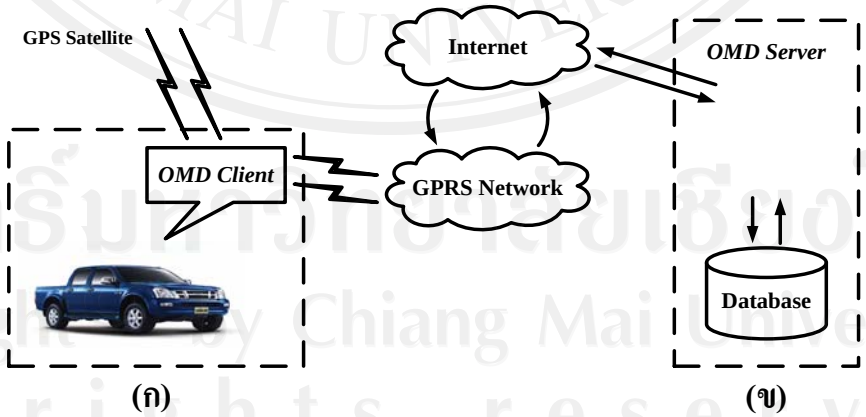
บทที่ 3
การออกแบบระบบ OMD

ระบบ OMD (Online Monitoring and Diagnosis system) คือระบบมอนิเตอร์และวินิจฉัยการทำงานของรถยนต์แบบออนไลน์ โดยเป็นการพัฒนาระบบฝังตัวให้สามารถวินิจฉัยการทำงานของรถยนต์และระบุตำแหน่งรถยนต์ด้วยเทคโนโลยีจีพีเอส จากนั้นรายงานข้อมูลผลการวินิจฉัยแบบออนไลน์ผ่านโครงข่ายจีพีอาร์เอส ส่งผลให้รถยนต์ที่ติดตั้งระบบ OMD สามารถถูกตรวจสอบและวิเคราะห์อาการเสียเบื้องต้นแบบออนไลน์ได้ ซึ่งมีประโยชน์ต่อการวางแผนซ่อมบำรุงรถยนต์ เพื่อช่วยลดข้อผิดพลาด ค่าใช้จ่ายและเวลาในการซ่อมบำรุงได้

รูป 3.1 แสดงโครงสร้างของระบบ OMD ประกอบด้วย 2 ส่วนหลัก ส่วนแรกคือ OMD Client ที่ถูกติดตั้งในรถยนต์ ทำหน้าที่มอนิเตอร์และวินิจฉัยการทำงานรถยนต์ โดยข้อมูลสถานะและผลการวินิจฉัยรถยนต์บางส่วนถูกส่งต่อให้ส่วนที่สองคือ OMD Server ซึ่งเป็นซอฟต์แวร์จัดการที่อยู่ในคอมพิวเตอร์เซิร์ฟเวอร์ของศูนย์บริการ เพื่อบันทึกข้อมูลดังกล่าวลงฐานข้อมูลและแสดงผลบนหน้าเว็บเพจ โดยองค์ประกอบทั้ง 2 ส่วนสามารถสื่อสารข้อมูลร่วมกันผ่านทางเครือข่ายอินเทอร์เน็ต



รูป 3.1 โครงสร้างของระบบ OMD

(ก) OMD Client (ข) OMD Server

ส่วน OMD Client มีสมบัติดังนี้

1) สามารถระบุตำแหน่งของรถยนต์โดยใช้เทคโนโลยีจีพีเอส

2) สามารถอ่านข้อมูลสถานะและวินิจฉัยการทำงานของรถยนต์

3) สามารถบันทึกข้อมูลด้วยหน่วยความจำประเภทมัลติมีเดียการ์ด MMC (MultiMediaCard)

4) มีส่วนติดต่อกับอินเทอร์เน็ตโดยผ่านเครือข่ายจีพีอาร์เอส เพื่อรับ-ส่งข้อมูลจาก OMD Server

5) มีส่วนแสดงผลและรับคำสั่งจากผู้ใช้งาน

6) มีโครงสร้างที่สามารถปรับเปลี่ยนหรือรองรับการเพิ่มเติมสมบัติได้ในอนาคต

และส่วน OMD Server มีสมบัติดังนี้

1) สามารถบันทึกข้อมูลที่ได้รับจาก OMD Client ลงในฐานข้อมูลของคอมพิวเตอร์เซิร์ฟเวอร์

2) สามารถแสดงผลการวินิจฉัยรถยนต์รวมถึงข้อมูลส่วนต่างๆ ที่เกี่ยวกับการทำงานของรถยนต์ที่หน้าเว็บเพจของศูนย์บริการ

สำหรับงานวิจัยนี้มุ่งเน้นที่การพัฒนาต้นแบบระบบ OMD Client ในเบื้องต้นจึงได้พัฒนาส่วน OMD Server เพื่อให้สามารถทดสอบการทำงานของระบบต้นแบบที่พัฒนาขึ้นเท่านั้น ซึ่งรายละเอียดการออกแบบถูกอธิบายไว้ในหัวข้อถัดไป

3.1 การออกแบบส่วน OMD Client

จากสมบัติของ OMD Client พิจารณาได้ว่าการพัฒนาฟังก์ชันทำงานของระบบควรคำนึงถึงองค์ประกอบหลายๆ ด้านทั้งปริมาณข้อมูล ความสำคัญของข้อมูล เงื่อนไขเวลาในการประมวลผล ตัวอย่างเช่น ข้อมูลที่เกี่ยวกับสถานะของรถยนต์ซึ่งมีผลต่อความปลอดภัยของผู้ขับขี่ดังนั้นจึงมีความสำคัญสูงและต้องประมวลผลให้เร็วที่สุด ส่วนข้อมูลพิกัดตำแหน่งของรถยนต์มีการเปลี่ยนแปลงช้าและใช้เวลาประมวลผลช้า เป็นต้น ทำให้การออกแบบและพัฒนาระบบส่วนนี้ต้องมีความยืดหยุ่นสูงทั้งฮาร์ดแวร์และเฟิร์มแวร์เพื่อให้รองรับเงื่อนไขดังกล่าวได้ นอกจากนี้ต้องสามารถเลือกใช้หรือรองรับการเพิ่มเติมสมบัติได้ในอนาคต

การใช้ระบบฝังตัวแบบกระจายจึงเป็นแนวทางหนึ่งที่ใช้สำหรับออกแบบส่วน OMD Client ระบบฝังตัวแบบกระจายมีลักษณะโครงสร้างที่ประกอบด้วยโหนดย่อยหลายๆ โหนดและเชื่อมต่อกันด้วยบัสสื่อสาร ซึ่งแนวคิดการออกแบบเป็นโหนดย่อยมีข้อดีดังนี้คือ

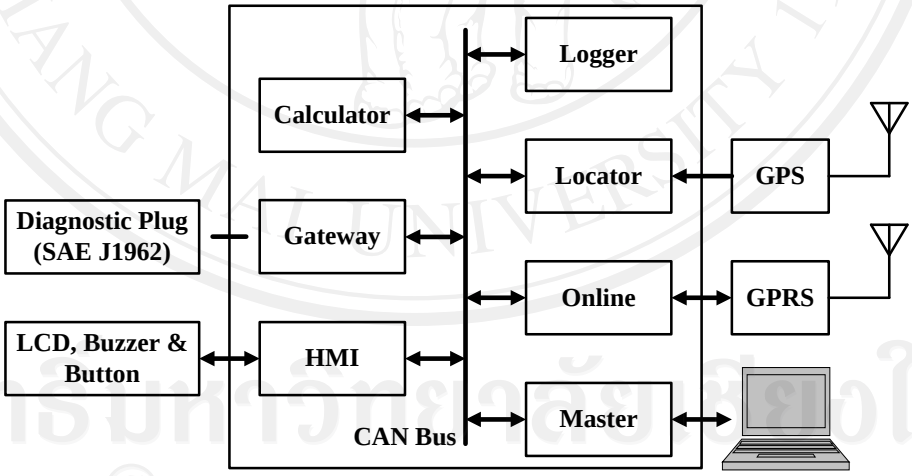
1) สามารถเพิ่มเติมโหนดใหม่เข้าไปในระบบทำงานเพื่อเพิ่มขีดความสามารถของระบบได้โดยไม่ต้องออกแบบระบบใหม่ทั้งหมดและไม่รบกวนการทำงานของโหนดที่มีอยู่ก่อนหน้านี้

2) สามารถพัฒนาวงจรและฟังก์ชันทำงานของแต่ละโหนด รวมถึงสามารถเลือกใช้อุปกรณ์และทรัพยากรต่างๆ ได้อย่างอิสระ

3) สนับสนุนให้ระบบสามารถแบ่งงานใหญ่ออกเป็นงานย่อยหลายๆ งาน จากนั้นกระจายให้แต่ละโหนดช่วยกันประมวลผลพร้อมกัน (Parallel Processing) โดยเลือกงานที่มีลักษณะใกล้เคียงกันไว้ที่โหนดเดียวกันส่งผลให้ระบบทำงานได้เร็วขึ้น

สิ่งสำคัญสำหรับระบบที่มีโครงสร้างแบบกระจายโหนดคือการเลือกใช้บัสสื่อสารข้อมูลที่มีความน่าเชื่อถือของข้อมูลสูงและมีส่วนจัดการกับข้อผิดพลาดที่เกิดจากการสื่อสาร นอกจากนี้เพื่อให้ได้ระบบฝังตัวที่มีประสิทธิภาพสูงตามต้องการ การออกแบบฮาร์ดแวร์ของระบบต้องมีความสอดคล้องกับฟังก์ชันการทำงานด้วย

ส่วน OMD Client ประกอบด้วยโหนดย่อยๆ ต่อทำงานร่วมกันดังแสดงในรูป 3.2 ซึ่งแบ่งออกเป็น 7 โหนดย่อยคือ โหนด Master โหนด Gateway โหนด Locator โหนด Online โหนด HMI โหนด Logger และโหนด Calculator



