ระบบตรวจจับความเร็วโดย OBDII
SYSTEMSPEEDDETECTORBY OBDII

กิตติคุณ นัยริยสัจ
KITTIKUN NAIRIYASAJ

วรศักดิ์ วิภาหัสน์
WORASAK WIPHAHUT

ศรราม ทองวิชิต
SRONRAM THONGWICHIT

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2557

ระบบตรวจจับความเร็วโดย OBDII
SYSTEM SPEED DETECTOR BY OBDII

โดย

กิตติคุณ นัยริยสัจ

วรศักดิ์ วิภาหัสน์

ศรราม ทองวิชิต

อาจารย์ที่ปรึกษา

ผศ.ดร.ภัทรพงษ์ ผาสุขกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2557

ปริญญานิพนธ์ปีการศึกษา 2557

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจจับความเร็วรถยนต์โดย OBD II

System speed detector by OBD II

ผู้จัดทำ นาย กิตติคุณ นัยริยสัจ รหัส 54010104

นาย วรศักดิ์ วิภาหัสน์ รหัส 54011136

นาย ศรราม ทองวิชิต รหัส 54011241

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

๓๑-๓๐

(ผศ.ดร.ภัทรพงษ์ ผาสุขกิจ)

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์	ระบบตรวจจับความเร็วรถยนต์โดย OBDII
นักศึกษา	นาย กิตติคุณ นัยริยสัจ
	นาย วรศักดิ์ วิภาหัสน์
	นาย ศรราม ทองวิชิต
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์
พ.ศ.	2557
อาจารย์ที่ปรึกษา	ผศ.ดร.ภัทรพงษ์ ผาสุขกิจ

บทคัดย่อ

ปริญญานิพนธ์นี้มีวัตถุประสงค์เพื่อการศึกษาการทำงานของอุปกรณ์ On-board diagnostics (OBDII) ซึ่งเป็นอุปกรณ์ที่ทำหน้าที่ตรวจเช็คนำข้อมูลจากกล่อง Electronics control unit (ECU) ของรถยนต์ผ่าน PORT OBD และตรวจจับความเร็วการขับขี่รถยนต์ โดยจะทำการแจ้งเตือนผ่าน Application จากโทรศัพท์มือถือระบบAndroid ที่รับข้อมูลจาก OBDII โดยการส่งผ่านแบบสัญญาณ Bluetooth โดยจะดูจากการเปลี่ยนแปลง ของความเร็ว ที่จะนำพาไปสู่การเกิดอุบัติเหตุ เพื่อที่จะเป็นประโยชน์แก่ผู้ขับขี่และช่วยลดการเกิดอุบัติเหตุบนท้องถนนได้

Thesis Title	System speed detector by OBDII
Student	Mr.Kittikun Nairiyasaj
	Mr.Worasak Wiphahut
	Mr.sronram Thongwichit
Program	Electronics Engineering
Year	2014
Project Adviser	Ass.Prof.Dr.Pattarapong Phasukkit

Abstract

The purpose of this thesis is to research The On-board diagnostics that check data from Electronics control unit box of car by PORT On-board diagnostics and capture speed then reply notification through application on Android system mobile that received from OBDII send by Bluetooth signal. it will check different of speed that will bring to the accident and give benefit to driver and can reduce an accident on the road.

กิตติกรรมประกาศ

ปริญญานิพนธ์เล่มนี้สำเร็จลุล่วงลงได้ด้วยความกรุณาของ ผศ.ดร.ภัทรพงษ์ ผาสุขกิจ อาจารย์ที่ปรึกษา ที่กรุณาให้คำปรึกษาและติดตามการดำเนินการอย่างใกล้ชิด ชี้แนะแนวทาง ข้อคิดเห็นและคำแนะตลอดจนการแก้ไข ซึ่งเป็นประโยชน์ต่อการดำเนินการวิจัย ข้าพเจ้าขอกราบขอบคุณเป็นอย่างสูง

ขอบคุณรุ่นพี่ และเพื่อนๆ ที่คอยให้กำลังใจและให้ความช่วยเหลือตลอดจนให้คำปรึกษาแก่ข้าพเจ้าในการทำปริญญานิพนธ์เล่มนี้ให้เสร็จสมบูรณ์

สุดท้ายนี้ขอกราบเท้าขอบพระคุณบิดา มารดา ผู้ให้กำเนิด เลี้ยงดู อบรมและให้โอกาสทางการศึกษาโดยสนับสนุนทุกๆด้านและให้กำลังใจด้วยดีเสมอมา

กิตติคุณ นัยริยสัจ
วรศักดิ์ วิภาหัสน์
ศรราม ทองวิชิต

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูปภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในยุคของโลกปัจจุบันนี้เทคโนโลยีที่ก้าวหน้าล้วนแล้วแต่ใช้อุปกรณ์อิเล็กทรอนิกส์เป็นส่วนมากและยังมีการพัฒนาและก้าวหน้าอย่างต่อเนื่อง รวมไปถึงยานพาหนะอย่าง รถยนต์ ก็ถือได้ว่าเป็นสิ่งที่มีความสำคัญมากในปัจจุบัน ปันจุบันอิเล็กทรอนิกส์เกี่ยวกับรถยนต์มีอย่างแพร่หลาย และรวมไปถึง Android Application ก็ถือว่ามีการพัฒนาอย่างมาก ซึ่งจะเห็นได้ว่าในชีวิตประจำวันของคนเรานั้นร่วมเต็มไปด้วยอิเล็กทรอนิกส์มากมาย

ในสังคมปัจจุบันนั้น การดื่มแอลกอฮอล์ในงานเลี้ยง งานสังสรรค์ ต่างๆ เป็นเรื่องปกติโดยเฉพาะนักศึกษา ส่วนมากแล้วการดื่มแอลกอฮอล์จนขาดสติและเกิดอุบัติเหตุทางการขับขี่นั้นมีเปอร์เซ็นมากในการเกิดอุบัติเหตุทางท้องถนน

ดังนั้นการทำ Speed detector system by OBDII (On-board diagnostics) เราได้ประยุกต์ความรู้เกี่ยวกับการออกแบบ Android Application บน Android Smartphone และเชื่อมต่อทาง Bluetooth กับ OBDII Bluetooth เพื่อรับค่าต่างๆจากเซ็นเซอร์บนรถยนต์ เช่น ความเร็วเครื่องยนต์ รอบเครื่องยนต์ เป็นต้น เพื่อตรวจจับ และ บันทึกค่าพฤติกรรมต่างๆของเครื่องยนต์ และจากการออกแบบ Application หลังจากเราได้ข้อมูลต่างๆที่ได้รับจาก OBDII แล้วเราได้ทำการพัฒนา Application ต่อโดยการนำค่าที่ได้มาใช้งานในการสร้าง Application

1.2 วัตถุประสงค์ของโครงงาน

1.เพื่อศึกษาและประยุกต์นำมาใช้งานจริงของ OBDII และ Android Application.

2.เพื่อศึกษาการทำงานของ OBDII.

3.เพื่อศึกษาและพัฒนา Android Application.

1.3 ทฤษฎีและแนวคิดที่ใช้ในโครงงาน

ในปัจจุบันอุบัติเหตุบนท้องถนนนั้นมีมากมายจำนวนมาก แต่สาเหตุหลักๆมากจากการประมาท การเมาแล้วขับ และ การหลับใน ซึ่งทำให้เกิดความสูญเสียมากทั้งทรัพย์สินและชีวิตเราจึงใช้สิ่งของที่ตอนนี้แพร่หลายนั้นคือโทรศัพท์มือถือระบบแอนดรอยด์ (Android) มาศึกษาการขับรถจะค่าพารามิเตอร์ต่างๆคือ

1.3.1 ความเร็ว (Speed)

ความเร็วนั้นเป็นตัวแปลสำคัญอย่างมาก เมื่อเราขับรถด้วยความเร็วนั้นตามทฤษี โมเมนตัม โมเมนตัม (P) = มวล (m) x ความเร็ว (V) ยิ่งความเร็ว (V) มากโมเมนตัม (P) ก็ยิ่งมาก ซึ่งตามกฎการอนุรักษ์พลังงาน พลังงานนั้นไม่มีทางสูญเสีย จาก แรงที่กระทำ = อัตราการเปลี่ยนแปลงโมเมนตัม เมื่อโมเมนตัม (P) มากจึงเกิดแรงในการชนมาก

1.3.2 จำนวนรอบต่อนาทีของเครื่องยนต์ (RPM)

RPM ย่อมาจาก Revolutions per Minute RPM หมายถึง จำนวนรอบต่อนาที (Revolutions per Minute) โดยหน่วยวัด RPM นั้นจะนิยมใช้ในการวัดอัตราการหมุนหรือจำนวนรอบการหมุนของเครื่องยนต์หรือมอเตอร์ ส่วนมากจะเขียนในรูปแบบที่แตกต่างกันไป เช่น rpm, RPM, r/min มักพบได้ในสินค้าบางประเภทเช่น ความเร็วรอบของเครื่องยนต์ที่หน้าปัดรถยนต์ หรือคุณสมบัติของฮาร์ดดิสก์ ตัวอย่างเช่นเขียนไว้ว่า 3,000r/min หมายถึงความเร็วในการหมุนอยู่ที่ 3,000 รอบต่อนาที

1.3.3 ลิ้นปีกผีเสื้อ (Throttle)

Air Throttle หรือ Throttle position sensor (TPS) หรือที่เรียกกันว่า ลิ้นปีกผีเสื้อ เพราะลักษณะของมันเป็นวงรีหรือปีกผีเสื้อที่กระพรือปิดเปิด มันมีหน้าที่ปิดเปิดให้อากาศที่บริสุทธิ์ผ่านไส้กรองอากาศ เข้าไปผสมกับน้ำมันในคาร์บิวฯ หรือ จากหัวฉีด โดยอาศัยกำลังดูดอากาศจากเครื่องยนต์ดูดเอาอากาศผ่านลิ้นนี้เข้าไป เหตุที่รถกินน้ำมันเจ้าตัวนี้ก็มีส่วนอยู่ด้วย เช่นสกปรกการเปิดหรือปิดไม่สนิท ปิดเปิดไปได้ตามจังหวะ ทำให้สมองกลสั่งน้ำมัน หรือ น้ำมันที่คาร์บิวถูกส่งเข้ามาเพื่อผสมให้ได้สัดส่วนการสันดาปไม่พอเหมาะ ทำให้กำลังเครื่องไม่ดี มีส่วนผสมหนาเกิดไปเผาไหม้ไม่หมด ส่วนนั้นก็จะถูกจังหวะดูดของวาวล์ไอเสียดูทิ้งไปออกทางท่อไอเสียเป็นน้ำมันส่วนเกิน มันก็สิ้นเปลืองโดยเปล่าประโยชน์ถึงต้องมีการล้างทำความสะอาด ถ้ามีการสึกหรอ เช่นหลวมก็ต้องปรับแต่งปรับปรุงเป็นต้น เพื่อให้แกนไม่หลวมจังหวะเปิดปิดแม่นยำ สะดวก ทุกอย่างของส่วนนี้ต้องสัมพันธ์กันหมด จะทำให้ ECU สั่งหรือ คาร์บิวฯ จ่ายน้ำมันและอากาศในสัดส่วนที่สมบูรณ์จึงจะได้กำลังแรงม้าที่ดี ทำให้ประหยัดน้ำมันขึ้น

1.4 ขอบเขตของโครงงาน

1. ศึกษาการขับรถในสถานการณ์ต่างๆ
2. หาความสัมพันธ์ของพารามิเตอร์ต่างๆ
3. สามารถสร้างแอพพลิเคชั่น แอนดรอยด์ (Android Application) ได้
4. แอพพลิเคชั่น (Application) สามารถบอกค่าพารามิเตอร์ต่างๆได้
5. แอพพลิเคชั่น (Application) สามารถเตือนได้เมื่อขับรถผิดปกติ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้เกี่ยวกับ ECU และ OBDII

2. สามารถนำความรู้เกี่ยวกับพารามิเตอร์ต่างๆไปใช้ในชีวิตประจำวันได้

3. มีความรู้ด้านการเขียนแอพพลิเคชั่น แอนดรอยด์ (Android Application) และสามารถนำไปประยุกต์ใช้ในชีวิตประจำวันได้

4. สามารถนำความรู้ที่ได้ไปศึกษาต่อยอดในการเขียนโปรแกรม และออกแบบแอพพลิเคชั่น (Application) ได้

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

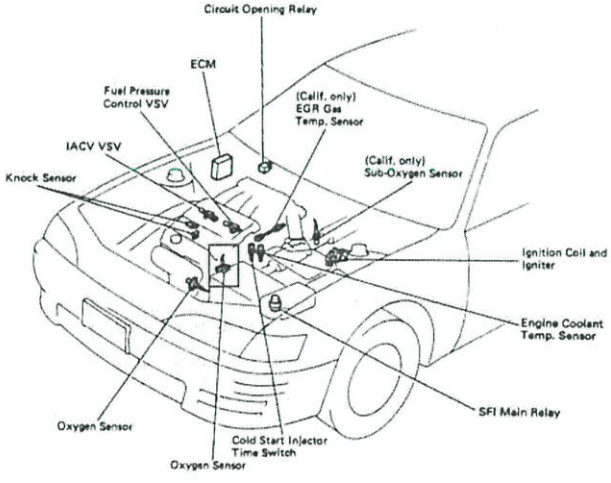
2.1. อุปกรณ์โอบีดีทู (OBDII)

อุปกรณ์โอบีดีทู ย่อมาจากคำว่า On-board diagnostics ซึ่งเป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่ตรวจเช็คและนำข้อมูลจากกล่อง ECU ของรถยนต์ส่งผ่านพอร์ตโอบีดี (Port OBD) มายังตัวโอบีดีทู (OBDII) และเราได้ใช้โอบีดีทู (OBDII) รุ่น ELM327 ใช้ระบบส่งข้อมูลแบบบลูทูธ (Bluetooth) ในกับอุปกรณ์ปลายทางเช่น Notebook smartphones และจากตัวอุปกรณ์ปลายทางสามารถเชื่อต่อสัญญาณบลูทูธ Bluetooth กับตัว ELM327 ได้จะสามารถใช้โปรแกรมหรือแอพพลิเคชั่นต่างๆในการอ่านค่าจากตัวโอบีดีทู (OBDII) ได้

รูปที่ 2.1 OBDII ELM327

2.1.1 เซ็นเซอร์ (Sensor) ต่างๆที่ใช้ศึกษา

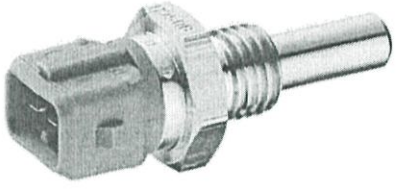
จากที่กล่าวข้างต้นว่าโอบีทีทู (OBDII) จะรับข้อมูลจากเซนเซอร์ (Sensor) ต่างๆ มากมายโดยเซ็นเซอร์ทั้งหมดในรถยนต์ ดังในรูปที่ 2.2



รูปที่ 2.2 เซนเซอร์ (Sensor) ต่างๆ ในรถยนต์

(ที่มา: http://www.clublexus.com/forums/attachments/es300-and-es330/151008d1251345781-92-es300-o2-sensor-location-the-hard-to-find-one-at-front-sensors.jpg)

