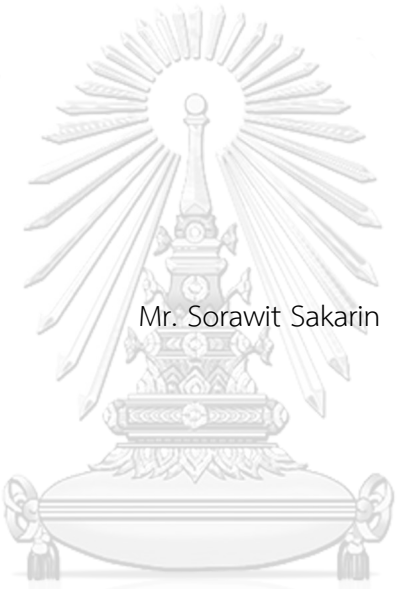
การพัฒนาระบบติดตามสถานะรถยนต์สำหรับธุรกิจแบ่งปันรถยนต์โดยใช้ช่อง OBD-II

นายสรวิชญ์ สาครินทร์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาระบบกายภาพที่เชื่อมประสานด้วยเครือข่ายไซเบอร์ ภาควิชาวิศวกรรมเครื่องกล
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

THE DEVELOPMENT OF CAR TRACKING SYSTEM FOR CAR SHARING BUSINESS BY USING OBD-II

Mr. Sorawit Sakarin



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Cyber-Physical System
Department of Mechanical Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2020

หัวข้อวิทยานิพนธ์	การพัฒนาระบบติดตามสถานะรถยนต์สำหรับธุรกิจแบ่งปันรถยนต์โดยใช้ช่อง OBD-II
โดย	นายสรวิชญ์ สาครินทร์
สาขาวิชา	ระบบกายภาพที่เชื่อมประสานด้วยเครือข่ายไซเบอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.กฤษฎา พนมเชิง

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.รัชทิน จันทร์เจริญ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.กฤษฎา พนมเชิง)

..... กรรมการ
(ดร.ณัฐพล ดำรงค์พลาสิทธิ์)

..... กรรมการ
(ดร.สุรัฐ ขวัญเมือง)

..... กรรมการภายนอกมหาวิทยาลัย
(ดร.กิตติกุล โกวิทางกูร)

สรวิชญ์ สาครินทร์ : การพัฒนาระบบติดตามสถานะรถยนต์สำหรับธุรกิจแบ่งปันรถยนต์โดยใช้ช่อง OBD-II. (THE DEVELOPMENT OF CAR TRACKING SYSTEM FOR CAR SHARING BUSINESS BY USING OBD-II) อ.ที่ปรึกษาหลัก : ผศ. ดร.กฤษฎา พนมเชิง

ในช่วงไม่กี่ปีที่ผ่านมานี้ ธุรกิจแบ่งปันรถเช่าทั่วโลกเติบโตขึ้นอย่างมาก แต่การเช่ารถยนต์นั้นยังมีปัญหาอยู่สองข้อคือความไม่สะดวกต่อการทำสัญญาเช่ารถยนต์และการที่ไม่สามารถติดตามสถานะของรถยนต์ได้ เพื่อที่จะติดตามสถานะของถยนต์ เราต้องการระบบที่สามารถดึงข้อมูลดิบจากรถยนต์ส่งขึ้นสู่คลาวเซิฟเวอร์ในเวลาจริง ในช่วงเวลาเดียวกัน เทคโนโลยีอินเทอร์เน็ตทุกสรรพสิ่งได้ถูกนำไปใช้ในหลายๆอุตสาหกรรมเพื่อเพิ่มศักยภาพในการทำงาน เช่นอุตสาหกรรมการแพทย์ การเกษตร และการขนส่งอัจฉริยะ ดังนั้นธุรกิจแบ่งปันรถเช่านั้นจึงยังมีช่องว่างสำหรับการพัฒนาเพิ่มเติมด้วยเทคโนโลยีนี้เช่นกัน งานวิจัยนี้ได้ทำการพัฒนาอุปกรณ์สำหรับเชื่อมต่อภายในรถยนต์เรียกว่ากล่องโคบอกซ์ อุปกรณ์สามารถดึงข้อมูลที่ต้องการจากรถยนต์และสั่งล้อคหรือปลดล้อคได้ด้วยช่องโอบีดีทู อุปกรณ์สามารถรับสัญญาณตำแหน่งจีพีเอสและส่งข้อมูลที่ต้องการสู่คลาวไฟร์เบสเซิฟเวอร์ได้บนเวลาจริง อย่างไรก็ตาม ระบบติดตามสถานะของรถยนต์นี้สามารถทำงานได้ดีในรถยนต์ Toyota Altis 2016 เท่านั้น ผู้วิจัยจึงมีแผนในการพัฒนาระบบติดตามเพื่อรองรับกับรถยนต์รุ่นต่างๆต่อไปในอนาคต

สาขาวิชา	ระบบกายภาพที่เชื่อมประสานด้วยเครือข่ายไซเบอร์	ลายมือชื่อนิสิต .....
ปีการศึกษา	2563	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6270374621 : MAJOR CYBER-PHYSICAL SYSTEM

KEYWORD: OBD-II, internet of things, Gps

Sorawit Sakarin : THE DEVELOPMENT OF CAR TRACKING SYSTEM FOR CAR SHARING BUSINESS BY USING OBD-II. Advisor: Asst. Prof. Gridsada Phanomchoeng

In the last few years, the car sharing businesses around the world are greatly growing. Nowadays, the main problems in renting cars are uncomfortably contracting and inconvenient tracking statuses of cars. In order to tracking cars, we need a system which receives raw data from cars and send them to cloud server real-time. Simultaneously, internet of thing technology is implemented to many industries like medical areas, agriculture and intelligent transportation. Hence, the car sharing business can be excelled by merging IOT technology with a traditional methodology. CoBox is an IOT device which designed to accomplish the tracking system. It can receive and transmit data to cars by connecting with OBD-II port. Monitoring, receiving locations from GPS module, managing power supply and connecting to Firebase server are controlled by using ESP32. With these abilities, the car tracking system can collect needed data and can control lock-unlock door real-time. However, the system only support on Toyota Altis2016 for now, we plan to verify with the other classes of cars in the future.

Field of Study: Cyber-Physical System Student's Signature .....

Academic Year: 2020 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ดำเนินการสำเร็จลุล่วงไปด้วยดีจากความกรุณาของผู้ช่วยศาสตรจารย์ ดร. กฤษฎา พนมเชิง อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้เสียสละเวลาให้คำปรึกษา ให้คำแนะนำ เสนอแนวทาง ให้ความอนุเคราะห์ ตลอดจนแก้ไขข้อบกพร่องต่างๆ ติดต่อผู้เชี่ยวชาญเฉพาะทางเพื่อรับคำปรึกษาเพิ่มเติม จนวิทยานิพนธ์ครั้งนี้เสร็จสมบูรณ์ ผู้วิจัยรู้สึกซาบซึ้งในความกรุณาเป็นอย่างยิ่ง จึงขอกราบขอบพระคุณเป็นอย่างสูง

กราบขอบพระคุณรองศาสตราจารย์ ดร.รัชทิน จันทร์เจริญ อาจารย์ ดร.สุรัฐ ขวัญเมือง และอาจารย์ดร.ณัฐพล ดำรงค์พลาสิทธิ์ ที่ได้เสียสละเวลา ติดตามและให้คำแนะนำทำให้วิทยานิพนธ์ฉบับนี้ครบถ้วนสมบูรณ์ และกราบขอบพระคุณดร. กิตติกุล โกวิทางกูร ที่กรุณาเป็นผู้ให้คำปรึกษาด้านการเขียนและติดต่อกับบริษัทเอกชน รวมทั้งเป็นกรรมการคุมสอบวิทยานิพนธ์ ผู้วิจัยขอกราบขอบพระคุณด้วยความเคารพอย่างสูง

กราบขอบพระคุณบริษัท กรุงไทยคาร์เร้นท์ แอนด์ ลีส จำกัด ที่ร่วมมือกันพัฒนาความต้องการของระบบติดตามสถานะของรถยนต์ อีกทั้งยังจัดเตรียมรถยนต์สำหรับทำการวิจัยและทดสอบเก็บข้อมูล และแนะนำถึงแนวทางแก้ไขและวางแผนในการปรับปรุงระบบในอนาคตต่อไป

กราบขอบคุณบริษัท โซลิแมค ออโตเมชั่น จำกัดและเพื่อนร่วมงานที่ให้ความรู้ ความร่วมมือและสนับสนุนการทำวิทยานิพนธ์ ช่วยกันขัดเกลาและแลกเปลี่ยนข้อมูลตลอดการทำวิทยานิพนธ์ในการเรียนปริญญาโท

สุดท้ายนี้ผู้วิจัยขอกราบขอบพระคุณบิดามารดา ที่ให้กำลังใจและสนับสนุนการศึกษาทำงานของผู้วิจัยเสมอมา อีกทั้งยังชี้แนะแนวทางและออกความเห็นให้กับการทำวิทยานิพนธ์ฉบับนี้จนครบถ้วนสมบูรณ์ ผู้วิจัยขอกราบขอบพระคุณด้วยความเคารพอย่างสูง

สรวิชญ์ สาครินทร์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ธุรกิจแบ่งปันรถยนต์ (car-sharing) เติบเป็นอย่างมากในช่วงสองสามปีที่ผ่านมา โดยมีมูลค่าตลาดทั่วโลกรวมกันมากกว่าสองพันห้าร้อนล้าน ดอลล่าสหรัฐในปี 2019 ซึ่งมีมูลค่าโดยรวมมากว่าในปี 2018 ถึง 39 เปอร์เซ็นต์ และมีแนวโน้มว่าจะโตต่อไปมากกว่า 24 เปอร์เซ็นต์ต่อปีจนถึงปี 2026[1] โดยปัจจัยหลักที่ส่งผลให้ธุรกิจแบ่งปันรถยนต์เติบโตอย่างรวดเร็วมีทั้งหมด 4 ปัจจัย ปัจจัยแรกคือจำนวนประชากรที่เพิ่มขึ้นภายในเมือง โดยประชากรโลกปัจจุบันมีประมาณเจ็ดพันเจ็ดร้อยล้านคน ซึ่งเพิ่มขึ้นกว่าหนึ่งพันล้านคนภายในเวลาสองร้อยปีและยังมีแนวโน้มที่ประชากรจะเพิ่มขึ้นอีก 1.05 เปอร์เซ็นต์ต่อปี หรือก็คือประมาณแปดสิบเอ็ดล้านคนต่อปี[2, 3] ความหนาแน่นของประชากรต่อพื้นที่ที่มากเป็นตัวดึงดูดการลงทุนของนักลงทุนเป็นอย่างดี ปัจจัยที่สองคือการครอบคลุมของระบบขนส่งสาธารณะ ประชากรจะเข้าถึงขนส่งสาธารณะได้ง่ายขึ้นส่งผลให้มีความต้องการการครอบครองรถยนต์ส่วนตัวลดลง[4] จึงนิยมใช้รถเช่าร่วมกันกับระบบขนส่งสาธารณะ ปัจจัยต่อมาคือนโยบายของภาครัฐที่ช่วยสนับสนุนให้ประชาชนหันมาใช้รถยนต์สาธารณะมากขึ้นเช่น ช่วยลดค่าใช้จ่ายหรือปรับปรุงคุณภาพของระบบขนส่งมวลชน ปัจจัยสุดท้ายคือการมีพื้นที่จอดรถส่วนตัวสำหรับบริการรถเช่า ในเมืองที่มีจำนวนประชากรหนาแน่นอย่างเช่นกรุงเทพมหานคร ประชากรมักจะประสบปัญหาเรื่องการหาพื้นที่สำหรับจอดรถยนต์ส่วนตัวยากส่งผลเสียต่อทั้งสุขภาพกายและสุขภาพจิต ดังนั้นธุรกิจรถเช่าจำนวนมากจึงมีบริการพื้นที่จอดรถยนต์เฉพาะเพื่อรองรับความต้องการส่วนนี้ตามพื้นที่สำคัญต่างๆ

ธุรกิจแบ่งปันรถยนต์นั้นมีรูปแบบหลักอยู่ 3 แบบด้วยกัน รูปแบบแรกคือรับรถตามสถานี (Station-based) โดยทางบริษัทรถเช่าจะทำการจัดเตรียมพื้นที่สำหรับจอดรถไว้ตามสถานที่ต่างๆ ผู้เช่าสามารถทำการเลือกรถได้ตามสถานที่ต่างๆ แต่จะต้องนำรถยนต์กลับมาคืนในสถานที่นั้นๆเช่นกัน โดยจะมีรูปแบบย่อยลงไปอีก 2 แบบ รูปแบบย่อยแรกคือราวด์ทริป (Round trip) เป็นรูปแบบที่ผู้เช่าจะต้องนำรถยนต์กลับมาจอดในพื้นที่เดียวกันกับสถานีที่ขับรถยนต์ออกไป และอีกรูปแบบคือ วันเวย์ทริป (One-way trip) ผู้เช่าสามารถคืนรถยนต์ได้ที่สถานีไหนก็ได้ตามที่ต้องการ รูปแบบธุรกิจแบ่งปันรถยนต์ถัดมาเรียกว่า เพียร์ทูเพียร์ (Peer to peer) โดยทางบริษัทรถเช่าจะเป็นเพียงแค่ตัวกลางในการติดต่อระหว่างผู้เช่า จะไม่มีการเตรียมสถานีจอดรถยนต์ ซึ่งผู้เช่าจะต้องจัดการเรื่อง

สถานีจอดรถยนต์กันเอง รูปแบบสุดท้ายเรียกว่าการเช่าอิสระ (Free-floating) จะมีรูปแบบการเช่าที่คล้ายคลึงกันกับรูปแบบก่อนหน้า แต่รูปแบบนี้จะไม่มีการตกลงเรื่องสถานีจอดรถกันก่อน ผู้เช่าสามารถรับและคืนรถยนต์ได้ทุกที่ตามต้องการตราบเท่าที่ไม่ผิดกฎระเบียบทางสังคม[5]

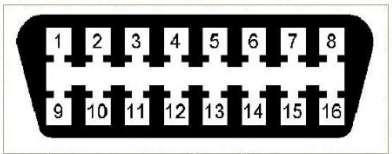
รถยนต์สำหรับการใช้งานเพื่อธุรกิจรถเช่านั้นมีด้วยกันหลากหลายประเภทขึ้นกับการใช้งานตามวัตถุประสงค์ โดยทั่วไปบริษัทรถเช่าจะนำรถยนต์ทั่วไปมาติดตั้งอุปกรณ์เสริมเฉพาะเพื่อเพิ่มฟังก์ชั่นการใช้งานให้เหมาะสมกับการเช่า อุปกรณ์เหล่านี้ทำหน้าที่เป็นตัวกลางการสื่อสารระหว่างผู้ขับและผู้ให้บริการโดยมีรูปแบบแตกต่างกันไปตามแต่แนวทางการให้บริการ

1.1.1 หน้าที่การทำงานหลักของอุปกรณ์ IOT (Internet Of Thing)

จากการหารือร่วมกับบริษัทรถเช่ารายใหญ่ในประเทศไทย รถยนต์หลังจากได้รับการติดตั้งอุปกรณ์ชุดนี้แล้ว จะต้องสามารถรับส่งข้อมูลเฉพาะระหว่างรถยนต์และผู้ควบคุมระบบได้ โดยจะต้องพึ่งระบบสื่อสารไร้สายอาทิเช่น อินเทอร์เน็ต หรือ บลูทูธ โดยข้อมูลที่รับมาจากรถยนต์นั้นจะสามารถนำไปวิเคราะห์เหตการณ์ต่างๆได้ ยกตัวอย่างเช่น การวิเคราะห์อัตราการเผาพลาญเชื้อเพลิงเกินความจำเป็นหรือการวิเคราะห์ความประพฤติบนท้องถนนของผู้ขับรถยนต์เป็นต้น โดยข้อมูลต่างๆบนรถยนต์นั้นสามารถดึงออกมาได้จากช่องโอบีดีทู (OBD-II) ของรถยนต์

1.1.2 ช่องเชื่อมต่อโอบีดีทู

ตามมาตรฐาน SAE J1979[6] รถยนต์ทุกคันบนท้องถนนจะต้องรองรับมาตรฐานนี้ร่วมกัน จะต้องมีช่องสื่อสารโอบีดีทู ไว้สำหรับดึงข้อมูลสำคัญออกจากรถยนต์ ตามมาตรฐานทั่วทั้งรถยนต์มีเซนเซอร์สำหรับตรวจสภาพรถยนต์ในยามจำเป็นยกตัวอย่างเช่น เซนเซอร์ตรวจจับความเร็วของยานพาหนะ เซนเซอร์วัดอุณหภูมิภายในรถ หรือ เซนเซอร์วัดระดับน้ำมันในถัง ช่องโอบีดีทูมีขาเชื่อมต่อทั้งหมด 16 ขาดังรูปที่ 1 ซึ่งแต่ละขาจะถูกนำไปใช้ตามแต่วัตถุประสงค์



PIN	DESCRIPTION	PIN	DESCRIPTION
1	Vendor Option	9	Vendor Option
2	J1850 Bus +	10	j1850 BUS
3	Vendor Option	11	Vendor Option
4	Chassis Ground	12	Vendor Option
5	Signal Ground	13	Vendor Option
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 Low
8	Vendor Option	16	Battery Power

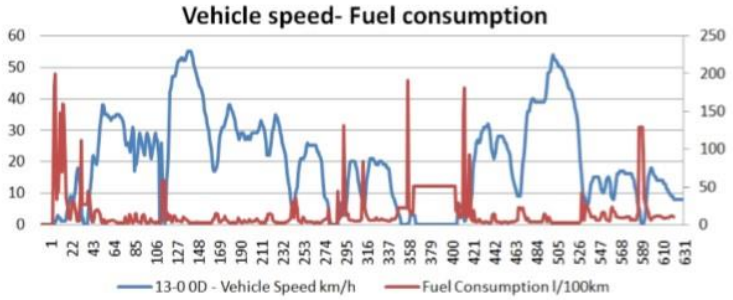
OBD-II Connector and Pinout

รูปที่ 1 ขาพินของโอบีดีทู

ผลิตภันท์ในท้องตลาดสำหรับการดึงข้อมูลออกจากรถยนต์นั้นมีค่อนข้างหลากหลาย มีข้อดีข้อเสียแตกต่างกันไปยกตัวอย่างการเปรียบเทียบระหว่าง OBD และ CAN Sampling ของ SensorHUB Framework ดังรูปที่ 2 เรื่องการเปรียบเทียบการดึงข้อมูลระหว่างใช้ OBD-II service กับ CAN BUS ทั้งสองตัวสามารถดึงข้อมูลออกมาได้ทุกตัวตามที่ต้องการ แต่ OBD-II service จะมีความถี่ในการส่งออกข้อมูลอยู่ที่ 1Hz ซึ่งในส่วนของ CAN BUS จะละเอียดที่สุดอยู่ที่ 5 HZ โดยสรุปแล้วหากต้องการความละเอียดของข้อมูลนั้นควรจะใช้การรับส่งข้อมูลโดยใช้ OBD-II service แต่จะมีข้อเสียเรื่องความปลอดภัยขณะการใช้งาน เพราะจะติดตั้งอุปกรณ์ผ่านทางช่องโอบีดีทูจะสามารถถูกถอดได้ง่าย[7]

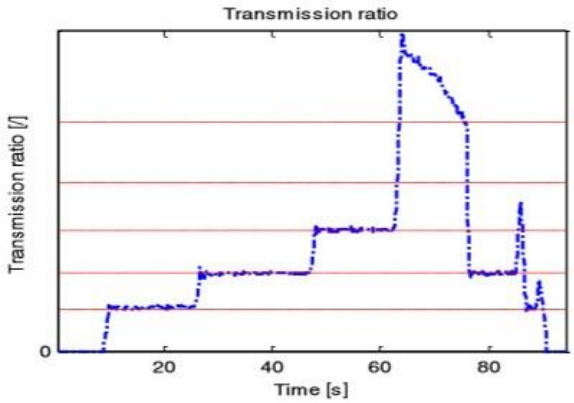
รูปที่ 2 OBD และ CAN Sampling ของ SensorHUB