รูป 3.2 โครงสร้างส่วน OMD Client

โหนดย่อยแต่ละโหนดมีหน้าที่แตกต่างกันไปและมีโหนด Master เป็นหน่วยควบคุมกลางทำหน้าที่สร้างและควบคุมการสื่อสารข้อมูลระหว่างโหนดอื่นๆ ในระบบ รวมถึงการติดต่อกับคอมพิวเตอร์ส่วนบุคคลเพื่อบันทึกหรือแก้ไขค่าพารามิเตอร์ทำงานของระบบ นอกจากนี้ด้วยสมบัติเด่นของบัส CAN [8] ดังได้กล่าวอธิบายในหัวข้อ 2.2 จึงเลือกใช้บัส CAN และพัฒนาโพร

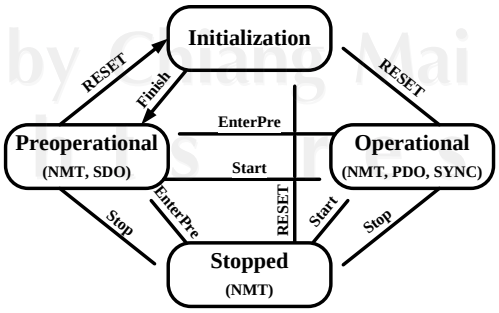
โทคอลชั้นสูงกว่าให้ควบคุมการทำงานของบัส CAN เพื่อใช้สื่อสารข้อมูลและเชื่อมต่อการทำงานของแต่ละโหนดเข้าด้วยกัน

3.1.1 หลักการทำงานของ OMD Client

การทำงานของ OMD Client เกิดจากการทำฟังก์ชันงานของทั้ง 7 โหนดย่อยร่วมกันเป็นฟังก์ชันงานรวมของระบบ โหนด Gateway ทำหน้าที่มอนิเตอร์และวินิจฉัยการทำงานของรถยนต์ที่ติดตั้งระบบ OMD Client ส่วนโหนด Locator ทำหน้าที่เป็นฐานเวลาจริงและบอกให้ทราบถึงตำแหน่งปัจจุบันของระบบ OMD Client หรือก็คือตำแหน่งของรถยนต์ด้วย จากนั้นข้อมูลสถานะต่างๆ ที่ได้จากการมอนิเตอร์ ผลของการวินิจฉัยรถยนต์และตำแหน่งของรถยนต์ถูกบันทึกด้วยโหนด Logger นอกจากนี้ข้อมูลบางส่วนถูกแสดงผลที่โหนด HMI และบางส่วนถูกรายงานให้ระบบ OMD Server ด้วยฟังก์ชันงานของโหนด Online ทั้งนี้การทำงานของทุกโหนดย่อยถูกควบคุมด้วยโหนด Master และโหนด Calculator ร่วมกัน

ปัจจัยสำคัญต่อการทำงานของ OMD Client คือการจัดการบัสสื่อสารข้อมูลระหว่างโหนด งานวิจัยนี้ได้พัฒนาโพรโทคอลชั้นสูงกว่า CAN โดยประยุกต์ใช้ความสามารถในการสื่อสารข้อมูลระหว่างอุปกรณ์ของโพรโทคอล CANopen เป็นต้นแบบ ซึ่งหลักการที่นำมาประยุกต์ใช้มีดังนี้

1) สถานะงานของโหนดสเลฟ ตามหลักการของโพรโทคอล CANopen หนึ่งโหนดในระบบถูกกำหนดให้เป็นโหนดมาสเตอร์ส่วนโหนดที่เหลือคือโหนดสเลฟ ในงานวิจัยนี้โหนด Master คือโหนดมาสเตอร์ ทำหน้าที่ควบคุมและตรวจสอบสถานะงานของโหนดสเลฟ โดยกำหนดสถานะงานของโหนดสเลฟไว้ 4 สถานะ คือ Initialization, Preoperational, Operational และ Stopped ซึ่งโหนด Master ใช้กลุ่มข้อความ NMT ประกอบด้วยข้อความ RESET, EnterPre, Start, Stop เพื่อเปลี่ยนสถานะงานของโหนดสเลฟดังผังสถานะในรูป 3.3 ทั้งนี้เพื่อลดความซับซ้อนในการพัฒนาเฟิร์มแวร์จึงได้กำหนดชนิดข้อความสื่อสารที่อนุญาตให้ใช้ได้ในแต่ละสถานะไว้แน่นอนดังที่เห็นในวงเล็บ



รูป 3.3 ผังสถานะของโหนดสเลฟ

2) บริการสื่อสาร SDO ในรูป 3.4 คือรูปแบบเฟรมร้องขอสำหรับสื่อสารข้อมูลด้วยบริการ SDO ตามข้อกำหนดของโพรโทคอล CANopen โดยใช้ตัวดรรชนีหลักขนาด 16 บิตและดรรชนีย่อยขนาด 8 บิตเพื่ออ้างอิงตำแหน่งข้อมูลที่ต้องการเข้าถึงในพจนานุกรมวัตถุ งานวิจัยนี้ออกแบบให้ใช้ตารางข้อมูลขนาดเล็กหลายๆ ตารางทำงานร่วมกันแทนการใช้พจนานุกรมวัตถุตามข้อกำหนดของโพรโทคอล CANopen โดยได้เปลี่ยนการใช้ตัวดรรชนีดังกล่าวเป็นรหัสฟังก์ชันขนาด 8 บิต (8-bits function code) และพารามิเตอร์ของฟังก์ชันขนาด 2 ไบต์ (16-bits function parameter) ดังแสดงในรูป 3.5 ซึ่งการใช้รหัสฟังก์ชันขนาด 8 บิตส่งผลให้เฟิร์มแวร์ใช้เวลาในการตัดสินใจเลือกทำฟังก์ชันสั้นลงกว่าเดิมและทำให้สามารถสร้างฟังก์ชันทำงานด้วยข้อความ SDO ได้ถึง 256 ฟังก์ชัน นอกจากนี้สามารถอ้างอิงถึงตำแหน่งของตารางที่ถูกเก็บไว้ในพื้นที่โปรแกรมของไมโครคอนโทรลเลอร์ด้วยการใช้พารามิเตอร์ของฟังก์ชันขนาด 2 ไบต์ (ตำแหน่ง 0x0000 ถึง 0xFFFF) ทั้งนี้ได้เตรียมฟังก์ชันพื้นฐานสำหรับอ่านและเขียนข้อมูลพื้นที่ส่วนโปรแกรมไว้ ทำให้สามารถแก้ไขโปรแกรมที่บรรจุในไมโครคอนโทรลเลอร์ด้วยการโปรแกรมข้อมูลผ่านบัสได้ทันที

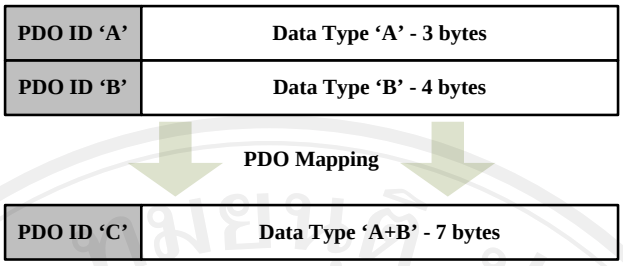
SDO ID	CS	16-bit Index	8-bit Sub-index	4 bytes
	1st	2nd – 3rd	4th	5th – 8th

รูป 3.4 รูปแบบเฟรมข้อมูลบริการ SDO ตามโพรโทคอล CANopen

SDO ID	CS	8-bit fnCode	16-bit fnParm	4 bytes
	1st	2nd	3rd - 4th	5th – 8th

รูป 3.5 รูปแบบเฟรมข้อมูลบริการ SDO ที่ใช้ในงานวิจัยนี้

3) บริการสื่อสาร PDO ตามมาตรฐานโพรโทคอล CANopen ข้อมูลที่ต้องการรับ-ส่งด้วยข้อความ PDO ต้องผ่านกระบวนการ PDO mapping [9] ก่อนเพื่อให้ได้หมายเลขข้อความและความหมายของข้อมูล รวมถึงตำแหน่งที่ใช้เก็บข้อมูลดังกล่าวในพจนานุกรมวัตถุ โดยมีรูปแบบอย่างง่ายดังรูป 3.6 ข้อมูลชนิดเดียวกันแต่คนละชุดหรือข้อมูลต่างชนิดกันที่รวมกันแล้วไม่เกิน 8 ไบต์ ถูกบรรจุให้อยู่ในข้อความเดียวกันและเก็บรายละเอียดต่างๆ ไว้ในพจนานุกรมวัตถุ ในงานวิจัยนี้ไม่ได้ใช้พจนานุกรมวัตถุและออกแบบให้หมายเลขข้อความ PDO มีความสัมพันธ์กับหมายเลขของโหนดแต่ละโหนด มีรูปแบบดังรูป 3.7 โดยไบต์แรกในส่วนข้อมูลถูกรหัสเป็นดรรชนี PDO ขนาด 8 บิต (8-bits PDO index) เพื่อใช้ระบุจำนวนข้อมูลในข้อความที่มีขนาด 0 ถึง 7 ไบต์ และใช้กำหนดฟังก์ชันทำงานจำนวน 32 ฟังก์ชันที่ถูกใช้ประมวลผลข้อมูลดังกล่าว ซึ่งการกำหนดให้หมายเลขข้อความ PDO มีความสัมพันธ์กับหมายเลขของโหนดขนาด 7 บิต ช่วยให้ผู้พัฒนาสามารถกำหนดรูปแบบการรับ-ส่งข้อมูลได้สะดวกขึ้น

