พาหนะขับเคลื่อนอัตโนมัติ

AUTONOMOUS VEHICLE

กันต์ ศิริงามเพ็ญ
คมสันติ์ เลิศบุญยพันธุ์
ภาณุวัฒน์ ปริญญาวิภาต
วีรยุทธ วชิรพรพงศา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2551

ปริญญานิพนธ์ปีการศึกษา 2551

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง พาหนะขับเคลื่อนอัตโนมัติ

AUTONOMOUS VEHICLE

ผู้จัดทำ

1. นายกันต์	ศิริงามเพ็ญ	รหัสนักศึกษา 48010036
2. นายคมสันติ์	เลิศบุญยพันธุ์	รหัสนักศึกษา 48010095
3. นายภาณุวัฒน์	ปริญญาวิภาต	รหัสนักศึกษา 48010673
4. นายวีรยุทธ	วชิรพรพงศา	รหัสนักศึกษา 48010852

อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์อภิเนตร อูนากูล)

พาหนะขับเคลื่อนอัตโนมัติ

นายกันต์ ศิริงามเพ็ญ 48010036

นายคมสันติ์ เลิศบุญยพันธุ์ 48010095

นายภาณุวัฒน์ ปริญญาวิภาต 48010673

นายวีรยุทธ วชิรพรพงศา 48010852

ผศ.อภิเนตร อูนากูล อาจารย์ที่ปรึกษา

ปีการศึกษา 2551

บทคัดย่อ

โครงงานนี้เป็นโครงงานที่พัฒนาให้ยานพาหนะที่มนุษย์เราใช้กันในปัจจุบันให้สามารถทำงานได้อย่างอัตโนมัติ โดยในโครงงานนี้ได้นำเอาระบบสมองกลฝังตัว (Embedded System) เข้ามาใช้ในการควบคุมการทำงานในส่วนประมวลผลกลางและขั้นตอนของการประมวลผลภาพ (Image Processing) โดยที่ยานพาหนะนั้นสามารถวิ่งบนถนนไปยังเป้าหมายที่กำหนดไว้ สามารถวิเคราะห์ป้ายสัญญาณจราจรได้อย่างถูกต้อง รวมทั้งสามารถวิเคราะห์เพื่อหาเส้นทางที่เหมาะสมที่จะไปยังจุดหมายจากแผนที่ที่มีอยู่ภายในได้ โดยโครงงานนี้จะใช้เอ็มเบ็ดเด็ดลินุกซ์บอร์ด (Embedded Linux Board) เป็นส่วนกลางที่คอยทำหน้าที่ติดต่อกับส่วนรับรู้ (Sensor) ต่างๆของยานพาหนะ ส่วนกลางนั้นจะประกอบด้วยส่วนโรบอทอินเตอร์เฟส (robot interface) และส่วนตัดสินใจ โดยส่วนโรบอทอินเตอร์เฟส นั้นโครงงานนี้ได้เลือกใช้เพลเยอร์ (Player) เพื่อให้สามารถเข้ากับระบบตัดสินใจอื่นๆที่เขียนขึ้นมาภายหลังได้ นอกจากส่วนกลางแล้วยังมีส่วนของการประมวลผลภาพ ซึ่งจะใช้เอ็มเบ็ดเด็ดพีซี (Embedded PC) ทำการประมวลผลภาพเพื่อหาส่วนของถนน หาป้ายสัญญาณจราจรแล้วนำป้ายสัญญาณที่พบไปวิเคราะห์ให้ทราบว่าเป็นสัญญาณใด แล้วจึงส่งข้อมูลที่วิเคราะห์มาได้ไปให้ยังส่วนกลางเพื่อประกอบการตัดสินใจ เมื่อส่วนกลางได้รับข้อมูลจากส่วนรับรู้เหล่านี้แล้ว ก็จะนำไปวิเคราะห์หาการกระทำที่เหมาะสมที่สุดเพื่อให้บรรลุเป้าหมาย จากนั้นจึงสั่งงานให้ยานพาหนะทำงานตามการกระทำนั้น

AUTONOMOUS VEHICLE

Mr. Khan	Siringampen	48010036
Mr. Komson	Lertboonyapun	48010095
Mr. Panuvat	Parinyavipart	48010673
Mr. Weerayoot	Wachirapornpongsa	48010852
Asst. Prof. Apinetr	Unakul	Advisor

Academic Year 2008

ABSTRACT

The project objective is to build a vehicle that is able to go to destination automatically. This project is using embedded system to control all sensor and actuator, make decision and process image. This vehicle can route to the destination by using internal map, recognizing traffic sign, analyzing road lane and run to the destination based on the route that calculated before. The central control part is run on embedded Linux board all data from sensors will send to this central control. There are two modules in central control part, robot interfaces and decision system. This robot interfaces are base on "Player" yield portable and compatible with other decision system. Furthermore there is image processing system run on Embedded PC for process image that acquits by camera and send the result to central control. The central control will use all data from other modules to control vehicle to reach the destination.