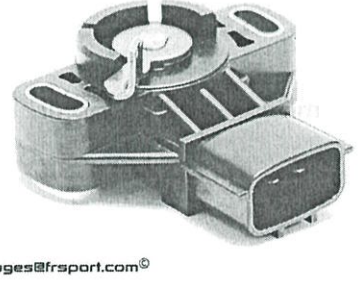
1.อุณหภูมิน้ำหล่อเย็นเซนเซอร์ (Coolant Temperature Sensor) มีหน้าที่วัดอุณหภูมิในระบบหล่อเย็นของเครื่องยนต์โดย ECU หรือ PCM จะใช้ข้อมูลจากเซนเซอร์ (sensor) ตัวนี้ควบควบการจุดระเบิดในรูปแบบต่างๆ การควบคุมการจ่ายน้ำมันเชื้อเพลิงและไอเสีย(มลพิษ) ตัวอย่างเช่น ขณะที่เครื่องยนต์เย็นอยู่ ส่วนผสมก็จะเข้มเพื่อให้เครื่องยนต์ทำงานและมีกำลังขับเคลื่อนตัวรถได้ แต่เมื่ออุณหภูมิของเครื่องยนต์สูงขึ้นได้ระดับหนึ่งแล้ว ECU ก็จะเริ่มใช้ข้อมูลจากอุณหภูมิน้ำหล่อเย็นเซนเซอร์ (Coolant Temperature Sensor)ปรับเปลี่ยนอัตราผสม ให้อยู่ในสภาวะการทำงานที่เหมาะสมแบบที่เรียกว่า วงจรปิด (Closed Loop Operation) ซึ่งจะทำให้มีระดับไอมลพิษต่ำสุด



รูปที่ 2.3 Coolant Sensor

(ที่มา: http://www.nzefi.com/wp-content/uploads/Engine-Coolant-Temperature-Sensor-e1361287218755.jpg)

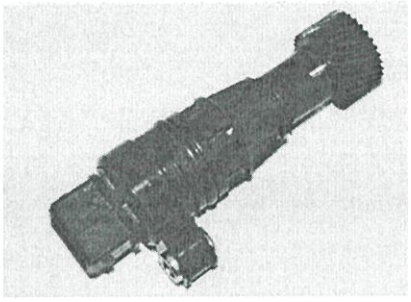
2. ค่าตำแหน่งของลิ้นปีกผีเสื้อ (Throttle Position Sensor) จะคอยบอก ECU ถึงตำแหน่ง (เปิด-ปิด) ของวาล์วหรือลิ้นปีกผีเสื้อ โดย ECU จะใช้ข้อมูลที่ได้รับ ทำการปรับเปลี่ยนระยะเวลาการจุดระเบิด และอัตราส่วนผสม เมื่อความต้องการกำลังของเครื่องยนต์เปลี่ยนแปลง แต่ในบางกรณีเวลาเร่งเครื่องอาจเกิดปัญหามีอาการวูบได้ (Flat spot) เหมือนกัน คล้ายกับเวลาปั๊มในคาร์บูเรเตอร์ (carburetor) เสีย



รูปที่ 2.4 Nissan 22620-53J01 Throttle Position Sensor(TPS)

(ที่มา: http://thumbs1.ebaystatic.com/d/l225/m/mWhD4sbnroVVGx9xTilCoHw.jpg)

3. ความเร็วของรถยนต์ที่วิ่งอยู่ (Vehicle speed sensor) จะเป็นตัวบอกให้ ECU รู้ว่า รถมีความเร็วเท่าไรเพื่อที่จะได้ใช้ไปกำหนดการทำงานในส่วนอื่นของเครื่องยนต์ เช่น torque converter lockup ในเกียร์ นอกจากนั้นสัญญาณยังถูกนำไปใช้ กับระบบอื่นๆ อีก รวมทั้ง ระบบกันล้อตายหรือที่เรียกว่า Anti-Brake Lock System (ABS) สำหรับใน Citroen ที่ระบบช่วงล่างเป็น Hydractive สัญญาณจาก Speed Sensor ยังถูกนำไปใช้โดย ECU Hydractive ด้วย เพื่อปรับให้ระบบช่วงล่างทำงาน ได้เหมาะตามสภาพในขณะนั้นๆ



รูปที่ 2.5 แสดงเซ็นเซอร์ (Sensor) วัดความเร็วรถ

(ที่มา: http://i00.i.aliimg.com/wsphoto/v0/32238326130/Car-Vehicle-Odometer-font-b-Speedometer-b-font-font-b-Sensor-b-font-83181-12020-For.jpg)

2.1.2 โหมด (MODE) การทำงานของโอบีดีทู (OBDII)

รูปแบบการทำงานของโอบีดีทู (OBDII) มีทั้งหมด 10 รูปแบบตามมาตรฐาน SAE J1979 คือ

ตารางที่ 2.1 ตารางแสดงโหมดการทำงานของโอบีดีทู (OBDII)

Mode (hex)	Description
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)

07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

โดยที่เราจะใช้ในการศึกษานั้นคือโหมด 01 นั้นคือโหมดใช้สำหรับให้โชว์ข้อมูลโดยสามารถแบ่งได้อีกเป็น ที่สำคัญคือ ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 ตารางแสดง PID OBDII

PID (hex)	Description	Min value	Max value	Units	Formula
05	Engine coolant temperature	-40	215	°C	A-40
0C	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
0D	Vehicle speed	0	255	km/h	A
0F	Intake air temperature	-40	215	°C	A-40
11	Throttle position	0	100	%	A*100/255

จากตารางที่ 2.2 สูตรในการนำไปใช้ในการคำนวณพารามิเตอร์นั้นจะแตกต่างกันออกไปตามค่าพารามิเตอร์นั้นๆ เช่น จาก Throttle position = A*100/255

โดย A คือ ค่าพารามิเตอร์ที่ 1 ที่จะได้รับจาก OBDII

255 คือ จำนวนเนื้อที่สำหรับเก็บค่า 1 ตัวอักษร = 2^8= 256

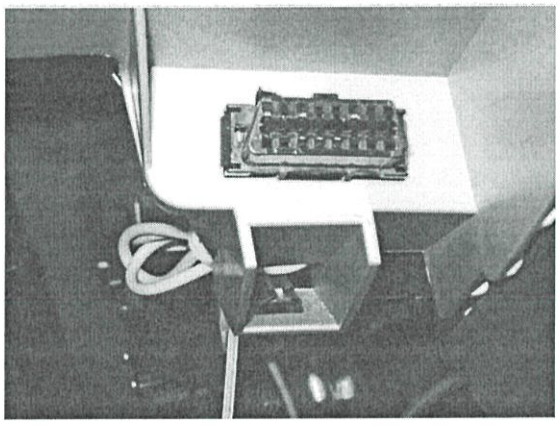
(เนื่องจากข้อมูลเริ่มจาก 0-255)

100 คือ ค่าเปอร์เซ็นต์ (%) ในการบอกถึงการเปิดปิดแบบสมบูรณ์

ซึ่งสูตรในการคำนวณค่าพารามิเตอร์ต่างๆ จะแตกต่างกันออกไปขึ้นอยู่กับสูตรนั้นๆ เช่น Engine RPM ก็จะใช้สูตร ((A*256)+B)/4 ในการคำนวณหาค่า RPM

2.2 พอร์ตโอบีดี (Port OBD)

ทำการต่อเข้ากับพอร์ตโอบีดี (Port OBD) ในเครื่องรถยนต์ซึ่งพอร์ตโอบีดี (Port OBD) นั้นจะรับส่งสัญญาณระหว่างเซนเซอร์ (sensor) ต่างๆในเครื่องยนต์ และอุปกรณ์โอบีดีทู (OBDII) เป็นอุปกรณ์สำหรับเชื่อมต่อกับ ECU รถยนต์ไร้สายผ่านทาง Bluetooth เพื่อใช้สำหรับดูข้อมูลต่างๆของเครื่องยนต์ที่ ECU ทราบ โดยสามารถใช้ได้กับรถยนต์รุ่นใหม่ๆส่วนมากหลังปี 2000 ที่มีพอร์ตโอบีดี (Port OBD) ซึ่งพอร์ต (Port) นี้ส่วนมากจะซ่อนอยู่ฝั่งคนขับใต้พวงมาลัยรถยนต์ดังรูป 2.6



รูปที่ 2.6 OBD Port

2.2.1 คุณสมบัติของพอร์ตโอบีดี (Port OBD)

1. ใช้สำหรับดูข้อมูลรหัสความผิดที่อีซียู ECU รถยนต์ได้ตรวจพบและบันทึกเก็บไว้ โดยเราจะสามารถนำค่านี้ไปค้นหาความผิดพลาดต่างๆของเครื่องยนต์ได้

2. สามารถลบค่าความผิดพลาดที่กล่อง ECU ตรวจพบและบันทึกไว้ได้ เมื่อเราซ่อมเสร็จไม่ต้องไปเสียเงินค่าลบที่ศูนย์หรือตามร้านค้า

3. สามารถลบไฟเตือน Check engine ที่ขึ้นโชว์ที่หน้าปัดได้

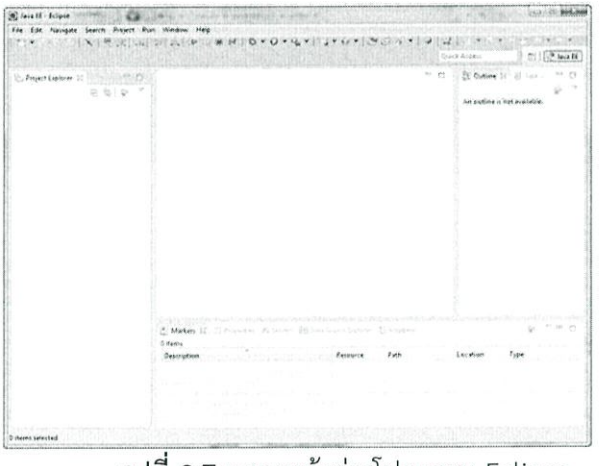
4. แสดงค่าสถานะเซ็นเซอร์ต่างๆ ที่ ECU มองเห็นแบบเรียลไทม์ได้ เช่น

- อัตราส่วนการผสมกันระหว่างอากาศกับเชื้อเพลิง (AF Ratio)
- ความเร็วรอบเครื่องยนต์ (Engine RPM)
- โหลดของเครื่องยนต์ที่อีซียู (ECU) คำนวณ (Calculated Load Value)
- อุณหภูมิน้ำหล่อเย็น (Coolant Temperature)
- สถานะของระบบจ่ายน้ำมัน (Fuel System Status)
- ความเร็วของรถยนต์ที่วิ่งอยู่ (Vehicle Speed)
- ค่าของทริมน้ำมันแบบอัพเดทเร็ว (Short Term Fuel Trim)
- ค่าของทริมน้ำมันแบบอัพเดทช้า (Long Term Fuel Trim)
- แรงดันในท่อร่วมไอดี (Intake Manifold Pressure)

- องศาไฟจุดระเบิด (Timing Advance)
- อุณหภูมิอากาศที่ท่อร่วมไอดี (Intake Air Temperature)
- อัตราการไหลของอากาศ (Air Flow Rate)
- ค่าตำแหน่งของลิ้นปีกผีเสื้อ (Absolute Throttle Position)
- แรงดันจากออกซิเจนเซ็นเซอร์ (Oxygen sensor voltages/associated short term fuel trims)
- แรงดันของน้ำมันเชื้อเพลิง (Fuel Pressure)

2.3 Eclipse Juno Program.

Eclipse เป็นเครื่องมือที่เรียกว่า integrated development environment (IDE) สำหรับพัฒนา Applications โดยใช้ java หรือภาษาอื่น ๆ เช่น C/C++, Python, PERL, Ruby ฯลฯ Eclipse มีการรองรับปลั๊กอินที่หลากหลาย ผู้พัฒนาที่ใช้ภาษา Java ในการพัฒนา application ต่าง ๆ สามารถใช้ Eclipse ในการพัฒนาได้ โดยตัวEclipse มีสภาวะแวดล้อมที่สมบูรณ์ คือมีเครื่องมือต่าง ๆ ให้ใช้พร้อม นอกจากนี้ Eclipse ยังสามารถใช้พัฒนาโปรแกรมภาษาอื่น ๆ ได้ถ้ามีตัว ปลั๊กอินนั้นอยู่ เช่น ถ้าเราต้องการพัฒนา application โดยใช้ภาษา php ถ้า Eclipse มีปลั๊กอินสำหรับภาษานี้ เราสามารถใช้ Eclipse ในการพัฒนาได้



รูปที่ 2.7 แสดงหน้าต่างโปรแกรม Eclipse

2.4 ยานพาหนะ

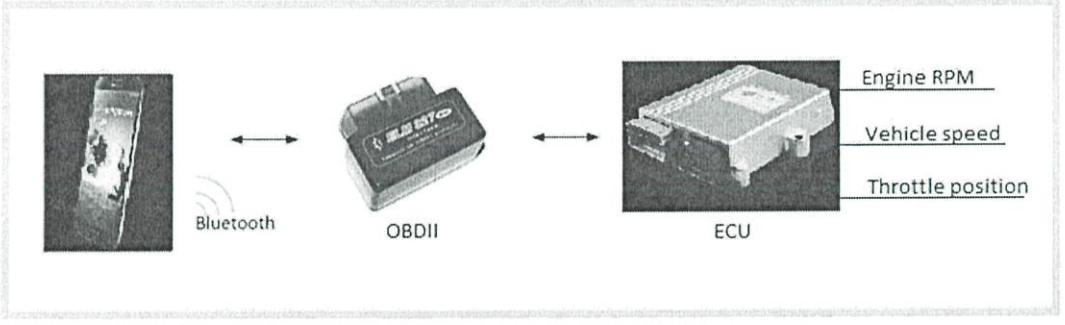
เนื่องจากเราทำการศึกษาเกี่ยวกับรถยนต์เราจึงต้องมีรถยนต์ที่ใช้ในการศึกษานั้นก็คือรถฟอร์ดเฟียสต้า (Ford Fiesta) ปี 2010 เครื่องยนต์ 1.6 cc turbo ดังในรูปที่ 2.8 มีคุณสมบัติดังนี้ เกียร์อัตโนมัติ (Automatic) 5 speed , เครื่องยนต์ 1600 ซีซี , speed ความเร็วสูงสุด 120 mph , ฐานล้อ: 98.0 นิ้ว. ความยาว: 155.6-155.9 in. ความกว้าง: 67.8 นิ้ว. ความสูง: 58.3 นิ้ว. น้ำหนัก: 2150-2250 ปอนด์. 5 ประตู

รูปที่ 2.8 แสดงรถฟอร์ด เฟียสต้า (Ford Fiesta) ที่ใช้ในการทดลอง
(ที่มา: http://www.moto123.com/ArtImages/111224/2010-ford-fiesta-i05.jpg)