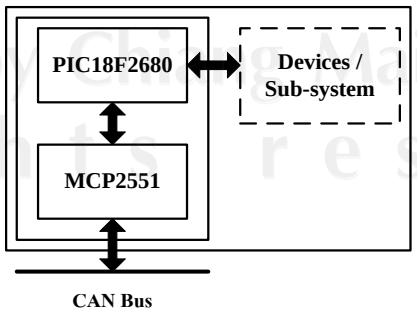


รูป 3.6 รูปแบบเฟรมข้อมูลบริการ PDO ตามโพรโทคอล CANopen

รูป 3.7 รูปแบบเฟรมข้อมูลบริการ PDO ที่ใช้ในงานวิจัยนี้

3.1.2 ฮาร์ดแวร์และเฟิร์มแวร์ของโหนดทั่วไป

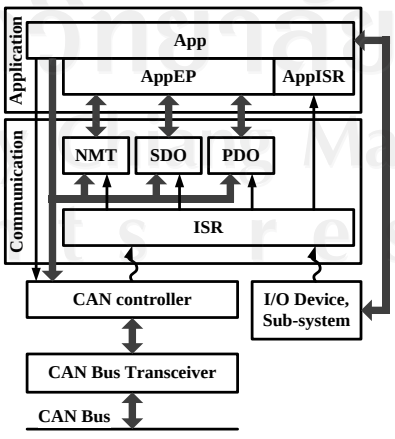
จากแนวคิดการออกแบบระบบฝังตัวโดยใช้โหนดย่อยๆ และสื่อสารข้อมูลระหว่างโหนดด้วยบัส CAN จึงแบ่งโครงสร้างของแต่ละโหนดออกเป็น 2 ส่วนคือ ส่วนควบคุมโหนดและส่วนควบคุมอุปกรณ์บริวาร โดยส่วนควบคุมโหนดมีองค์ประกอบทางฮาร์ดแวร์และเฟิร์มแวร์เหมือนกันทุกโหนด เพื่อทำหน้าที่ติดต่อกับบัส CAN และควบคุมการทำงานของส่วนควบคุมอุปกรณ์บริวารซึ่งแตกต่างกันไปในแต่ละโหนด ดังนั้นในการสร้างส่วนควบคุมโหนดจึงต้องพิจารณาองค์ประกอบทางฮาร์ดแวร์ 3 ส่วนสำคัญคือ (ก) ไมโครคอนโทรลเลอร์ (ข) CAN คอนโทรลเลอร์และ (ค) CAN ทรานซีฟเวอร์ งานวิจัยนี้เลือกใช้ไมโครคอนโทรลเลอร์เบอร์ PIC18F2680 (8 บิต) ที่มีมอดูล CAN คอนโทรลเลอร์ภายในตัว [14] และ CAN ทรานซีฟเวอร์เบอร์ MCP2551 [15] ของบริษัทไมโครชิพจำกัด (Microchip Inc.) ซึ่งอุปกรณ์ทั้งสองตัวนี้มีสมบัติทางฮาร์ดแวร์เพียงพอต่อการสร้างส่วนควบคุมโหนด รูป 3.8 แสดงโครงสร้างฮาร์ดแวร์ของโหนดทั่วไปโดยกรอบสี่เหลี่ยมใหญ่คือโครงสร้างของทั้งโหนด ส่วนกรอบสี่เหลี่ยมเล็กคือส่วนควบคุมโหนดและกรอบสี่เหลี่ยมเล็กเส้นประคือส่วนควบคุมอุปกรณ์บริวารที่มีแตกต่างกันไปแต่ละโหนด



รูป 3.8 โครงสร้างฮาร์ดแวร์ของโหนดทั่วไป

ไมโครคอนโทรลเลอร์ PIC18F2680 มีพื้นที่โปรแกรมขนาด 64 กิโลไบต์และมีหน่วยความจำ 3 กิโลไบต์ จุดเด่นที่สำคัญคือมี CAN คอนโทรลเลอร์ภายในตัวที่สนับสนุนมาตรฐาน ISO 11898 (CAN 2.0A, CAN 2.0B) โดยมีบัฟเฟอร์สำหรับรับ-ส่งข้อมูลขนาด 14 ไบต์จำนวน 2 และ 3 ชุด ตามลำดับ ซึ่งเพียงพอต่อการทำงานในงานวิจัยนี้และใช้งานได้ง่าย ทั้งนี้สามารถควบคุมการทำงานและตรวจสอบข้อผิดพลาดในการสื่อสารผ่านทางรีจีสเตอร์ที่เกี่ยวข้องได้ทันที นอกจากนี้ ไมโครคอนโทรลเลอร์ PIC18F2680 ยังประกอบด้วยมอดูลทำงานต่างๆ เช่น พอร์ตอินพุต-เอาต์พุต (I/O port) ไทม์เมอร์ (Timers) ส่วนติดต่อ UART (Universal Asynchronous Receiver Transmitter) ส่วนติดต่อบัส SPI (Serial Peripheral Interface) และ I^2C (Inter-Integrated Circuit) เป็นต้น ส่งผลให้สามารถประยุกต์ใช้ไมโครคอนโทรลเลอร์ PIC18F2680 ควบคุมการทำงานของอุปกรณ์บริวารที่มีส่วนติดต่อตามรูปแบบดังกล่าวได้อย่างหลากหลาย (ตั้งแต่เนื้อหาส่วนนี้ไปผู้วิจัยขอใช้คำว่า “ไมโครคอนโทรลเลอร์” แทนการใช้คำว่า “ไมโครคอนโทรลเลอร์ PIC18F2680”)

จากหลักการพัฒนาโพรโทคอลชั้นสูงกว่า CAN ดังที่ได้นำเสนอจึงได้พัฒนาเป็นเฟิร์มแวร์ที่บรรจุในไมโครคอนโทรลเลอร์ เพื่อควบคุมการทำงานของแต่ละโหนดซึ่งแบ่งเฟิร์มแวร์ออกเป็น 2 ส่วนหลักคือ เฟิร์มแวร์ส่วนคอมมิวนิเคชันและเฟิร์มแวร์ส่วนประยุกต์ เฟิร์มแวร์ส่วนคอมมิวนิเคชันทำหน้าที่ควบคุมการสื่อสารระหว่างโหนดผ่านบัส CAN และทำหน้าที่ควบคุมการทำงานของเฟิร์มแวร์ส่วนประยุกต์ ทั้งนี้เฟิร์มแวร์ส่วนคอมมิวนิเคชันเป็นส่วนที่มีหน้าที่เหมือนกันในทุกโหนดประกอบด้วยรูทีนย่อยที่สำคัญ 5 รูทีนดังผังโครงสร้างเฟิร์มแวร์ในรูป 3.9 ซึ่งทิศทางการเคลื่อนที่ของข้อมูล สัญญาณขัดจังหวะและการเรียกใช้รูทีนถูกแทนด้วยลูกศรเงา ลูกศรคลื่นและลูกศรตรง ตามลำดับ นอกจากนี้เนื่องจากใช้บัส CAN เป็นบัสสื่อสารข้อมูล ทำให้รูทีนย่อยในเฟิร์มแวร์ส่วนคอมมิวนิเคชันประกอบด้วยการทำงานที่เกี่ยวกับการประมวลผลเฟรมข้อมูล CAN เป็นหลัก



รูป 3.9 โครงสร้างเฟิร์มแวร์ของส่วนควบคุมโหนด

เฟิร์มแวร์ส่วนคอมมิวนิเคชันประกอบด้วยรูทีนย่อย 5 รูทีนดังนี้

1) รูทีน ISR (Interrupt Service Routine) คือส่วนตอบสนองอินเทอร์รัพที่เกิดขึ้นจาก CAN คอนโทรลเลอร์และจากอุปกรณ์บริวาร รูทีน ISR มีหน้าที่พิจารณาว่าสัญญาณอินเทอร์รัพที่เกิดขึ้นควรส่งต่อให้รูทีนย่อยส่วนใดประมวลผล โดยกำหนดให้สัญญาณอินเทอร์รัพที่เกิดจาก CAN คอนโทรลเลอร์มีความสำคัญสูงสุดและถูกพิจารณาเป็นอันดับแรก ส่วนอินเทอร์รัพที่เกิดจากอุปกรณ์บริวารจะเรียกใช้รูทีน AppISR เพื่อประมวลผลอีกที

2) รูทีน NMT คือส่วนตอบสนองข้อความ NMT เพื่อรายงานและเปลี่ยนสถานะงานของโหนดสเลฟตามที่โหนด Master ร้องขอ

3) รูทีน PDO คือส่วนตอบสนองข้อความ PDO โดยพิจารณาจากหมายเลขของโหนดผู้ส่งข้อความและดรรชนี PDO แล้วทำงานตามฟังก์ชันที่ถูกอ้างถึงในส่วนรูทีน AppEP

4) รูทีน SDO คือส่วนตอบสนองข้อความ SDO โดยพิจารณาจากรหัสฟังก์ชันและพารามิเตอร์ของฟังก์ชันที่ถูกร้องขอ แล้วทำงานตามฟังก์ชันที่ถูกระบุในรูทีน AppEP

5) รูทีน SYNC คือส่วนตอบสนองการเกิดข้อความ SYNC จากโหนด Master เพื่อให้โหนดสเลฟส่งข้อความ PDO ให้บัสตามคาบเวลาที่ได้กำหนดไว้

สำหรับเฟิร์มแวร์ส่วนประยุกต์มีองค์ประกอบของแต่ละรูทีนแตกต่างกันขึ้นอยู่กับการทำงานของอุปกรณ์บริวารที่ต่อพ่วงกับโหนดและหน้าที่ของแต่ละโหนดในระบบ ประกอบด้วยรูทีนย่อย 3 รูทีนดังนี้

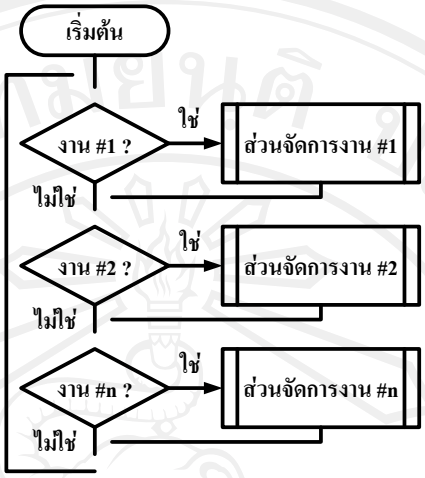
1) รูทีน AppISR คือส่วนตอบสนองอินเทอร์รัพที่เกิดจากอุปกรณ์บริวาร

2) รูทีน AppEP คือจุดให้บริการเพื่อตัดสินใจว่าแต่ละเฟรมข้อมูลที่ได้รับการร้องขอต้องทำฟังก์ชันใดและถ่ายโอนข้อมูลให้ฟังก์ชันนั้นเพื่อเตรียมการประมวลผล ทั้งนี้รูทีน AppEP ประกอบด้วยจุดบริการสำหรับข้อความ PDO, SDO, NMT และ SYNC