ค่าที่สามารถดึงออกมาได้จากโอบีดีทูนั้นมีเยอะมาก ค่าหลักๆที่เหมาะสำหรับมาวิเคราะห์การทำงานของการเผาผลาญน้ำมันรถยนต์มีดังนี้ Engine coolant temperature, short term fuel trim, Engine RPM, Vehicle speed, Air flow rate จากการทดลองขับรถเป็นระยะทาง 5 กิโลเมตร บันทึกค่าตัวแปรดังกล่าว[8] สามารถวิเคราะห์ออกมาได้ว่าค่าการเผาผลาญเชื้อเพลิงนั้น ใกล้เคียงกับค่าจากทางผู้ผลิต มีปัจจัยหลักสำคัญที่ทำให้ค่าคลาดเคลื่อนคือพฤติกรรมของผู้ขับ หากขับรถยนต์ด้วยอัตราเร็วคงที่มีการเผาผลาญเชื้อเพลิงที่ต่ำกว่าการขับหยุดขับหยุด ดังรูปที่ 3 จะเห็นว่าอัตราการเผาผลาญเชื้อเพลิง(สีแดง)จะพุ่งสูงขึ้นเมื่อความเร็วรถยนต์(สีฟ้า)ลดลง จากการทดลองขับรถยนต์ขึ้นทางด่วนมีอัตราการเผาผลาญเชื้อเพลิงต่ำที่สุดเนื่องจากขับรถยนต์ด้วยความเร็วคงที่ตลอด



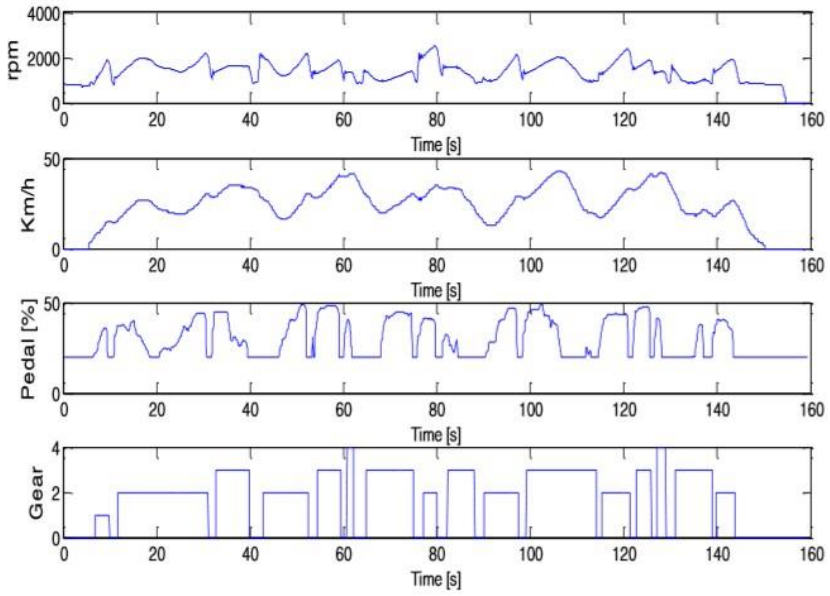
รูปที่ 3 กราฟความสัมพันธ์ระหว่างความเร็วและการเผาผลาญเชื้อเพลิง

ค่าเซนเซอร์จากโอบีดีทูนั้นไม่ได้ครอบคลุมหมดทุกค่าที่เราต้องการ บางค่าไม่มีการเตรียมไว้รองรับ ยกตัวอย่างเช่นตำแหน่งของเกียร์กระปุกในรถยนต์ที่ไม่ได้มีการรองรับ การรับรู้สถานะของเกียร์นั้นมีประโยชน์อย่างมากในการไปวิเคราะห์ประสิทธิภาพการส่งกำลังและการควบคุมของรถยนต์ ตัวแปรที่จำเป็นสำหรับการคำนวนสถานะของเกียร์มี ตำแหน่งของคันเร่งเท้า ความเร็วรถยนต์ และความเร็วรอบเครื่อง ซึ่งสามารถดึงข้อมูลออกมาได้จากช่องโอบีดีทู ค่าอัตราทดของกำลังส่ง (transmission ratios)[9] สามารถคำนวนได้จากสมการ $R = \frac{\omega_w}{\omega_e} = \frac{V60}{3.6 r_w 2\pi rpm}$ โดย R คือ transmission ratios และ r_w คือรัศมีของล้อรถยนต์ สามารถนำค่ามาวาดกราฟได้ดังรูปที่ 4



รูปที่ 4 กราฟความสัมพันธ์ระหว่างอัตราทดต่อเวลา

โดยเส้นสีแดงในรูปที่ 4 คือตำแหน่งเกียร์ของรถยนต์ จากการเก็บข้อมูล นำมาแสดงกราฟเปรียบเทียบดังรูปที่ 5 จะเห็นว่าข้อมูลทั้งสี่ตัวนั้นมีความสัมพันธ์สอดคล้องไปในทางเดียวกัน

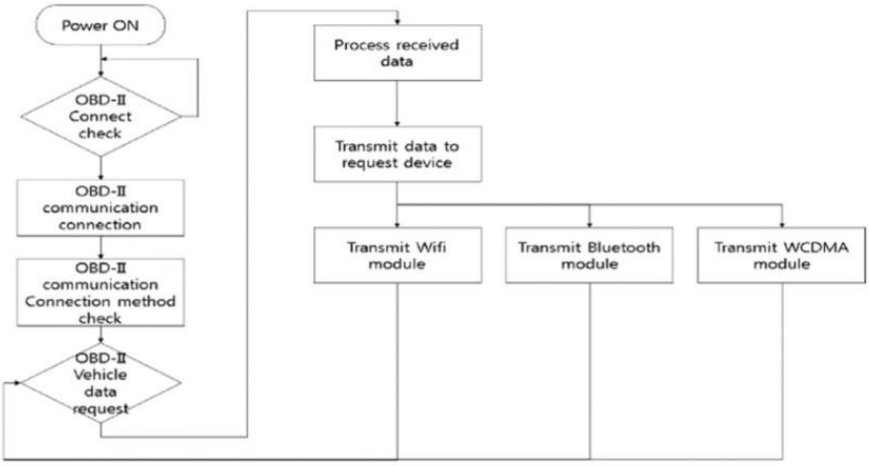


รูปที่ 5 กราฟแสดงความเร็วรอบเครื่อง ความเร็วรถยนต์ เปอร์เซ็นเหยียบคันเร่ง และเกียร์บนเวลาเดียวกัน

1.1.3 การสื่อสารไร้สาย

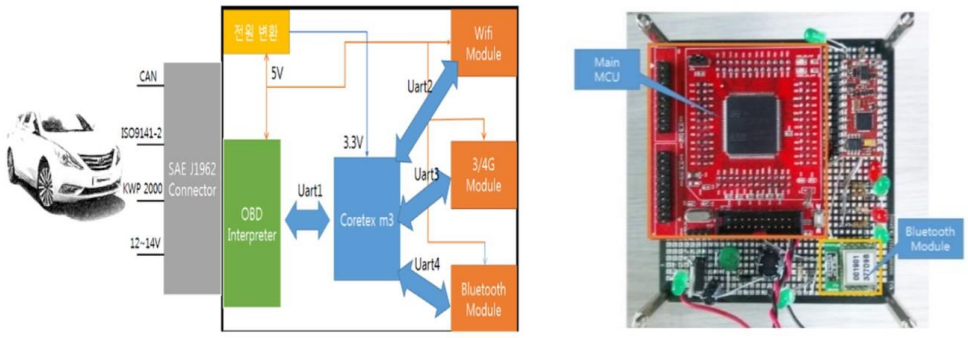
โอบีดีทูเป็นช่องทางการสื่อสารจากรถยนต์แบบใช้สาย เหมาะสำหรับกับการใช้งานในรูปแบบงานวิเคราะห์ย้อนหลัง แต่ปัจจุบันการรับรู้ถึงสถานะของรถยนต์แบบตามเวลาจริง (real time) มีความจำเป็นอย่างมากเพราะสามารถวิเคราะห์และคาดการณ์สถานการณ์ต่างๆได้ทันท่วงที เป็นเหตุให้ต้องมีการพัฒนาระบบการส่งข้อมูลแบบไร้สายเพื่อทำให้รับรู้ถึงสถานะของรถยนต์ตามเวลาจริง

การสื่อสารแบบไร้สายนั้นมีด้วยกันหลากหลายวิธีขึ้นกับการใช้งาน เช่น Bluetooth, Nb iot หรือ สัญญาณ 3G และ 4G จากการศึกษาบทความ Implementation of integrated OBD-II connector with external network[10] ได้ทำการออกแบบการสื่อสารไร้สายสำหรับดึงข้อมูลผ่านโอบีดีทูดัง รูปที่ 6 โดยเริ่มจาก เก็บข้อมูลที่ต้องการจากรถยนต์ จากนั้นตรวจสอบว่าในระบบต้องการทำการส่งข้อมูลด้วยวิธีทางใด โดยจะมีการสื่อสารออกสู่อุปกรณ์ภายนอก 3 ช่องทาง คือสื่อสารผ่านไวไฟ สื่อสารผ่านบลูทูธ และสื่อสารด้วยสัญญาณสามจี



รูปที่ 6 แผนผังการออกแบบโดยใช้การสื่อสารไร้สาย

จากการศึกษาได้ออกแบบระบบการสื่อสารไร้สายสามช่องทางดังนี้ สื่อสารผ่านทางบลูทูธสำหรับการสื่อสารสู่มือถือในช่วงใกล้ สื่อสารผ่านทาง WIFI module และ WCDMA module สำหรับการสื่อสารไร้สายทางไกล หลังจากออกแบบระบบการสื่อสารทางไกลสำเร็จ จึงออกแบบอุปกรณ์สำหรับการใช้งาน โดยมีโครงร่างอุปกรณ์ดังรูปที่ 7 โดยใช้ตัวควบคุมเป็น STM32 coretex-3 เนื่องจากอุปกรณ์ที่ออกแบบนั้นเป็นอุปกรณ์ที่ใช้ภายนอกรถยนต์ทั้งหมด หากอุปกรณ์มีปัญหาหรือได้รับความเสียหายจากการใช้งาน ทางส่วนของรถยนต์จะไม่มีความเสียหายตามตัวอุปกรณ์แต่อย่างใด



รูปที่ 7 ไดอะแกรม และ อุปกรณ์ต้นแบบสำหรับการสส่งสัญญาณไร้สาย

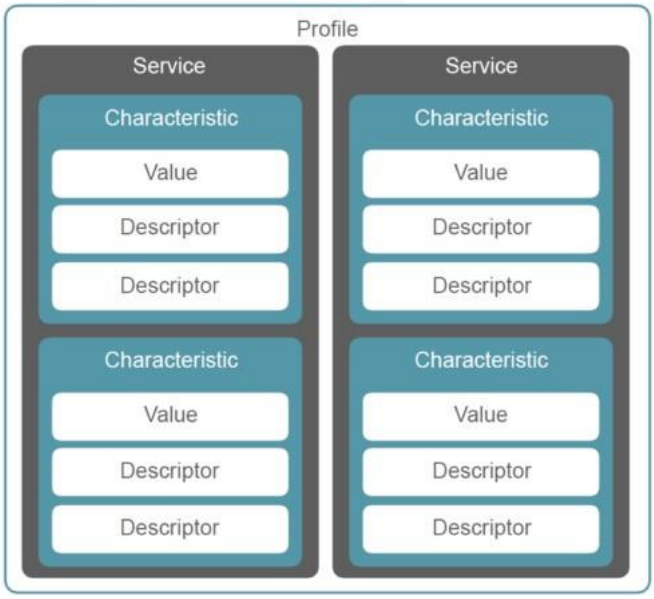
1.1.3.1 การสื่อสารด้วยบลูทูธพลังงานต่ำ

การสื่อสารกันระหว่างมือถือและอปกรณ์ IOT แบบไร้สายนั้นมีด้วยกันหลากหลายูปแบบขึ้นกับการนำไปใช้ประโยชน์ตามวัตถุประสงค์ Bluetooth Low Energy (BLE) คือการสื่อสารไร้สายประเภทหนึ่งที่เหมาะสำหรับอุปกรณ์ที่ต้องการประหยัดพลังงานและไม่จำเป็นต้องส่งข้อมูลในปริมาณมาก โดยจะมีองค์กรผู้กำหนดมาตฐานคือ Bluetooth SIG มาตรฐานเครือข่าย IEEE 802.15.1[11]

ตารางที่ 1 รายละเอียดของการสื่อสารไร้สายประเภทบลูทูธ และ เอ็นเอฟซี

	บลูทูธ 2.1	บลูทูธพลังงานต่ำ	เอ็นเอฟซี
โหมด RFID	active	active	มาตรฐาน ISO 18000-3
องค์กรผู้กำหนดมาตรฐาน	Bluetooth SIG	Bluetooth SIG	ISO/IEC
มาตรฐานเครือข่าย	IEEE 802.15.1	IEEE 802.15.1	ISO 13157 เป็นต้น
ประเภทของเครือข่าย	WPAN	WPAN	Point-to-point
การเข้ารหัส	ใช้ได้	ใช้ได้	ใช้ไม่ได้กับ RFID
ระยะทาง	~ 30 เมตร (คลาส 2)	~50 เมตร	< 0.2 เมตร
ความถี่	2.4-2.5 GHz	2.4-2.5 GHz	13.56 MHz
อัตราการส่งข้อมูล	1-3 Mbit/s	~200 kbit/s	424 kbit/s
เวลาเริ่มต้นทำงาน	< 6 วินาที	< 0.003 วินาที	< 0.1 วินาที

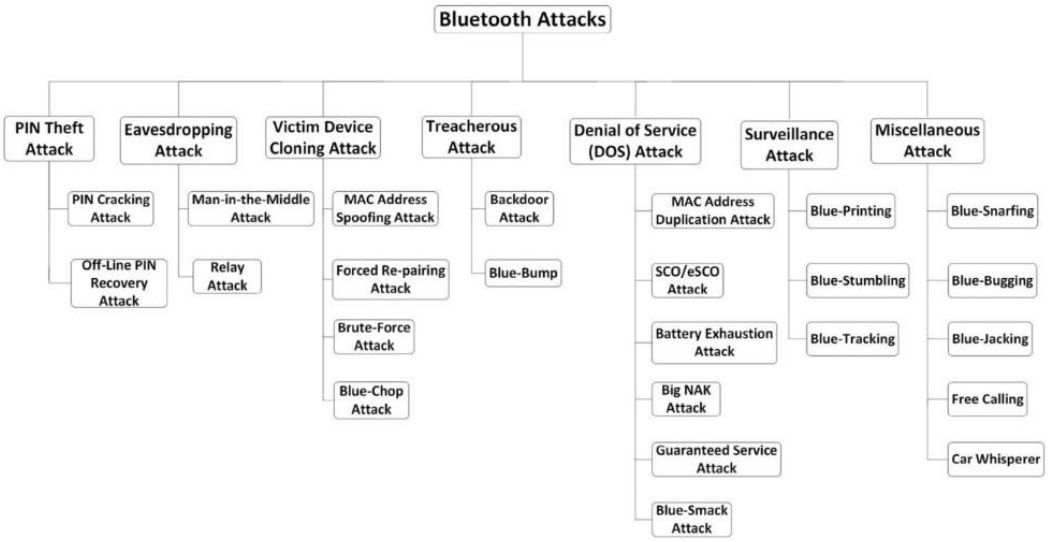
ในตัวอุปกรณ์ ESP32 จะมีชิปบลูทูธพลังงานต่ำอยู่ บลูทูธพลังงานต่ำจะมีรูปแบบการรับส่งข้อมูลที่แตกต่างกับบลูทูธทั่วไปโดยไม่จำเป็นต้องออกแบบโปรโตคอลเอง จะมีความยืดหยุ่นในการใช้งานมากกว่า การเชื่อมต่อบลูทูธพลังงานต่ำนั้นจะใช้รูปแบบการกำหนด Service ซึ่งในแต่ละ Service จะมีสิ่งที่เรียกว่า Characteristic ที่แตกต่างกันเปรียบเหมือนตัวลูกสำหรับการรับส่งข้อมูล ดังรูปที่ 8



รูปที่ 8 โครงสร้างทั่วไปของบลูทูธพลังงานต่ำ

โดยชั้น Profile นอกสุดจะเปรียบเสมือนเป็นชั้นส่วนของตัวอุปกรณ์ โดย ESP32 จะมีหมายเลข Mac Address ที่ต่างกันและไม่สามารถแก้ไขใดๆได้ ดังนั้นในการเชื่อมต่อระหว่างกันของบลูทูธพลังงานต่ำจะใช้หมายเลข Mac Address ในการอ้างอิงถึงอุปกรณ์ที่จะเชื่อมต่อด้วย ทั้งนี้ชั้น Profile สามารถกำหนดตัวชื่อของบลูทูธพลังงานต่ำเองได้ ลำดับต่อมาคือชั้น Service ในอุปกรณ์หนึ่งตัวสามารถสร้างชั้น Service ขึ้นมาได้หลายตัวขึ้นกับวัตถุประสงค์การใช้งาน โดยการสร้าง Service นั้นจะใช้การเข้ารหัสที่เรียกว่า Universally unique identifier (UUID) มักอยู่ในรูปของเลขฐานสิบหกจำนวนหนึ่งไบต์ ซึ่งในแต่ละการสร้าง Service นั้นรหัส UUID จะต้องไม่ซ้ำกัน ภายใต้ชั้นService จากรูป จะเห็นว่าในแต่ละ Service สามารถมี Characteristic ได้หลายตัว ซึ่งการสร้างตัว Characteristic นั้นจะทำด้วยวิธีเดียวกันกับการสร้าง Service จะต้องสร้างรหัส UUID ที่ไม่ซ้ำกันกับชั้น Service และตัว Characteristic ตัวอื่น โดยการรับส่งข้อมูลแต่ละครั้ง จะทำการตรวจสอบรหัส UUID ของส่วน Characteristic ว่าตรงกับส่วนที่ต้องการสื่อสารหรือไม่ ภายในส่วนของ Characteristic นั้นจะสามารถบันทึกข้อมูลในรูปของ Value และ Descriptor อื่นๆได้

แม้ว่าการสื่อสารด้วยวิธีการบลูทูธจะแพร่หลายและได้รับการยอมรับมาโดยตลอด แต่รายงานด้านปัญหาของบลูทูธมักจะเกิดขึ้นบ่อยๆโดยเฉพาะการโดนโจมตีจากผู้ไม่หวังดี โดยจะสามารถแบ่งเป็นกลุ่มหลักๆได้ทั้งหมด 7 รูปแบบดังรูปที่ 9 ซึ่งการป้องกันการถูกโจมตีในแต่ละวิธีจะแตกต่างกันไป[12]

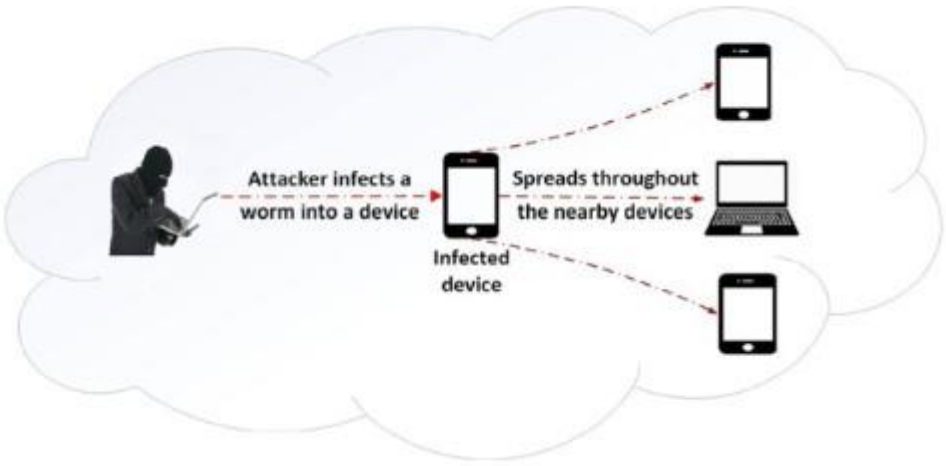


รูปที่ 9 วิธีการโจมตีทางบลูทูธเจ็ดรูปแบบ

ปัญหาอีกเรื่องที่ถูกพบเจอบ่อยๆคือมัลแวร์ (Malware) มัลแวร์คือ ซอฟท์แวร์ที่ถูกพัฒนาเพื่อโจมตีระบบต่างๆโดยเฉพาะ ซึ่งแต่ละมัลแวร์จะมีรูปแบบการโจมตีหรือก่อกวนที่ต่างกัน