บทที่ 3
การออกแบบและหลักการทำงาน

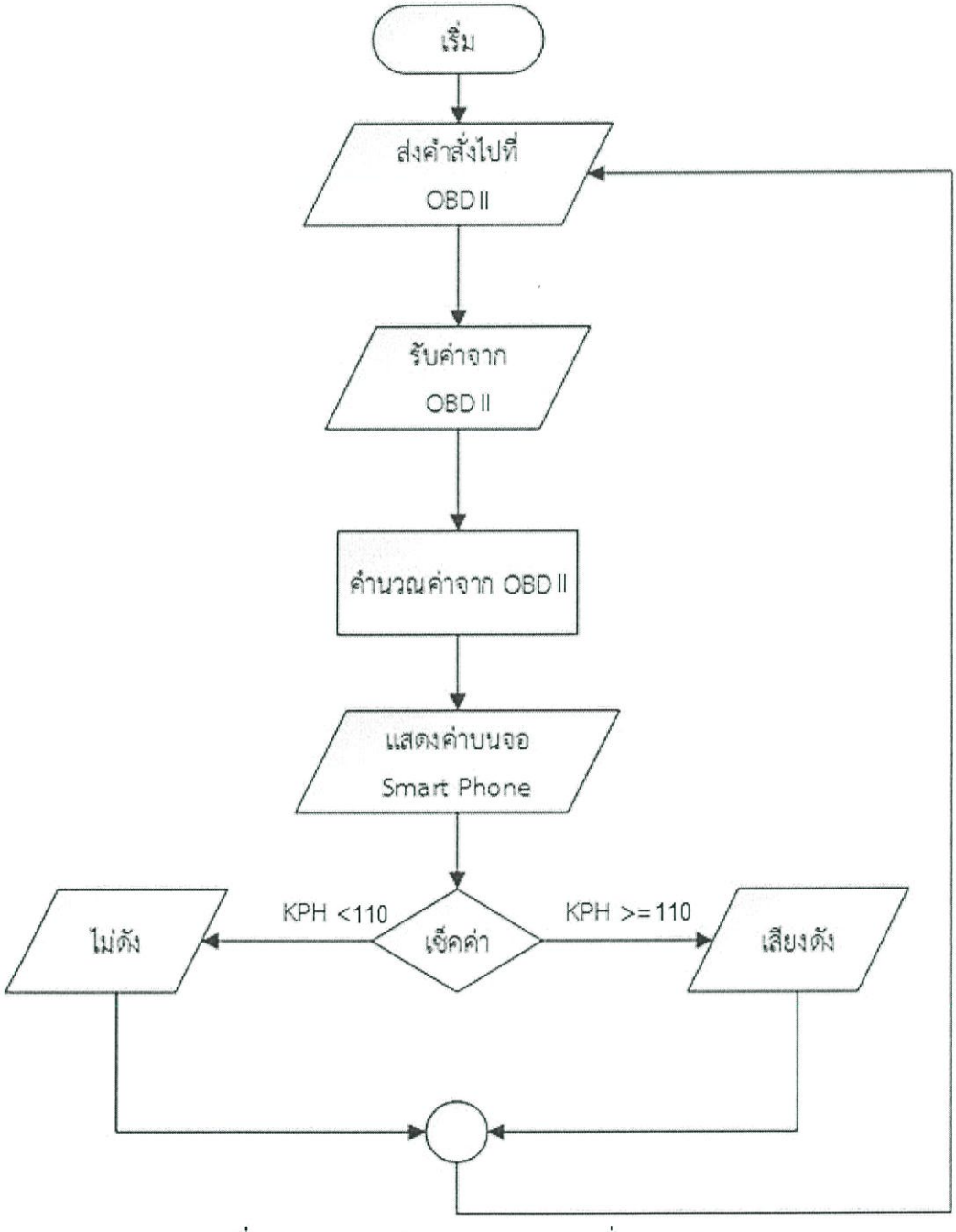
3.1 การออกแบบ

จากโครงงาน เราได้ออกแบบแอพพลิเคชั่นแอนดรอยด์เพื่อตรวจจับค่า Intake Temperature , Throttle Position Sensor (TPS), Coolant Temperature, รอบเครื่องยนต์ (RPM), ความเร็ว (Speed) ของเครื่องยนต์เพื่อทำการแจ้งเตือนเมื่อค่าต่างๆมีความเสี่ยง ผ่านทางโอบีดีทู (OBDII) นั้นเองเราจะทำการติดตั้งโอบีดีทู (OBDII) บน พอร์ตโอบีดีทู (OBDII) บนรถยนต์ ซึ่งรถยนต์ทุกคันที่สร้างหลังจากปี 1980 จะมี OBDII Port ทุกคัน หลังจากเราติดตั้ง OBDII แล้วเราจะทำการเชื่อมต่อแอพพลิเคชั่นแอนดรอยด์และ OBDII ผ่านทางบลูทูธ (Bluetooth) และหลังจากทำการเชื่อต่อกับบลูทูธ (Bluetooth) เสร็จแล้ว เราสามารถเปิดใช้งานแอพพลิเคชั่นได้ทันทีเนื่องจากการแจ้งเตือนจะทำงานอัตโนมัติเมื่อเชื่อมต่อเสร็จ โดยค่าพารามิเตอร์ต่างๆจะถูกตั้งไว้ตามค่าที่มา จากการเก็บสถิติของพฤติกรรมคนขับนั้นเอง โดยรูปของการทำงานอย่างง่าย ดังในรูปที่ 3.1



รูปที่ 3.1 โครงสร้างการทำงานอย่างง่าย

3.1.1 ออกแบบโปรแกรมการทำงานของแอพพลิเคชั่น (Application)



รูปที่ 3.2 แสดงการทำงานของแอพพลิเคชั่น (Application)

จากรูปที่ 3.2 แสดงการทำงานของแอพพลิเคชั่นเริ่มจากแอพพลิเคชั่น (Application) ส่งคำสั่งไปยังโอดีบีทู (OBDII) เพื่อไปสั่งให้อีซียู (ECU) ส่งข้อมูลกลับมาที่โอดีบีทู (OBDII) เพื่อส่งไปยังแอพพลิเคชั่น (Application) เพื่อนำข้อมูลนั้นมาคำควณและโชว์ในจอแสดงผล และทำการวิเคราะห์ข้อมูลที่ได้มาตามเงื่อนไข

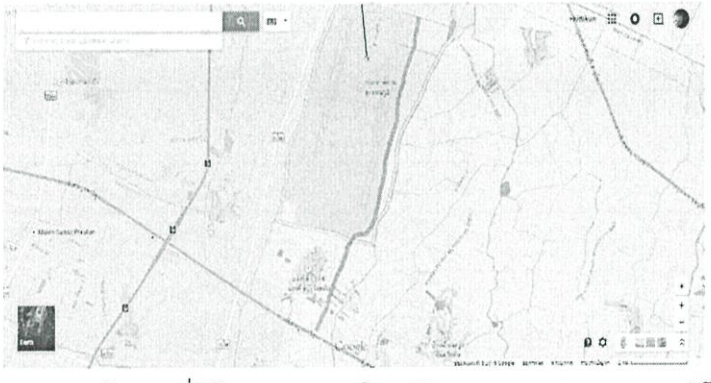
3.2 หลักการทำงาน

จากการออกแบบที่กล่าวมาเราได้ทำการติดตั้งโอบีดีทู (OBDII) เข้ากับพอร์ตโอบีดีทู (OBDII) ในรถยนต์ ฟอร์ด เฟียสต้า รุ่นปี 2010 โดยมีรายละเอียดดังนี้ เกียร์อัตโนมัติ (Automatic) 5 speed , เครื่องยนต์ 1600 ซีซี , speed ความเร็วสูงสุด 120 mph , ฐานล้อ: 98.0 นิ้ว. ความยาว: 155.6-155.9 in. ความกว้าง: 67.8 นิ้ว. ความสูง: 58.3 นิ้ว. น้ำหนัก: 2150-2250 ปอนด์. จากนั้นจะเป็นส่วนของการเขียน Application เพื่อเชื่อมต่อกับ โอบีดีทู (OBDII)

3.2.1 ทำการศึกษาพฤติกรรมการขับรถของคนปกติ และ คนที่ไม่ปกติ โดยทดลอง 2 ส่วน คือ

1. การทดลองความสัมพันธ์ของค่าพารามิเตอร์แต่ละชนิด โดยเก็บตัวอย่างจากคนขับ 6 คน แบ่งเป็น ชาย 4 คน หญิง 2 โดยใช้รถที่ติดตั้งโอบีดีทู (OBDII) เข้ากับพอร์ตโอบีดีทู (OBDII) ในรถยนต์ ฟอร์ด เฟียสต้า (Ford Fiesta) ข้างต้น โดยใช้เส้นทาง ดังรูปที่ 3.2

2. การทดลองขับรถในสภาวะต่างๆที่ผิดปกติ โดยการทดลองนี้เราจะทดสอบโดยการให้ผู้ทดลอง 3 คน ทำการขับรถในสถานะที่ต่างกัน คือ อดนอน กินยาที่ทำให้มีอาการง่วงนอน และ ดื่มเครื่องดื่มแอลกอฮอล์ โดยใช้เส้นทางดังรูปที่ 3.3



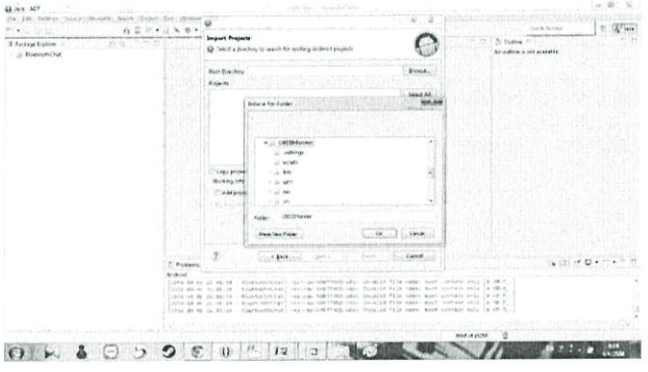
รูปที่ 3.3 แสดงเส้นทางที่ใช้ในการทดลองไปกลับระยะทางประมาณ 20 กิโลเมตร

3.2.2 การเขียนแอพพลิเคชั่น (Application)

จากการออกแบบแอพพลิเคชั่น (Application) เราทำการนำโค้ด มาทำการพัฒนาเขียนเงื่อนไขต่างๆตามที่เราต้องกการและออกแบบเสียงเตือนให้กับแอพพลิเคชั่น (Application) โดยเขียนโปรแกรมแอพพลิเคชั่น (Application) โดยใช้ Java ทำการเปิดโปรแกรมขึ้นมา และทำการ Import ไฟล์แอพพลิเคชั่น (Application) โดยการไปที่ File > > Import ตามรูปที่ 3.4

รูปที่ 3.4 การ Import ไฟล์แอพพลิเคชั่น (Application)

หลังจากนั้นให้กดที่ Browse แล้วเลือกไฟล์ Application code ที่เราออกแบบไว้ตามรูปที่ 3.5

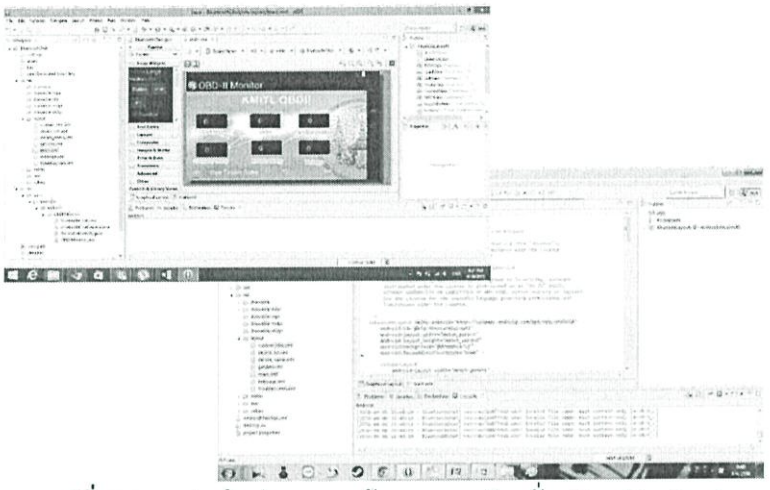


รูปที่ 3.5 เลือกไฟล์ Application code

เมื่อเราทำการ Import code ที่เราออกแบบแอพพลิเคชั่น (Application) เบื้องต้นไว้เข้ามาในโปรแกรมแล้วจะได้ตามรูปที่ 3.5 ซึ่งจะเป็นcode ในส่วนของคำสั่งต่างๆ และอีกส่วนที่สำคัญคือการออกแบบหน้าตาของ แอพพลิเคชั่น (Application) จากการออกแบบเบื้องต้นของเราจะได้ตามรูปที่ 3.6

รูปที่ 3.6 code ในส่วนของคำสั่งต่างๆ

ทำการเขียนโค้ดเพื่อให้แอพพลิเคชั่น (Application) รับข้อมูลพารามิเตอร์ 6 ชนิด คือ Intake Temperature, Throttle Position Sensor (TPS), Coolant Temperature, รอบเครื่องยนต์ (RPM), ความเร็ว (Speed)แล้วนำค่าที่ได้มาแสดงผลในแอพพลิเคชั่น (Application) โดยการออกแบบหน้าตา แอพพลิเคชั่น (Application) นี้เราสามารถออกแบบโดยการวาดออกมาได้เลย และในส่วนของ code หน้าตา แอพพลิเคชั่น (Application) นั้นจะไปอยู่หน้าต่างของ main.xml จะเป็นส่วน code นั้นเอง ตามรูปที่ 3.7



รูปที่ 3.7 code ในส่วนของหน้าตาแอพพลิเคชั่น (Application)

```
            final TextView TX = (TextView)
findViewById(R.id.TXView2);

            switch(messagenumber) {

            case 1:
                  sendMessage("01 0C" + '\r'); //get RPM
                  TX.setText("01 0C");
                  messagenumber++;
                  break;
            case 2:
                  sendMessage("01 0D" + '\r'); //get speed
                  TX.setText("01 0D");
                  messagenumber++;
                  break;
            case 3:
                  sendMessage("01 11" + '\r'); //Throttle position
                  TX.setText("01 11");
                  messagenumber++;
                  break;
            case 4:
                  sendMessage("01 05" + '\r'); //get Coolant
Temperature
                  TX.setText("01 05");
                  messagenumber++;
                  break;
            case 5:
                  sendMessage("01 0F" + '\r'); //get Intake
Temperature
                  TX.setText("01 0F");
                  messagenumber++;
                  break;
            case 6:
                  sendMessage("AT RV" + '\r'); //get Voltage
                  TX.setText("AT RV");
                  messagenumber++;
                  break;
            default: ;
            }
      }
```

รูปที่ 3.8 แสดงโค้ดการทำงานในส่วนของการรับข้อมูล ทั้ง 6 ตัว

จากรูปที่ 3.8 ยกตัวอย่าง case 1 คือการรับข้อมูลของค่ารอบเครื่องต่อนาที (RPM) จากคำสั่ง sendMessage (“01 0C” + ‘r’);

ทำการเขียนโค้ดเพื่อให้แอพพลิเคชั่น (Application) เพื่อนำค่าที่ได้จากโอบีดีทู (OBDII) ทั้ง 6 ค่ามาคำนวณเพื่อแสดงผลดังรูปภาพที่ 3.9

```
switch(PID) {

            case 15://PID(0F): Intake Temperature
                value = value - 40; //Formula for Intake Temperature
                String displayIntakeTemp = String.valueOf(value);
                intakeTemperature.setText(displayIntakeTemp);
                break;
            case 17://PID(11): Throttle position
                value = (value * 100 ) / 255;
                String displayThrottle = String.valueOf(value);
                engineLoad.setText(displayThrottle);
                double c= Double.parseDouble(displayThrottle);
                  if (c>=70) {
                       mpAudio.start();
                       }
                   else{
                       mpAudio.pause();
                       }
                break;
            case 5://PID(05): Coolant Temperature

                  value = value - 40;
                  String displayCoolantTemp = String.valueOf(value);
                  coolantTemperature.setText(displayCoolantTemp);
                break;
            case 12: //PID(0C): RPM
                int RPM_value = (value*256)/4;

                String displayRPM = String.valueOf(RPM_value);
                RPM.setText(displayRPM);
                double b= Double.parseDouble(displayRPM);
                if (b>=3500) {
                   mpAudio.start();

                 }
                  else{
                   mpAudio.pause();
                }
                break;

            case 13://PID(0D): MPH
                value = (value*5)/5; //convert KPH to MPH
               String displayMPH = String.valueOf(value);
                MPH.setText(displayMPH);
                double a= Double.parseDouble(displayMPH);
                 if (a>=50) {
                   mpAudio.start();

                   }
                  else{
                  mpAudio.pause();
                  }
                   break;
```