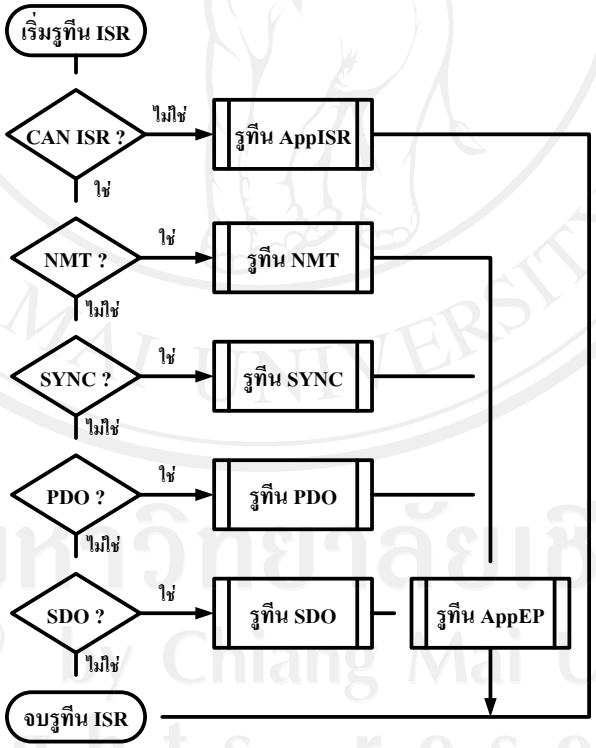
3) รูทีน APP คือโปรแกรมส่วนประยุกต์เพื่อสร้างงานตามหน้าที่ของแต่ละโหนดในระบบ แบ่งออกเป็นรูทีนหรืองาน (Task) ย่อยๆ หลายส่วนทำงานร่วมกัน

การออกแบบและพัฒนาเฟิร์มแวร์ทั้งส่วนคอมมิวนิเคชันและส่วนประยุกต์ ใช้หลักการแบ่งแต่ละรูทีนหรืองานออกเป็นหลายๆ ขั้นตอนทำให้ใช้เวลาประมวลผลแต่ละขั้นตอนสั้น และมีส่วนจัดการทำหน้าที่ควบคุมลำดับการทำงานอีกที (State Machine) ส่งผลให้เสมือนว่าทำทุกงานเสร็จสิ้นพร้อมกัน (Multi-tasking) นอกจากนี้ยังใช้หลักการ look-up-table ในการพัฒนาฟังก์ชันทำงานที่เกี่ยวข้องกับตารางข้อมูล จึงทำให้การเปิดหรือเรียกดูตารางข้อมูลใช้เวลาสั้น ในรูป 3.10 แสดงผังทำงานปกติของโหนดทั่วไป ซึ่งในหนึ่งรอบการทำงานของโหนดแต่ละขั้นตอนย่อยของทุกงาน

จะถูกเรียกใช้งาน โดยมีส่วนจัดการงานคอยลำดับขั้นตอนของแต่ละงาน ส่วนรูป 3.11 แสดงผังทำงานของรูทีน ISR ร่วมกับรูทีนย่อยอื่นๆ เมื่อเกิดสัญญาณอินเทอร์รัพ



รูป 3.10 ผังทำงานปกติของโหนดทั่วไป

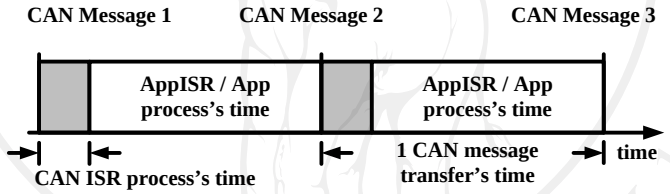


รูป 3.11 ผังทำงานของรูทีน ISR

ในการพัฒนาเฟิร์มแวร์ของโหนดทั่วไป รูทีน ISR ที่ดีควรออกแบบให้ใช้เวลาประมวลผลและจัดการกับข้อมูลที่ได้รับให้สั้นที่สุด สำหรับงานวิจัยนี้ใช้บัส CAN เป็นบัสสื่อสารข้อมูลหลัก สัญญาณอินเทอร์รัพจาก CAN คอนโทรลเลอร์จึงมีความสำคัญสูงสุดและเกิดขึ้นบ่อย โดยเกิดขึ้น

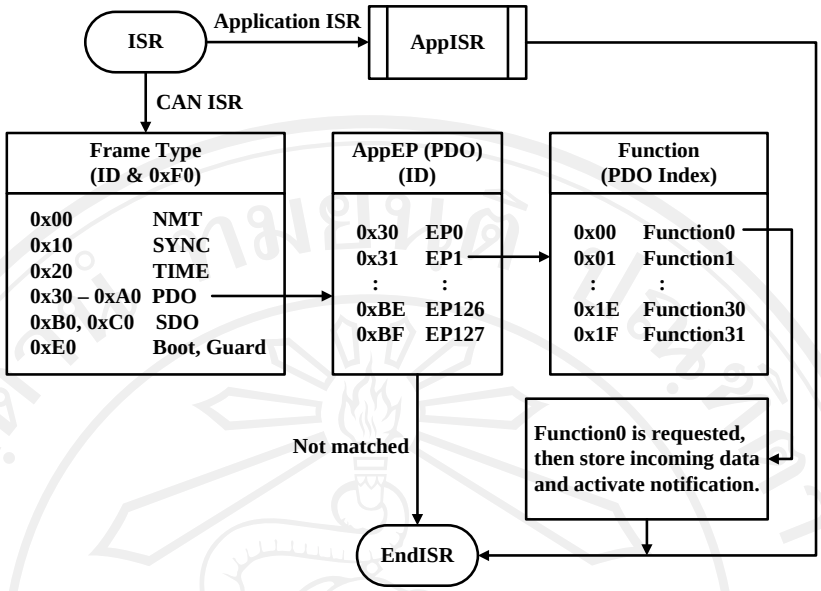
เมื่อบัฟเฟอร์สำหรับรับเฟรมข้อมูลได้รับเฟรมข้อมูลที่ถูกต้องและครบถ้วน ซึ่งจากจำนวนบัฟเฟอร์สำหรับรับเฟรมข้อมูล CAN จำนวน 2 ชุดและความสามารถทางด้านฮาร์ดแวร์ของไมโครคอนโทรลเลอร์ที่ใช้ทำให้ขณะที่รูทีน ISR กำลังจัดการกับเฟรมข้อมูลที่บัฟเฟอร์ชุดที่ 1 ได้รับ บัฟเฟอร์ชุดที่ 2 สามารถรับอีกเฟรมข้อมูล CAN ได้ทันที ดังนั้นระยะเวลาระหว่างการเกิดอินเทอร์รัพหรือระยะเวลาในการสื่อสารแต่ละเฟรมข้อมูล CAN จึงมีความสำคัญและถูกพิจารณาเพื่อพัฒนารูทีน ISR

ในรูป 3.12 แสดงการทำงานเมื่อโหนดได้รับเฟรมข้อมูล CAN มากกว่า 2 เฟรมต่อเนื่องกัน ส่วนที่แรเงาคือช่วงเวลาที่รูทีน ISR ใช้ประมวลผลหรือจัดการเฟรมข้อมูล CAN ที่ได้รับซึ่งถ้ารูทีน ISR ใช้เวลาประมวลผลสั้น จะส่งผลให้สัดส่วนเวลาที่สามารถใช้ประมวลผลรูทีนย่อยอื่นๆ มีเพิ่มขึ้น ทั้งนี้การพัฒนาเฟิร์มแวร์ด้วยการแบ่งงานออกเป็นหลายๆ ขั้นตอนและใช้เวลาทำแต่ละงานสั้น ช่วยลดความผิดพลาดของการทำงานที่อาจเกิดขึ้นเนื่องจากการถูกขัดจังหวะด้วยสัญญาณอินเทอร์รัพจาก CAN คอนโทรลเลอร์ได้



รูป 3.12 ผังเวลาเมื่อเกิดอินเทอร์รัพอย่างต่อเนื่องจาก CAN คอนโทรลเลอร์

นอกจากระยะเวลาที่ใช้สื่อสารเฟรมข้อมูล CAN แล้ว ระยะเวลาที่ทำงานร่วมกับรูทีนอื่นก็มีความสำคัญเช่นกันสำหรับการพัฒนารูทีน ISR พิจารณาตัวอย่างการสื่อสารข้อมูลด้วยข้อความ PDO ซึ่งมีความสำคัญสูงและมีเงื่อนไขทางเวลา รูทีน ISR ต้องทำงานผสานกับรูทีน PDO และรูทีน AppEP มีลำดับการทำงานแสดงดังรูป 3.13 เมื่อโหนดได้รับข้อความ PDO และเกิดสัญญาณอินเทอร์รัพจาก CAN คอนโทรลเลอร์ แล้วรูทีน ISR ใช้การเปิดตารางด้วยหมายเลขข้อความที่ได้รับเพื่อแยกชนิดการร้องขอบริการ จากนั้นเมื่อพบว่าเป็นข้อความ PDO จึงส่งต่อให้รูทีน PDO ต่อไปในทำนองเดียวกันรูทีน PDO ใช้การเปิดตารางด้วยหมายเลขข้อความอีกครั้งเพื่อหาตำแหน่งจุดบริการที่เชื่อมต่อกับรูทีน AppEP ซึ่งในท้ายที่สุดข้อมูลที่ได้รับจะถูกประมวลผลด้วยฟังก์ชันที่ระบุด้วยดรรชนี PDO หรือส่งต่อให้ฟังก์ชันย่อยอื่นอีกที จากนั้นจึงเป็นการสิ้นสุดการทำงานของรูทีน ISR ทั้งนี้หากไม่พบจุดบริการในรูทีน AppEP ก็จบการทำงานของรูทีน ISR ทันที