รูปแบบแรกคือโทรจันเป็นมัลแวร์ประเภทหนึ่งที่หลอกให้ผู้ใช้งานบลูทูธอนุญาตเข้าถึงฟังก์ชันต่างๆ และจะส่งข้อมูลกลับไปทางผู้ประสงค์ร้าย โดยโทจันเป็นชื่อที่ได้รับมาจากม้าไม้ในสงครามเมืองทรอย ยกตัวอย่างโทรจัน เช่น Skull หากมือถือติดมัลแวร์ตัวนี้จะทำให้ไอคอนทุกอย่างบนมือถือถูกเปลี่ยนเป็นรูปหัวกะโหลก อีกทั้งไม่สามารถใช้งานโปรแกรมต่างๆได้ โดยมัลแวร์ตัวนี้ถูกแพร่กระจายอย่างกว้างขวางในปี 2006 ผ่านทางบลูทูธและ SMS

อีกรูปแบบคือ เวิร์มเป็นมัลแวร์ที่สามารถแพร่กระจายได้ด้วยตัวเอง โดยมันสามารถส่งต่อให้ผู้ที่อยู่ข้างเคียงได้ผ่านทางบลูทูธ มัลแวร์ตัวนี้มัลแวร์ตัวนี้พบเป็นอย่างมากในระบบปฏิบัติการ Symbian ยกตัวอย่างเวิร์มเช่น Comm-warrior หากมือถือหรือคอมพิวเตอร์ถูกติดมัลแวร์ตัวนี้จะไม่มีวันรู้ตัวจนกระทั่งวันที่ 14 ของทุกเดือน เวิร์มจะทำลบข้อมูลทุกอย่างบนอุปกรณ์อีกทั้งยังสามารถสร้างตัวเองซ่อนไว้ในเครื่องและส่งต่อเครื่องอื่นๆผ่านทางบลูทูธและ SMS ได้ดังรูปที่ 10

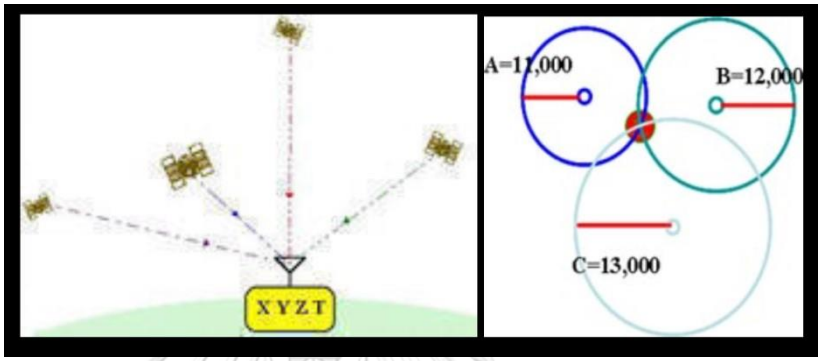


รูปที่ 10 แผนผังการโจมตีของเวิร์ม

1.1.4 การระบุตำแหน่งด้วยจีพีเอส

Global Positioning System (GPS) คือการระบุตำแหน่งของอุปกรณ์โดยพึ่งพาดาวเทียมทำงานโดยการรับสัญญาณจากดาวเทียมแต่ละดวง โดยสัญญาณดาวเทียมนี้ประกอบไปด้วยข้อมูลที่ระบุตำแหน่งและเวลาขณะส่งสัญญาณ ตัวเครื่องรับสัญญาณจีพีเอสจะต้องประมวลผลความแตกต่างของเวลาในการรับสัญญาณเทียบกับเวลาจริง ณ ปัจจุบันเพื่อแปรเป็นระยะทางระหว่างเครื่องรับ

สัญญาณกับดาวเทียมแต่ละดวง ซึ่งได้ระบุตำแหน่งของมันมากับสัญญาณดังกล่าวข้างต้น เพื่อให้เกิดความแม่นยำในการค้นหาตำแหน่งด้วยดาวเทียม ต้องมีดาวเทียมอย่างน้อย 4 ดวง เพื่อบอกตำแหน่งบนผิวโลก ซึ่งระยะห่างจากดาวเทียมทั้ง 3 กับเครื่องจีพีเอส(ที่จุดสีแดง) ดังรูปที่ 11 จะสามารถระบุตำแหน่งบนผิวโลกได้ หากพื้นโลกอยู่ในแนวระนาบแต่ในความเป็นจริงพื้นโลกมีความโค้งเนื่องจากสัณฐานของโลกมีลักษณะกลม ดังนั้นดาวเทียมดวงที่ 4 จะทำให้สามารถคำนวณเรื่องความสูงเพื่อทำให้ได้ตำแหน่งที่ถูกต้องมากขึ้น[13]

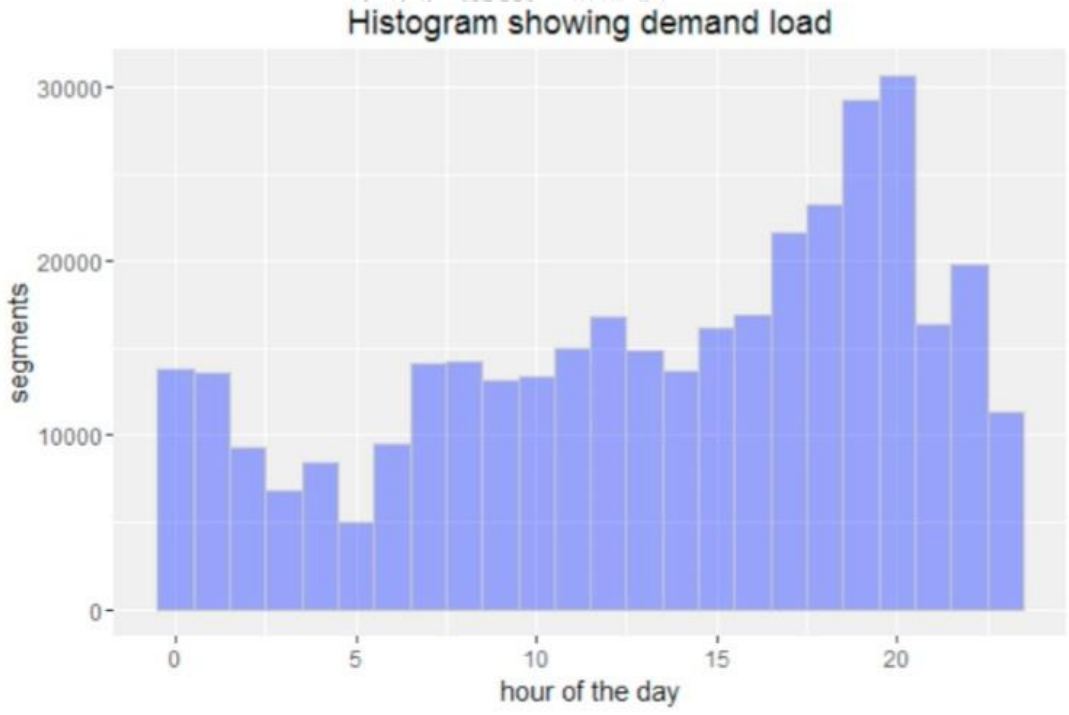


รูปที่ 11 หลักการทำงานของจีพีเอสอย่างง่าย

การนำตำแหน่งที่ได้รับจากจีพีเอสสามารถนำไปประยุกต์ใช้ได้หลากหลายมาก ไม่ว่าจะอุตสาหกรรมการขนส่ง อุตสาหกรรมโรงงาน หรืออุตสาหกรรมการแพทย์[14] ในเมือง ดาร์ เอส ซาลาม ประเทศแทนซาเนียได้ทำการสำรวจคนขับรถมอเตอร์ไซค์แทกซี่สองล้อและสามล้อเป็นจำนวนทั้งหมด 39 คน แบ่งพื้นที่การวิ่งเป็น 6 ส่วน โดยทำการติดเครื่องติดตามตำแหน่งผ่านจีพีเอสไปทั้งหมด 483 ชั่วโมง ดังตารางที่ 2 ผลทดสอบการวิ่งรถทั้ง 39 คน Segmentation คือจำนวนครั้งในการวิ่งของผู้ขับ โดยจะแบ่งแยก Segment เมื่อพบว่าตำแหน่งของผู้ขับอยู่ที่เดิมนานกว่า 120 วินาที เมื่อเก็บข้อมูลครบทั้งหมด ทำการทำความสะอาดข้อมูลโดยกรองข้อมูลที่ใช้ไม่ได้ออก จากนั้นนำข้อมูลที่ใช้ได้มาสร้างแผนภูมิดังรูปที่ 12 แผนภูมิการใช้รถตลอดวัน จะเห็นว่าช่วงที่มีการใช้รถมอเตอร์ไซค์แทกซี่จะเริ่มเพิ่มตั้งแต่เวลาหกโมงเช้า เพิ่มขึ้นเรื่อยๆจนสูงสุดในเวลาหนึ่งทุ่ม กราฟนี้ทำให้เห็นได้ชัดกว่าการสังเกตด้วยตาเปล่ามาก

ตารางที่ 2 ผลทดสอบการวิ่งรถทั้ง 39 คน

	Drivers	Valid	Segments
Mlimani City	10	9	547
Kimara Suka	7	6	238
Kigamboni	8	5	120
Mikocheni	3	2	88
Sinza	3	2	89
Kimara Stopover	7	4	73
Free flow	1	1	33
Total	39	29	1188
Two-wheelers			431
Three-wheelers			757



รูปที่ 12 แผนภูมิการใช้รถตลอดวัน

1.1.5 Firebase database

ระบบการทำงานของอุปกรณ์ IOT นั้น จำเป็นต้องมีส่วนส่งข้อมูล และส่วนรับข้อมูล อุปกรณ์ที่ทำหน้าที่สำหรับส่งข้อมูลนั้นจะถูกติดตั้งภายในรถยนต์เพื่อดึงค่าสถานะต่างๆออกมา จากนั้นจะส่ง

ข้อมูลผ่านเครือข่ายอินเทอร์เน็ตและเก็บข้อมูลเหล่านั้นไว้ในเซิฟเวอร์ดังนั้นการเลือกเซิฟเวอร์ที่เหมาะสมกับการใช้งานนั้นเป็นสิ่งที่จำเป็น

Firebase เป็น Platform จาก Google ที่มีความสามารถหลากหลายมาก สามารถนำไปประยุกต์ใช้กับงานได้หลากหลายรูปแบบ Firebase มีผลิตภัณฑ์ทั้งหมดถึง 18 อย่างและแบ่งออกเป็น 3 หมวดหมู่ ดังนี้ **[15]**

1.1.5.1 ส่งเสริมให้แอพพลิเคชันดียิ่งขึ้น

1.1.5.1.1. Realtime Database คือบริการฐานข้อมูล NoSQL ใช้วิธีการเก็บข้อมูลเป็น JSON Tree ขนาดใหญ่ และสามารถเชื่อมต่อสถานะข้ามไคลแอนท์ได้แบบ Realtime กล่าวคือ หากเชื่อมต่อฐานข้อมูลเดียวกัน 2 ที่ เมื่อใดที่ที่นึงมีการอัพเดตข้อมูล อีกที่นึงก็จะมีการอัพเดตข้อมูลให้เหมือนกันโดยอัตโนมัติ และสามารถทำงานแบบออฟไลน์ได้บนแอพพลิเคชั่นทั้ง Android และ iOS

1.1.5.1.2. Authentication คือบริการตรวจสอบผู้ใช้ โดยสามารถตรวจสอบได้หลายวิธี เช่น Email/Password, เบอร์โทรศัพท์, บัญชี Google, Facebook, Twitter, Github เป็นต้น มีฐานข้อมูลเป็นของตนเองไม่ต้องสร้างใหม่หรือออกแบบวิธีการเก็บซึ่ง สามารถดูได้ว่าสมัครด้วยวิธีไหน สมัครเมื่อไหร่ และเข้าใช้ระบบครั้งล่าสุดเมื่อไหร่

1.1.5.1.3. Hosting คือบริการฝากไฟล์เช่น HTML, CSS, JS, JPG เพื่อให้ผู้ใช้อื่น เข้าใช้งานเว็บของเราได้ นิยมใช้ในการฝากไฟล์ที่ได้จากการ Build ของ JavaScript Framework ต่าง ๆ เช่น Angular, React, Vue สังเกตว่าจะได้ไฟล์ HTML, CSS, JS ไม่จำเป็นต้องใช้ Framework ก็ได้เหมือนกัน อีกทั้งมีบริการ CDN และ SSL เพื่อให้ผู้ใช้ได้รับประสบการณ์การใช้งานที่ปลอดภัย เชื่อถือได้และไม่มีความล่าช้าแม้ว่าจะอยู่ที่ไหนก็ตาม โดยทุกเว็บมี Domain Name ของ Firebase ให้อัตโนมัติ

1.1.5.1.4. Cloud Functions คือบริการสำหรับ Deploy Function ที่พัฒนาด้วย JavaScript หรือ TypeScript เพื่อทำงานตาม Tigger ที่เกิดขึ้นบน Firebase

1.1.5.1.5. Cloud Storage คือบริการเก็บไฟล์รูปภาพ ไฟล์เสียง และวิดีโอเพื่อใช้บนแอพพลิเคชันเช่น รูปภาพประจำตัวสมาชิกและวิดิโอสอนการใช้งานโปรแกรม เป็นต้น

1.1.5.1.6. Cloud Firestore (Beta) คือ Realtime Database รุ่นใหม่มาพร้อมการค้นหาและการปรับขนาดอัตโนมัติที่มีประสิทธิภาพมากขึ้น ปรับปรุงวิธีการเก็บข้อมูลใหม่เป็น Collections และสามารถทำงานแบบออฟไลน์บนเว็บไซต์ได้เช่นกัน

1.1.5.1.7. ML Kit (Beta) คือ Machine Learning SDK ที่ช่วยให้แอพพลิเคชันมือถือสามารถใช้ความสามารถของ ML ได้ง่ายยิ่งขึ้น สามารถทำงานได้ทั้งแบบออนไลน์และออฟไลน์

1.1.5.2 ส่งเสริมคุณภาพของแอพพลิเคชัน

1.1.5.2.1. Crashlytics คือบริการตรวจจับและแจ้งเตือนหากแอพพลิเคชันเกิดอาการ Crash ขึ้นแบบ Realtime เพื่อให้แอพพลิเคชันเสถียรอยู่เสมอ โดยจะทำการแจ้งให้ทราบถึงข้อผิดพลาดและผลกระทบ ผ่านทาง E-mail และ Firebase Console เพื่อการแก้ปัญหาที่รวดเร็วและตรงจุด

1.1.5.2.2. Performance Monitoring คือบริการตรวจสอบคุณภาพของแอพพลิเคชัน เพื่อให้แอพพลิเคชันตอบสนองได้เร็วอยู่เสมอ โดยสามารถตรวจสอบเวลาและรายละเอียดการทำงานต่างๆได้ เช่น เวลาที่ใช้ในการเปิดแอพพลิเคชัน เวลาที่ใช้การเปลี่ยนหน้าการใช้งาน และขนาดข้อมูลที่ดาวน์โหลด

1.1.5.2.3. Test Lab คือบริการทดสอบแอพพลิเคชันบนฮาร์ดแวร์จริง ๆ เพื่อให้มั่นใจว่าแอพพลิเคชันสามารถรองรับฮาร์ดแวร์ที่ต้องการได้จริง โดยสามารถระบุรุ่นและเวอร์ชันที่ต้องการได้ แล้วระบุรูปแบบการทดสอบต่างๆ เพื่อทดสอบและรายงานผลกลับมา โดยไม่จำเป็นต้องมีซื้อโทรศัพท์จริง

1.1.5.3. ตอบโจทย์ต่อการพัฒนาธุรกิจ

1.1.5.3.1. In-App Messaging คือบริการแสดงข้อความป็อปอัพ (popup) ภายในแอพพลิเคชัน เช่น โฆษณา การแจ้งเตือน หรือข่าวสาร

1.1.5.3.2. Google Analytics คือบริการแสดงข้อมูลสถิติต่างๆ ของแอพพลิเคชัน เช่น มือถือทำงานด้วยระบบปฏิบัติการอะไร จำนวนเท่าไหร่ มีผู้ใช้งาน ณ ปัจจุบันกี่คน หรือใช้งานส่วนไหนบ้างเป็นต้น เพื่อวิเคราะห์กลุ่มเป้าหมาย หรือรับทราบพฤติกรรมของผู้ใช้งานต่างๆ

1.1.5.3.3. Predictions คือบริการวิเคราะห์ข้อมูลการใช้งานแอพพลิเคชัน ช่วยให้รู้ว่าผู้ใช้ใช้งานส่วนใดบ้างในแอพพลิเคชัน ช่วยให้รู้ว่าส่วนใดตอบสนองได้ดี ส่วนใดควรปรับปรุง หรืออาจต้องการที่จะหยั่งรู้พฤติกรรมในอนาคตของผู้ใช้งานแอพพลิเคชัน เพื่อวางแผนกลยุทธ์ทั้งรุกและรับ รวมทั้งสร้างประสบการณ์ที่น่าประทับใจให้กับผู้ใช้

1.1.5.3.4. Cloud Messaging คือบริการส่งการแจ้งเตือนไปยังมือถือหรือเว็บของเรา เพื่อแจ้งข้อความไปยังผู้ใช้แม้ว่าจะปิดแอปไปแล้วก็เช่น การแจ้งเตือนจาก facebook line หรือ instagram เป็นต้น

1.1.5.3.5. Remote Config คือความสามารถที่จะเปลี่ยนลักษณะการทำงานและลักษณะที่ปรากฏของแอพพลิเคชันได้ทันทีจากหน้าเว็บ Firebase โดยไม่ต้องรอการอนุมัติจาก App Store

1.1.5.3.6. Dynamic Links คือลิงค์การเชื่อมโยงไปยังแอพพลิเคชันมือถือใช้สำหรับแสดงบนหน้าเว็บเพื่อให้ผู้ใช้งานติดตั้งแอพพลิเคชันมือถือผ่านลิงค์นี้ อีกทั้งยังสามารถแนบข้อมูลต่างๆของผู้ใช้ที่อยู่บนเว็บไซต์ได้

1.1.5.3.7. App Indexing คือการปรับแต่งแอพพลิเคชันให้แสดงผลข้อมูลภายในแอพพลิเคชันบน Google Search ได้เช่น ค้นชื่อร้านอาหารแล้วปรากฏแอพพลิเคชันวงใน (Wong Nai) ขึ้นมาให้ดูรายละเอียดและรีวิว เป็นต้น

1.1.5.3.8. A/B Testing (Beta) คือความสามารถในการแสดงผลแอพพลิเคชันหลายรูปแบบเพื่อทดสอบการแสดงผลหรือการทำงาน ว่าสิ่งไหนจะมอบประสบการณ์การใช้งานที่ดีกว่าให้แก่ผู้ใช้งาน เช่น การวางปุ่มกดแบบไหนที่ผู้ใช้งานใช้สะดวก สมมุติว่ามีผู้ใช้งาน 100 คน อาจจะมี 50 คนได้ปุ่มที่อยู่มุมบน อีก 50 คน