รูปที่ 3.9 แสดง code การคำนวณค่าต่างๆ และเข้าเงื่อนไข

จากรูปที่ 3.9 ยกตัวอย่างการทำงาน case 17 คือ

```
case 17://PID(11): Throttle position
   value = (value * 100 ) / 255;
   String displayThrottle = String.valueOf(value);
   engineLoad.setText(displayThrottle);
   double c= Double.parseDouble(displayThrottle);
 if (c>=70) {
    mpAudio.start();
    }
  else{
    mpAudio.pause();
    }
   break;
```

Code จะทำงานรับค่า TPS จากโอบีดีทู (OBDII) แล้วนำมาคำนวณหาให้เป็นเปอร์เซ็นต์ จากนั้น จะดูว่าค่าที่ได้นั้นมีค่าเกิน ที่ 70 หรือไม่ ซึ่งถ้าเกิน จะสั่งให้แอพพลิเคชั่น (Application) ส่งเสียงเตือน

หลังจากนั้นเราจะทำการ เพิ่มเงื่อนไขตามที่เราต้องการแอพพลิเคชั่น (Application) ออกแบบโดยกำหนดให้แอพพลิเคชั่น ร้องเตือนเมื่อ ขับที่ความเร็ว เกิน 50 km/h หรือเมื่อค่าตำแหน่งปีกผีเสื้อ (TPS) มีค่ามากกว่าหรือเท่ากับ 70 %

และเมื่อเราเขียน Code เสร็จ เราจะติดตั้ง แอพพลิเคชั่น (Application) ลง สมาร์ทโฟน (Smartphone) และทำการทดสอบ แอพพลิเคชั่น (Application) ในขั้นตอนต่อไป

บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลองพฤติกรรมคนขับรถ

การทดลอง การตรวจจับพฤติกรรมเครื่องยนต์จากบุคคลต่างๆ ในแต่ละกรณี จากการทดลองนี้เราจะทำการบันทึกค่าพฤติกรรมรถยนต์ ของคนขับรถยนต์โดยแบ่งเป็น 2 การทดลอง

4.1.1 การทดลองความสัมพันธ์ของค่าพารามิเตอร์แต่ละชนิด

โดยการทดลองแรกทดลองกับผู้ทดลองทั้งหมด 6 คน ชาย 4 หญิง 2 ขับรถในเส้นทางที่กำหนดเพื่อนำค่าที่ได้ของแต่ละคนมาศึกษาดูความสัมพันธ์ของค่าพารามิเตอร์ต่างๆ ลำดับขั้นตอนการทดลองและจากการเก็บค่าพารามิเตอร์ต่างๆ เราต้องนำค่าที่ได้มาทำการเปรียบเทียบ หาความถูกต้องของค่าต่างๆตามสูตรจากตารางที่ 2.2 เช่น การคำนวณความถูกต้องขอค่า RPM เมื่อเราส่งขอมูล 01 0C ไปยังโอบีดีทู (OBDII) และ โอบีดีทู (OBDII) จะส่งข้อมูลกลับมาเป็นเลขฐาน 16 เช่น 24 0F ซึ่งเราจะเอามาคำนวณโดยให้ 0F คือ A และ 23 คือ B ซึ่งเราต้องแปลเป็นฐาน 10 คือจะได้ A = 15 B = 36 แล้วนำมาเข้าสูตร ((A*256)+B)/4 ได้ ((15*256)+36)/4 ได้ 969 รอบ/นาที แล้วเป็นข้อมูลที่ถูกต้อง

1. ทำการติดตั้ง OBDII บน OBD Port บนรถยนต์ตามรูปที่ 4.1

รูปที่ 4.1 รูปแสดงการเชื่อมต่อ OBDII กับ รถยนต์

2. ทำการเชื่อมต่อ OBDII Bluetooth และ Android Bluetooth

3. ทำการเปิดแอพพลิเคชั่น (Application) ตามรูปที่ 4.3 และเริ่มขับขี่เพื่อทำการบันทึกค่าต่อไป

รูปที่ 4.2 แสดงแอพพลิเคชั่น (Application) ที่ใช้ในการเก็บข้อมูลการทดลอง

4. ทำการเก็บข้อมูลการขับขี่โดยจะเก็บข้อมูลทุกๆ 30 วินาที จะได้ตามตารางที่ 4.1, 4.2, 4.3 และ 4.4 และนำไปเขียนกราฟได้ตามรูปที่ 4.3, 4.4, 4.5, 4.6, 4.7 และ 4.8

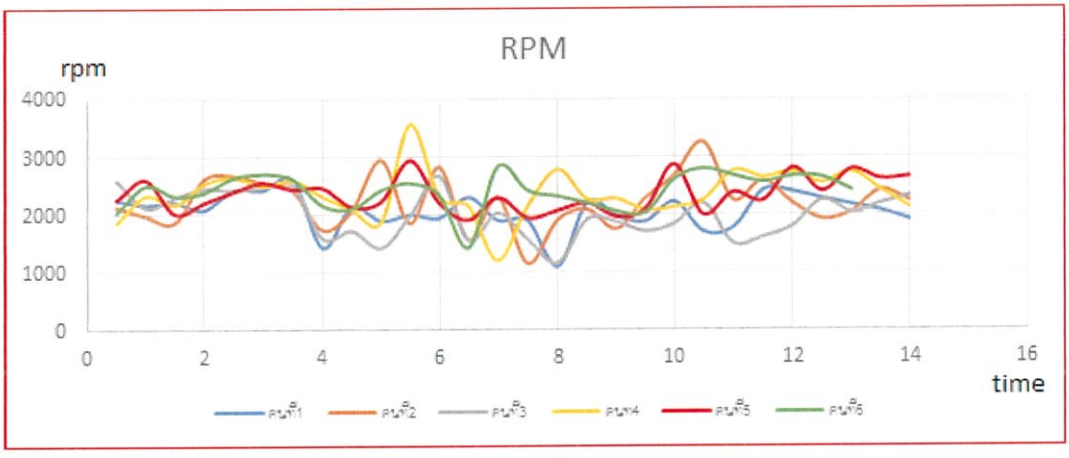
ตารางที่ 4.1 ตารางแสดงค่า RPM ในเวลาต่างๆ

เวลา / ชื่อ	0.5	1	1.5	2	2.5	3	3.5	4	4.5
คนที่ 1	2252	2169	2210	2084	2429	2425	2613	1427	2136
คนที่ 2	2128	1980	1863	2615	2678	2555	2427	1734	2075
คนที่ 3	2591	2125	2296	2452	2418	2482	2507	1583	1714
คนที่ 4	1853	2328	2171	2537	2646	2500	2589	2314	2081
คนที่ 5	2261	2605	2018	2206	2399	2556	2436	2461	2130
คนที่ 6	2042	2503	2324	2386	2641	2704	2616	2157	2100

เวลา / ชื่อ	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
คนที่ 1	1890	2001	1941	2302	1899	1895	1098	2203	1970	1876
คนที่ 2	2946	1851	2823	1563	2301	1148	1908	2087	1743	2275
คนที่ 3	1422	1981	2672	1555	2027	1570	1160	1918	1870	1707
คนที่ 4	1868	3563	2297	2148	1202	2140	2779	2262	2272	2074
คนที่ 5	2228	2943	2200	1904	2292	1942	2069	2194	1961	2086
คนที่ 6	2425	2542	2330	1426	2836	2416	2318	2198	2054	2034

เวลา / ชื่อ	10	10.5	11	11.5	12	12.5	13	13.5	14
คนที่ 1	2222	1698	1769	2422	2394	2272	2169	2068	1908
คนที่ 2	2674	3255	2249	2592	2192	1919	2042	2418	2236
คนที่ 3	1838	2195	1498	1608	1799	2256	2033	2190	2324
คนที่ 4	2126	2252	2754	2644	2736	2548	2740	2422	2119
คนที่ 5	2865	1993	2380	2240	2807	2400	2783	2616	2656
คนที่ 6	2616	2796	2674	2565	2670	2644	2420	-	-

จากตารางที่ 4.1 จะสังเกตได้ว่าผู้ทดลอง 6 คนนั้นค่าที่ได้นั้นแตกต่างกันโดยจะมีค่าอยู่ที่ตั้งแต่ 1000 – 3500 ซึ่งค่าส่วนมากจะอยู่ที่ประมาณ 1500 - 3000 ดังแสดงในรูปที่ 4.3



รูปที่ 4.3 กราฟแสดงความสัมพันธ์ระหว่าง RPM กับ เวลา

จากรูปที่ 4.3 กราฟแสดงความสัมพันธ์ระหว่าง RPM กับ เวลา กราฟจะรวมกันอยู่ที่ประมาณ 1500 – 3000 ซึ่งเป็นการขับในสภาวะปกติโดยที่ไม่มีการแซงเพื่อเร่งความเร็ว แต่จากกราฟที่เห็นเมื่อเราแซงกราฟจะพุ่งขึ้นสูงเกิน 3000 รอบ/นาที ไปจนเกือบถึง 4000 รอบ/นาที

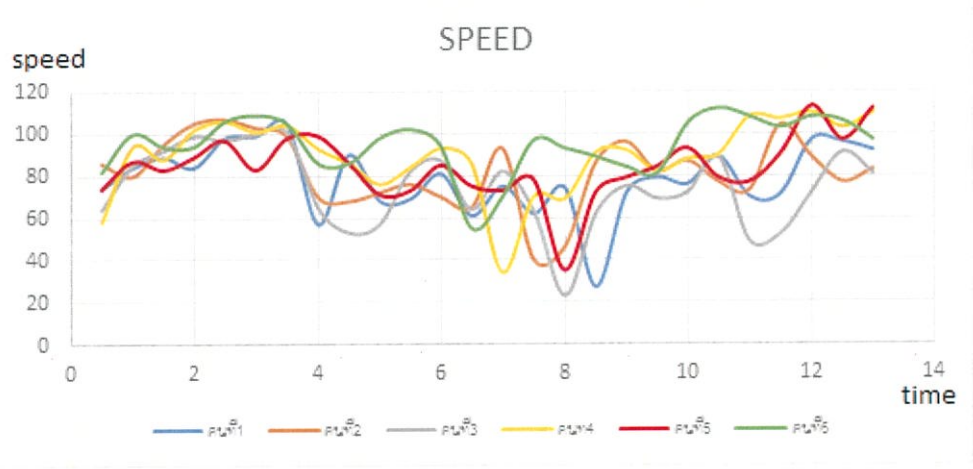
ตารางที่ 4.2 แสดงค่าความเร็ว (Speed) ในเวลาต่างๆ

เวลา / ชื่อ	0.5	1	1.5	2	2.5	3	3.5	4	4.5
คนที่ 1	74	84	89	84	98	99	105	57	90
คนที่ 2	86	80	95	105	107	103	99	70	68
คนที่ 3	64	85	92	99	97	100	101	65	53
คนที่ 4	58	94	88	102	106	101	104	93	86
คนที่ 5	74	87	83	89	97	83	98	99	86
คนที่ 6	82	100	94	94	106	109	105	86	86

เวลา / ชื่อ	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
คนที่ 1	68	69	81	61	75	62	74	27	72	79
คนที่ 2	72	76	70	65	93	40	46	86	96	82
คนที่ 3	57	82	87	64	82	63	23	62	75	69
คนที่ 4	76	84	93	86	34	70	69	91	92	82
คนที่ 5	71	73	85	75	73	78	35	72	79	84
คนที่ 6	98	102	94	55	70	97	93	89	84	81

เวลา / ชื่อ	10	10.5	11	11.5	12	12.5	13	13.5
คนที่ 1	76	89	70	71	97	96	92	88
คนที่ 2	87	77	73	104	89	77	83	97
คนที่ 3	72	89	49	52	72	91	81	88
คนที่ 4	88	90	108	107	110	103	110	92
คนที่ 5	93	79	77	90	113	97	112	105
คนที่ 6	105	112	108	103	108	106	97	-

จากตารางที่ 4.2 แสดงความเร็วในเวลาต่างๆของผู้ทดลอง จะเห็นได้ว่า ความเร็วของผู้ทดลองจะอยู่ที่ประมาณตั้งแต่ 60 – 110 กิโลเมตร/ชั่วโมง ขึ้นอยู่กับแต่ละคน จะขับความเร็วมากหรือน้อย



รูปที่ 4.4 กราฟแสดงความสัมพันธ์ระหว่างความเร็ว กับ เวลา

จากรูปที่ 4.4 กราฟแสดงความสัมพันธ์ระหว่างความเร็วกับเวลา จะเห็นได้ว่ากราฟมีความสอดคล้องกัน โดยช่วงแรก ความเร็วกราฟค้อนข้างจะคงที่ และเริ่มตกและเพิ่มจากการเข้าโค้งๆต่างๆ ซึ่งความเร็วที่มากๆจะอยู่ในช่วงทางตรง

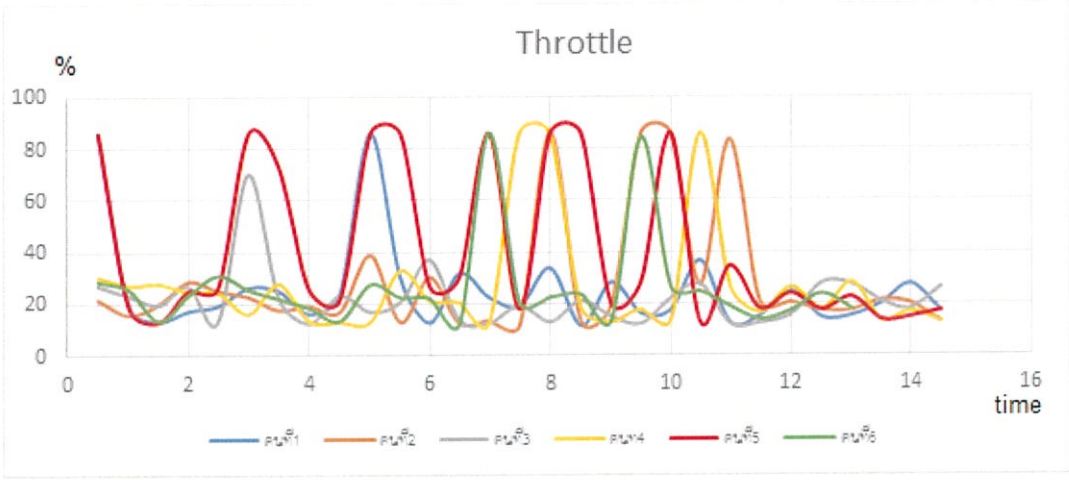
ตารางที่ 4.3 แสดงค่า Throttle Position ในเวลาที่ต่างกัน

เวลา / ชื่อ	0.5	1	1.5	2	2.5	3	3.5	4	4.5
คนที่1	85.9	19.2	13.3	17.3	19.3	26.3	25.1	16.1	23.9
คนที่2	21.6	15.7	20	28.6	25.1	22.7	17.6	19.2	16.9
คนที่3	27.1	23.1	19.6	26.7	13.3	70.2	22.4	12.5	23.1
คนที่4	30.2	27.1	27.8	24.7	24.3	16.1	27.8	13.3	12.9
คนที่5	85.9	18.4	12.9	25.1	26.3	85.9	73.3	25.5	21.6
คนที่6	28.6	25.5	13.3	23.5	31	25.5	22	18.4	13.3