รูป 3.13 ผังทำงานของรูทีน ISR รูทีน PDO และรูทีน AppEP

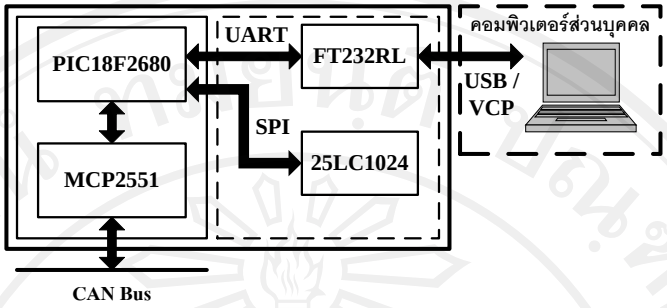
สำหรับการสื่อสารเฟรมข้อมูล CAN ที่อัตราบิตเท่ากับ 500 กิโลบิตต่อวินาทีนั้น 1 เฟรมข้อมูล CAN ใช้เวลาสื่อสารประมาณ 94 – 260 ไมโครวินาที [16] ด้วยเฟิร์มแวร์ที่พัฒนาขึ้นในงานวิจัยนี้ พบว่าจากตัวอย่างที่กล่าวถึงรูทีน ISR รูทีน PDO และรูทีน AppEP ใช้เวลาประมวลผลเฟรมข้อมูลดังกล่าวโดยใช้ค่าน้อยที่สุดประมาณ 49 รอบทำงานของไมโครคอนโทรลเลอร์ PIC18F2680 ซึ่งทำงานด้วยสัญญาณนาฬิกาเท่ากับ 11.0592 เมกะเฮิรตซ์หรือคิดเป็นประมาณ 18 ไมโครวินาทีโดยใช้วิธีการคำนวณเวลาการทำงานตาม [14] จึงทำให้มีเวลามากพอเพื่อประมวลผลงานส่วนอื่นๆ ของโหนด นอกจากนี้เฟิร์มแวร์ส่วนคอมมิวนิเคชันที่พัฒนาขึ้นใช้พื้นที่โปรแกรมขนาด 2 กิโลไบต์ และใช้หน่วยความจำขนาด 256 ไบต์ พบว่ามีทรัพยากรเพียงพอเพื่อพัฒนาเฟิร์มแวร์ส่วนประยุกต์ที่ฟังก์ชันทำงานมีหลายขั้นตอนหรือมีความซับซ้อนได้

3.1.3 โหนด Master

โหนด Master ทำหน้าที่เป็นตัวกลางในการสื่อสารข้อมูลระหว่างระบบ OMD กับคอมพิวเตอร์ส่วนบุคคล เพื่อตั้งค่าตารางและพารามิเตอร์ทำงานต่างๆ รวมถึงใช้ทดสอบการทำงานของระบบ OMD

1) ฮาร์ดแวร์ของโหนด Master อุปกรณ์บริวารของโหนด Master ประกอบด้วยส่วนติดต่อวงจรแปลงข้อมูล UART เป็นยูเอสบี โดยได้เลือกใช้ไอซีเบอร์ FT232RL [17] จากการทำงานของไอซีนี้ทำให้คอมพิวเตอร์มองเห็นพอร์ตยูเอสบีเป็น COM Port เสมือนจริง (Virtual COM Port, VCP) ช่วยให้คอมพิวเตอร์สามารถติดต่อกับไมโครคอนโทรลเลอร์ผ่านทางมอดูล

UART ภายในตัวได้ นอกจากนี้ยังประกอบด้วยหน่วยความจำอีอีพรอมเบอร์ 25LC1024 และมีผังโครงสร้างของโหนดดังรูป 3.14



รูป 3.14 ผังโครงสร้างของโหนด Master

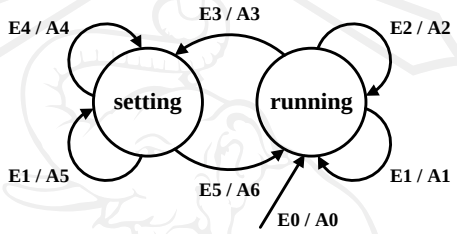
2) เฟิร์มแวร์ส่วนประยุกต์ของโหนด Master มี 2 รูทีนหลักประกอบด้วย

2.1) *รูทีน PC Interface* ซึ่งทำหน้าที่รับคำสั่งจากคอมพิวเตอร์และรายงานข้อมูลให้กับคอมพิวเตอร์ โดยได้ออกแบบแพ็คเก็ตสำหรับรับ-ส่งข้อมูลไว้มีรูปแบบดังแสดงในตาราง 3.1 หนึ่งแพ็คเก็ตข้อมูล UART ประกอบด้วยส่วนหัว (HEADER) ความยาวข้อมูล (LEN) ไบต์ข้อมูล (DATA) และไบต์ผลการ XOR ข้อมูลตั้งแต่ส่วนหัวจนถึงไบต์สุดท้ายของข้อมูล (Check-sum, CHKSUM) ตัวอย่างเช่น แพ็คเก็ตที่ใช้เป็นคำสั่งมีความยาวข้อมูลเพียง 1 ไบต์เท่านั้น ส่วนแพ็คเก็ตที่ใช้รับและรายงานเฟรมข้อมูล CAN มีความยาวข้อมูลเท่ากับผลรวมของไบต์หมายเลขประจำตัวขนาด 2 ไบต์และจำนวนข้อมูลในเฟรม CAN เป็นต้น

ตาราง 3.1 รูปแบบและตัวอย่างแพ็คเก็ตข้อมูล UART

UART packet format	<HEADER><LEN>[<$DATA_0$><$DATA_1$>…<$DATA_{LEN-1}$>]<CHKSUM>	
Descriptions	<…>	Byte (Hexadecimal)
	[…]	Data field (Hexadeimal)
	HEADER	Packet's header, value is 0xF5
	LEN	Length of data of packet
	DATAn	Data of packet, n {0 to (LEN-1)}
	CHKSUM	Packet's check-sum, calculate using XOR from <HEADER> to <$DATA_{LEN-1}$>
Examples	Command: Request setting mode ('S') <F5><01>[<53>]<A7>	
	CAN Frame : Request NMT RESET for all node <F5><04>[<00><00><81><00>]<70>	

2.2) *รูทีน Manager* ทำหน้าที่ประมวลผลและตอบสนองตามเงื่อนไขของเฟรมข้อมูล CAN ที่ได้รับจากบัสสื่อสาร CAN โดยรูทีนนี้ทำงานร่วมกับเฟิร์มแวร์ส่วนคอมมิวนิเคชัน แบ่งโหมดการทำงานของโหนดได้ 2 โหมดหลักคือ โหมดตั้งค่า (setting) และโหมดทำงาน (running) โหมดตั้งค่าใช้การเชื่อมต่อกับคอมพิวเตอร์เพื่อกำหนดค่าตารางหรือแก้ไขพารามิเตอร์ของโหนดอื่นๆ รวมถึงใช้ทดสอบและวินิจฉัยการทำงานของระบบ OMD Client ส่วนโหมดทำงานคือโหมดที่โหนด Master ประมวลผลการทำงานตามที่ได้โปรแกรมไว้ ซึ่งสามารถแสดงทิศทางการเคลื่อนข้อมูลของรูทีนนี้ดังรูป 3.15 และมีรายชื่ออีเวนท์และแอคชันสำหรับใช้อธิบายทิศทางการเคลื่อนข้อมูลของโหนด Master ดังตาราง 3.2



รูป 3.15 ทิศทางการเคลื่อนข้อมูลของโหนด Master

ตาราง 3.2 รายการอีเวนท์และแอคชันของรูทีน Manager

No	Event	Action
0	Power-on	Initialize working registers
1	Receive CAN data frame	Process CAN data, if (fMonitor) report to PC
2	Get internal trigger (timer, event, defined message)	Process trigger
3	Request setting mode	Store working parameter, change to setting mode
4	Request send CAN data frame	Adjust incoming UART packet, transmit CAN data frame
5	Request running mode	Adjust incoming CAN data frame, report UART packet to PC
6		Store working parameter, change to running mode

3.1.4 โหนด Gateway

โหนด Gateway ทำหน้าที่เชื่อมต่อการทำงานระหว่าง OMD Client กับระบบวินิจฉัยการทำงานภายในรถยนต์ เพื่อทำให้ส่วน OMD Client สามารถอ่านค่าสถานะต่างๆ และผลการวินิจฉัยการทำงานของรถยนต์ได้

1) ฮาร์ดแวร์ของโหนด Gateway จากสมบัติการทำงานของไอซี ELM327 [18] ซึ่งเป็นผลิตภัณฑ์สแกนทูลที่สามารถวินิจฉัยรถยนต์ได้และจากการศึกษาการทำงานของระบบวินิจฉัยรถยนต์พบว่าสามารถพัฒนาสแกนทูลสำหรับติดต่อทำงานกับระบบวินิจฉัยรถยนต์ได้เองเพื่อใช้กับระบบ OMD Client โดยเฉพาะ ซึ่งสแกนทูลที่พัฒนาขึ้นเองมีสมบัติดังนี้

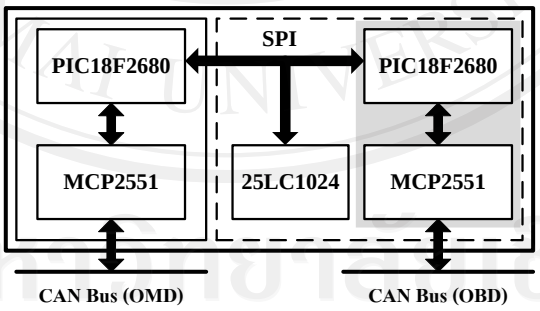
ก) สามารถติดต่อกับระบบวินิจฉัยรถยนต์ที่มีบัสสื่อสาร CAN ผ่านทางตัวเชื่อมต่อ SAE J1962 (รูป 2.1)

ข) รองรับโหมดการทำงาน 01, 03 และ 04 ตามข้อกำหนด SAE J1979

ค) สามารถรายงานผลรหัส DTC ตามข้อกำหนด SAE J2012

ง) ควบคุมการทำงานด้วยคำสั่งผ่านทางบัส SPI (Serial Peripheral Interface)