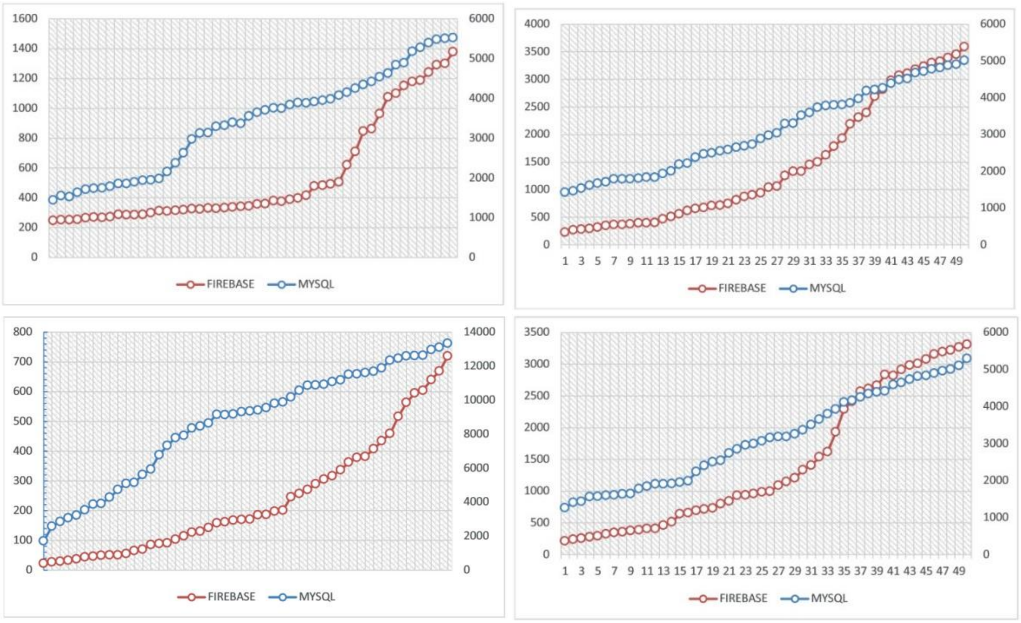
ได้ปุ่มอยู่มุมล่าง หากว่ามีการใช้งานแบบไหนมากกว่ากันก็อาจจะสรุปผลและเลือกใช้แบบนั้นกับทุกคนในท้ายที่สุด

เปรียบเทียบศักยภาพการทำงานระหว่าง Firebase Realtime Database กับ MySQL Database ด้วยวิธี CRUD operation[16] เนื่องจากความต้องการของระบบติดตามรถยนต์นั้นจำเป็นต้องปรับข้อมูลให้เป็นปัจจุบันมากที่สุด ดังนั้นเซิฟเวอร์ที่ใช้ในระบบจะต้องสามารถทำงานแบบ Realtime ได้ CRUD operation คือการทดสอบระบบ 4 รูปแบบโดย C = Create คือทดสอบการสร้างข้อมูลในระบบ R = Read คือทดสอบการเข้าถึงข้อมูลในระบบ U = Update คือทดสอบการปรับปรุงข้อมูลในระบบ และ D = Delete คือทดสอบการลบข้อมูลออกจากระบบ โดยข้อมูลแต่ละชุดนั้นจะถูกจำลองให้มีหน้าตาดัง รูปที่ 13

```
{
  "child_id": "LXY8GZSoX84ExkwNaEy",
  "child_weight": 5,
  "child_age": 9,
  "status_score": 1.2,
  "status_gizi": "Gizi Baik",
  "date_save": "1-1-2019"
}
```

รูปที่ 13 ตัวอย่างข้อมูลที่ใช้ในการทดสอบ

ซึ่งการทดสอบแต่ละครั้งนั้นจะทำการทดสอบกับข้อมูลทั้งหมด 50 ชุด โดยข้อมูลชุดละ 3000 ข้อมูล จากการทดลองทั้ง 4 แบบ สามารถสรุปได้ว่า Firebase Realtime Database สามารถทำงานได้เร็วกว่า MySQL database ดังรูปที่ 14 โดยแกนนอนคือจำนวนข้อมูลทั้ง 50 ชุด แกนตั้งทางซ้ายคือระยะเวลาทำงานของ Firebase database แกนตั้งทางขวาคือระยะเวลาทำงานของ MySQL database



รูปที่ 14 กราฟการเปรียบเทียบระหว่าง Firebase และ MySQL ด้วยวิธี CRUD

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อพัฒนาระบบติดตามสถานะของรถยนต์สำหรับธุรกิจรถเช่า

1.2.2 เพื่อออกแบบอุปกรณ์สำหรับติดตั้งภายในรถยนต์และรองรับกับโมบายแอพพลิเคชั่นในอนาคต

1.3 ขอบเขตของงานวิจัย

1.3.1 พัฒนาอุปกรณ์ที่ใช้ร่วมกับรถยนต์ Toyota altis 2016

1.3.2 ใช้โปรแกรม Arduino ide สำหรับการเขียนโปรแกรม

1.4 ระเบียบขั้นตอนของงานวิจัย

1.4.1 หารือร่วมกับบริษัทรถเช่าเพื่อหาฟังก์ชั่นที่จำเป็นของอุปกรณ์

1.4.2 ศึกษาระบบติดตามรถยนต์ที่มีอยู่แล้วในท้องตลาดทั้งภายในประเทศและต่างประเทศ

1.4.3 ศึกษางาน วิจัย บทความทางวิชาการที่เกี่ยวข้องกับธุรกิจรถเช่าและอุปกรณ์ที่จำเป็น

1.4.4 ออกแบบอุปกรณ์สำหรับติดตั้งภายในรถยนต์และระบบที่สามารถประมวลผลออนไลน์ได้

1.4.5 สร้างอุปกรณ์จำลองและส่งให้บริษัทรถเช่าเพื่อทำการทดลองโดยการใช้งานจริง

1.4.6 สรุปผลงานวิจัยและหาข้อเสนอแนะ

1.4.7 จัดทำรูปเล่มวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

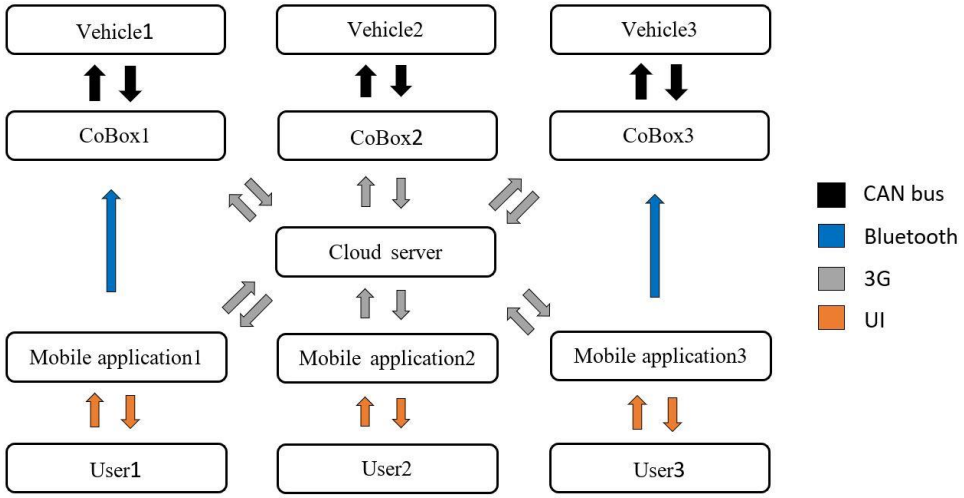
งานวิจัยนี้เป็นการศึกษาและออกแบบอุปกรณ์ใหม่เพื่อติดตามสถานะของรถยนต์ โดยระบบติดตามจะต้องมีความสามารถที่มากกว่าระบบในท้องตลาดและสามารถนำไปใช้ได้จริงในธุรกิจรถเช่า

บทที่ 2
ระเบียบวิธีดำเนินงาน

ในบทนี้จะเล่าถึงการออกแบบอย่างละเอียด โดยจะแยกหัวข้อหลักออกเป็นสามส่วนคือส่วนการออกแบบระบบการทำงานของอุปกรณ์ติดตามรถยนต์ และส่วนต่อมาการออกแบบตัวอุปกรณ์ที่ต้องการติดเข้าไปในรถยนต์ หลังจากนี้จะเรียกอุปกรณ์นี้ว่า CoBox และส่วนสุดท้ายคือการเขียนโปรแกรมสำหรับการทำงานภายในกล่อง CoBox

2.1 ออกแบบระบบ

จากกระบวนการศึกษาความต้องการของระบบ และศึกษาวรรณกรรมย้อนหลัง จึงได้ทำการออกแบบระบบติดตามสถานะของรถยนต์ได้ดังรูปที่ 15 โดยมีส่วนประกอบหลัก 5 ส่วนด้วยกันคือ รถยนต์ อุปกรณ์ติดตาม คลาวด์เซิฟเวอร์ แอพพลิเคชั่นบนมือถือ และผู้ใช้งาน



รูปที่ 15 แผนผังการทำงานของระบบติดตามรถยนต์

รถยนต์ในปัจจุบันนั้นจะใช้มาตรฐานร่วมกันสำหรับการเตรียมข้อมูลในโอบีดีทูซึ่งรถแต่ละยี่ห้อจะมีรหัสสำหรับการเข้าถึงค่าจากเซนเซอร์แต่กต่างกัน แต่รถภายในรุ่นเดียวกันจะใช้รหัสที่ใกล้เคียงกัน จากการศึกษาวรรณกรรม[7] เราจะทำการดึงข้อมูลออกจากช่องโอบีดีทู โดยการดึงสัญญาณจากช่อง CAN BUS ข้อมูลทั้งหมดจากเซนเซอร์ในรถยนต์จะเข้าสู่อุปกรณ์ที่ถูกตอดตั้ง จากนั้นทำการถอดรหัสข้อมูลเหล่านั้นว่าเป็นข้อมูลที่จำเป็นต่อระบบติดตามสถานะรถยนต์หรือไม่ กรองข้อมูลที่ไม่สำคัญออกและเก็บไว้เฉพาะข้อมูลที่จำเป็น ภายในอุปกรณ์ CoBox จะมีการวิเคราะห์ข้อมูลเบื้องต้น และจะทำการรับส่งชุดข้อมูลออกไปสู่เซิฟเวอร์ เพื่อเก็บข้อมูลต่อๆไป

คลาวด์เซิฟเวอร์สำหรับระบบติดตามรถยนต์นั้นเป็นอีกหนึ่งส่วนประกอบที่สำคัญมาก หากขาดส่วนนี้ไป ผู้ใช้งานจะไม่สามารถใช้งานระบบติดตามสถานะรถยนต์ได้ เซิฟเวอร์ทำหน้าที่เก็บข้อมูลจำนวนมหาศาลที่ถูกส่งมาจาก CoBox หลายๆตัวเก็บมารวมไว้ที่เดียว ระบบติดตามรถยนต์นั้นจะใช้ Firebase real-time database เป็นเซิฟเวอร์[16] ซึ่งเซิฟเวอร์สามารถทำงานแบบ real-time ได้อีกทั้งโครงสร้างยังเหมาะสำหรับการพัฒนาแอพพลิเคชั่นต่อไป นอกจากเซิฟเวอร์จะทำหน้าที่รับส่งข้อมูลจำนวนมากกับ CoBox แล้ว เซิฟเวอร์จะต้องทำหน้าที่รับส่งข้อมูลต่อไปยังแอพพลิเคชั่นบนมือถือในเวลาเดียวกันด้วย

แอพพลิเคชั่นบนมือถือนั้นถือเป็นหน้าตาหลักของระบบเพราะเป็นเพียงส่วนเดียวที่ผู้ใช้งานระบบนี้จะได้สัมผัส โดยแอพพลิเคชั่นนี้จะทำหน้าที่เป็นตัวกลางการสื่อสารระหว่างผู้ใช้และรถยนต์ โดยผู้ใช้นั้นจะสามารถมองเห็นสถานะของรถยนต์ผ่านทางแอพพลิเคชั่นบนมือถือได้ด้วยการเชื่อมต่อผ่านอินเทอร์เน็ต อีกทั้งสามารถสั่งคำสั่งเปิด-ปิดล้อคประตูรถยนต์อีกด้วย

2.2 ออกแบบอุปกรณ์ต้นแบบ

อุปกรณ์ IOT สำหรับติดตั้งภายในรถยนต์นี้ถือเป็นหัวใจสำคัญหลักของระบบติดตามรถยนต์ หน้าที่สำคัญของอุปกรณ์ตัวนี้จะต้องสามารถดึงข้อมูลจากรถยนต์ผ่านทางช่องแคนบัส (CAN BUS) รับสัญญาณตำแหน่งจากโมดูลจีพีเอสรับส่งข้อมูลผ่านทางอินเทอร์เน็ตเพื่อสื่อสารกับคลาวด์เซิฟเวอร์ และรับข้อมูลคำสั่งจากแอพพลิเคชั่นบนมือถือผ่านทางบลูทูธ

ในส่วนรับข้อมูลจากรถยนต์ผ่านสัญญาณแคนบัสเลือกใช้ MCP2515 ดังรูปที่ 16.ก เป็นตัวรับส่งข้อมูล โดยตัวโมดูลจะรับข้อมูลแคนบัสออกจากรถยนต์ที่ความเร็ว 1000 Kbps เมื่อข้อมูลเข้าสู่ตัวโมดูลแล้ว ตัวโมดูลจะแปลสัญญาณข้อมูลแล้วส่งให้กับทางตัวควบคุมเป็นสัญญาณ SPI(Serial Peripheral Interface) โดยจะส่งสัญญาณเป็น 4 ช่อง คือ SCK ใช้สำหรับส่งสัญญาณนาฬิกา MISO (ย่อมาจาก Master In Slave Out) ใช้สำหรับรับข้อมูลจาก Slave MOSI (ย่อมาจาก Master Out Slave In) ใช้สำหรับส่งข้อมูลจาก Master ไปยัง Slave และสุดท้าย SS/CS (ย่อมาจาก Slave Select/Chip Select) ใช้สำหรับเลือก Slave ที่ต้องการใช้งาน

ในส่วนรับส่งข้อมูลตำแหน่งของอุปกรณ์เลือกใช้โมดูลจีพีเอส Quectel L86 ดังรูปที่ 16.ข โดยโมดูลจะทำหน้าที่รับสัญญาณตำแหน่งจากดาวเทียมเป็นพิกัดละจิจูดและลองจิจูด โดยภายในโมดูลจะต้องมีถ่านสำหรับเลี้ยงระบบให้รับสัญญาณตำแหน่งจากดาวเทียมได้ตลอดเวลา จากนั้นจะทำการส่งสัญญาณตำแหน่งให้ตัวควบคุมเป็นสัญญาณซีเรียล(Serial) ข้อเสียของโมดูลจีพีเอสโดยทั่วไป

คือไม่สามารถรับสัญญาณตำแหน่งได้ในจุดอับสัญญาณหรือจุดที่มีสัญญาณอ่อน นั่นคือเมื่อทำการติดตั้งโมดูลจีพีเอสภายในรถยนต์ อาจทำให้สัญญาณตำแหน่งส่งมาถึงได้อยากขึ้น จึงต้องติดเสาสัญญาณออกมาภายนอกเพื่อให้รับสัญญาณได้ชัดยิ่งขึ้น ดังรูปที่ 16.ค

อีกหนึ่งหน้าที่สำคัญของอุปกรณ์คือการสื่อสารระหว่างอินเทอร์เน็ต เลือกใช้โมดูลสามจี Quectel UC15 ดังรูปที่ 16.ง สามจีโมดูลตัวนี้จะใช้ AT command ในการสื่อสาร โดยจะสื่อสารกับตัวควบคุมโดยใช้ Serial โดยสามจีโมดูลนี้จะทำหน้าที่เป็นตัวกลางการสื่อสารระหว่างตัวควบคุม และ Fire base database โดยในการส่งข้อมูลขึ้นเซิฟเวอร์จะใช้คำสั่ง PUT หรือ PUSH ส่วนรับข้อมูลจากเซิฟเวอร์จะใช้คำสั่ง GET และหากต้องการรบข้อมูลออกจากเซิฟเวอร์จะใช้คำสั่ง DELETE ในที่นี้จะใช้ซิมเติมเงินของทรูในการรับสัญญาณสามจี

รูปที่ 16.ก MCP2515

รูปที่ 16.ข โมดูลจีพีเอส

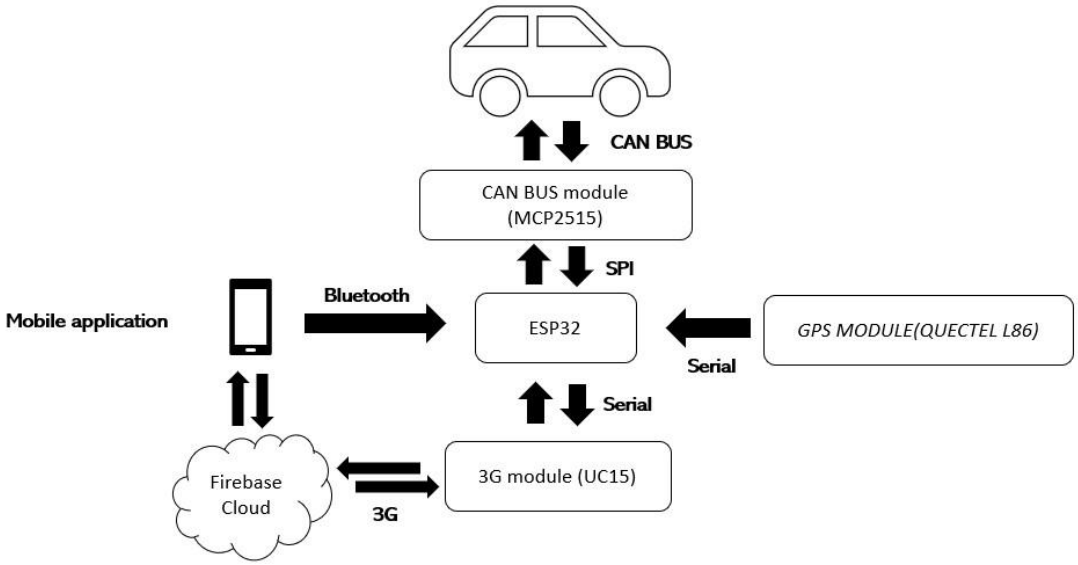
รูปที่ 16.ค เสาอากาศสำหรับจีพีเอส

รูปที่ 16.ง สามจีโมดูล

รูปที่ 16.จ ESP32

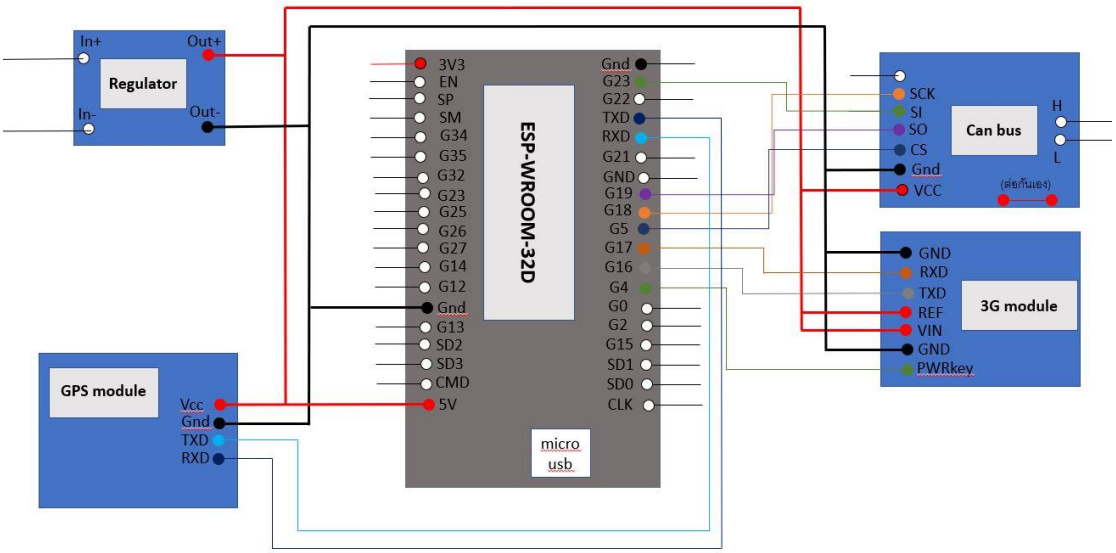
รูปที่ 16 โมดูลภายในกล่องต้นแบบ

ตัวควบคุมเป็นเหมือนสมองของระบบ คอยสั่งการการทำงานให้แก่โมดูลต่างๆที่เชื่อมต่อหากัน โดยเลือกใช้ ESP32 เป็นตัวควบคุมดังรูปที่ 16.จ มีหน้าที่หลัก 5 หน้าที่ 1. กรองข้อมูลจากโมดูลแคนบัส ข้อมูลที่ต้องใช้จากรถยนต์คือ ความเร็วรถยนต์ ความเร็วรอบเครื่อง ระดับน้ำมัน และเลขไมล์ โดยทั่วไปข้อมูลทั้งหมดจะถูกส่งผ่านโมดูลแคนบัส ดังนั้นการกรองข้อมูลที่ไม่จำเป็นออกจึงเป็นหน้าที่สำคัญ 2. รับข้อมูลตำแหน่งจากโมดูลจีพีเอส 3. ถอดรหัส CAN ID ให้เป็นข้อมูลที่สามารถเข้าใจได้โดยง่าย และรวบรวมข้อมูลที่ต้องใช้มัดรวมเป็นข้อมูลหนึ่งชุด 4. รับส่งข้อมูลกับสามจีโมดูลโดยใช้ AT command สั่งการผ่านทางช่องซีเรียล 5. รับข้อมูลจากแอพพลิเคชั่นมือถือผ่านทางบลูทูธ โดยใช้บลูทูธพลังงานต่ำภายใน บอร์ดESP32 โดยแผนผังการทำงานของตัวอุปกรณ์มีดังรูปที่ 17