5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
85.9	29.8	12.7	31.8	22.4	19.2	33.7	11.4	28.2	16.1
38.8	12.9	30.2	12.5	13.3	11.4	85.9	12.9	19.2	85.9
16.9	20.8	36.9	13.7	12.5	19.2	12.9	21.2	14.9	12.2
12.9	32.9	21.2	20.4	12.2	85.9	85.9	18.4	12.9	17.6
85.9	85.9	26.7	30.2	85.9	18	85.9	85.9	20	28.2
27.5	22.4	22	12.2	85.9	20.4	22.4	23.5	13.3	84.3

10	10.5	11	11.5	12	12.5	13	13.5	14	14.5
17.6	36.5	12.5	15.3	24.7	14.5	15.3	20	27.5	16.9
85.9	27.5	83.5	20.4	20.4	17.3	17.3	21.2	20	12.9
22	27.5	12.2	12.5	15.7	27.8	27.4	20.8	18	26.3
13.3	85.9	26.7	16.5	26.3	17.6	28.2	13.3	16.9	12.9
85.9	12.9	34.5	18	23.9	17.6	22.4	13.7	14.7	17.3
26.3	24.7	18.8	13.7	17.3	23.7	17.6	-	-	-

จากตารางที่ 4.3 แสดงค่า Throttle Position ในเวลาต่างๆของผู้ทดลอง จะเห็นได้ว่า ค่า Throttle Position ของผู้ทดลองจะอยู่ที่ประมาณตั้งแต่ 12 – 30% แต่มีค่า Throttle Position กระโดดขึ้นมา 85.9% เกิดจากการเหยียบคันเร่ง ทำให้ค่า Throttle Position มีค่าเปอร์เซ็นต์สูงมาก ดังแสดงในรูปที่ 4.5



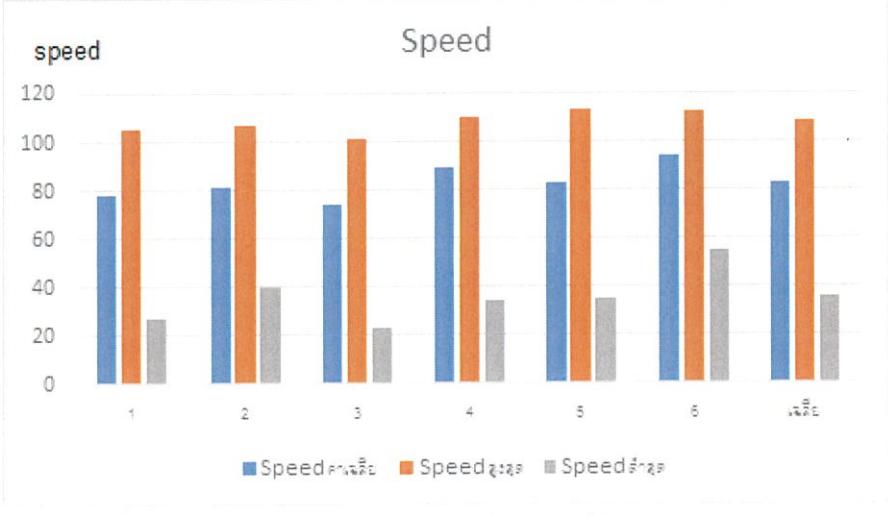
รูปที่ 4.5 กราฟแสดงความสัมพันธ์ระหว่าง Throttle Position กับ เวลา

จากรูปที่ 4.5 กราฟแสดงความสัมพันธ์ระหว่าง Throttle Position กับ เวลา จะเห็นว่า กราฟสวนใหญ่จะอยู่ในช่วง 10 – 20 % แล้วจะมีช่วงที่กระโดดสูง ขึ้นมา คือ ช่วง 70-90 % นั้นคือ ช่วงเวลาที่มีการเร่งความเร็ว

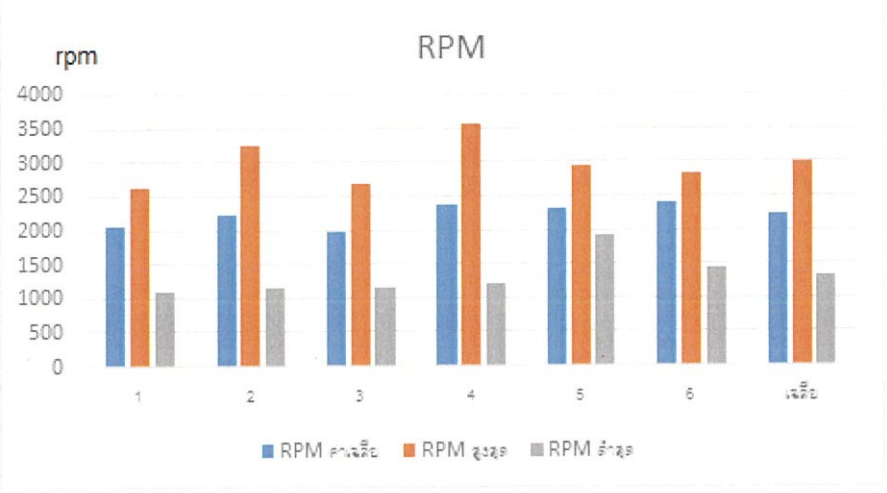
ตารางที่ 4.4 ค่าเฉลี่ยของแต่ละบุคคล

บุคคล	Speed			RPM			Throttle Position		
	ค่าเฉลี่ย	สูงสุด	ต่ำสุด	ค่าเฉลี่ย	สูงสุด	ต่ำสุด	ค่าเฉลี่ย	สูงสุด	ต่ำสุด
1	78	105	27	2067	2613	1098	25	85.9	11.4
2	81	107	40	2216	3255	1148	30	85.9	11.4
3	74	101	23	1972	2672	1160	21	70.2	12.2
4	89	110	34	2362	3563	1202	28	85.9	12.2
5	83	113	35	2321	2943	1904	26	85.9	12.9
6	94	112	55	2401	2836	1426	26	85.9	12.2
เฉลี่ย	83	108	35	2223	2980	1323	26	83.3	12

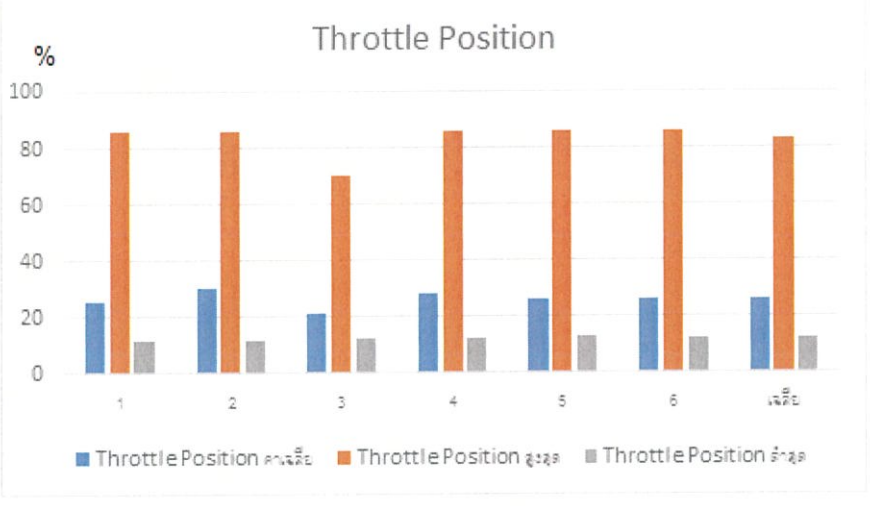
จากตารางที่ 4.4 แสดงให้เห็นถึงค่าเฉลี่ย สูงสุด ต่ำสุด ของค่าความเร็ว RPM และ Throttle Position จากตารางความเร็ว (Speed) ความเร็วเฉลี่ยของผู้ทดลองอยู่ที่ประมาณ 83 กิโลเมตร/ชั่วโมง ความเร็วสูงสุดเฉลี่ยอยู่ที่ประมาณ 108 กิโลเมตร/ชั่วโมง ความเร็วต่ำสุดเฉลี่ยอยู่ที่ประมาณ 35 กิโลเมตร/ชั่วโมง ดังรูปที่ 4.6 จากตาราง RPM ค่า RPM เฉลี่ยของผู้ทดลองอยู่ที่ประมาณ 2223 รอบ/นาที ค่า RPM สูงสุดเฉลี่ยอยู่ที่ประมาณ 2980 รอบ/นาที ค่า RPM ต่ำสุดเฉลี่ยอยู่ที่ประมาณ 1323 รอบ/นาที ดังรูปที่ 4.7 จากตาราง Throttle Position ค่า Throttle Position เฉลี่ยของผู้ทดลองอยู่ที่ประมาณ 26% ค่าThrottle Position สูงสุดเฉลี่ยอยู่ที่ประมาณ 83.3% ค่า Throttle Position ต่ำสุดเฉลี่ยอยู่ที่ประมาณ 12% ดังรูปที่ 4.8



รูปที่ 4.6 แผนภูมิแสดงค่าความเร็วเฉลี่ย สูงสุด ต่ำสุด ของแต่ละบุคคล



รูปที่ 4.7 แผนภูมิแสดงค่า RPM เฉลี่ย สูงสุด ต่ำสุด ของแต่ละบุคคล



รูปที่ 4.8 แผนภูมิแสดงค่า TPS เฉลี่ย สูงสุด ต่ำสุด ของแต่ละบุคคล

4.1.2 การทดลองขับรถในสภาวะต่างๆที่ผิดปกติ

โดยการทดลองนี้เราจะทดสอบโดยการให้ผู้ทดลอง 3 คน ทำการขับรถในสถานะที่ต่างกัน คือ อดนอนกินยาที่ทำให้มีอาการง่วงนอน และ ดื่มเครื่องดื่มแอลกอฮอล์

ลำดับขั้นตอนการทดลอง

1. ให้ผู้ทดลองคนที่ 1 อดนอนเป็นเวลาหนึ่ง จนทำให้มีอาการง่วงนอน ให้ผู้ทดลองคนที่ 2 ให้กินยาแก้แพ้ซึ่งมีผลทำให้เกิดอาการง่วงนอน ให้ผู้ทดลองคนที่ 3 ให้ กินเครื่องดื่มแอลกอฮอล์ไปประมาณหนึ่ง แล้วใช้เครื่องเป่าแอลกอฮอล์ ตรวจดู ดังรูปที่ 4.9 และ 4.10

รูปที่ 4.9 รูปแสดงผู้ทดลองเป่าเครื่องเป่าแอลกอฮอล์

รูปที่ 4.10 รูปแสดงเครื่องเป่าแอลกอฮอล์บอกค่าของแอลกอฮอล์

2. ทำการเชื่อมต่อโทรศัพท์กับโอบีดีทู (OBDII)

3. เปิดแอพพลิเคชั่น (Application) เพื่อทำการเก็บข้อมูล

4. ทำการเก็บข้อมูลการขับขี่โดยจะเก็บข้อมูลทุกๆ 30 วินาที จะได้ตามตารางที่ 4.5, 4.6, 4.7 และ 4.8 และนำไปเขียนกราฟได้ตามรูปที่ 4.11, 4.12, 4.13, 4.14, 4.15 และ 4.16

ตารางที่ 4.5 แสดงค่า RPM ของผู้ทดลอง ในเวลาต่างๆ

เวลา / อาการ	0.5	1	1.5	2	2.5	3	3.5	4	4.5
อดนอน	1821	1621	2288	2406	2524	2492	2346	2396	2105
เมา	2425	2899	2992	3079	2968	2776	1978	2401	2916

เวลา / อาการ	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
อดนอน	1760	1161	1320	2268	1756	1251	1919	1528	1631	1478
เมา	2456	3012	1799	2546	2416	2505	2600	1997	2762	3163

เวลา / อาการ	10	10.5	11	11.5	12	12.5	13	13.5	14
อดนอน	2180	1770	1948	1850	2072	1846	1838	2094	2214
เมา	3276	3112	3214	2630	-	-	-	-	-

จากตารางที่ 4.5 แสดงค่ารอบเครื่องยนต์ต่อนาที (RPM) จะเห็นว่าคนที่เมานั้นมีค่า RPM ที่สูงเนื่องจากมีการเร่งเครื่องบ่อย กว่า กรณีอื่นๆ ซึ่งมีค่าที่ต่ำกว่าแสดงได้ดังรูปที่ 4.11

รูปที่ 4.11 กราฟแสดงความสัมพันธ์ระหว่าง RPM กับ เวลา

ตารางที่ 4.6 แสดงค่าความเร็วของผู้ทดลอง ในเวลาต่างๆ

เวลา / อาการ	0.5	1	1.5	2	2.5	3	3.5	4	4.5
อดนอน	59	54	92	97	102	100	95	96	85
เมา	79	116	120	124	120	112	81	92	99

เวลา / อาการ	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
อดนอน	70	40	56	58	57	41	61	48	63	52
เมา	99	98	74	63	98	100	105	80	91	126

เวลา / อาการ	10	10.5	11	11.5	12	12.5	13	13.5	14
อดนอน	36	69	79	74	88	76	74	84	84
เมา	132	125	130	106	-	-	-	-	-

จากตารางที่ 4.6 แสดงค่าความเร็วของรถยนต์จะเห็นได้ว่าคนที่มีอาการเมานั้นมีการขับรถที่ความเร็วสูงกว่ากรณีอื่นๆมาก โดยอยู่ช่วงประมาณ 100 – 130 กิโลเมตร/ชั่วโมง ซึ่งต่างจากกรณีอื่นๆ ที่ความเร็วอยู่ช่วงประมาณ 60 – 90 กิโลเมตร/ชั่วโมง ดังในรูปที่ 4.12

รูปที่ 4.12 กราฟแสดงความสัมพันธ์ระหว่างความเร็ว กับ เวลา

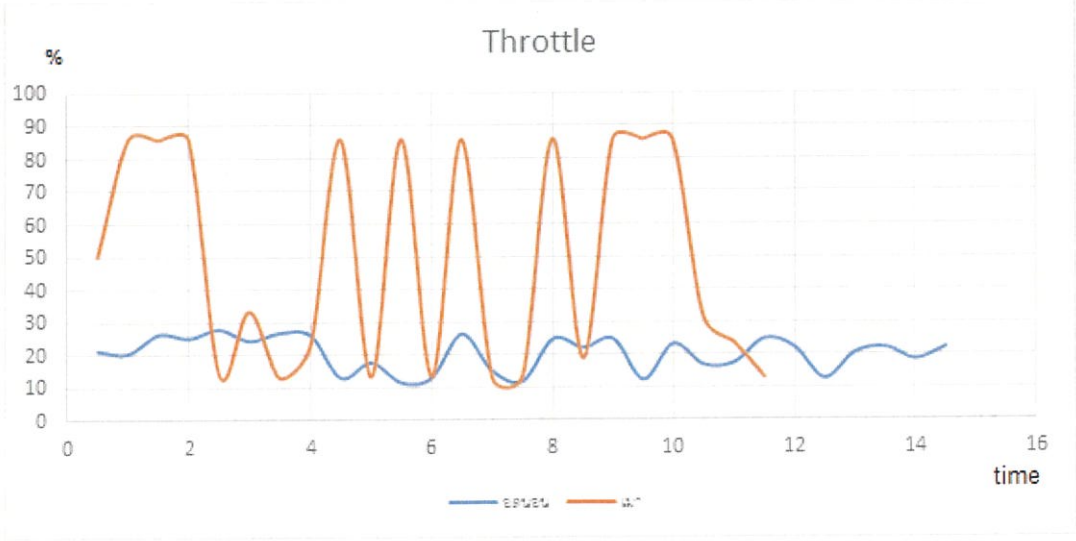
ตารางที่ 4.7 แสดงค่าตำแหน่งปีกผีเสื้อ (Throttle Position Sensor) ของผู้ทดลอง ในเวลาต่างๆ