สแกนทูลที่พัฒนาขึ้น มีฮาร์ดแวร์เหมือนส่วนควบคุมโหนด แต่มีส่วนติดต่อบัส SPI เพื่อใช้รับ-ส่งข้อมูลรวมถึงใช้ควบคุมการทำงานของสแกนทูล และได้ออกแบบแพ็คเก็ตข้อมูล SPI สำหรับการทำงานดังกล่าว รูป 3.16 แสดงผังโครงสร้างของโหนด Gateway ที่มีส่วนประกอบของสแกนทูลที่พัฒนาขึ้น (ส่วนที่แรเงา) และอีอีพรอมเบอร์ 25LC1024

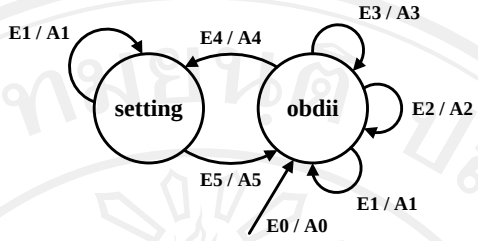


รูป 3.16 ผังโครงสร้างของโหนด Gateway

2) เฟิร์มแวร์ส่วนประยุกต์ของโหนด Gateway ประกอบด้วย 2 รูทีนหลัก

2.1) *รูทีน OBD Interface* ทำหน้าที่เกี่ยวกับการมอนิเตอร์สถานะและวินิจฉัยการทำงานรถยนต์ ในรูป 3.17 แสดงทิศทางการเคลื่อนข้อมูลของสแกนทูลที่พัฒนาขึ้นซึ่งถูกอธิบายด้วยรายการอีเวนท์และแอคชันในตาราง 3.3 โดยโหมด obdii สามารถแบ่งการทำงานออกเป็น 2 ลักษณะคือ แบบแมนนวล (Manual Diagnosis) และแบบออโต้ (Auto Monitor) แบบออโต้

สามารถรองรับเฉพาะการร้องขอดูข้อมูลสถานะของรถยนต์ผ่านโหมด 01 เท่านั้น ซึ่งทำการร้องขอด้วย PID ตามตารางที่บันทึกไว้ในทุกๆ คาบตามกำหนดเวลา



รูป 3.17 ทิศทางการเคลื่อนข้อมูลทำงานของสแกนทูล

ตาราง 3.3 รายการอีเวนท์และแอคชันของรูทีน OBD Interface

No	Event	Action
0	Power-on	Initialize working registers
1	Receive SPI command byte	Receive all SPI packet, process command
2	Get internal trigger (timer, event)	Process trigger
3	Found number of DTCs	Request read DTCs code, store in prepared RAM
4	Request setting mode	Store working parameter, change to setting mode
5	Request obdii mode	Store working parameter, change to obdii mode

2.2) *รูทีน SPI Interface* ใช้สำหรับสื่อสารข้อมูลระหว่างส่วนควบคุมโหนดและสแกนทูล รวมถึงการแก้ไขตั้งค่าตาราง PID และตารางคาบเวลาทำงาน โดยมีรูปแบบการสื่อสารด้วยแพ็คเก็ต SPI ดังตาราง 3.4

ตาราง 3.4 รูปแบบแพ็คเก็ต SPI

SPI packet format	<CMD>, <CMD>[<$PARAM_0$><$PARAM_1$>], <CMD>[<$DATA_0$>…<$DATA_{127}$>]	
Descriptions	<…>	Byte (Hexadecimal)
	[…]	Data field (Hexadeimal)
	CMD	Command
	$PARAM_n$	Parameter of command
	DATA	Data of packet

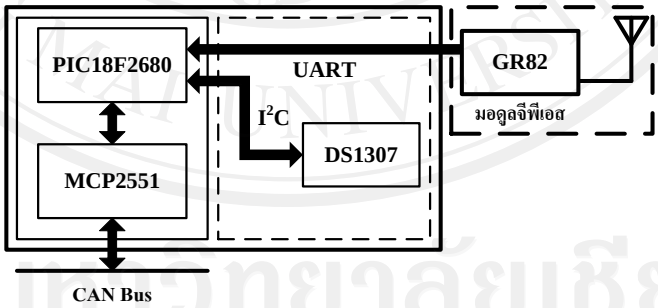
ตาราง 3.4 รูปแบบแพ็คเก็ต SPI (ต่อ)

Example	Auto request diagnostic (All or PID) <AOBD>[< All | PID>]
	Manual request diagnostic (SVid and PID) <MOBD>[<SVid><PID>]
	Read diagnostic results (All or PID) <ROBD>[<All | PID>]

3.1.5 โหนด Locator

โหนด Locator ทำหน้าที่สร้างข้อความตำแหน่งและข้อความฐานเวลาให้บัส CAN เพื่อเป็นอินพุตให้ฟังก์ชันของโหนดอื่นๆ ในระบบ ข้อความตำแหน่งสร้างโดยใช้ข้อมูลพิกัดละติจูดและลองจิจูดซึ่งได้รับจากมอดูลรับสัญญาณจีพีเอสแล้วนำมาแปลงเป็นค่าตำแหน่ง (Position) ส่วนข้อความฐานเวลาได้รับข้อมูลจากไอซีที่ทำหน้านี่รักษาเวลาจริง

1) ฮาร์ดแวร์ของโหนด Locator รูป 3.18 แสดงผังโครงสร้างของโหนด Locator ซึ่งอุปกรณ์บริวารประกอบด้วยส่วนติดต่อกับวงจรสร้างฐานเวลาจริงและส่วนติดต่อกับมอดูลรับสัญญาณจีพีเอส ทั้งนี้มอดูลที่เลือกใช้คือ มอดูล GR82 สามารถสื่อสารข้อมูลผ่านทางมอดูล UART ของไมโครคอนโทรลเลอร์และให้ข้อมูลพิกัดละติจูด-ลองจิจูดที่อยู่ในรูปแบบข้อมูลอนุกรมตามมาตรฐาน NMEA โดยมีความคลาดเคลื่อนไม่เกิน 20 เมตร [13]



รูป 3.18 ผังโครงสร้างของโหนด Locator

2) เฟิร์มแวร์ประยุกต์ของโหนด Locator ประกอบด้วยรูทีนย่อยที่สำคัญคือ

2.1) *รูทีนฐานเวลาจริง* ใช้ควบคุมการทำงานของไอซีเบอร์ DS1307 [19] ซึ่งสามารถรักษาฐานเวลาจริงได้ (Real Time Clock) รูทีนนี้มีฟังก์ชันสำหรับตั้งค่าวัน-เวลา อ่านค่าวัน-เวลา อ่านและบันทึกค่าในหน่วยความจำภายในของไอซี รวมถึงฟังก์ชันสร้างข้อความที่บรรจุด้วยค่าวัน-เวลา (ข้อความ TIME) เพื่อส่งให้กับบัส CAN ทุกๆ คาบเวลาที่กำหนดไว้ ค่าวัน-เวลามีขนาด 6 ไบต์ประกอบด้วยข้อมูล วินาที นาที ชั่วโมง วันที่ เดือน และปี

2.2) *รูทีน GPS* สำหรับรับข้อมูลพิกัดตามมาตรฐาน NMEA ด้วยมอดูล UART แล้วแปลงข้อมูลดังกล่าวเป็นข้อความตำแหน่ง (Position) ก่อนส่งให้บัส CAN ทุกๆ คาบข้อความ SYNC ที่กำหนดไว้ โดยข้อมูลพิกัดละติจูด-ลองจิจูดถูกแปลงให้อยู่ในรูปแบบชุดข้อมูลตำแหน่งก่อน เพื่อให้สามารถสื่อสารชุดข้อมูลตำแหน่งด้วยข้อความ CAN เพียงหนึ่งข้อความได้ ข้อมูลละติจูด-ลองจิจูดที่ได้จากเรคอร์ด RMC ตามมาตรฐาน NMEA ของมอดูลจีพีเอส มีรูปแบบเรคอร์ดดังตัวอย่างต่อไปนี้

$GPRMC,203157.000,A,1847.7838,N,09857.1092,E,0.00,,100108,,,A*7D

ข้อมูลพิกัดละติจูด คือ 1847.7838, N (18 องศาเหนือ 47.7838 ลิปดา)

ข้อมูลพิกัดลองจิจูด คือ 09857.1092, E (98 องศาตะวันออก 57.1092 ลิปดา)

ทั้งนี้ประเทศไทยตั้งอยู่ในช่วงพิกัดละติจูดที่ 5 ถึง 20 องศาเหนือและช่วงพิกัดลองจิจูดที่ 97 ถึง 105 องศาตะวันออก จากขอบเขตข้อมูลดังกล่าวสามารถแปลงเป็นข้อมูลตำแหน่งจำนวน 6 ไบต์ โดยมีรูปแบบเป็นการเข้ารหัสแบบบิต [A7A6…F1F0] ดังตาราง 3.5 และจากตัวอย่างเรคคอร์ดดังกล่าวสามารถแปลงเป็นชุดข้อมูลตำแหน่งได้มีค่าเท่ากับ **0xDBE7268E455C**

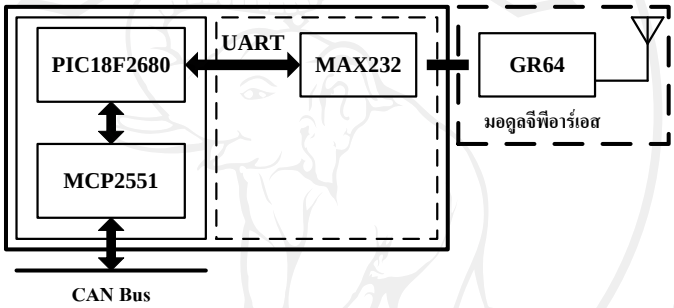
ตาราง 3.5 ตัวอย่างการเข้ารหัสข้อมูลพิกัดละติจูดและลองจิจูด