รูปที่ 17 แผนผังการทำงานของแต่ละโมดูลภายในอุปกรณ์ต้นแบบ

หลังจากศึกษาการใช้งานอุปกรณ์ภายใน CoBox จึงเริ่มทำการประกอบชิ้นส่วนเข้าด้วยกัน เนื่องจากไฟเลี้ยงภายในรถยนต์จะมีแรงดันอยู่ในช่วง 11-14 v โดยปกติจะอยู่ที่ประมาณ 12 v จึงต้องมีตัวปรับแรงดันลงให้อยู่ในระดับใช้งานซึ่งในที่นี้จะปรับลงมาให้เหลือที่แรงดัน 5 v โดยไฟที่ออกจากตัวปรับแรงดันนี้จะเป็นไฟเลี้ยงสำหรับการทำงานของทุกโมดูลใน CoBox โดยเชื่อมต่อขา Out+ และ Out- ของตัวปรับแรงดันเข้ากับขา Vcc และ Gnd ของทุกโมดูลตามลำดับ หลังจากนั้นเริ่มต้นประกอบจากประกอบโมดูลแคนบัส(MCP2515) เข้ากลับบอร์ด ESP32 อุปกรณ์สองชิ้นนี้สื่อสารผ่านทาง SPI ซึ่งจะต้องทำการเชื่อมต่อกันสี่เส้น ต่อสายสัญญาณเวลา (SCK) เข้ากับขาที่18 ของESP32 ต่อสาย MOSI เข้ากับขาที่23 ของบอร์ด ESP32 ต่อสาย MISO เข้ากับขาที่19 ของ ESP32 ต่อสาย CS เข้ากับขาที่5 ของ ESP32 ซึ่งจะต้องตั้งค่าอีกครั้งในส่วนการเขียนโปรแกรม ต่อไปทำการเชื่อมต่อกับโมดูลสามจียูซี15 กับบอร์ด ESP32 เริ่มจากต่อสายสัญญาณซีเรียล RXD และ TXD เข้ากับขาที่17 และ 16 ของESP32 ซึ่งจะต้องตั้งค่าอีกครั้งใน่ส่วนของการเขียนโปรแกรม ส่วนขา REF จะต่อกับขา 5v ของESP32 ส่วนขาสุดท้าย PWRkey คือขาสำหรับสั่งเปิด-ปิดการทำงานของโมดูลสามจี จะทำการต่อกับขาที่4 ของ ESP32 สุดท้ายทำการประกอบโมดูลจีพีเอสกับบอร์ด ESP32 โดยต่อสัญญาณซีเรียล TXD และ RXD ของโมดูลจีพีเอสกับขา RXD และ TXD ของบอร์ด ESP32 ตามลำดับดังรูปที่ 18



รูปที่ 18 แผนผังการเชื่อมต่อโมดูลภายใน CoBox

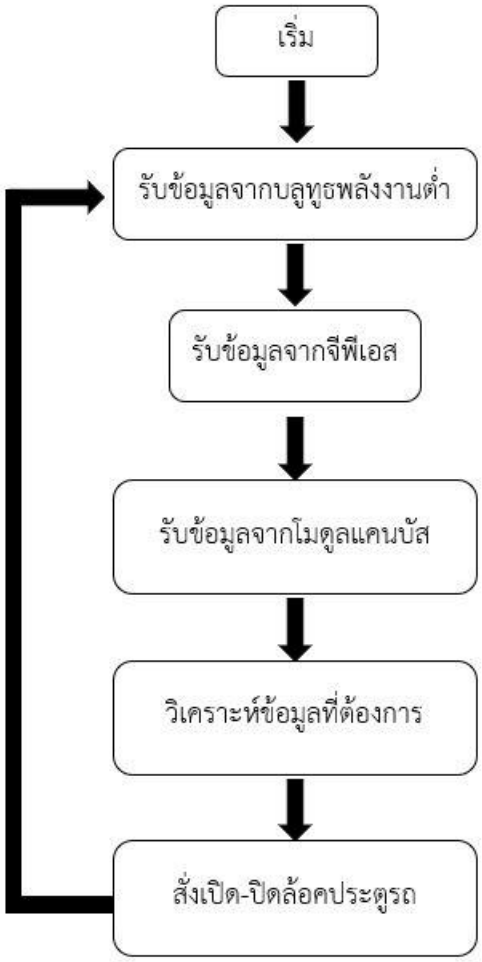
2.3 เขียนโปรแกรมสำหรับการทำงานของกล่อง CoBox

การเขียนเฟิร์มแวร์สำหรับการทำงานนั้นมีปัจจัยมากมายที่ต้องคำนึง การที่จะทำให้กล่องสามารถทำงานได้ตามที่ต้องการนั้นควรเริ่มจากการออกแบบผังการทำงานของโปรแกรม เพื่อให้เห็นภาพชัดและไม่เกิดการซ้อนทับกันระหว่างทำงาน หลังจากหารือกับทางบริษัทรถเช่าและอาจารย์ผู้เชี่ยวชาญ การทำงานของกล่องจะแสดงให้เห็นชัดดังนี้

2.3.1 แผนผังการทำงานของโปรแกรม

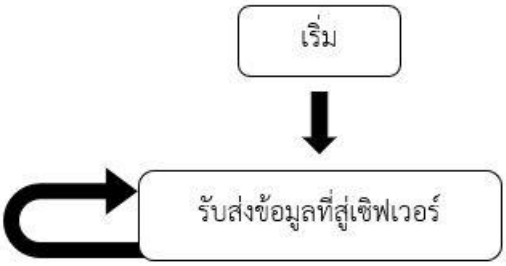
เนื่องจากระบบการทำงานของกล่อง CoBox นั้นจะค่อนข้างซับซ้อน จึงเลือกที่จะแยกการทำงานออกเป็นสองส่วน โดยส่วนแรกจะเน้นไปที่การทำงานฝั่งรับและวิเคราะห์ข้อมูลที่ได้มาจากรถยนต์ และอีกส่วนจะเป็นส่วนที่รับส่งข้อมูลผ่านทางสัญญาณสามจี

2.3.1.1 ส่วนแรกเป็นส่วนโปรแกรมที่รับข้อมูล โดยเมื่อเข้าสู่วังวนทำงาน จะเริ่มจากการตั้งค่าการทำงานของบลูทูธพลังงานต่ำในบอร์ด ESP32 จากนั้นเขียนโปรแกรมเช็คข้อมูลตำแหน่งจากโมดูลจีพีเอส หากรับข้อมูลตำแหน่งได้จะทำการบันทึกข้อมูลลงไปในตัวแปรโดยจะรับเป็นค่าละจิจูดและลองจิจูด หลังจากนั้นจะทำการรับข้อมูลจากโมดูลแคนบัส เมื่อเขียนส่วนรับข้อมูลจากทั้งสองโมดูลแล้วจึงนำข้อมูลมากรองและวิเคราะห์ข้อมูลเบื้องต้น ทั้งหมดนี้จบในส่วนของการรับข้อมูลจากรถยนต์ สุดท้ายนี้จะเป็นโปรแกรมส่วนเช็คว่ามีคำสั่งเปิด-ปิดล้อคประตูรถยนต์หรือไม่ดังรูปที่ 19



รูปที่ 19 แผนผังการทำงานส่วนรับข้อมูลจากรถยนต์

2.3.1.2 ส่วนนี้จะแยกออกมาจากส่วนที่เขียนโปรแกรมรับข้อมูลออกมาเนื่องจากช่วงเวลาการทำงานสำหรับการรับส่งข้อมูลเข้าสู่ Firebase database จะใช้เวลาค่อนข้างนานเมื่อเทียบกับเวลาในส่วนการรับส่งข้อมูลกับรถยนต์ ดังนั้นหากเขียนโปรแกรมรวมกันจะทำให้เกิดการตกหล่นของข้อมูลจากรถยนต์ ดังนั้นเริ่มเขียนโปรแกรมส่วนนี้เขียนในส่วนการรับส่งข้อมูลกับเซิฟเวอร์ผ่านโมดูลสามจีผ่านสัญญาณซีเรียลโดยจะมีแผนผังดังรูปที่ 20 โดยรายละเอียดของการทำงานระหว่างบอร์ด ESP32 และโมดูลสามจีจะถูกอธิบายอย่างละเอียดในหัวข้อถัดไป



รูปที่ 20 แผนผังการทำงานส่วนรับส่งข้อมูลผ่านอุปกรณ์ไร้สาย

2.3.2 การเขียนโปรแกรมแบบ Multitasking ของ Free RTOS

จากในหัวข้อก่อนหน้านี้ได้ทำการแบ่งการทำงานของโปรแกรมภายในบอร์ด ESP32 ออกเป็นสองส่วน ซึ่งโปรแกรมทั้งสองส่วนนี้จะทำงานพร้อมกันโดยการเขียน Multitasking ของ Free RTOS หลายๆไมโครคอนโทรลเลอร์นั้นไม่สามารถเขียนการทำงานในรูปแบบนี้ได้เนื่องจากมีตัวประมวลผลเพียงแค่ตัวเดียว ในทางกลับกันบอร์ด ESP32 สามารถทำงานสองส่วนในเวลาเดียวกันได้เนื่องจากมีตัวประมวลผลภายในบอร์ดอยู่สองตัว ในการใช้งานทั่วไปแล้วหากไม่มีการตั้งค่าการทำงานของตัวประมวลผลอีกตัวแล้วการทำงานทั้งหมดจะทำงานโดยใช้เพียงตัวประมวลผลตัวเดียวเท่านั้น ข้อดีของการเขียนแบบ Multitasking คือจะสามารถเขียนโปรแกรมให้ทำงานพร้อมกันได้แบบไม่ต้องมีความสัมพันธ์ใดๆร่วมกัน

การเขียนโปรแกรมมัลติทาสกิงเริ่มจากการประกาศตัวแปรต่างๆที่ต้องใช้ดังรูปที่ 21 โดย Task1code และ Task2code จะเป็นเหมือนฟังก์ชันวนสำหรับการทำงานสองส่วน

```
115
116  xTaskCreatePinnedToCore(
117                    Task1code,   /* Task function. */
118                    "Task1",     /* name of task. */
119                    10000,       /* Stack size of task */
120                    NULL,        /* parameter of the task */
121                    2,           /* priority of the task */
122                    &Task1,      /* Task handle to keep track of created task */
123                    0);          /* pin task to core 0 */
124  delay(500);
125
126  //create a task that will be executed in the Task2code() function, with priority 1 and executed on core 1
127  xTaskCreatePinnedToCore(
128                    Task2code,   /* Task function. */
129                    "Task2",     /* name of task. */
130                    10000,       /* Stack size of task */
131                    NULL,        /* parameter of the task */
132                    10,           /* priority of the task */
133                    &Task2,      /* Task handle to keep track of created task */
134                    1);          /* pin task to core 1 */
135    delay(500);
136
137 }
```

รูปที่ 21 ประกาศตัวแปรสำหรับการทำงานมัลติทาสกิ้ง

หลังจากตั้งค่าการทำงานเรียบร้อย ส่วนสำคัญคือการเขียนโปรแกรมภายในลูป โดยจะเขียนโปรแกรมส่วนแรกลงไปในลูปของ Task1code และโปรแกรมอีกส่วนลงไปใน Task2code และจะปล่อยให้ Void loop ว่างดังรูปที่ 22

```
140
141 void Task1code( void * pvParameters ){
142   for(;;)
143   {
144
145   //โปรแกรมส่วนที่หนึ่ง
146
147   }
148
149 }
150
151 void Task2code( void * pvParameters ){
152   for(;;)
153   {
154
155   //โปรแกรมส่วนที่สอง
156
157   }
158
159 }
160
161 void loop(){
162
163   //ปล่อยว่างไว้...
164
165 }
166
```

รูปที่ 22 ตัวอย่างโครงสร้างการเขียนโปรแกรมมัลติทาสกิ้ง

2.3.3 การเขียนโปรแกรมสำหรับควบคุมโมดูลแคนบัส

โปรแกรมในส่วนนี้ถูกเขียนลงในโปรแกรมส่วนที่หนึ่ง โดยโปรแกรมส่วนควบคุมโมดูลแคนบัสดังรูปที่ 16.ก ให้ดึงข้อมูลออกจากรถยนต์ได้ตามที่ต้องการผ่านช่องโอบีดีทู ก่อนที่จะเริ่มเขียนโปรแกรมส่วนควบคุมโมดูลนี้จะต้องรู้หลักการทำงานของโมดูลเบื้องต้นเสียก่อน โดยโมดูลลจะรับส่งข้อมูลออกจากช่องโอบีดีทูผ่านทางขา CAN high และขา CAN low เข้าสู่โมดูลในขา H และ L ตามลำดับ เมื่อดึงข้อมูลจากรถยนต์เข้าโมดูลแล้วจะส่งข้อมูลออกไปสู่บอร์ด ESP32 ผ่านทาง SPI การเขียนโปรแกรมจะแบ่งออกเป็นสองส่วนคือส่วนรับข้อมูลจากรถยนต์และส่วนส่งข้อมูลให้รถยนต์เพื่อสั่งการล้อคและปลดล้อครถยนต์

เริ่มจากโปรแกรมส่วนรับข้อมูลจากรถยนต์ เริ่มการเขียนจากตั้งค่าตัวแปรและการทำงานของขาพิน CS ไว้ที่ขาที่ 5 ของบอร์ด ESP32 จากนั้นตั้งค่าให้โมดูลรับส่งข้อมูลอยู่ที่ความถี่ 1000 Kbps ดังรูปที่ 23

```
#include <SPI.h>
#include "mcp_can.h"
long unsigned int rxId;
unsigned long rcvTime;
unsigned char len = 0;
unsigned char buf[8];
const int SPI_CS_PIN = 5;
MCP_CAN CAN(SPI_CS_PIN);                                    // Set CS pin
void setup()
{
    Serial.begin(115200);
    SPI.begin();

    while (CAN_OK != CAN.begin(CAN_1000KBPS))              // init can bus : baudrate = 500k
    {
        Serial.println("CAN BUS Module Failed to Initialized");
        Serial.println("Retrying....");
        delay(200);
    }
  }

```

รูปที่ 23 โปรแกรมควบคุมโมดูลแคนบัสส่วนการตั้งค่า

หลังจากนั้นอัพโหลดโปรแกรมลงไปในบอร์ดESP32 แล้วเช็คการทำงานของโมดูลแคนบัสผ่านซีเรียลมอนิเตอร์(Serial monitor) เมื่อโปรแกรมเริ่มทำงานทาง Library จะเช็คการต่อสายสัญญาณระหว่างบอร์ดESP32 และโมดูลแคนบัสว่าผิดพลาดหรือมีปัญหาหรือไม่ หากผิดพลาดทางโปรแกรมจะติดอยู่ใน While loop ในบรรทัดที่14 รูปที่ 23 โปรแกรมควบคุมโมดูลแคนบัสส่วนการตั้งค่า ไม่สามารถทำงานต่อได้จนกว่าจะแก้ไขการต่อสายสัญญาณ

หลังจากตั้งค่าและเชื่อมต่อโมดูลแคนบัสเสร็จสิ้นจึงเขียนโปรแกรมในส่วนรับข้อมูลจากรถยนต์ดังรูปที่ 24 โดยเริ่มจากเช็คว่ามีข้อมูลถูกส่งเข้าไปในโมดูลหรือไม่ หากมีข้อมูลส่งมาจะทำการเก็บข้อมูลชุดนั้นเอาไว้ และบันทึกเวลาในขณะนั้นเช่นกัน การสื่อสารด้วยแคนบัสภายในรถยนต์จะสื่อสารในรูปของเลขฐานสิบหก ดังนั้นในการเขียนโปรแกรมเพื่อให้เข้าใจง่ายจึงต้องปรับเปลี่ยนเป็นเลขฐานสิบในบรรทัดที่ 50 57 และ 61 ดังรูปที่ 24

```
void loop()
{

    if(CAN_MSGAVAIL == CAN.checkReceive())            // check if data coming
    {
        rcvTime = millis();
        CAN.readMsgBuf(&len, buf);    // read data,  len: data length, buf: data buf

        rxId= CAN.getCanId();

        Serial.print(rcvTime);
        Serial.print("\t\t");
        Serial.print("0x");
        Serial.print(rxId, DEC);
        Serial.print("\t");

        for(int i = 0; i<len; i++)    // print the data
        {
            if(buf[i] > 15){
              Serial.print("0x");
              Serial.print(buf[i], DEC);
            }
          else{
              Serial.print("0x0");
              Serial.print(buf[i], DEC);
          }

            Serial.print("\t");
        }
        Serial.println();
    }

}
```

รูปที่ 24 โปรแกรมควบคุมโมดูลแคนบัสส่วนดึงข้อมูล

เมื่อโมดูลเริ่มทำงาน บอร์ดESP32 จะแสดงค่าที่ได้รับจากโมดูลแคนบัสผ่านทางจอซีเรียลมอนิเตอร์ดังรูปที่ 25

```
1645407         0x705   0x08    0x00    0x00    0xFF    0x9A    0xC0    0x00    0x2C
1645412         0x611   0x07    0xFD    0x1F    0x00    0xB4    0x80    0xC3
1645416         0x548   0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
1645421         0x610   0x00    0x00    0x00    0x01    0x6A
1645425         0x180   0x00    0x00    0x00    0x00    0x00    0x00    0x00    0xBC
1645430         0x865   0x00    0x22    0x00    0x00    0x00    0x00    0x00    0x00
1645435         0x170   0x1A    0x6F    0x1A    0x6F    0x1A    0x6F    0x1A    0x6F
1645440         0x1017  0x51    0x78    0x26    0x02    0x00    0x00    0x00    0xF5
1645445         0x170   0x1A    0x6F    0x1A    0x6F    0x1A    0x6F    0x1A    0x6F
1645450         0x452   0x03    0xDA    0x28    0x08    0x3A    0x00    0x7F    0x93
1645455         0x456   0x80
```

รูปที่ 25 ตัวอย่างข้อมูลที่ถูกดึงจากรถยนต์ในช่วงเวลา 1645407-1645455 มิลลิวินาที

จากรูปที่ 25 จะเห็นว่าในทุกบรรทัดจะโดนแบ่งสิบคอลลัม ข้อมูลที่ว่างหมายความว่าไม่มีข้อมูล ณ ขนะนั้น โดยคอลลัมที่หนึ่งจะแสดงเวลาที่ข้อมูลถูกดึงโดยมีหน่วยเป็นมิลลิวินาที โดยเวลาจะเริ่มนับจากเมื่อบอร์ดESP32 เริ่มใช้งาน คอลลัมที่สองเป็นข้อมูลที่แสดงค่าของแคนไอดี(CAN ID) แคนไอดีคือรหัสเฉพาะของตัวแปรหนึ่งๆที่แตกต่างกัน ซึ่งจากทำการหารือก่อนหน้านี้เราต้องการดึงข้อมูล

จากรถคือระดับน้ำมัน ความเร็วรถยนต์ ความเร็วรอบเครื่อง และเลขไมล์ ซึ่งค่าแคนไอดีของแต่ละตัวคือดังตารางที่ 3

ตารางที่ 3 รหัสแคนไอดีของข้อมูลที่ดึงออกจากรถยนต์

ข้อมูลภายในรถยนต์	รหัสแคนไอดี
ความเร็วรอบเครื่องยนต์	452
ความเร็วรถยนต์	1552
เลขไมล์(Odometer)	1553
ระดับน้ำมัน	1559