เวลา / อาการ	0.5	1	1.5	2	2.5	3	3.5	4	4.5
อดนอน	21.2	20.4	26.3	25.1	27.8	24.3	26.7	25.9	12.9
เมา	50.2	85.9	85.9	85.9	14.1	33.3	12.9	23.1	85.9

เวลา / อาการ	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
อดนอน	17.6	11.4	12.9	26.3	14.9	11.8	24.7	22	24.7	12.2
เมา	13.3	85.9	13.3	85.9	12.9	13.3	85.9	18.8	85.9	85.9

เวลา / อาการ	10	10.5	11	11.5	12	12.5	13	13.5	14	14.5
อดนอน	23.1	16.7	17.3	24.7	22	12.5	20.4	22	18.4	22
เมา	85.9	31.4	23.5	12.9	-	-	-	-	-	-

จากตารางที่ 4.7 แสดงถึงค่าตำแหน่งปีกผีเสื้อ (Throttle Position Sensor) จะเห็นได้ว่าค่านั้นมีอยู่ 2 ช่วงใหญ่ๆ คือช่วงปกตินั้นคือช่วง 10 – 30 % และอีกช่วงที่เร่งความเร็วคือ 85.9 % ดังในรูปที่ 4.13

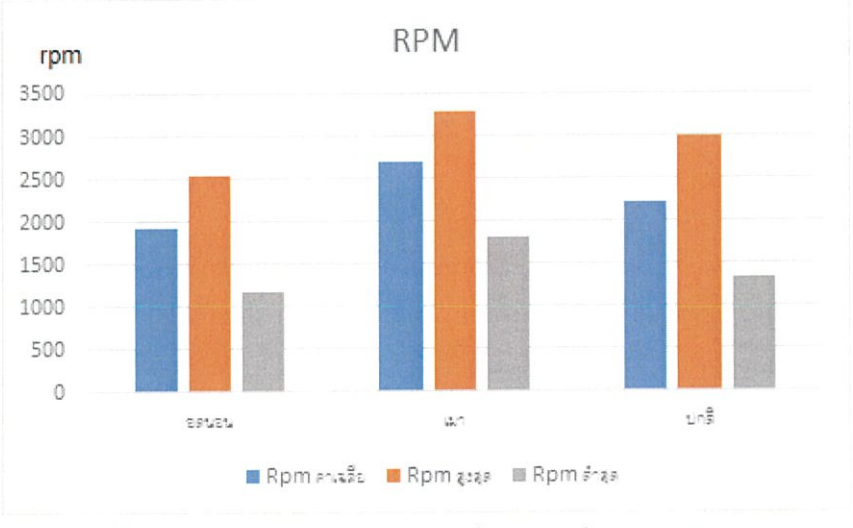


รูปที่ 4.13 กราฟแสดงความสัมพันธ์ระหว่าง TPS กับ เวลา

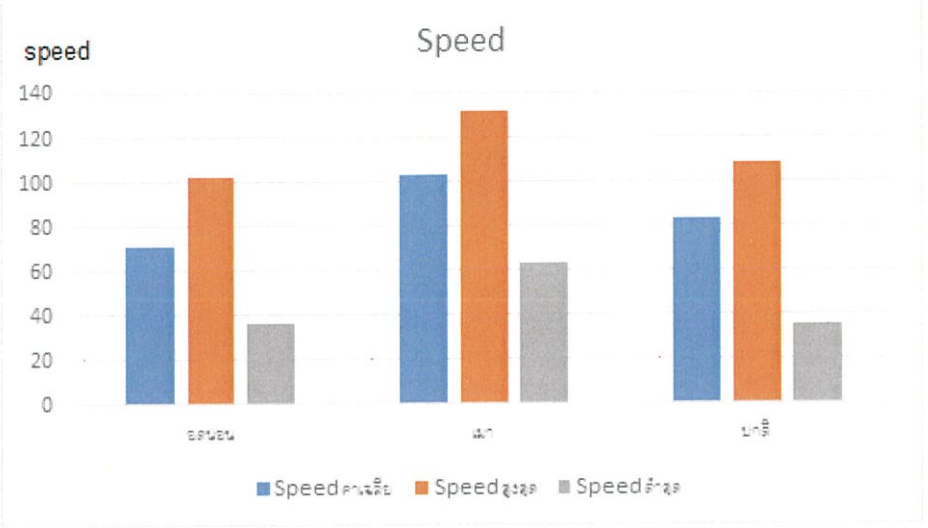
ตารางที่ 4.8 แสดงค่าเฉลี่ย สูงสุด ต่ำสุด ของการทดลอง

บุคคล	Rpm			Speed			Throttle Position		
	ค่าเฉลี่ย	สูงสุด	ต่ำสุด	ค่าเฉลี่ย	สูงสุด	ต่ำสุด	ค่าเฉลี่ย	สูงสุด	ต่ำสุด
ง่วงนอน	1924	2524	1161	71	102	36	20.2	27.8	11.4
เมา	2692	3276	1799	103	132	63	49.2	85.9	12.9
ปกติ	2223	2980	1323	83	108	35	26	83.3	12

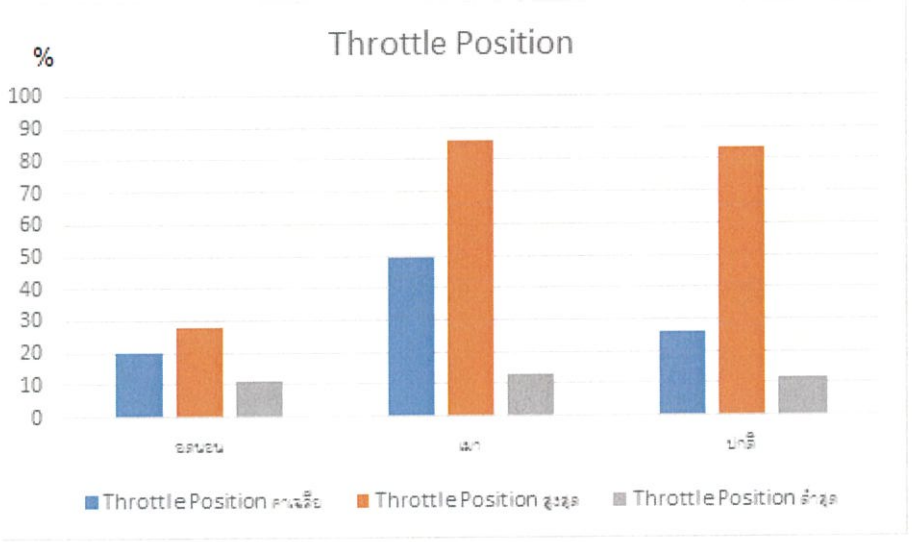
จากตารางที่ 4.8 แสดงค่าเฉลี่ย สูงสุด ต่ำสุด จะเห็นว่าค่า RPM เฉลี่ยของคนง่วงนอน จะต่ำกว่าคนปกติ ส่วนคนเมามีค่า RPM เฉลี่ยมากกว่าคนปกติ และ RPM สูงสุดเฉลี่ยของคนง่วงนอน มีค่าต่ำกว่าคนปกติ ส่วนคนเมามีค่า RPM เฉลี่ยมากกว่าคนปกติ ดังแสดงในรูปที่ 4.14 ค่าความเร็ว (Speed) จะเห็นว่าค่าความเร็ว (Speed) เฉลี่ย คนง่วงนอน จะต่ำกว่าคนปกติ ส่วนคนเมามีค่าความเร็ว (Speed) เฉลี่ยมากกว่าคนปกติ ดังแสดงในรูปที่ 4.15 ค่า Throttle Position จะเห็นว่า ค่า Throttle Position เฉลี่ยของคนง่วงนอน มีค่าต่ำกว่าคนปกติ ส่วนคนเมามีค่า Throttle Position เฉลี่ยมากกว่าคนปกติ ดังแสดงในรูปที่ 4.16



รูปที่ 4.14 แผนภูมิแสดงค่า RPM เฉลี่ย สูงสุด ต่ำสุด ของผู้ทดลอง



รูปที่ 4.15 แผนภูมิแสดงค่าความเร็วเฉลี่ย สูงสุด ต่ำสุด ของผู้ทดลอง

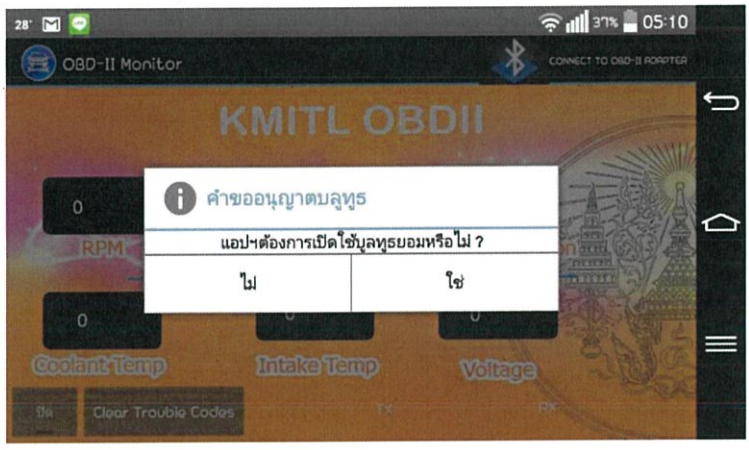


รูปที่ 4.16 แผนภูมิแสดงค่า TPS เฉลี่ย สูงสุด ต่ำสุด ของผู้ทดลอง

4.2 การทดลองแอพพลิเคชั่นแอนดรอยด์ (Android Application)

ขั้นตอนการทดลอง

1.แอพพลิเคชั่นแอนดรอยด์ (Android Application) ที่ได้ทำขึ้นมา ในแอพพลิเคชั่น (Application) ของเราสามารถวัดค่าได้ คือ RPM, Speed, Throttle position, Coolant Temp, Intake Temp, Voltage ดังในรูปที่ 4.17



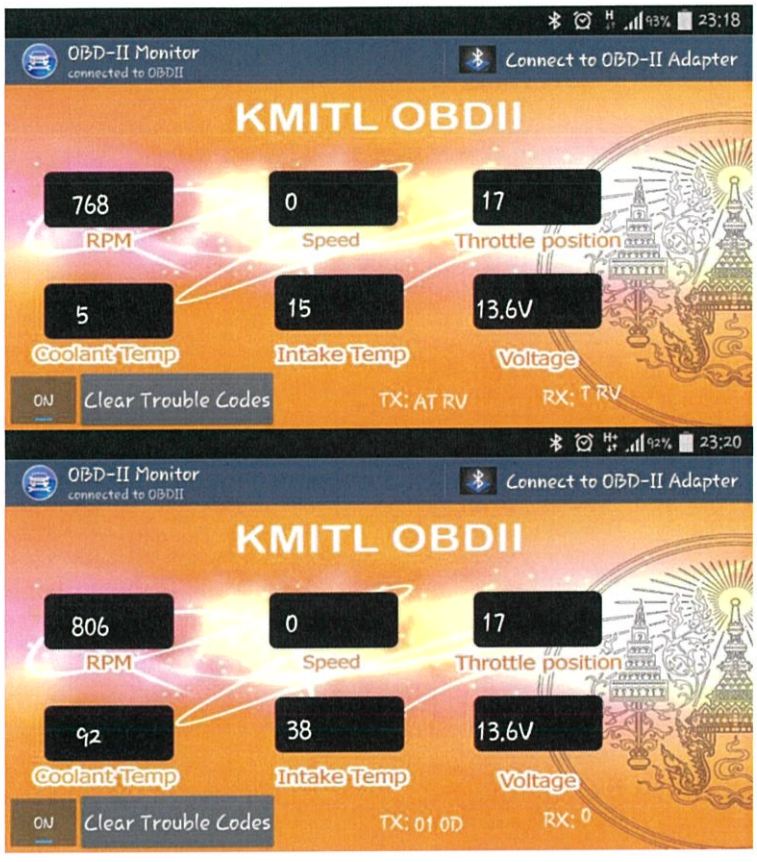
รูปที่ 4.17 แสดงหน้าตาแอพพลิเคชั่น (Application)

2. ทำการต่อกับโอบีดีทู (OBDII) โดยกดที่มุมขวาของแอพพลิเคชั่น (Application) Connect to OBD-II Adapter แล้วเลือกที่ OBDII เมื่อเชื่อมต่อเสร็จแล้วด้านมุมซ้ายจะขึ้นว่า OBD-II Monitor connected to OBDII ดังในรูปที่ 4.18

รูปที่ 4.18 แสดงการเชื่อมต่อแอพพลิเคชั่น (Application) กับ โอบีดีทู (OBDII)

3. ทดสอบวัดค่าต่างๆ

ในแอพพลิเคชั่น (Application) ของเราสามารถวัดค่าได้ คือ RPM, Speed, Throttle position, Coolant Temp, Intake Temp, Voltage



รูปที่ 4.19 แสดงการวัดค่าต่างๆจากแอพพลิเคชั่น (Application)

4. ทดลองการแจ้งเตือนโดยการขับรถที่ทดลอง ดังรูปที่ 4.20 เพื่อให้ค่าที่ ความเร็ว เกินตามกำหนด โดยตั้งกำหนดความเร็วไว้ที่ 50 กิโลเมตร ต่อชั่วโมง เมื่อขับรถเกินกว่ากำหนดแอพพลิเคชั่นก็จะมีเสียงเตือนขึ้นมา

รูปที่ 4.20 รูปแสดงการทดสอบขับรถ

บทที่ 5
วิเคราะห์และสรุปผลการทดลอง

5.1 วิเคราะห์ผลการทดลอง

จากการทดลองพฤติกรรมคนขับรถ จะเห็นได้ว่า จะได้ผลการทดลองในรูปแบบต่างๆทั้งที่เป็นไปตามต้องการและมีทั้งที่เกิดปัญหาขึ้นมาเราจึงนำมาวิเคราะห์หาสาเหตุต่างๆได้ดังนี้

1. คนที่มีพฤติกรรมต่างกันการขับรถที่ไม่เหมือนกัน โดยผลที่ได้นั้นแบ่งเป็น 3 ประเภท คือ คนปกติ คนง่วงนอน และ คนเมา ผลปรากฏว่า ค่าความเร็วเฉลี่ยของคนปกติ คือ 83 กิโลเมตร/ชั่วโมง ความเร็วเฉลี่ยของคนง่วงนอน คือ 70 กิโลเมตร/ชั่วโมง ความเร็วเฉลี่ยของคนเมา 103 กิโลเมตร/ชั่วโมง ค่ารอบเครื่องยนต์ต่อนาที (RPM) เฉลี่ยของคนปกติ คือ 2223 รอบ/นาที ค่ารอบเครื่องยนต์ต่อนาที (RPM) เฉลี่ยของง่วงนอน คือ 1924 รอบ/นาที ค่ารอบเครื่องยนต์ต่อนาที (RPM) เฉลี่ยของเมา คือ 2692 รอบ/นาที และค่าตำแหน่งปีกผีเสื้อ (Throttle Position) เฉลี่ยของคนปกติ คือ 26 % ค่าตำแหน่งปีกผีเสื้อ (Throttle Position) เฉลี่ยของคนง่วงนอน คือ 20.2 % ค่าตำแหน่งปีกผีเสื้อ (Throttle Position) เฉลี่ยของคนเมา คือ 49.2 % จะเห็นว่าความเร็วเฉลี่ยของคนง่วงนอนจะนอนกว่าคนปกติ ส่วนคนเมานั้นจะมากกว่าปกติ เช่นเดียวกับรอบเครื่องยนต์ต่อนาที (RPM) และค่าตำแหน่งปีกผีเสื้อ (Throttle Position Sensor)