ไบต์ข้อมูล	ความหมาย	ค่าต่ำสุด	ค่าสูงสุด
A (บิต 7-4)	องศาของพิกัดละติจูด	0x0 (องศาที่ 5)	0x0F (องศาที่ 20)
A (บิต 3-0) และ B (บิต 7-6)	ลิปดาของพิกัดละติจูด	0x00 (ลิปดาที่ 0)	0x3B (ลิปดาที่ 59)
B (บิต 5-0) และ C (บิต 7)	ทศนิยมหลักที่ 1 และ 2 ของลิปดาของพิกัดละติจูด	0x00 (ค่า 0)	0x63 (ค่า 99)
C (บิต 6-0)	ทศนิยมหลักที่ 3 และ 4 ของลิปดาของพิกัดละติจูด	0x00 (ค่า 0)	0x63 (ค่า 99)
D (บิต 7-4)	องศาของพิกัดลองจิจูด	0x0 (องศาที่ 90)	0x0F (องศาที่ 105)
D (บิต 3-0) และ E (บิต 7-6)	ลิปดาของพิกัดลองจิจูด	0x00 (ลิปดาที่ 0)	0x3B (ลิปดาที่ 59)
E (บิต 5-0) และ F (บิต 7)	ทศนิยมหลักที่ 1 และ 2 ของลิปดาของพิกัดลองจิจูด	0x00 (ค่า 0)	0x63 (ค่า 99)
F (บิต 6-0)	ทศนิยมหลักที่ 3 และ 4 ของลิปดาของพิกัดลองจิจูด	0x00 (ค่า 0)	0x63 (ค่า 99)

3.1.6 โหนด Online

โหนด Online ทำหน้าที่รับ-ส่งข้อมูลระหว่างระบบ OMD Client กับ OMD Server โดยการสร้างส่วนติดต่อกับอินเทอร์เน็ตผ่านทางเครือข่ายเทคโนโลยีจีพีอาร์เอสในทุกๆ คาบเวลาที่กำหนดไว้และสื่อสารข้อมูลด้วยวิธี GET ของโพรโทคอล HTTP

1) ฮาร์ดแวร์ของโหนด Online รูป 3.19 คือผังโครงสร้างโหนด Online ประกอบด้วยส่วนควบคุมโหนดและมีส่วนติดต่อกับมอดูลจีพีอาร์เอสผ่านมอดูล UART เป็นส่วนอุปกรณ์บริวารมอดูลจีพีอาร์เอสที่เลือกใช้คือมอดูล GR64 [20] สามารถสื่อสารข้อมูลผ่านอินเทอร์เน็ตด้วยโพรโทคอล HTTP และ FTP ที่อัตราดาวน์โหลดข้อมูลสูงสุดถึง 48 กิโลบิตต่อวินาทีและอัตราอัปโหลดข้อมูลสูงสุด 12 กิโลบิตต่อวินาที

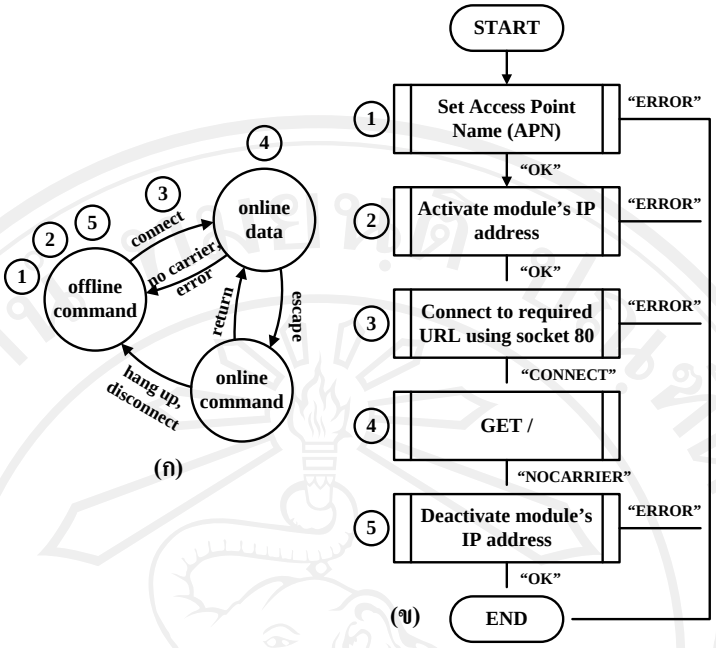


รูป 3.19 ผังโครงสร้างของโหนด Online

2) เฟิร์มแวร์ประยุกต์ของโหนด Online มีรูทีนย่อยที่สำคัญคือ

2.1) *รูทีน GPRS Interface* มีหน้าที่สร้างการติดต่อกับอินเทอร์เน็ตผ่านเครือข่ายจี-พีอาร์เอสทุกๆ คาบเวลาที่กำหนดไว้ ซึ่งสามารถสั่งให้มอดูล GR64 ทำงานโดยใช้กลุ่มคำสั่ง AT (AT command) ผ่านทางมอดูล UART ของไมโครคอนโทรลเลอร์ ในรูป 3.20 (ก) แสดงผังสถานะงานของมอดูลจีพีอาร์เอส เมื่อเชื่อมต่อกับอินเทอร์เน็ต ประกอบด้วยสถานะงานดังนี้

- offline command คือสถานะที่การทำงานของมอดูลจีพีอาร์เอสถูกควบคุมด้วยกลุ่มคำสั่ง AT เพื่อกำหนดค่าการพารามิเตอร์ทำงานต่างๆ ที่จำเป็นในการสร้างการติดต่อกับอินเทอร์เน็ต
- online data คือสถานะงานเมื่อมีการเชื่อมต่อกับอินเทอร์เน็ตแล้ว ข้อมูลทุกตัวที่รับ-ส่งกับมอดูลมีรูปแบบและกระบวนวิธีขึ้นอยู่กับโพรโทคอลสื่อสารที่ใช้เชื่อมต่อเช่น HTTP หรือ FTP เป็นต้น
- online command คือสถานะงานที่สามารถใช้กลุ่มคำสั่ง AT เพื่อควบคุมการทำงานของมอดูลได้โดยตรงในขณะที่ยังไม่ตัดการเชื่อมต่อกับอินเทอร์เน็ต



รูป 3.20 ผังสถานะงานและขั้นตอนการทำงานของมอดูลจีพีอาร์เอส
(ก) ผังสถานะงานแบบง่าย (ข) ขั้นตอนสื่อสารข้อมูลผ่านโพรโทคอล HTTP

ส่วนรูป 3.20 (ข) คือขั้นตอนการทำงานของมอดูลจีพีอาร์เอส เพื่อร้องขอข้อมูลผ่านโพรโทคอล HTTP ซึ่งสรุปขั้นตอนตามหมายเลขกำกับได้ดังนี้

- ขั้นตอนที่ 1 คือการตั้งค่า Access Point Name (APN) ของผู้ให้บริการเครือข่ายจีพี-อาร์เอสตามซิมการ์ดที่เลือกใช้เช่น ค่า APN ของ DTAC คือ www.dtac.co.th เป็นต้น
- ขั้นตอนที่ 2 คือการขอหมายเลข IP ของมอดูลจีพีอาร์เอสจากเครือข่ายของผู้ให้บริการ
- ขั้นตอนที่ 3 คือการสร้างส่วนติดต่อไปยังเซิร์ฟเวอร์ที่ต้องการ โดยระบุหมายเลข IP ของคอมพิวเตอร์เซิร์ฟเวอร์และโพรโทคอลที่ใช้สำหรับสื่อสารข้อมูล ซึ่งการเลือกใช้หมายเลขซ็อกเก็ต 80 คือการร้องขอแลกเปลี่ยนข้อมูลด้วยโพรโทคอล HTTP
- ขั้นตอนที่ 4 คือการร้องขอข้อมูลจากคอมพิวเตอร์ที่เชื่อมต่อด้วยวิธี GET ของโพรโทคอล HTTP (ภาคผนวก ก)
- ขั้นตอนที่ 5 คือการถอนมอดูลจีพีอาร์เอสออกจากเครือข่ายจีพีอาร์เอส

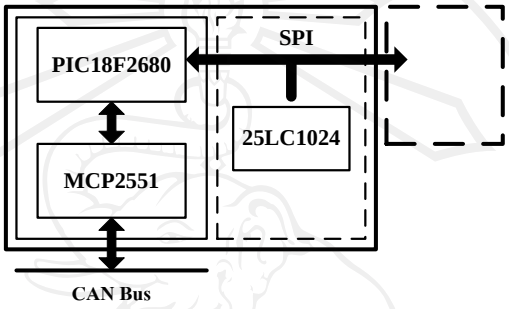
ขั้นตอนการทำงานกับมอดูลจีพีอาร์เอสดังที่ได้กล่าวถึงเป็นการสั่งงานจากมนุษย์ซึ่งหากพบข้อผิดพลาด มนุษย์สามารถตัดสินใจแก้ปัญหาได้ ดังนั้นเมื่อเปลี่ยนจากมนุษย์เป็นไมโครคอนโทรลเลอร์ (Machine-to-Machine) จึงต้องพัฒนากระบวนวิธีทำงานให้สามารถป้องกันปัญหาที่คาดว่าจะพบให้ได้มากที่สุด การสร้างส่วนติดต่ออินเทอร์เน็ตจึงเป็นงานที่ใช้ทรัพยากรมาก

ทั้งนี้สามารถศึกษาคำสั่ง AT และขั้นตอนการทำงานเพิ่มเติมได้ในภาพผนวกหรือตามแหล่งบรรณานุกรมที่แนบไว้

3.1.7 โหนด Logger

โหนด Logger ทำหน้าที่บันทึกข้อมูลบางส่วนของระบบลงในการ์ดหน่วยความจำ

1) ฮาร์ดแวร์ของโหนด Logger สามารถเชื่อมต่อกับมัลติมีเดียการ์ดได้โดยตรงผ่านบัสสื่อสารข้อมูลอนุกรม SPI มีผังโครงสร้างของโหนดดังรูป 3.21



รูป 3.21 ผังโครงสร้างของโหนด Logger

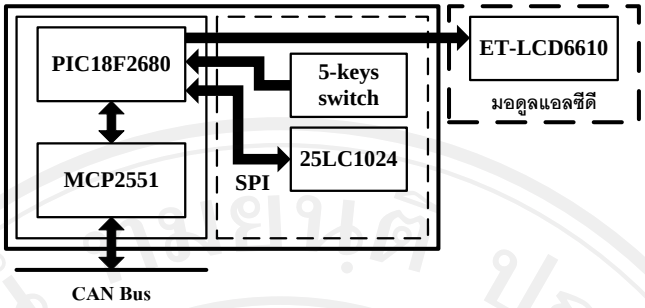
2) เฟิร์มแวร์ประยุกต์ของโหนด Logger ประกอบด้วยรูทีนย่อยที่สำคัญคือ