โดยข้อมูลแต่ละตัวจะมีความถี่ที่แสดงผลแตกต่างกันยกตัวอย่างเช่น ความเร็วของรถยนต์จะถูกแสดงออกที่ความถี่ทุกหนึ่งวินาทีและระดับน้ำมันจะถูกแสดงออกมาทุกสองวินาที จากรูปที่ 25 ในคอลลัมที่สามถึงคอลลัมที่สิบจะเป็นข้อมูลที่แสดงรายละเอียดในข้อมูลในแต่ละชุดนั้นๆ โดยจะมีค่าที่แสดงออกมาสองถึงแปดค่าเป็นเลขฐานสิบหก ยกตัวอย่างเช่นในรูปที่ 25 ณ เวลาที่ 1645450 มิลลิวินาที ค่าระดับน้ำมัน(แคนไอดี 452) แสดงข้อมูลออกมาแปดตัวคือ 03,DA,28,08,3A,00,7F,93 ข้อมูลชุดนี้ค่าสองตัวแรกจะแสดงถึงความเร็วรอบเครื่องของรถยนต์ในเวลานั้น จากชุดข้อมูลข้างต้นรถยนต์มีความเร็วรอบเครื่องคือ RPM = 3*16^2+13*16+10 = 986 รอบต่อนาที

เพื่อความง่ายต่อการเขียนโปรแกรมในส่วนรวบรวมข้อมูลในส่วนถัดไป การเขียนโปรแกรมในส่วนควบคุมโมดูลแคนบัสนี้จึงเขียนโปรแกรมในส่วนคำนวนค่าของข้อมูลที่แท้จริงไปพร้อมกัน จากการเขียนโปรแกรมในรูปที่ 24 และรหัสแคนไอดีของตารางที่ 3 จึงเขียนโปรแกรมเพื่อกรองเก็บเฉพาะข้อมูลที่ต้องการเท่านั้น จากนั้นจะทำการคำนวนค่าของข้อมูลให้ออกมาในรูปแบบของเลขฐานสิบออกมาในเวลาเดียวกันโดยที่ค่าของระดับน้ำมันคือค่าของข้อมูลในคอลลัมที่สี่และห้า ค่าของความเร็วรถยนต์คือค่าของข้อมูลในคอลลัมที่สอง ค่าของเลขไมล์คือค่าของข้อมูลในคอลลัมที่ห้า หกและเจ็ดและสุดท้ายค่าของความเร็วรอบเครื่องคือค่าของข้อมูลในคอลลัมที่ศูนย์และหนึ่งดังที่ได้แสดงในรูปที่รูปที่ 26

จากรูปที่ 26 ในช่วงบรรทัดที่ 229-235 จะเห็นว่ามีการบวกเพิ่มค่าของ r ทุกครั้งที่เจอค่าของความเร็วรอบเครื่อง เนื่องจากความเร็วรอบเครื่องจะมีความถี่ในการส่งข้อมูลออกมามากกว่าข้อมูล

อื่นๆดังกล่าวมากกว่าถึง 25 เท่า ในโปรแกรมส่วนนี้จึงทำการเขียนเงื่อนไขเพื่อให้ข้อมูลที่เข้ามาเก็บในบอร์ด ESP32 ไม่มากเกินไป

```
218  if(CAN_MSGAVAIL == CAN.checkReceive())            // check if data coming
219      {
220        rcvTime = millis();
221        CAN.readMsgBuf(&len, buf);    // read data,  len: data length, buf: data buf
222
223        rxId= CAN.getCanId();
224
225        if (rxId == 1559){
226          fuel_val = buf[4]*16*16+buf[5];
227
228        }
229        else if (rxId == 452){
230          r++;
231          if  (r>25){
232          engine_val = buf[0]*16*16+buf[1];
233          r=0;
234          time_startt = millis();
235          }
236        }
237        else if (rxId == 1552){
238
239          speed_val = buf[2];
240          distance += speed_val;
241
242        }
243         else if (rxId == 1553){
244
245          odo_val = buf[5]*16*16*16*16+buf[6]*16*16+buf[7];
246
247        }
248
249    }
```

รูปที่ 26 โปรแกรมควบคุมโมดูลแคนบัสส่วนคำนวนค่าของข้อมูล

2.3.4 การเขียนโปรแกรมสำหรับควบคุมโมดูลจีพีเอส

โปรแกรมชุดนี้จะเขียนต่อจากชุดโปรแกรมควบคุมโมดูลแคนบัสโดยโปรแกรมจะยังคงอยู่ในโปรแกรมส่วนที่หนึ่ง โมดูลจีพีเอสจะมีความเฉพาะต่างกับโมดูลอื่นๆเป็นโมดูลที่ต้องมีไฟเลี้ยงตลอดเวลาเนื่องจากมีโมดูลสำหรับจับนาฬิกาอยู่ภายในด้วยทำให้โมดูลจีพีเอสจะต้องใส่ถ่านเพื่อจ่ายไฟเลี้ยงอยู่ตลอดเวลา หลังจากจ่ายไฟและเชื่อมต่อโมดูลจีพีเอสเข้ากับบอร์ด ESP32 ตามหัวข้อที่ 2.2 จะทำการดึงข้อมูลตำแหน่งจากโมดูลจีพีเอสด้วยการอ่านสัญญาณซีเรียล บอร์ด ESP32 มีช่องสำหรับเชื่อมต่อสัญญาณซีเรียสู่ภายนอกอยู่สองพอร์ทซึ่งทั้งคู่เป็นฮาร์ดแวร์ซีเรียล โดยจะให้พอร์ทซีเรียลที่หนึ่งเป็นขาสื่อสารกับโมดูลจีพีเอส โดยการรับข้อมูลจากโมดูลจีพีเอสจะใส่ไลบรารี่ TinyGPS++ เพื่อความสะดวกในการเขียนโปรแกรม เริ่มเขียนโปรแกรมจากการตั้งค่า Baud rate สำหรับการสื่อสารซีเรียลกับโมดูลจีพีเอสไว้ที่ 9600 จากนั้นให้ทำการเช็คว่ามีข้อมูลส่งเข้ามาในบอร์ด ESP32ผ่านซีเรียล

ช่องที่หนึ่งหรือไม่ เพื่อทำการบันทึกค่าของละจิจูดและลองจิจูดในรูปแบบข้อมูลประเภทสตริงดังรูปที่ 27

```
while (Serial.available() > 0)
  if (gps.encode(Serial.read()))
  {
    if (gps.location.isValid())
    {
      latitude = String((gps.location.lat()));

      longtitude =String((gps.location.lng()));
    }
  }

```

รูปที่ 27 โปรแกรมดึงข้อมูลตำแหน่งผ่านการสื่อสารประเภทซีเรียล

การใช้งานสำหรับโมดูลจพีเอสในครั้งแรกจะใช้เวลาในการจับสัญญาณจากดาวเทียมเป็นช่วงเวลา 1-5 นาที หากตัวรับสัญญาณอยู่ในจุดอับสัญญาณจะไม่สามารถระบุตำแหน่งของตัวรถได้ชัดเจนหรือไม่สามารถระบุได้อย่างถูกต้อง[13]

2.3.5 การเขียนโปรแกรมสำหรับรวมข้อมูล

การรวบรวมข้อมูลข้อมูลทั้งหมดเข้าด้วยกันเป็นส่วนที่สำคัญมากสำหรับโปรแกรมส่วนที่หนึ่ง การเขียนโปรแกรมส่วนนี้จะต้องคำนึงถึงรูปแบบในการส่งต่อให้กับทางเซิฟเวอร์เช่นกัน เนื่องจากการส่งข้อมูลขึ้นเซิฟเวอร์จะใช้เวลาค่อนข้างเยอะ เพราะฉะนั้นการส่งข้อมูลแต่ละชุดจึงควรมีข้อมูลครบทุกประเภทอยู่ในนั้น

จากหัวข้อที่ 2.3.3 และ 2.3.4 จึงได้ข้อมูลครบทุกตัวที่ต้องการ เริ่มการรวบรวมข้อมูลโดยการเปลี่ยนประเภทข้อมูลเป็นประเภทสตริง จากนั้นนำข้อมูลแต่ละตัวมาต่อกันโดยใช้คอมม่าเป็นตัวกั้นระหว่างข้อมูล แล้วนำข้อมูลชุดเก็บไว้ในตัวแปร val ดังรูปที่ 28 เพื่อเตรียมตัวนำไปส่งขึ้นเซิฟเวอร์ในโปรแกรมชุดต่อไป

```
val = String(speed_val)+ "," + String(odo_val)+ "," + String(engine_val)+ "," + String(fuel_val) +","+ latitude +"," + longtitude;
```

รูปที่ 28 โปรแกรมรวมข้อมูลจากโมดูลแคนบัสและโมดูลจีพีเอส

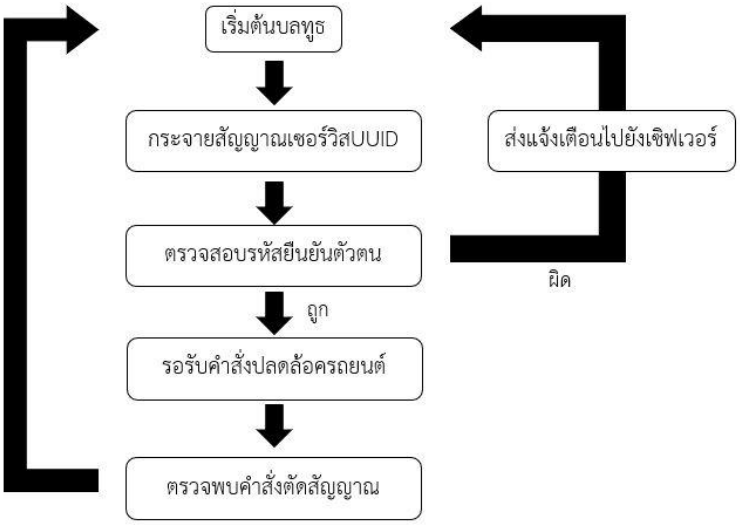
2.3.6 การเขียนโปรแกรมสำหรับควบคุมบลูทูธพลังงานต่ำ

ชิปบลูทูธพลังงานต่ำมีทั่วไปในบอร์ดESP32 สามารถใช้งานได้โดยที่ไม่ต้องทำการติดตั้งอุปกรณ์ใดเพิ่มก่อนที่จะเริ่มเขียนโปรแกรมสำหรับควบคุมบลูทูธต้องศึกษาการทำงานของบลูทูธเสียก่อน การทำงาน

ของบลูทูธพลังงานต่ำจะมีส่วนสำคัญอยู่สองส่วน ส่วนแรกคืออุปกรณ์สำหรับการทำงานแบบบลูทูธเซิฟเวอร์ (Bluetooth Server) เปรียบเสมือนเป็นเจ้าบ้านสำหรับการสื่อสารที่รอให้อุปกรณ์ตัวอื่นเข้าไปทำการเชื่อมต่อ โดยบลูทูธเซิฟเวอร์จะกระจายสัญญาณ (Advertise)สำหรับรอการเชื่อมต่อ อีกส่วนคือบลูทูธไคลแอนท์ (Bluetooth Client) คืออุปกรณ์ฝั่งที่เข้าไปเชื่อมต่อกับบลูทูธเซิฟเวอร์ โดยบลูทูธไคลแอนท์จะทำการแสกนว่ามีสัญญาณที่ต้องการเชื่อมต่อหรือไม่

จากรูปที่ 8 โครงสร้างของการสื่อสารบลูทูธชั้นนอกสุดคือชั้นเซอร์วิส การที่อุปกรณ์สองตัวจะสื่อสารกันได้ผ่านบลูทูธจะต้องเริ่มจากการเชื่อมต่อของสัญญาณเดียวกัน โดยจะมีการกำหนดรหัสของชั้นเซอร์วิส เริ่มจากฝั่งบลูทูธเซิฟเวอร์จะกระจายสัญญาณที่มีหมายเลขเซอร์วิสเฉพาะตัว จากนั้นหากมีอุปกรณ์ที่กำลังค้นหาสัญญาณเซอร์วิสเดียวกันนี้จะสามารถทำการเชื่อมต่อกันได้ หลังจากการเชื่อมต่อสำเร็จจะสามารถสื่อสารกันได้โดยจะแยกเป็นหมวดหมู่ตามชั้นของคาแรกเทอริสติก (Characteristic) ซึ่งรหัสของเซอร์วิสและคาแรกเทอริสติกจะถูกกำหนดด้วยรหัสUUID (Universally Unique Identifier) โดยรหัสUUID เป็นรหัส 128 บิต ประกอบไปด้วยตัวอักษรและตัวเลข 32 ตัว และแบ่งออกเป็น 4 ชุด หรือในรูปแบบนี้ xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx เช่น 428912a6-29fd-4aea-ac33-7f52e4069726

จากการหารือกับฝั่งพัฒนาแอพพลิเคชั่นบนมือถือ โดยทั่วไปมือถือจะใช้งานบลูทูธเป็นตัวค้นหา(บลูทูธไคลแอนท์) เพราะสะดวกต่อการเขียนโปรแกรม นั่นคือฝั่งบอร์ด ESP32 จึงต้องเป็นบลูทูธเซิฟเวอร์ เริ่มต้นการเขียนโปรแกรมโดยตั้งค่าการทำงานของบลูทูธเซิฟเวอร์โดยกำหนดรหัสUUID ที่แตกต่างกันของส่วนเซอร์วิสและคาแรกเทอริสติก จากนั้นเขียนโปรแกรมให้บอร์ด ESP32 กระจายสัญญาณเพื่อรองรับการเชื่อมต่อแบบอัตโนมัติ(Auto pairing) หลังจากบอร์ด ESP32 สามารถเชื่อมต่อกับแอพพลิเคชันมือถือได้แล้ว บอร์ดESP32 จะทำการตรวจสอบว่ามือถือที่ทำการเชื่อมต่อคือมือถือที่ทำการจองรถเช่าผ่านระบบหรือไม่ หากไม่ใช่ทางบอร์ด ESP32 จะทำการแจ้งไปยังเซิฟเวอร์ว่ามีมือถือที่ทำการเชื่อมต่อมาโดยไม่ได้รับอนุญาต จากนั้นจะทำการตัดสัญญาณและรีสตาร์ทตัวบอร์ดหนึ่งนาที หากตรวจสอบแล้วพบว่ารหัสตรงกับมือถือที่ทำการจองผ่านระบบ ตัวบอร์ด ESP32 จะทำการรอรับคำสั่งสัญญาณล้อคและปลดล้อคจากแอพพลิเคชันมือถือซึ่งเป็นรหัสที่ทำการใส่รหัสเฉพาะมาแล้วเช่นกันดังรูปที่ 29



รูปที่ 29 ผังการทำงานของบลูทูธเซิฟเวอร์

ในทำนองเดียวกัน ฝั่งแอพพลิเคชันมือถือจะทำการค้นหาสัญญาณบลูทูธของบอร์ด ESP32 หลังจากเชื่อมต่อสำเร็จจะทำการส่งรหัสยืนยันตัวตนว่าคือมือถือที่ได้รับ เมื่อได้รับการยืนยันจากบอร์ด ESP32 จะสามารถส่งสัญญาณคำสั่งปลดล็อครถยนต์ได้

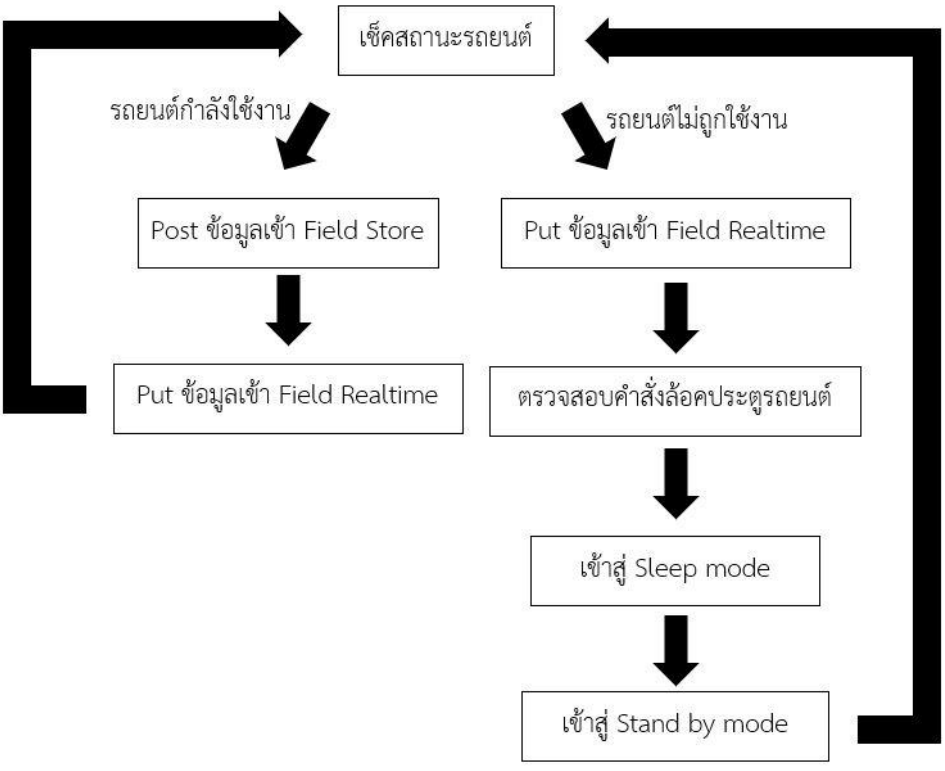
2.3.7 การเขียนโปรแกรมสำหรับควบคุมโมดูลสามจี

โมดูลสามจีจะเป็นตัวทำการสื่อสารหลักระหว่าง CoBox และ Firebase database โดยการสั่งการทำงานของโมดูลสามจีจะใช้เอทีคอมมานด์ (AT command) ซึ่งเป็นคำสั่งเดียวกันกับการทำงานของโทรศัพท์มือถือ นั่นหมายความว่าโมดูลสามจีสามารถเชื่อมต่ออินเทอร์เน็ต โทรเข้าออกและส่งข้อความแบบโทรศัพท์มือถือได้ ซึ่งการใช้งานของโมดูลในส่วนนี้จะเพียงแค่ส่วนเชื่อมต่อกับอินเทอร์เน็ตเพื่อรับส่งข้อมูลจากเซิฟเวอร์

การเขียนโปรแกรมส่วนนี้จะเขียนอยู่ในโปรแกรมส่วนที่สอง จะทำงานในเวลาเดียวกันกับโปรแกรมส่วนแรก เริ่มการเขียนโปรแกรมจากตั้งค่าการเชื่อมต่ออินเทอร์เน็ตของโมดูลสามจีโดยเชื่อมกับสัญญาณสามจีของทรูมูฟ หลังจากเชื่อมต่อสำเร็จจะทำการเชื่อมต่อกับ FIrebase realtime database โดยจะต้องใส่รหัสอนุญาตการเข้าถึง(Authentication key) และ Host URL ของฐานข้อมูลที่ต้องการเชื่อมต่อ หลังจากเชื่อมต่อกับฐานข้อมูลของเซิฟเวอร์ได้แล้ว บอร์ด ESP32จะดึงข้อมูล value ในรูปที่ 28 แล้วทำการส่งให้กับโมดูลสามจีเป็นเอทีคอมมานด์ผ่านช่องซีเรียลที่สอง เมื่อโมดูลสามจีได้รับคำสั่งจากบอร์ด ESP32 จะทำตามคำสั่งตามเอทีคอมมานด์นั้น หากทำสำเร็จโมดูล

สามจีจะส่งสัญญาณกลับไปบอร์ดว่าทำงานสำเร็จ แต่หากทำงานไม่สำเร็จจะส่งกลับไปว่าทำงานล้มเหลว โดยกระบวนการรับส่งข้อมูลถึงเซิฟเวอร์ในแต่ละรอบจะใช้เวลาอยู่ที่ 1 ถึง 2 วินาที