2. เนื่องจากการศึกษาจำเป็นต้องใช้รถเป็นตัวแปลในการศึกษาทำให้มีความลำบากในการศึกษา อาทิเช่นรถยนต์มีขนาดใหญ่ไม่สามารถนำไปทดลองในห้องทดลองได้ การทดลองพฤติกรรม จำเป็นต้องใช้ถนนจริงในการทดลองซึ่งก็จะมีตัวแปลที่ควบคุมไม่ได้ต่างๆเขามาด้วย อาทิ จำนวนรถซึ่งอาจทำให้รถติดทำให้ไม่ได้พฤติกรรมที่แท้จริงได้

3. จากแอพพลิเคชั่น (Application) ที่พัฒนาค่าที่ได้มานั้นไม่ใช่ค่าที่ปัจจุบัน มีความล่าช้าจึงทำให้ ผลที่ได้มีความคลาดเคลื่อน

4. แอพพลิเคชั่น (Application) เวลารับค่าจากโอบีดีทู (OBDII) มาช่วงเวลาหนึ่งจะเกิดการค้างจึงทำให้ต้องรีเซต (Reset) แอพพลิเคชั่น (Application) ใหม่เนื่องจากโค้ดที่ใช้นั้นยังไม่เสถียร

5.2 สรุปผลการทดลอง

จากโครงงานจะเห็นได้ว่าเราได้ทำ การออกแบบแอพพลิเคชั่น (Application) ในการแจ้งเตือนผู้ขับขี่นั้น จะเห็นได้ว่าการสร้างเงื่อนไขการทำงานของแอพพลิเคชั่น (Application) นั้นจากข้อมูลที่ได้มาทำให้เรารู้และออกแบบแอพพลิเคชั่น (Application) โดยตั้งเตือนที่ความเร็ว 110 กิโลเมตร/ชั่วโมง ซึ่งมีความเร็วที่สูงและเกินกว่าที่กฎหมายกำหนด และเตือนเมื่อค่าตำแหน่งปีกผีเสื้อ (Throttle Position) ขึ้นสูงเกิน 70 % นั้นหมายความว่ากำลังเร่งด้วยความเร็วอาจทำให้เกิดอันตรายได้และจากโครงงานนี้เราได้ใช้ค่าพารามิเตอร์ทั้งหมด 6 ค่าในการสร้าง แอพพลิเคชั่น (Application)นี้

และแอพพลิเคชั่นยังสามารถบอกค่าได้อีก 3 พารามิเตอร์ คือ Intake Temperature, Coolant Temperature และ Voltage จากการทดลองที่ผ่านมานั้น ปรากฏว่า

1. การอ่านค่าของแอพพลิเคชั่น ยังไม่ใช่ค่าที่ปัจจุบัน มีความล่าช้า และค่ามีการกระโดด จึงทำให้การเตือนมีความล่าช้า

2. แอพพลิเคชั่น (Application) ไม่เสถียรมีอาการค้างบ่อย ทำให้ต้องมาคอยรีเซต แอพพลิเคชั่น (Application) บ่อยๆ

5.3 แนวทางการแก้ไข

เนื่องจากการทดลองจำเป็นต้องศึกษาการทำงานจากการขับรถจริงซึ่งถ้าจะทดลองได้ที่ได้ผลแท้จริงอาจต้องไปทำการทดลองในสนามที่มีความพร้อม และมีเครื่องมือในการศึกษาที่มากพอ

บรรณานุกรม

[1] ดร.จักรชัย โสอินทร์, พงษ์ศธร จันทร์ยอย, ณัฐณิชา วีระมงคลเลิศ, คู่มือพัฒนาแอพพลิเคชัน Android อย่างมืออาชีพ, พิมพ์ครั้งที่ 1 , นนทบุรี: ไอดีซีฯ, 2555

ภาคผนวก ก

โปรแกรมภาษา จาวา ที่ใช้ในโครงงาน

```
package com.example.android.OBDIIMonitor;

import java.io.File;

import java.io.FileOutputStream;

import java.io.OutputStreamWriter;

import java.util.Date;

import com.example.android.OBDIIMonitor.R;

import android.media.MediaPlayer;

import android.net.Uri;

import android.annotation.SuppressLint;

import android.app.ActionBar;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.content.Intent;

import android.os.Bundle;

import android.os.Environment;

import android.os.Handler;

import android.os.Message;

import android.text.format.DateFormat;

import android.util.Log;

import android.view.KeyEvent;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.inputmethod.EditorInfo;

import android.webkit.MimeTypeMap;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ListView;
```

```
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.view.View.OnClickListener;
import android.view.animation.LinearInterpolator;
import android.view.animation.RotateAnimation;

@SuppressLint("HandlerLeak")
public class BluetoothChat<ImageView> extends Activity {
        MediaPlayer mpAudio;

  // Debugging
  private static final String TAG = "BluetoothChat";
  private static final boolean D = true;

  // Message types sent from the BluetoothChatService Handler
  public static final int MESSAGE_STATE_CHANGE = 1;
  public static final int MESSAGE_READ = 2;
  public static final int MESSAGE_WRITE = 3;
  public static final int MESSAGE_DEVICE_NAME = 4;
  public static final int MESSAGE_TOAST = 5;
  // Key names received from the BluetoothChatService Handler
  public static final String DEVICE_NAME = "device_name";
  public static final String TOAST = "toast";

  // Intent request codes
  private static final int REQUEST_CONNECT_DEVICE_SECURE = 1;
  private static final int REQUEST_CONNECT_DEVICE_INSECURE = 2;
```

```
private static final int REQUEST_ENABLE_BT = 3;

// Layout Views
private ListView mConversationView;
private EditText mOutEditText;
private Button mSendButton;

// Name of the connected device
private String mConnectedDeviceName = null;
// Array adapter for the conversation thread
private ArrayAdapter<String> mConversationArrayAdapter;
// String buffer for outgoing messages
private StringBuffer mOutStringBuffer;
// Local Bluetooth adapter
private BluetoothAdapter mBluetoothAdapter = null;
// Member object for the chat services
private BluetoothChatService mChatService = null;


@Override
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    if(D) Log.e(TAG, "+++ ON CREATE +++");

    // Set up the window layout
    setContentView(R.layout.main);
```

```
        // Get local Bluetooth adapter
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        // If the adapter is null, then Bluetooth is not supported
        if (mBluetoothAdapter == null) {
            Toast.makeText(this, "Bluetooth is not available", Toast.LENGTH_LONG).show();
            finish();
            return;
        }
        mpAudio = MediaPlayer.create(this, R.raw.alarm);
        mpAudio.setLooping(true);
    }

    @Override
    public void onStart() {
        super.onStart();
        if(D) Log.e(TAG, "++ ON START ++");

        // If BT is not on, request that it be enabled.
        // setupChat() will then be called during onActivityResult
        if (!mBluetoothAdapter.isEnabled()) {
            Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
        // Otherwise, setup the chat session
        }

else {
            if (mChatService == null) setupChat();
        }
```

```
}

@Override
public synchronized void onResume() {
  super.onResume();
  if(D) Log.e(TAG, "+ ON RESUME +");

  // Performing this check in onResume() covers the case in which BT was
  // not enabled during onStart(), so we were paused to enable it...
  // onResume() will be called when ACTION_REQUEST_ENABLE activity returns.
  if (mChatService != null) {
    // Only if the state is STATE_NONE, do we know that we haven't started already
    if (mChatService.getState() == BluetoothChatService.STATE_NONE) {
      // Start the Bluetooth chat services
      mChatService.start();
    }
  }
}

//keep track of current PID number
int message_number = 1;

private void setupChat() {
  Log.d(TAG, "setupChat()");

  // Initialize the array adapter for the conversation thread
  mConversationArrayAdapter = new ArrayAdapter<String>(this, R.layout.message);
```

```
mConversationView = (ListView) findViewById(R.id.in);
mConversationView.setAdapter(mConversationArrayAdapter);

// Initialize the BluetoothChatService to perform bluetooth connections
mChatService = new BluetoothChatService(this, mHandler);

// Initialize the buffer for outgoing messages
mOutStringBuffer = new StringBuffer("");

//---Get Vehicle Data Button---
final ToggleButton getDataButton = (ToggleButton) findViewById(R.id.toggleButton1);
getDataButton.setOnClickListener(new View.OnClickListener()
{

   public void onClick(View v) {

     if(getDataButton.isChecked()) {
               startTransmission();
     }
     else {
               message_number = 0;
     }

   }
});


//---Clear Trouble Codes Button---
Button getCodesButton = (Button) findViewById(R.id.button2);
```

```
  getCodesButton.setOnClickListener(new View.OnClickListener()
  {

    public void onClick(View v) {

      clearCodes();
    }
  });

}

public void startTransmission() {

      sendMessage("01 00" + '\r');

}
      public void getData(int messagenumber) {

            final TextView TX = (TextView) findViewById(R.id.TXView2);
            switch(messagenumber) {
      case 1:
            sendMessage("01 0C" + '\r'); //get RPM
            TX.setText("01 0C");
            messagenumber++;

            break;
      case 2:
            sendMessage("01 0D" + '\r'); //get MPH
            TX.setText("01 0D");
```

```
                messagenumber++;
                break;
        case 3:
                sendMessage("01 11" + '\r'); //Throttle position
                TX.setText("01 11");
                messagenumber++;
                break;
        case 4:
                sendMessage("01 05" + '\r'); //get Coolant Temperature
                TX.setText("01 05");
                messagenumber++;
                break;
        case 5:
                sendMessage("01 0F" + '\r'); //get Intake Temperature
                TX.setText("01 0F");
                messagenumber++;
                break;
        case 6:
                sendMessage("AT RV" + '\r'); //get Voltage
                TX.setText("AT RV");
                messagenumber++;
                break;

        default: ;
                }
}
public void clearCodes() {
        final TextView TX = (TextView) findViewById(R.id.TXView2);
```

```
    if(mConnectedDeviceName != null) {
        sendMessage("04" + '\r'); //send Clear Trouble Codes Command
        TX.setText("Clear Codes");
        Toast.makeText(getApplicationContext(), "OBD Trouble Codes Cleared",
Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText(getApplicationContext(), "OBD Adapter NOT CONNECTED",
Toast.LENGTH_SHORT).show();
    }
  }
  @Override
  public synchronized void onPause() {
    super.onPause();
    if(D) Log.e(TAG, "- ON PAUSE -");
  }
  @Override
  public void onStop() {
    super.onStop();
    if(D) Log.e(TAG, "-- ON STOP --");
  }
  @Override
  public void onDestroy() {
    super.onDestroy();
    // Stop the Bluetooth chat services
    if (mChatService != null) mChatService.stop();
    if(D) Log.e(TAG, "--- ON DESTROY ---");
  }
  /*private void ensureDiscoverable() {
    if(D) Log.d(TAG, "ensure discoverable");
    if (mBluetoothAdapter.getScanMode() !=
```

```
        BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE) {
        Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
        discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
        startActivity(discoverableIntent);
    }
}*/
/**
 * Sends a message.
 * @param message  A string of text to send.
 */
private void sendMessage(String message) {
    // Check that we're actually connected before trying anything
    if (mChatService.getState() != BluetoothChatService.STATE_CONNECTED) {
        Toast.makeText(this, R.string.not_connected, Toast.LENGTH_SHORT).show();
        return;
    }
    // Check that there's actually something to send
    if (message.length() > 0) {
        // Get the message bytes and tell the BluetoothChatService to write
        byte[] send = message.getBytes();
        mChatService.write(send);
        // Reset out string buffer to zero and clear the edit text field
        mOutStringBuffer.setLength(0);
        //mOutEditText.setText(mOutStringBuffer);
    }
}
// The action listener for the EditText widget, to listen for the return key
```

```
private TextView.OnEditorActionListener mWriteListener =
    new TextView.OnEditorActionListener() {
    public boolean onEditorAction(TextView view, int actionId, KeyEvent event) {
        // If the action is a key-up event on the return key, send the message
        if (actionId == EditorInfo.IME_NULL && event.getAction() == KeyEvent.ACTION_UP) {
            String message = view.getText().toString();
            sendMessage(message);
        }
        if(D) Log.i(TAG, "END onEditorAction");
        return true;
    }
};
private final void setStatus(int resId) {
    final ActionBar actionBar = getActionBar();
    actionBar.setSubtitle(resId);
}
private final void setStatus(CharSequence subTitle) {
    final ActionBar actionBar = getActionBar();
    actionBar.setSubtitle(subTitle);
}
// The Handler that gets information back from the BluetoothChatService
private final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {



        //<--------- Initialize Data Display Fields ---------->//
        final TextView RPM = (TextView) findViewById(R.id.RPMView);
        final TextView MPH = (TextView) findViewById(R.id.MPHView);
        final TextView engineLoad = (TextView) findViewById(R.id.LoadView);
```

```
  final TextView coolantTemperature = (TextView) findViewById(R.id.CoolantView);
  final TextView intakeTemperature = (TextView) findViewById(R.id.IntakeView);
  final TextView voltage = (TextView) findViewById(R.id.voltView);
  final TextView RX = (TextView) findViewById(R.id.RXView2);
  //<-------- Initialize Needle Animations --------->//
  String dataRecieved;
  int value = 0;
  int value2 = 0;
  int PID = 0;
switch (msg.what) {
case MESSAGE_STATE_CHANGE:
   if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
   switch (msg.arg1) {
   case BluetoothChatService.STATE_CONNECTED:
      setStatus(getString(R.string.title_connected_to, mConnectedDeviceName));
      mConversationArrayAdapter.clear();
      break;
   case BluetoothChatService.STATE_CONNECTING:
      setStatus(R.string.title_connecting);
      break;
   case BluetoothChatService.STATE_LISTEN:
   case BluetoothChatService.STATE_NONE:
      setStatus(R.string.title_not_connected);
      break;
   }
   break;
case MESSAGE_WRITE:
   byte[] writeBuf = (byte[]) msg.obj;
   // construct a string from the buffer
```

```
            String writeMessage = new String(writeBuf);

            //mConversationArrayAdapter.add("Me:  " + writeMessage);
            break;

        case MESSAGE_READ:
            byte[] readBuf = (byte[]) msg.obj;
            // construct a string from the valid bytes in the buffer
            String readMessage = new String(readBuf, 0, msg.arg1);

            // ------- ADDED CODE FOR OBD -------- //
            dataRecieved = readMessage;
            RX.setText(dataRecieved);

            if((dataRecieved != null) && (dataRecieved.matches("\\s*[0-9A-Fa-f]{2} [0-9A-Fa-
f]{2}\\s*\r?\n?" ))) {
                        dataRecieved = dataRecieved.trim();
                        String[] bytes = dataRecieved.split(" ");
                        if((bytes[0] != null)&&(bytes[1] != null)) {
                                PID = Integer.parseInt(bytes[0].trim(), 16);
                                value = Integer.parseInt(bytes[1].trim(), 16);
                                }


switch(PID) {
case 15://PID(0F): Intake Temperature
value = value - 40; //Formula for Intake Temperature
String displayIntakeTemp = String.valueOf(value);
```

```
intakeTemperature.setText(displayIntakeTemp);
break;
case 17://PID(11): Throttle position
value = (value * 100 ) / 255;
String displayThrottle = String.valueOf(value);
engineLoad.setText(displayThrottle);
double b= Double.parseDouble(displayThrottle);
if (b >= 70) {
mpAudio.start();
}
else{
 mpAudio.pause();
}
break;
case 5://PID(05): Coolant Temperature
value = value - 40;
String displayCoolantTemp = String.valueOf(value);
coolantTemperature.setText(displayCoolantTemp);
break;
case 12: //PID(0C): RPM
int RPM_value = (value*256)/4;
String displayRPM = String.valueOf(RPM_value);
RPM.setText(displayRPM);
break;
case 13://PID(0D): MPH
value = value;
String displayMPH = String.valueOf(value);
MPH.setText(displayMPH);
double a= Double.parseDouble(displayMPH);
```

```
if (a >= 110) {
mpAudio.start();
}
else{
 mpAudio.pause();
}
break;
        default: ;
}
}
else if((dataRecieved != null) && (dataRecieved.matches("\\s*[0-9A-Fa-f]{1,2} [0-9A-Fa-f]{2} [0-9A-Fa-f]{2}\\s*\r?\n?" ))) {
dataRecieved = dataRecieved.trim();
String[] bytes = dataRecieved.split(" ");
if((bytes[0] != null)&&(bytes[1] != null)&&(bytes[2] != null)) {
PID = Integer.parseInt(bytes[0].trim(), 16);
value = Integer.parseInt(bytes[1].trim(), 16);
value2 = Integer.parseInt(bytes[2].trim(), 16);
}
//PID(0C): RPM
if(PID == 12) {
int RPM_value = ((value*256)+value2)/4;
String displayRPM = String.valueOf(RPM_value);
RPM.setText(displayRPM);
}
}
        else if((dataRecieved != null) && (dataRecieved.matches("\\s*[0-9]+(\\.[0-9]?)?V\\s*\r*\n*" )))
{
```

```
          dataRecieved = dataRecieved.trim();
          String volt_number = dataRecieved.substring(0, dataRecieved.length()-1);
          double needle_value = Double.parseDouble(volt_number);
          voltage.setText(dataRecieved);

        }
        else if((dataRecieved != null) && (dataRecieved.matches("\\s*[0-9]+(\\.[0-
9]?)?V\\s*V\\s*>\\s*\r*\n*" ))) {

          dataRecieved = dataRecieved.trim();
          String volt_number = dataRecieved.substring(0, dataRecieved.length()-1);
          double needle_value = Double.parseDouble(volt_number);
          needle_value = (((needle_value - 11)*21) /0.5) - 100;
          int volt_value = (int)(needle_value);
          voltage.setText(dataRecieved);

        }
        else if((dataRecieved != null) && (dataRecieved.matches("\\s*[ .A-Za-z0-
9\\?*>\r\n]*\\s*>\\s*\r*\n*" ))) {

          if(message_number == 7) message_number = 1;
          getData(message_number++);

        }

        else {

                ;
          }
```

```
        //mConversationArrayAdapter.add(mConnectedDeviceName+":  " + readMessage);
        break;
      ////////////////////////////////////////
      case MESSAGE_DEVICE_NAME:
        // save the connected device's name
        mConnectedDeviceName = msg.getData().getString(DEVICE_NAME);
        Toast.makeText(getApplicationContext(), "Connected to "
                       + mConnectedDeviceName, Toast.LENGTH_SHORT).show();
        break;
      case MESSAGE_TOAST:
        Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
                       Toast.LENGTH_SHORT).show();
        break;
      }
    }


};

public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(D) Log.d(TAG, "onActivityResult " + resultCode);
    switch (requestCode) {
    case REQUEST_CONNECT_DEVICE_SECURE:
        // When DeviceListActivity returns with a device to connect
```

```
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data, true);
        }
        break;
    case REQUEST_CONNECT_DEVICE_INSECURE:
        // When DeviceListActivity returns with a device to connect
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data, false);
        }
        break;
    case REQUEST_ENABLE_BT:
        // When the request to enable Bluetooth returns
        if (resultCode == Activity.RESULT_OK) {
            // Bluetooth is now enabled, so set up a chat session
            setupChat();
        } else {
            // User did not enable Bluetooth or an error occurred
            Log.d(TAG, "BT not enabled");
            Toast.makeText(this, R.string.bt_not_enabled_leaving, Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}

private void connectDevice(Intent data, boolean secure) {
    // Get the device MAC address
    String address = data.getExtras()
        .getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
    // Get the BluetoothDevice object
```

```
    BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
    // Attempt to connect to the device
    mChatService.connect(device, secure);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.option_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Intent serverIntent = null;
    switch (item.getItemId()) {
    case R.id.secure_connect_scan:
        // Launch the DeviceListActivity to see devices and do scan
        serverIntent = new Intent(this, DeviceListActivity.class);
        startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE_SECURE);
        return true;
    /* case R.id.insecure_connect_scan:
        // Launch the DeviceListActivity to see devices and do scan
        serverIntent = new Intent(this, DeviceListActivity.class);
        startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE_INSECURE);
        return true;
    case R.id.discoverable:
        // Ensure this device is discoverable by others
        ensureDiscoverable();
```

```
        return true;*/
      }
      return false;
    }
}
```

