อำเภอนครไทย จังหวัดพิษณุโลก

2. ดำรงตำแหน่งเหรัญญิกชมรมภูมิศาสตร์

3. งาน GeoHackathon ณ หมู่บ้านนครชุม อำเภอนครไทย จังหวัดพิษณุโลก

4. เข้าอบรมการใช้เครื่องมือ GNSS ณ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

5. เข้าร่วมการเรียนรู้นอกห้องเรียนด้วยการออกภาคสนามจริง ณ สถานีปฏิบัติการบึงราชนก โดยใช้ Trimble GNSS RTK ทำการปักพิกัด GCPs และถ่ายภาพจากโดรน Multispectral Camera อำเภอวังทอง จังหวัดพิษณุโลก

6. เข้าอบรมโครงการพัฒนาทักษะภาษาอังกฤษสำหรับภูมิศาสตร์ประจำปี 2567 ณ มหาวิทยาลัยนเรศวร

กิจกรรมที่เข้าร่วม(ต่อ)

7. งานประชุมวิชาการและงานวิจัยนักภูมิศาสตร์ประจำปี 2568

8. เข้าฟัง Special Seminar Theme: From Climate Change to Sustainable Futures: Geoinformatics in Action for SDGs ณ มหาวิทยาลัยนเรศวร

9. เข้าอบรม การบรรยายและอบรมเชิงปฏิบัติการ หัวข้อ “IoT และ LoRawan Gateways”

10. เข้ารับฟังบรรยายพิเศษหัวข้อ “Leveraging Geospatial Technologies Using Free & Open Source Software and Open Data” และพูดคุยแลกเปลี่ยนความร่วมมือทางวิชาการระหว่างมหาวิทยาลัยนเรศวรและ Osaka Metropolitan University (OMU) ณ มหาวิทยาลัยนเรศวร