ในแต่ละรอบของการทำงานโปรแกรมชุดที่สอง จะเริ่มต้นจากตรวจสอบสถานะการจองของรถยนต์จากเซิฟเวอร์ โดยสถานะของรถยนต์จะมีสองสถานะ สถานะที่หนึ่งคือรถยนต์ถูกใช้งาน เมื่อ CoBox ตรวจพบว่ารถยนต์อยู่ในสภาพถูกใช้งาน บอร์ด ESP32 จะทำการดึงข้อมูล value ดังรูปที่ 28 จากโปรแกรมส่วนที่หนึ่ง จากนั้นจะส่งข้อมูลดังกล่าวขึ้นเซิฟเวอร์ผ่านโมดูลสามจีสองครั้ง ครั้งที่หนึ่งจะส่งข้อมูลแบบ POST ลงไปในฟิลด์ที่ไม่ต้องการให้ข้อมูลถูกลบ นั่นคือข้อมูลชุดต่อๆไปจะถูกส่งเข้ามาและซ้อนต่อไปจากข้อมูลก่อนหน้า ข้อมูลจะไม่หายไปจนกว่าจะโดนลบด้วยคำสั่ง DELETE การส่งข้อมูลครั้งที่สองจะส่งข้อมูลแบบ PUT ลงไปในฟิลด์ Realtime เพื่อที่จะแสดงค่า ณ ขณะนั้นได้ทันที ซึ่งข้อมูลในฟิลด์นี้จะถูกเขียนทับไปเรื่อยๆ จะไม่สามารถดูข้อมูลตัวก่อนหน้าได้ ในกรณีที่ตรวจพบว่ารถยนค์อยู่ในสถานะไม่ถูกใช้งาน ทางบอร์ด ESP32 จะทำการส่งข้อมูลแบบ PUT ลงไปในฟิลด์ Realtime จากนั้นจะดึงข้อมูลจากเซิฟเวอร์ว่ารถยนต์คันดังกล่าวมีคำสั่งล้อคหรือปลดล้อคประตูหรือไม่ จากนั้นจะเข้าสู่รูปแบบนอนหลับ(Sleep mode) และตื่นขึ้นมาทำงานต่อในเวลา 1 นาที ส่วนนี้จะถูกอธิบายอย่างละเอียดในบทถัดไป โดยแผนผังการทำงานแสดงไว้ดังรูปที่ 29



รูปที่ 30 แผนผังการทำงานจองโมดูลสามจี

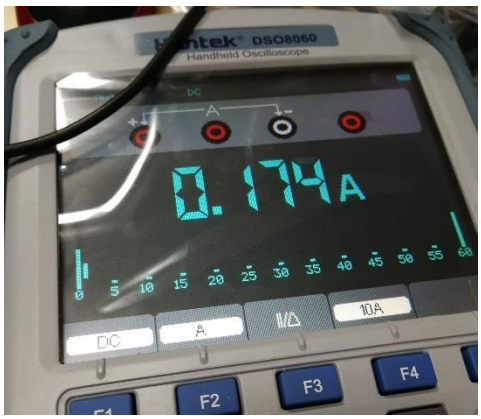
2.3.8 การเขียนโปรแกรมสำหรับควบคุมการใช้พลังงาน

จากการเขียนโปรแกรมในส่วนก่อนหน้านี้ทั้งหมดนั้นเพื่อให้การทำงานของกล่อง CoBox ทำงานได้ตามจุดประสงค์ แต่หลังจากการนำกล่องต้นแบบไปทดสอบกลับพบว่าการทำงานของกล่องใช้พลังงานค่อนข้างมาก ส่งผลให้แบตเตอร์รี่ภายในรถยนต์ถูกดึงเอาไปใช้งานจนหมดภายในเวลาไม่นานในกรณีที่รถยนต์ถูกจอดทิ้งไว้โดยไม่ได้ใช้งาน ซึ่งจำเป็นต้องมีการปรับเปลี่ยนให้เหมาะแก่การใช้งานจริงในลำดับต่อมา

เริ่มต้นจากทดสอบจำนวนการใช้ปริมาณของกระแสไฟฟ้าขณะใช้งาน โดยต่อเครื่องวัดกระแสไฟฟ้าแบบอนุกรมเข้ากับกล่อง CoBox จากนั้นทำการจ่ายไฟแรงดัน12โวลต์เหมือนกับรถยนต์ได้ผลดังรูปที่ 31 นั่นคือ 245mA จากการปรึกษากับช่างในอู่รถยนต์พบว่าแบตเตอรี่รถยนต์ที่ใช้ทำการติดตั้งมีขนาด 50A นั่นคือหากกล่องCoBox ทำงานตลอดเวลาจะสามารถใช้งานได้ประมาณ $\frac{50}{0.245} = 204$ ชั่วโมง หรือประมาณแปดวันครึ่ง จะเห็นว่าไม่เพียงพอต่อการใช้งานจริงๆดังนั้นจึงมีรูปแบบประหยัดพลังงานเพื่อให้กล่องทำงานได้นานขึ้น

รูปที่ 31 ปริมาณกระแสไฟฟ้าขณะกล่อง CoBox ทำงาน

รูปแบบประหยัดพลังงาน(Sleep mode) คือรูปแบบที่จะทำให้กล่องอยู่ในสภาพหลับ ไม่มีการทำงานใดๆ แต่กล่องพร้อมจะทำงานในเวลาต่อมา โดยขณะที่กล่องอยู่ในสภาพหลับนั้นจะใช้กระแสไฟฟ้าสำหรับเลี้ยงโมดูลไว้น้อยกว่าขณะทำงานมาก โดยใช้ปริมาณกระแสประมาณ 174mA ดังรูปที่ 32 นั่นคือหากกล่องอยู่ในสภาพหลับตลอดเวลากล่องจะสามารถอยู่ในสภาพนี้ได้ประมาณ $\frac{50}{0.174} = 287$ ชั่วโมง หรือประมาณสิบสองวัน จะเห็นว่าสามารถใช้งานได้นานกว่าแบบแรกมากกว่าสามวัน



รูปที่ 32 ปริมาณกระแสไฟฟ้าขณะกล่อง CoBox หลับ

ดังนั้นการเขียนโปรแกรมจะพิจารณาว่า หากรถยนต์อยู่ในสถานะใช้งาน กล่อง CoBox จะทำงานเต็มประสิทธิ์ภาพ ซึ่งในกรณีนี้รถยนต์จะถูกใช้งานนั่นหมายความว่าแบตเตอร์รี่รถยนต์จะถูกชาร์จกลับทำให้ปริมาณกระแสในรถยนต์ไม่ลดลง กรณีที่รถยนต์อยู่ในสถานะไม่ถูกใช้งาน กล่อง CoBox จะอยู่ในสภานะหลับเพื่อรอการทำงานในลำดับต่อไปดังรูปที่ 29 ดังนั้นหากรถยนต์ไม่ถูกใช้งานเลยภายในสิบสองวัน แบตเตอรี่รถยนต์อาจจะถูกกล่อง CoBox ดึงไปใช้งานจนหมดได้

บทที่ 3

การทดลอง

3.1 ติดตั้งอุปกรณ์

หลังจากออกแบบและเขียนโปรแกรมสำหรับกล่อง CoBox เสร็จสิ้น จึงทำการประกอบอุปกรณ์ทั้งหมดเข้าด้วยกันแล้วใส่ลงไปกล่องดังรูปที่ 33 โดยจะมีสายไฟออกมาจากกล่องทั้งหมด 4 เส้น เส้นสีแดงคือสายไฟที่ต่อกับขั้วบวกของแบตเตอร์รี่รถยนต์ 12 โวลต์ เส้นสีดำคือสายไฟที่ต่อกับกราวด์ของรถยนต์ เส้นสีน้ำเงินและสีขาวคือสายสัญญาณต่อกับสัญญาณแคนไฮก์ (CAN High) และแคนโลว์ (CAN Low) ของรถยนต์ตามลำดับ อีกทั้งยังมีเสาสัญญาณไว้สำหรับรับสัญญาณอินเทอร์เน็ตสามจีและมีสายเสาสัญญาณยาว 1.5 เมตรสำหรับรับสัญญาณโมดูลจีพีเอส

รูปที่ 33 กล่อง CoBox

ทำการติดตั้งกล่อง CoBox โดยการใส่ไว้ใต้คอนโซลของรถยนต์ ในที่นี้จะไม่ใช้หัวต่อโอบีดีทูตัวผู้ (Male OBD-II Connecter) เนื่องจากจะทำให้ง่ายต่อการถูกถอดออกเช่นกัน ดังนั้นจึงใช้ตลับต่อสายไฟดังรูปที่ 34 ในการต่อสายไฟแต่ละเส้น ในส่วนของตำแหน่งสำหรับติดตั้งกล่องจะอยู่ใกล้กับช่องต่อโอบีดีทูและมีอุณหภูมิใกล้เคียงกับภายในห้องโดยสารจึงเป็นตำแหน่งที่เหมาะสม แต่ตำแหน่งดังกล่าวรับสัญญาณจีพีเอสได้ไม่ดีเท่าที่ต้องการ จึงต้องลากสายสัญญาณจีพีเอสออกมาติดตั้งภายนอกเพื่อให้รับสัญญาณได้ดียิ่งขึ้น การติดตั้งภายใต้คอนโซลควรจะถอดเฟรมบางชิ้นออกก่อนเพื่อความสะดวกในการติดตั้งดังรูปที่ 35 และทำให้ยากต่อการถูกถอดออกอีกด้วย

รูปที่ 34 ตลับต่อสายไฟ

รูปที่ 35 ตำแหน่งติดตั้ง CoBox ภายในคอนโซล

3.2 ข้อมูลบนเซิฟเวอร์

หลังจากติดตั้งกล่อง CoBox และทำการใช้งานจริงไปช่วงเวลาหนึ่ง จึงมีข้อมูลจากรถยนต์ส่งขึ้นมาเก็บไว้บนเซิฟเวอร์จำนวนมาก จากนั้นทำการดึงข้อมูลออกจากเซิฟเวอร์เพื่อวิเคราะห์ต่อไป โดยดึงออกมาในรูปแบบเจสัน (JSON) ดังรูปที่ 36 แล้วส่งเข้าโปรแกรมไมโครซอฟท์เอ็กเซล (Microsoft Excel) เพื่อสะดวกต่อการวิเคราะห์


```
"val_stored100052" : "6,170872,966,158,24723,136964150.00,1005358983.33",
"val_stored100053" : "3,170872,973,167,24818,136964566.67,1005359983.33",
"val_stored100054" : "5,170872,944,139,24848,136964883.33,1005360566.67",
"val_stored100055" : "4,170872,947,146,24926,136965383.33,1005361483.33",
"val_stored100056" : "7,170872,890,122,24980,136965583.33,1005362066.67",
"val_stored100057" : "6,170872,1100,142,25033,136965616.67,1005362533.33",
"val_stored100058" : "0,170872,1007,85,25108,136965866.67,1005363083.33",
"val_stored100059" : "31,170872,1854,177,25395,136966683.33,1005365066.67",
"val_stored100060" : "43,170872,1752,165,26164,136970850.00,1005372866.67",
"val_stored100061" : "44,170873,1999,170,27078,136974233.33,1005381200.00",
"val_stored100062" : "37,170873,1345,138,27834,136977833.33,1005389000.00",
"val_stored100063" : "33,170873,1397,146,28508,136982466.67,1005396133.33",
"val_stored100064" : "17,170873,1391,152,29056,136985933.33,1005400533.33",
"val_stored100065" : "15,170873,1565,147,29389,136988600.00,1005403550.00",
"val_stored100066" : "24,170873,1277,164,29782,136991433.33,1005405916.67",
```

รูปที่ 36 ตัวอย่างข้อมูลจากรถยนต์ในรูปเจสัน

เมื่อข้อมูลทั้งหมดอยู่ในโปรแกรมไมโครซอฟท์เอ็กเซลแล้ว ข้อมูลเหล่านั้นจะถูกรวมไว้ในคอลลัมเดียวเพราะทางโปรแกรมมองว่าข้อมูลเป็นข้อมูลเพียงตัวเดียวเท่านั้น จึงต้องทำการแยกข้อมูลออกมาโดยทำการเลือกข้อมูลในคอลลัมที่ต้องการแยกข้อมูลออกมา จากนั้นกด Text to columns แล้วเลือกตัวแบ่งเป็นเครื่องหมายคอมม่า จากนั้นกดตกลง กระบวนการดังกล่าวจะทำการแยกข้อมูลออกมาเป็นแต่ละคอลลัมตามที่ต้องการดังรูปที่ 37 โดยคอลลัมA B C D E F G H คือ จำนวนข้อมูล ความเร็วรถยนต์(กิโลเมตร/ชั่วโมง) เลขไมล์(กิโลเมตร) ความเร็วรอบเครื่องยนต์(รอบ/นาที) ระดับน้ำมัน(10*ลิตร) ระยะทางทั้งหมดที่วิ่ง(เมตร) ละติจูดและลองจิจูดตามลำดับ

	A	B	C	D	E	F	G	H
1	val_stored100041	0	170872	993	156	20786	136949950	1005319017
2	val_stored100042	0	170872	1623	126	20786	136949950	1005319017
3	val_stored100043	23	170872	1278	153	21118	136951066.7	1005321650
4	val_stored100044	27	170872	1456	161	21622	136952766.7	1005326367
5	val_stored100045	30	170872	1425	151	22190	136955400	1005332700
6	val_stored100046	24	170872	1372	115	22784	136957750	1005338817
7	val_stored100047	27	170872	1543	153	23285	136959683.3	1005343600
8	val_stored100048	25	170872	1425	161	23790	136961483.3	1005349367
9	val_stored100049	16	170872	1346	156	24115	136962550	1005352533
10	val_stored100050	8	170872	1574	164	24328	136963233.3	1005354750
11	val_stored100051	12	170872	1659	155	24539	136963533.3	1005356750
12	val_stored100052	6	170872	966	158	24723	136964150	1005358983
13	val_stored100053	3	170872	973	167	24818	136964566.7	1005359983
14	val_stored100054	5	170872	944	139	24848	136964883.3	1005360567
15	val_stored100055	4	170872	947	146	24926	136965383.3	1005361483

รูปที่ 37 ตัวอย่างข้อมูลจากรถยนต์ในโปรแกรมไมโครซอฟท์เอ็กเซล

3.3 ทดสอบปลดล้อคประตู

การสั่งการเพื่อล้อคหรือปลดล้อคประตูนั้นมีสองรูปแบบคือการสั่งการผ่านบลูทูธผ่านแอพพลิเคชันบนมือถือและการสั่งการผ่าน Firebase database

3.3.1 สั่งการผ่านบลูทูธคอนเนคชัน

หลังจากติดตั้งกล่อง CoBox ในรถยนต์แล้ว ส่วนบลูทูธพลังงานต่ำของบอร์ด ESP32 จะเริ่มทำงานทันทีในสถานะของบลูทูธเซิฟเวอร์เพื่อรอการเชื่อมต่อจากแอพพลิเคชันบนมือถือ

ในส่วนของแอพพลิเคชันบนมือถือนั่นจะสามารถใช้งานส่วนนี้ได้ก็ต่อเมื่อทำการจองรถยนต์แล้วเท่านั้น โดยเริ่มจากทำการค้นหาสัญญาณบลูทูธที่มีรหัสเซอร์วิสและรหัสคาแรกเทอริสติกที่

ต้องการเชื่อมต่อ หลังจากเชื่อมต่อสำเร็จทางแอพพลิเคชันส่งรหัสยืนยันตัวตนว่าใช่ผู้ที่ทำการจองรถยนต์ตัวจริง เมื่อยืนยันเสร็จสิ้นจึงสามารถทำการส่งรหัสเฉพาะเพื่อสั่งล้อคและปลดล้อครถยนต์ได้

3.3.2 สั่งการผ่านอินเทอร์เน็ตคอนเนคชัน

การสั่งการล้อคหรือปลดล้อครถยนต์ผ่านทางอินเทอร์เน็ตนั้นสามารถทำได้จากทางผู้ดูแลระบบเท่านั้น จากรูปที่ 30 จะเห็นว่าทางฝั่งกล่อง CoBox จะมีสถานะสำหรับตรวจสอบคำสั่งล้อคหรือปลดล้อคจากเซิฟเวอร์ในขณะที่สถานะของรถยนต์ไม่ถูกใช้งาน ดังนั้นทางผู้ดูแลระบบจะสามารถสั่งการได้ก็ต่อเมื่อรถยนต์คันนั้นๆไม่ได้ถูกใช้งานในเวลานั้น

การสั่งล้อคหรือปลดล้อครถยนต์นั้นสามารถทำได้โดยแก้ไขข้อมูลในฟิลด์ door ของรถยนต์คันนั้นๆดังรูปที่ 38 โดยเมื่อเขียนให้สถานะของฟิลด์ door เป็น Unlock รถยนต์จะทำการปลดล้อคประตูของรถยนต์ และเมื่อแก้ให้สถานะของฟิลด์ door เป็น Lock รถยนต์จะทำการล้อคประตูของรถยนต์

รูปที่ 38 สั่งล้อครถยนต์ผ่านอินเทอร์เน็ตขณะรถยนต์ไม่ถูกใช้งาน

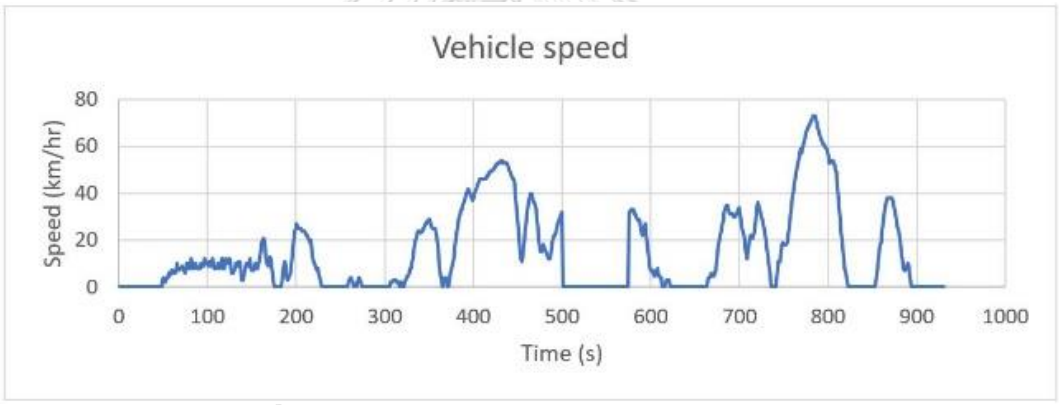
บทที่ 4

ผลการทดลอง

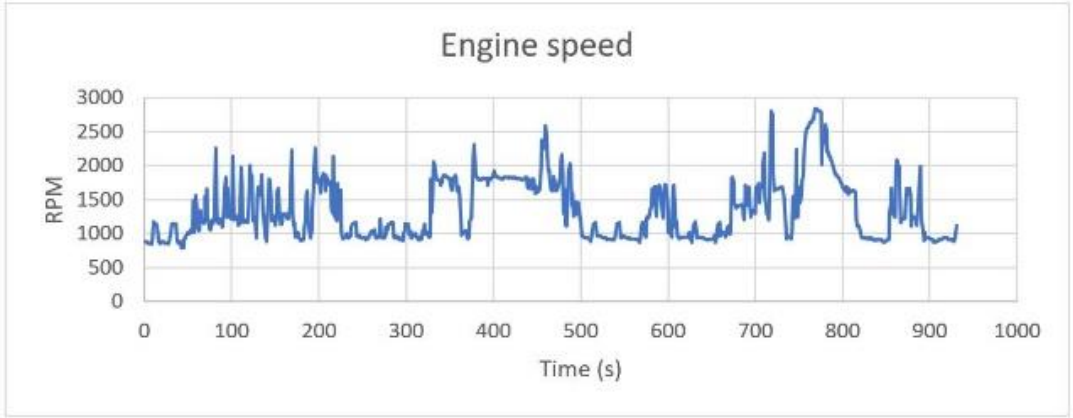
4.1 วิเคราะห์ข้อมูลจากFirebase realtime database