ภาคผนวก ข

ตาราง mode OBDII PIDs

Mode 01 Show current data

PID (hex)	Data bytes returned	Description	Min value	Max value	Units	Formula
00	4	PIDs supported [01 - 20]				Bit encoded [A7..D0] == [PID $01..PID $20] See below
01	4	Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)				Bit encoded. See below
02	2	Freeze DTC				
03	2	Fuel system status				Bit encoded. See below
04	1	Calculated engine load value	0	100	%	A*100/255
05	1	Engine coolant temperature	-40	215	°C	A-40
06	1	Short term fuel % trim—Bank 1	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
07	1	Long term fuel % trim—Bank 1	-100 Subtracting Fuel	99.22 Adding Fuel	%	(A-128) * 100/128

			(Rich Condition)	(Lean Condition)		
08	1	Short term fuel % trim—Bank 2	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
09	1	Long term fuel % trim—Bank 2	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
0A	1	Fuel pressure	0	765	kPa (gauge)	A*3
0B	1	Intake manifold absolute pressure	0	255	kPa (absolute)	A
0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
0D	1	Vehicle speed	0	255	km/h	A
0E	1	Timing advance	-64	63.5	° relative to #1 cylinder	(A-128)/2
0F	1	Intake air temperature	-40	215	°C	A-40
10	2	MAF air flow rate	0	655.35	grams/sec	((A*256)+B) / 100
11	1	Throttle position	0	100	%	A*100/255
12	1	Commanded secondary air status				Bit encoded. See below

13	1	Oxygen sensors present				[A0..A3] == Bank 1, Sensors 1-4. [A4..A7] == Bank 2...
14	2	Bank 1, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
15	2	Bank 1, Sensor 2: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
16	2	Bank 1, Sensor 3: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
17	2	Bank 1, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
18	2	Bank 2, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
19	2	Bank 2, Sensor 2:	0	1.275	Volts %	A/200 (B-128) *

		Oxygen sensor voltage, Short term fuel trim	-100(lean)	99.2(rich)		100/128 (if B==$FF, sensor is not used in trim calc)
1A	2	Bank 2, Sensor 3: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
1B	2	Bank 2, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
1C	1	OBD standards this vehicle conforms to				Bit encoded. See below
1D	1	Oxygen sensors present				Similar to PID 13, but [A0..A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]
1E	1	Auxiliary input status				A0 == Power Take Off (PTO) status (1 == active) [A1..A7] not used
1F	2	Run time since engine start	0	65,535	seconds	(A*256)+B
20	4	PIDs supported [21 - 40]				Bit encoded [A7..D0] == [PID $21..PID

						$40] See below
21	2	Distance traveled with malfunction indicator lamp (MIL) on	0	65,535	km	(A*256)+B
22	2	Fuel Rail Pressure (relative to manifold vacuum)	0	5177.265	kPa	((A*256)+B) * 0.079
23	2	Fuel Rail Pressure (diesel, or gasoline direct inject)	0	655,350	kPa (gauge)	((A*256)+B) * 10
24	4	O2S1_WR_lambda(1): Equivalence Ratio Voltage	0 0	1.999 7.999	N/A V	((A*256)+B)*2/65535 or ((A*256)+B)/32768 ((C*256)+D)*8/65535 or ((C*256)+D)/8192
25	4	O2S2_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
26	4	O2S3_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
27	4	O2S4_WR_lambda(1):	0 0	2 8	N/A V	((A*256)+B)*2/65535

		Equivalence Ratio Voltage				((C*256)+D)*8/65535
28	4	O2S5_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
29	4	O2S6_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
2A	4	O2S7_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
2B	4	O2S8_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
2C	1	Commanded EGR	0	100	%	A*100/255
2D	1	EGR Error	-100	99.22	%	(A-128) * 100/128
2E	1	Commanded evaporative purge	0	100	%	A*100/255
2F	1	Fuel Level Input	0	100	%	A*100/255
30	1	# of warm-ups since codes cleared	0	255	N/A	A

31	2	Distance traveled since codes cleared	0	65,535	km	(A*256)+B
32	2	Evap. System Vapor Pressure	-8,192	8,192	Pa	((A*256)+B)/4 (A and B are two's complement sig ned)
33	1	Barometric pressure	0	255	kPa (Absolute)	A
34	4	O2S1_WR_lam bda(1): Equivalence Ratio Current	0 -128	1.999 127.99	N/A mA	((A*256)+B)/32,7 68 ((C*256)+D)/256 - 128
35	4	O2S2_WR_lam bda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,7 68 ((C*256)+D)/256 - 128
36	4	O2S3_WR_lam bda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/327 68 ((C*256)+D)/256 - 128
37	4	O2S4_WR_lam bda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,7 68 ((C*256)+D)/256 - 128
38	4	O2S5_WR_lam bda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,7 68 ((C*256)+D)/256 - 128
39	4	O2S6_WR_lam	0	2	N/A	((A*256)+B)/32,7

		bda(1): Equivalence Ratio Current	-128	128	mA	68 ((C*256)+D)/256 - 128
3A	4	O2S7_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
3B	4	O2S8_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
3C	2	Catalyst Temperature Bank 1, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 - 40
3D	2	Catalyst Temperature Bank 2, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 - 40
3E	2	Catalyst Temperature Bank 1, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 - 40
3F	2	Catalyst Temperature Bank 2, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 - 40
40	4	PIDs supported [41 - 60]				Bit encoded [A7..D0] == [PID $41..PID $60] See below
41	4	Monitor status				Bit

		this drive cycle				encoded. See below
42	2	Control module voltage	0	65.535	V	((A*256)+B)/1000
43	2	Absolute load value	0	25,700	%	((A*256)+B)*100/255
44	2	Fuel/Air commanded equivalence ratio	0	2	N/A	((A*256)+B)/32768
45	1	Relative throttle position	0	100	%	A*100/255
46	1	Ambient air temperature	-40	215	°C	A-40
47	1	Absolute throttle position B	0	100	%	A*100/255
48	1	Absolute throttle position C	0	100	%	A*100/255
49	1	Accelerator pedal position D	0	100	%	A*100/255
4A	1	Accelerator pedal position E	0	100	%	A*100/255
4B	1	Accelerator pedal position F	0	100	%	A*100/255
4C	1	Commanded throttle actuator	0	100	%	A*100/255

4D	2	Time run with MIL on	0	65,535	minutes	(A*256)+B
4E	2	Time since trouble codes cleared	0	65,535	minutes	(A*256)+B
4F	4	Maximum value for equivalence ratio, oxygen sensor voltage, oxygen sensor current, and intake manifold absolute pressure	0, 0, 0, 0	255, 255, 255, 2550	, V, mA, kPa	A, B, C, D*10
50	4	Maximum value for air flow rate from mass air flow sensor	0	2550	g/s	A*10, B, C, and D are reserved for future use
51	1	Fuel Type				From fuel type table see below
52	1	Ethanol fuel %	0	100	%	A*100/255
53	2	Absolute Evap system Vapor Pressure	0	327.675	kPa	((A*256)+B)/200
54	2	Evap system vapor pressure	-32,767	32,768	Pa	((A*256)+B)-32767
55	2	Short term secondary oxygen sensor trim bank 1 and bank 3	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
56	2	Long term secondary	-100	99.22	%	(A-128)*100/128 (B-128)*100/128

		oxygen sensor trim bank 1 and bank 3				
57	2	Short term secondary oxygen sensor trim bank 2 and bank 4	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
58	2	Long term secondary oxygen sensor trim bank 2 and bank 4	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
59	2	Fuel rail pressure (absolute)	0	655,350	kPa	((A*256)+B) * 10
5A	1	Relative accelerator pedal position	0	100	%	A*100/255
5B	1	Hybrid battery pack remaining life	0	100	%	A*100/255
5C	1	Engine oil temperature	-40	210	°C	A - 40
5D	2	Fuel injection timing	-210.00	301.992	°	(((A*256)+B)-26,880)/128
5E	2	Engine fuel rate	0	3212.75	L/h	((A*256)+B)*0.05
5F	1	Emission requirements to which vehicle is designed				Bit Encoded
60	4	PIDs supported [61 - 80]				Bit encoded [A7..D0] == [PID

						$61..PID $80] See below
61	1	Driver's demand engine - percent torque	-125	125	%	A-125
62	1	Actual engine - percent torque	-125	125	%	A-125
63	2	Engine reference torque	0	65,535	Nm	A*256+B
64	5	Engine percent torque data	-125	125	%	A-125 Idle B-125 Engine point 1 C-125 Engine point 2 D-125 Engine point 3 E-125 Engine point 4
65	2	Auxiliary input / output supported				Bit Encoded
66	5	Mass air flow sensor				
67	3	Engine coolant temperature				
68	7	Intake air temperature sensor				
69	7	Commanded EGR and EGR Error				
6A	5	Commanded Diesel intake				

		air flow control and relative intake air flow position				
6B	5	Exhaust gas recirculation temperature				
6C	5	Commanded throttle actuator control and relative throttle position				
6D	6	Fuel pressure control system				
6E	5	Injection pressure control system				
6F	3	Turbocharger compressor inlet pressure				
70	9	Boost pressure control				
71	5	Variable Geometry turbo (VGT) control				
72	5	Wastegate control				
73	5	Exhaust pressure				
74	5	Turbocharger RPM				

75	7	Turbocharger temperature				
76	7	Turbocharger temperature				
77	5	Charge air cooler temperature (CACT)				
78	9	Exhaust Gas temperature (EGT) Bank 1				Special PID. See below
79	9	Exhaust Gas temperature (EGT) Bank 2				Special PID. See below
7A	7	Diesel particulate filter (DPF)				
7B	7	Diesel particulate filter (DPF)				
7C	9	Diesel Particulate filter (DPF) temperature				
7D	1	NOx NTE control area status				
7E	1	PM NTE control area status				
7F	13	Engine run time				
80	4	PIDs supported [81 - A0]				Bit encoded [A7..D0] == [PID $81..PID

						$A0] See below
81	21	Engine run time for Auxiliary Emissions Control Device(AECD)				
82	21	Engine run time for Auxiliary Emissions Control Device(AECD)				
83	5	NOx sensor				
84		Manifold surface temperature				
85		NOx reagent system				
86		Particulate matter (PM) sensor				
87		Intake manifold absolute pressure				
A0	4	PIDs supported [A1 - C0]				Bit encoded [A7..D0] == [PID $A1..PID $C0] See below
C0	4	PIDs supported [C1 - E0]				Bit encoded [A7..D0] == [PID $C1..PID $E0] See below
C3	?	?	?	?	?	Returns numerous data,

						including Drive Condition ID and Engine Speed*
C4	?	?	?	?	?	B5 is Engine Idle Request B6 is Engine Stop Request*