2.1) *รูทีน MMC Interface* มีหน้าที่อ่านและเขียนข้อมูลที่ต้องการในมัลติมีเดียการ์ด สำหรับการเขียนข้อมูลต้องสำรองทรัพยากรส่วนหน่วยความจำปริมาณหนึ่งเพื่อเรียงข้อมูลให้ได้ขนาดตามที่การ์ดกำหนด ก่อนเขียนข้อมูลลงการ์ดจริงๆ และในทำนองเดียวกันกับการเขียนการอ่านข้อมูลก็ต้องทำเช่นเดียวกันซึ่งปริมาณหน่วยความจำที่เตรียมไว้อย่างน้อยที่สุดต้องมากกว่า 512 ไบต์และมีจำนวนไม่ต่ำกว่า 2 ชุด (ตัวเลขขึ้นอยู่กับความจุของการ์ดที่ใช้)

3.1.8 โหนด HMI

โหนด HMI ทำหน้าที่แสดงผลข้อมูลบางส่วนของระบบรวมถึงทำหน้าที่รับอินพุตจากผู้ใช้ด้วยปุ่มกด ข้อมูลที่ถูกแสดงผลได้จากการสื่อสารด้วยข้อความ PDO จากโหนดต่างๆ

1) ฮาร์ดแวร์ของโหนด HMI ประกอบด้วยมอดูลแสดงผลแบบกราฟิกแอลซีดีรุ่น ET-LCD6610 ของบริษัทอีทีทีจำกัด [21] และส่วนรับอินพุตด้วยสวิตซ์ปุ่มกดจำนวน 5 ตัว มีผังโครงสร้างของโหนดดังรูป 3.22



รูป 3.22 ผังโครงสร้างของโหนด HMI

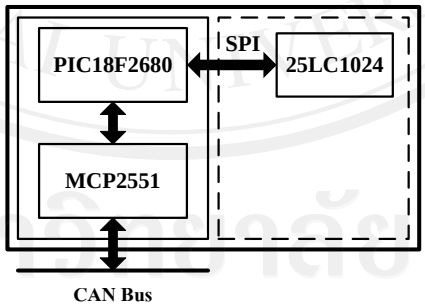
2) เฟิร์มแวร์ส่วนประยุกต์ของโหนด HMI มีรูทีนย่อยที่สำคัญคือ

2.1) *รูทีน GLCD Interface* ทำหน้าที่ควบคุมการแสดงผลของมอดูล ET-LCD6610 ซึ่งได้พัฒนาการแสดงผลตัวอักษรภาษาอังกฤษจากฟอนท์ Tahoma โดยมีความสูงของตัวอักษรเท่ากับ 16 พิกเซลและสามารถแสดงผลข้อความได้ 8 บรรทัดต่อหนึ่งหน้าจอ

3.1.9 โหนด Calculator

โหนด Calculator ทำหน้าที่ตัดสินใจเพื่ออนุญาตให้โหนดอื่นๆ ทำฟังก์ชันงานพิเศษเช่น ตัดสินใจเมื่อรถยนต์วิ่งครบระยะทางในการบำรุงรถยนต์แล้วส่งสัญญาณให้โหนด Master และโหนด HMI เพื่อแจ้งเตือนแก่ผู้ใช้งานเป็นต้น

1) ฮาร์ดแวร์ของโหนด Calculator ประกอบด้วยอีอีพรอมเบอร์ 25LC1024 มีผังโครงสร้างของโหนดดังรูป 3.23



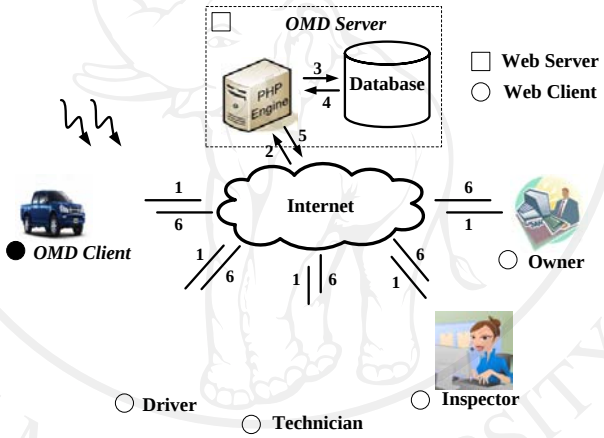
รูป 3.23 ผังโครงสร้างของโหนด Calculator

2) เฟิร์มแวร์ส่วนประยุกต์ของโหนด Calculator มีรูทีนที่สำคัญคือ

2.1) *รูทีน CAL* ทำหน้าที่เปรียบเทียบข้อความหรือเหตุการณ์ตามที่ต้องการ (เท่ากับ ไม่เท่ากับ มากกว่าและน้อยกว่า เป็นต้น)

3.2 การออกแบบส่วน OMD Server

ระบบ OMD ส่วน Server ประกอบด้วยเว็บเพจที่ใช้สำหรับแสดงเนื้อหาและเว็บเพจสำหรับประมวลผล รวมถึงเว็บเพจสำหรับอ่านและบันทึกฐานข้อมูล ในงานวิจัยนี้ได้เลือกใช้ภาษา PHP [11] และโปรแกรมฐานข้อมูล MySQL [22] ในการพัฒนาเว็บเพจดังกล่าว โดยใช้หลักการของ Client-Server (ผู้ขอใช้บริการ-ผู้ให้บริการ) เป็นแนวทางในการพัฒนาทำให้สามารถรองรับการร้องขอข้อมูลพร้อมๆ กันจากหลายๆ เว็บไคลเอนท์ ทั้งนี้จากจุดเด่นของเว็บเพจที่พัฒนาด้วยภาษา PHP ทำให้เนื้อหาของหน้าเว็บเพจมีการเปลี่ยนแปลงไปตามเงื่อนไขการร้องขอ ดังนั้นผลลัพธ์ที่ได้สำหรับแต่ละเว็บไคลเอนท์จึงมีความแตกต่างกัน ส่งผลให้สามารถกำหนดระดับและประเภทของผู้ใช้งานได้เช่น ช่าง พนักงาน ผู้ตรวจสอบ เจ้าของกิจการและผู้ใช้รถยนต์ เป็นต้น รูป 3.24 แสดงตัวอย่างลำดับขั้นตอนการเรียกดูข้อมูลของรถยนต์จากผู้ใช้งานประเภทต่างๆ



รูป 3.24 ภาพการทำงานของ OMD Server

นอกจากเว็บเพจสำหรับใช้แสดงผลข้อมูลต่างๆ ของรถยนต์แล้ว ในงานวิจัยนี้ต้องมีเว็บเพจสำหรับรับและประมวลผลข้อมูลที่มาจากระบบ OMD Client ด้วย โดยได้เลือกใช้การส่งข้อมูลจาก OMD Client มายัง OMD Server ผ่านทางวิธี GET ของโพรโทคอล HTTP ซึ่งอาศัยการส่งข้อมูลที่ต้องการเป็นอักขระต่อท้ายมากับ URL ของเว็บเพจที่กำหนดไว้ (กล่าวถึงในหัวข้อ 2.6) ข้อมูลที่ส่งจาก OMD Client แบ่งออกเป็น 4 กลุ่มใหญ่คือ (ก) ชุดข้อมูลตำแหน่ง (ข) ชุดข้อมูลสถานะ (ค) ชุดข้อมูลรหัสปัญหาที่ตรวจพบ และ (ง) เวลาของ OMD Client ขณะสุ่มข้อมูล มีรูปแบบการจัดข้อมูลดังตาราง 3.6 โดยทุกกลุ่มข้อมูลใช้อักขระ “&” ในการเชื่อมต่อกัน เมื่อรวมทุกชุดข้อมูลและ URL ของเว็บเพจจึงมีรูปแบบการส่งข้อมูลให้เว็บเพจดังตัวอย่างข้างล่างนี้

www.yourweb.com/querypage.php?pt=DBE7268E455C&st=04-45,05-53,...49-2F,&dtc=02,0102,0107,&cts=004214290311

ตาราง 3.6 รายละเอียดข้อมูลที่ส่งจาก OMD Client

ชื่อชุดข้อมูล	รายละเอียด
ชุดข้อมูลตำแหน่ง	• ข้อมูลเลขฐานสิบหกในรูปแบบอักขระขนาด 12 ตัว • ชุดข้อมูลตำแหน่งได้จากการเข้ารหัสดังตาราง 3.5 • ใช้ตัวอักษร “pt=” นำหน้าชุดข้อมูล • ตัวอย่าง “pt=DBE7268E455C”
ชุดข้อมูลสถานะ	• ข้อมูลเลขฐานสิบหกในรูปแบบอักขระขนาด 128 ตัว • ชุดข้อมูลสถานะจากการร้อง PID 15 ชนิด มีรูปแบบคือ PID1-value1,PID2-value2,…PID15-value15, • ใช้ตัวอักษระ “st=” นำหน้าชุดข้อมูล • ตัวอย่างเช่น st=04-45,05-53,0B-1B,0C-1340,0D-00,”
ชุดข้อมูลรหัสปัญหาที่ตรวจพบ	• ข้อมูลเลขฐานสิบหกในรูปแบบอักขระขนาด 128 ตัว • ชุดข้อมูลรหัสปัญหาที่ตรวจพบ มีรูปแบบคือ no,code1,code2,…code(no), • ใช้ตัวอักษระ “dtc=” นำหน้าชุดข้อมูล • ตัวอย่างเช่น “dtc= 02,0102,0107,”
เวลาของ OMD Client ขณะสุ่มข้อมูล	• ข้อมูลเลขฐานสิบหกในรูปแบบอักขระขนาด 12 ตัว • ประกอบด้วยข้อมูล วินาที นาที ชั่วโมง วันที่ เดือน และปี ทั้งหมดอย่างละ 2 อักขระ • ใช้ตัวอักษร “cts=” นำหน้าชุดข้อมูล • ตัวอย่าง “cts=004214290311”