4.1.1 อภิปรายคุณภาพของข้อมูล

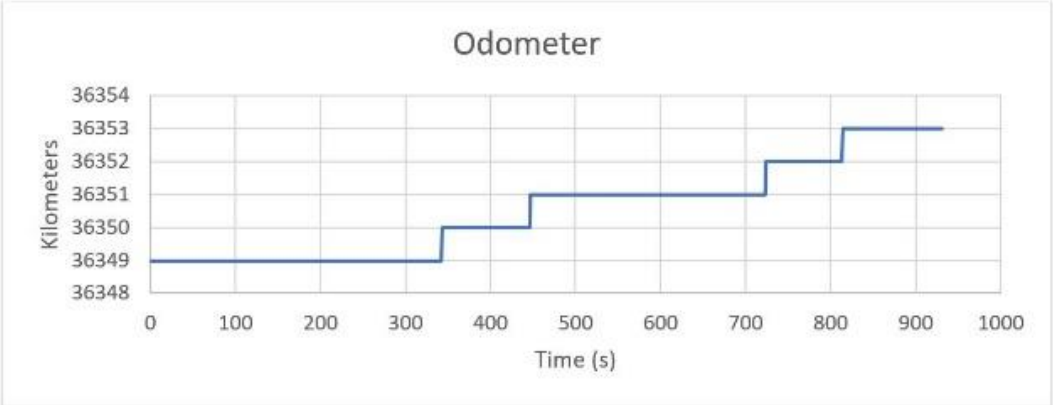
หลังจากทำการทดสอบขับรถยนต์เพื่อทำการเก็บข้อมูล ทำให้ได้ข้อมูลมาเก็บไว้ในเซิฟเวอร์จำนวนมาก จึงทำการดึงข้อมูลบางส่วนมาวิเคราะห์ถึงแนวโน้มในช่วงข้อมูลต่อเวลา โดยเลือกข้อมูลช่วงการขับรถยนต์จากที่พักไปเติมน้ำมันที่ปั๊มน้ำมันเพราะจะสามารถสังเกตพฤกรรมการเปลี่ยนแปลงของระดับน้ำมันได้ง่าย หลังจากดึงข้อมูลในช่วงเหตุการณ์ดังกล่าวออกจากเซิฟเวอร์ไปใส่ในโปรแกรมไมโคตรซอฟท์เอ็กเซลโดยแยกประเภทข้อมูลออกในแต่ละคอลลัม จากนั้นทำการวาดกราฟโดยใช้ข้อมูลเหล่านั้นเทียบกับเวลาได้ดังรูปที่ 39 รูปที่ 40 รูปที่ 41 และรูปที่ 42 ส่วนค่าของพิกัดละติจูดและลองจิจูดนำไปวาดกราฟสามมิติบนแผนที่ได้ดังรูปที่ 43



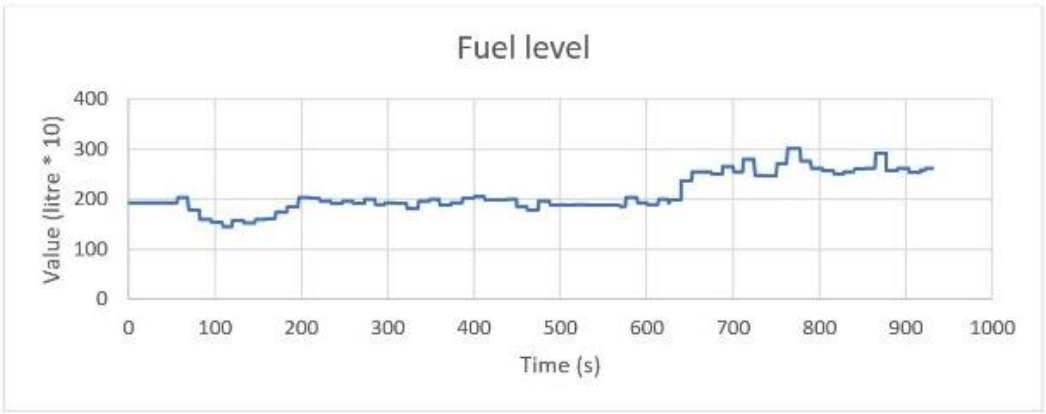
รูปที่ 39 กราฟความสัมพันธ์ของความเร็วรถยนต์ต่อเวลา



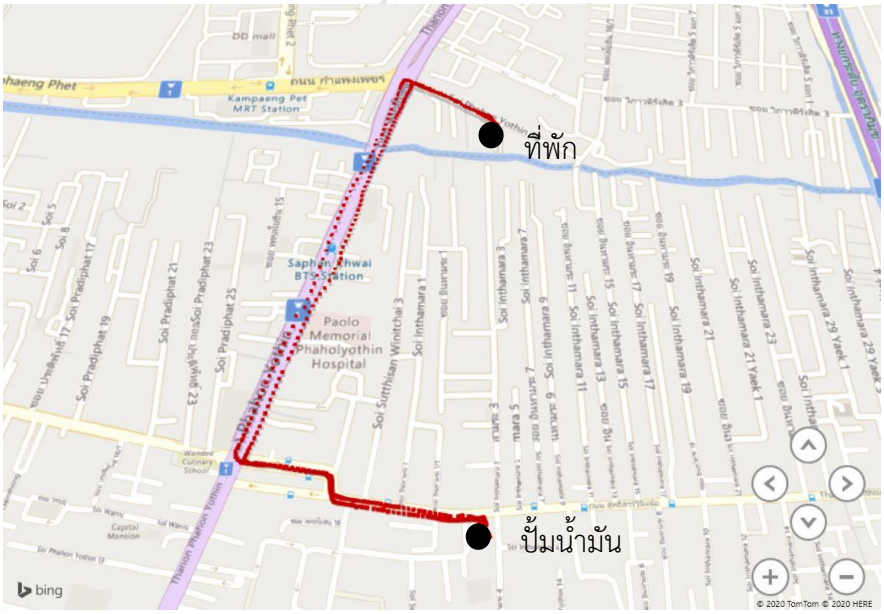
รูปที่ 40 กราฟความสัมพันธ์ของความเร็วรอบเครื่องต่อเวลา



รูปที่ 41 กราฟความสัมพันธ์ของเลขไมล์ต่อเวลา



รูปที่ 42 กราฟความสัมพันธ์ของระดับน้ำมันต่อเวลา



รูปที่ 43 พิกัดการเดินทางระหว่างขับรถยนต์ไปเติมน้ำมัน

รูปภาพที่แสดงมาดังกล่าวมาทั้ง5 รูปได้แสดงค่าของข้อมูลให้เห็นได้ชัดเจนตามที่หวังไว้ จากรูปที่ 39 และ รูปที่ 40 จะเห็นว่าช่วงเวลาที่รถยนต์ไม่เคลื่อนที่ กราฟของความเร็วรถยนต์และความเร็วรอบเครื่องอยู่ที่จุดต่ำสุดทั้งคู่(เช่นในช่วงเวลา 230-300วินาที) และช่วงที่ความเร็วสูงสุดจะเป็นช่วงที่รอบเครื่องมีการทำงานสูงสุดเช่นกัน(ที่เวลา 785วินาที) จากรูปที่ 41 ขณะเริ่มขับรถยนต์มีเลขไมล์อยู่ที่36349 กิโลเมตร หลังจากขับกลับจาเติมน้ำมันมาที่พัก รถยนต์บอกเลขไมล์อยู่ที่ 36353 กิโลเมตร นั่นคือรถยนต์ขับเป็นระยะทางประมาณ 4 กิโลเมตร ซึ่งเป็นการบอกระยะทางแบบคร่าวๆ เท่านั้น การคำนวนระยะทางที่แท้จริงของรถยนต์สามารถทำได้โดยการนำความเร็วที่บันทึกไว้ในช่วงเวลาดังกล่าวมาบวกรวมกัน แล้วแปลงหน่วยให้อยู่ในรูปของ SI ดังสมการ

$$Distance_{total} = \sum_{i_{Start}}^{i_{End}} V_i \,/3.6$$

โดย Vi คือค่าของความเร็วในขณะนั้นๆ(กิโลเมตร/ชั่วโมง) และ $Distance_{total}$ คือระยะทางรวม(เมตร) จากสถานการณ์ข้างต้นจะสามารถคำนวนระยะทางรวมจากความเร็วได้ 3837.8 เมตร ใกล้เคียงกับระยะทางที่อ่านจากเลขไมล์

เนื่องจากข้อมูลชุดนี้ถูกเลือกมาด้วยเหตุผลการเติมน้ำมัน จากรูปที่ 42 จะเห็นว่าระดับน้ำมันของรถยนต์ในช่วงท้ายจะมีปริมาณมากกว่าในช่วงแรก โดยช่วงเวลาที่ทำการเติมน้ำมันอยู่ในช่วงเวลาประมาณ 620-660 วินาที โดยในขณะที่วินาทีที่ 620 วินาที และ 660 วินาที วัดระดับน้ำมันได้ 19.4 ลิตร และ 26.2 ลิตรตามลำดับ นั่นคือมีระดับน้ำมันเพิ่มขึ้นมา 6.8 ลิตร

แต่จะเห็นว่าค่าของปริมาณน้ำมันในรถยนต์จะมีความแกว่งค่อนข้างมาก เนื่องจากภายในรถยนต์ใช้เซนเซอร์วัดระดับน้ำมันจากผิวของปริมาณของเหลวในตัวถัง เมื่อทำการขับรถยนต์จะส่งผลให้น้ำมันกระเพื่อมไปพร้อมกับตัวรถยนต์จึงเป็นสาเหตุที่ทำให้ปริมาณน้ำมันมีความไม่แน่นอนในขณะขับรถยนต์

จากรูปที่ 43 จะเห็นว่าพิกัดที่วาดไว้บนแผนที่จะค่อนข้างแม่นยำมากในช่วงระหว่างขับจากที่พักเข้าสู่ถนนใหญ่(เส้นสีม่วงอ่อนในรูปที่ 43) และขับจากถนนใหญ่เข้าสู่ปั้มน้ำมัน แต่ในส่วนระหว่างขับรถอยู่ในถนนใหญ่ จะมีบางช่วงที่พิกัดจะคลาดเคลื่อนออกนอกบริเวณถนน เนื่องจากถนนเส้นนี้เป็นถนนใต้แนวรถไฟฟ้าบีทีเอส (BTS Sky train) ทำให้โมดูลจีพีเอสภายในกล่อง CoBox ไม่สามารถจับสัญญาณได้แม่นยำจึงส่งผลให้เกิดการคลาดเคลื่อนดังที่ปรากฏ

4.1.2 อภิปรายคุณภาพของการส่งข้อมูล

ในส่วนของการอภิปรายคุณภาพของการส่งข้อมูลจะแบ่งการอภิปรายเป็นสองส่วน โดยส่วนแรกจะอภิปรายความล่าช้าในการส่งข้อมูล และส่วนที่สองจะอภิปรายการสูญหายระหว่างการส่งข้อมูล

เริ่มจากทำการเก็บข้อมูลหลังจากติดตั้งกล่อง CoBox เป็นเวลาแปดชั่วโมงเต็ม ในช่วงเวลา 8.00 น. ถึง 16.00 น. โดยตั้งค่าการส่งข้อมูลจากกล่อง CoBox ขึ้นสู่เซิฟเวอร์ในทุก 1 วินาที หลังจากเก็บข้อมูลครบแปดชั่วโมง พบว่ามีข้อมูลถูกส่งเข้าสู่เซิฟเวอร์ทั้งหมด 27609 ชุด โดยมีข้อมูล 221 ชุดที่ถูกส่งมาไม่สมบูรณ์หรือสูญหายไป หากพิจารณาความล่าช้าในการส่งข้อมูลจะพบว่า ภายในเวลา 28800 วินาที (8 ชั่วโมง) มีข้อมูลถูกส่งมาเพียง 27609 ชุดเท่านั้น พบว่ามีอัตราเร็วในการส่งข้อมูลอยู่ที่ 1.04314 ชุดต่อวินาที หรือก็คือการส่งข้อมูลเกิดการล่าช้าอยู่ที่ 4.314 เปอร์เซ็น

ในตลอดการส่งข้อมูลทั้ง 27609 ชุด มีข้อมูลที่สูญหายหรือไม่สามารถนำมาใช้ได้อยู่ที่ 221 ชุด นั่นคือเกิดการสูญหายของการส่งข้อมูลอยู่ที่ 0.800 เปอร์เซ็น

4.2 รายการตรวจสอบจากบริษัทรถเช่า

หลังจากสร้างกล่อง CoBox ตัวต้นแบบเรียบร้อยจึงส่งกล่องให้ทางบริษัทรถเช่าที่ร่วมมือกัน 2 กล่อง เพื่อทำการทดสอบประสิทธิ์ภาพการทำงาน โดยสิ่งที่ทางบริษัทต้องการทดสอบระบบติดตามรถยนต์มีอยู่ 6 ข้อด้วยกันดังรูปที่ 44 ซึ่งหัวข้อที่ 2 3 4 5 และ 6 เป็นหัวข้อสำหรับทดสอบความสามารถของกล่อง CoBox ส่วนหัวข้อที่ 1 เป็นหัวข้อสำหรับทดสอบฝั่งแอพพลิเคชันมือถือ

	Toyota Yaris 8กภ 6576	Toyota Yaris 8กภ 6607	หมายเหตุ
1. การใช้ระบบ Log in เพื่อเข้าใช้งาน Application KOKAR	✓	✓	
2. การใช้ระบบ lock และ unlock ประตูรถยนต์	✓	✓	ระยะ ms lock และ unlock ห้องสูงในระยะที่ใช้งานได้
3. การแสดงผลระดับน้ำมัน	✓	✓	
4. การแสดงความเร็วของรถยนต์	✓	✓	
5. การแสดง Mileage ของแต่ละทริป (มีการรีเซ็ตเป็น 0 ทุกครั้งที่สตาร์ทรถ)	✓	✓	
6. การแสดงระยะเวลาชับ	✓	✓	

หมายเหตุ กรุณาใส่สถานภาพของการทดสอบด้านบน

ผู้ช่วยวิจัย

เจ้าหน้าที่ตรวจงาน

รูปที่ 44 ตารางรายการตรวจสอบของบริษัทรถเช่า

บทที่ 5
บทสรุป

บทนี้คือบทสุดท้ายของงานวิจัยชิ้นนี้ ก่อนจะสรุปงานทั้งหมดจึงขอย้อนกลับไปที่วัตถุประสงค์ของงานวิจัยชิ้นนี้อีกครั้ง งานวิจัยนี้จัดทำขึ้นเพื่อพัฒนาระบบติดตามสถานะของรถยนต์สำหรับธุรกิจรถเช่า และเพื่อออกแบบอุปกรณ์สำหรับติดตั้งภายในรถยนต์และรองรับกับแอพพลิเคชั่นมือถือในอนาคต โดยใช้แค่ช่องโอบีดีทูสำหรับเชื่อมต่อรถยนต์เท่านั้น

โดยเมื่อออกแบบและประดิษฐ์อุปกรณ์ต้นแบบกล่อง CoBox ได้สำเร็จ จึงทำการติดตั้งกล่องลงในรถยนต์ Toyota altis 2016 ทำการเก็บข้อมูลเป็นเวลาหนึ่งสัปดา โดยข้อมูลที่ดึงออกมาจากรถยนต์คือ ความเร็วรถยนต์ ความเร็วรอบเครื่อง เลขไมล์ และระดับน้ำมัน อีกทั้งยังมีพิกัดจากโมดูลจีพีเอส ข้อมูลจะถูกส่งขึ้นคลาวเซิฟเวอร์ทุก 1 วินาที หลังจากนั้นทำการวิเคราะห์ข้อมูลที่ทำการเก็บมาพบว่าข้อมูลทั้งหมดสามารถถูกส่งขึ้นเซฟเวอร์ได้ตามที่ต้องการ โดยมีความล่าช้าของการส่งข้อมูลอยู่ที่ 4.31% และมีอัตราการสูญหายของข้อมูลอยู่ที่ 0.80% ข้อมูลของความเร็วรถยนต์ ความเร็วรอบเครื่อง และเลขไมล์จากรถยนต์เป็นข้อมูลที่สามารถนำไปใช้ประโยชน์ในการวิเคราะห์ต่อๆไปได้ทันที แต่ระดับน้ำมันและพิกัดของตำแหน่งรถยนต์เป็นข้อมูลที่จะต้องมีการกรองหรือปรับแต่งเบื้องต้นก่อนนำไปใช้วิเคราะห์ต่อๆไป

ในส่วนของการสั่งล้อคหรือปลดล้อคประตูรถยนต์สามารถทำได้ดีในการสั่งการทำงานผ่านอินเทอร์เน็ต ส่วนการสั่งการทำงานผ่านบลูทูธในบางครั้งไม่สามารถทำการเชื่อมต่อได้เนื่องจากแอพพลิเคชันมือถือหาสัญญาณบลูทูธจากรถยนต์ไม่เจอ หากเชื่อมต่อบลูทูธได้สำเร็จก็สามารถทำทำการสั่งล้อคและปลดล้อคได้ดีเช่นกัน

อย่างไรก็ตาม ระบบติดตามสถานะของรถยนต์นี้สามารถทำงานได้ดีในรถยนต์ Toyota altis 2016 เท่านั้น ผู้วิจัยจึงมีแผนในการพัฒนาระบบติดตามเพื่อรองรับกับรถยนต์รุ่นต่างๆต่อไปในอนาคต

บรรณานุกรม

1. Preeti Wadhwani, P.S., *Car Sharing Market Size By Model (P2P, Station-Based, Free-Floating), By Business Model (Round Trip, One Way), By Application (Business, Private), Industry Analysis Report, Regional Outlook, Application Potential, Price Trend, Competitive Market Share & Forecast, 2020 – 2026*. 2020. p. 225.
2. Max Roser, H.R.a.E.O.-O. *World Population Growth*. 2019; Available from: https://ourworldindata.org/world-population-growth.
3. Worldometer. *Current World Population*. 2020; Available from: https://www.worldometers.info/world-population/.
4. Ikezoe, K., E. Kiriyama, and S. Fujimura, *Car-sharing intention analysis in Japan by comparing the utility of car ownership for car-owners and non-car owners.* Transport Policy, 2020. **96**: p. 1-14.
5. Movmi. *CARSHARING MARKET & GROWTH ANALYSIS 2019*. 2019; Available from: https://movmi.net/carsharing-market-growth-2019/.
6. Engineers, S.o.A., *Diagnostic Test Modes J1979_201202*. 2004, SAE.
7. Sik, D., et al., *Comparing OBD and CAN Sampling on the go with the SensorHUB Framework.* Procedia Engineering, 2016. **168**: p. 39-42.
8. Rimpas, D., A. Papadakis, and M. Samarakou, *OBD-II sensor diagnostics for monitoring vehicle operation and consumption.* Energy Reports, 2020. **6**: p. 55-63.
9. D'Agostino, M., M. Naddeo, and G. Rizzo, *Development and validation of a model to detect active gear via OBD data for a Through-The-Road Hybrid Electric Vehicle.* IFAC Proceedings Volumes, 2014. **47**(3): p. 6618-6623.
10. Baek, S.-h. and J.-W. Jang, *Implementation of integrated OBD-II connector with external network.* Information Systems, 2015. **50**: p. 69-75.
11. Adafruit. *Introduction to Bluetooth Low Energy*. GATT 2020; Available from: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt.
12. Hassan, S.S., et al., *Security threats in Bluetooth technology.* Computers &

Security, 2018. **74**: p. 308-322.

13. Global5thailand. *GPS knowledge*. 2006; Available from: https://www.global5thailand.com/thai/gps.htm.
14. Goletz, M. and D. Ehebrecht, *How can GPS/GNSS tracking data be used to improve our understanding of informal transport? A discussion based on a feasibility study from Dar es Salaam.* Journal of Transport Geography, 2020. **88**: p. 102305.
15. Sirawit. ทำความรู้จัก *Firebase* และผลิตภัณฑ์ต่าง ๆ ในช่วงต้นปี *2019* กัน. 2019; Available from: https://medium.com/@sirawit/firebase-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-%E0%B8%97%E0%B8%B3%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81-firebase-%E0%B9%83%E0%B8%99%E0%B8%8A%E0%B9%88%E0%B8%A7%E0%B8%87%E0%B8%95%E0%B9%89%E0%B8%99%E0%B8%9B%E0%B8%B5-2019-%E0%B8%81%E0%B8%B1%E0%B8%99-473a8e8699fb.
16. Ohyver, M., et al., *The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test.* Procedia Computer Science, 2019. **157**: p. 396-405.

ประวัติผู้เขียน

ชื่อ-สกุล	สรวิชญ์ สาครินทร์
วัน เดือน ปี เกิด	6 กันยายน 2539
สถานที่เกิด	สงขลา
วุฒิการศึกษา	จุฬาลงกรณ์มหาวิทยาลัย
ที่อยู่ปัจจุบัน	208 ถนนนางงาม ตำบลบ่อยาง อำเภอเมือง จังหวัดสงขลา