กิตติกรรมประกาศ

โครงงานนี้ไม่อาจสำเร็จได้ด้วยดี หากขาดการช่วยเหลือ สนับสนุนและให้คำปรึกษาจาก ผศ. อภิเนตร อูนากูล ซึ่งเป็นผู้ควบคุมดูแลในการทำโครงงานในครั้งนี้ ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์ของอาจารย์ และขอกราบขอบพระคุณท่านเป็นอย่างสูง

ขอกราบขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกๆท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า รวมไปถึงบุคลากรทุกท่านที่คอยอำนวยความสะดวกและประสานงานในเรื่องต่างๆ ขอขอบคุณ ดร. วัชระ ฉัตรวิริยะ ที่ให้คำปรึกษาทางด้านต่างๆ ขอขอบคุณอาจารย์ศมิทธิ์ เอมสมบัติ ที่ให้คำปรึกษาทางด้านฮาร์ดแวร์ ขอขอบคุณ ผศ. ดร. อรฉัตร จิตต์โสภักตร์ และพี่ก้องเกียรติ เรืองไทย ที่ให้คำปรึกษาทางด้านการประมวลผลภาพ ขอขอบคุณพี่ขวัญพงศ์ เมืองสมุทรนาวี พี่พิชิต รินทระ และพี่อภิรมณ์ บุญประสิทธิ์ ซึ่งเป็นพี่ๆ ห้อง ESL ที่ให้คำปรึกษาและช่วยเหลือทางด้านฮาร์ดแวร์ พี่นิพิฐพนธ์ ชูวงษ์ธนนาถ และพี่สุเมธ แซ่ลี้ ที่บริษัท NorhTec ที่ให้คำปรึกษาและช่วยเหลือเกี่ยวกับเอ็มเบ็ดเด็ดพีซี และขอขอบคุณพี่ๆ เพื่อนๆ ที่ห้องวิจัย ESL และห้องวิจัย การประมวลผลภาพ ที่อนุเคราะห์ให้คำแนะนำต่างๆมาโดยตลอด

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยให้กำลังใจและให้คำแนะนำที่ดีเสมอมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจ และให้การสนับสนุนช่วยเหลือในทุกๆเรื่อง คุณค่าและประโยชน์อันพึงมาจากรายงานเล่มนี้ ข้าพเจ้าขอมอบให้แก่ผู้มีพระคุณทุกท่าน

นายกันต์ ศิริงามเพ็ญ

นายคมสันติ์ เลิศบุญยพันธุ์

นายภาณุวัฒน์ ปริญญาวิภาต

นายวีรยุทธ วชิรพรพงศา

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

จากปัญหาการจราจรบนท้องถนนที่เกิดขึ้นในปัจจุบัน ส่วนใหญ่เกิดจากการขาดวินัย ความประมาทหรือสภาพร่างกายของผู้ขับขี่ยวดยานบนท้องถนน ส่งผลให้เกิดปัญหาการจราจรติดขัด อุบัติเหตุทางรถยนต์ รวมถึงการเลือกเส้นทางเดินทางที่ไม่เหมาะสม ทำให้สิ้นเปลืองน้ำมัน โครงงานนี้มุ่งหวังที่จะพัฒนารถยนต์ที่ขับเคลื่อนอย่างอัตโนมัติ โดยการตัดสินใจบนข้อมูลที่ได้จากเซ็นเซอร์ต่างๆ ผู้ใช้งานเพียงแค่ระบุจุดหมายปลายทางที่จะไป จากนั้นระบบจะคำนวณหาเส้นทางที่ดีที่สุด และขับเคลื่อนไปยังจุดหมายปลายทางที่ต้องการอย่างประหยัดและปลอดภัย

1.2 วัตถุประสงค์ของโครงงาน

- เพื่อศึกษาทฤษฎีและหลักการต่างๆ เกี่ยวกับการควบคุมรถยนต์อย่างอัตโนมัติ
- เพื่อศึกษาทฤษฎีและหลักการต่างๆ เกี่ยวกับการประมวลผลภาพเพื่อใช้ในการขับเคลื่อนรถยนต์อย่างอัตโนมัติ
- เพื่อสร้างระบบขับเคลื่อนรถยนต์อัตโนมัติ โดยใช้การประมวลผลภาพ การตีความจากเซ็นเซอร์ต่างๆ และแผนที่ที่บรรจุอยู่ภายใน

1.3 ประโยชน์ที่คาดว่าจะได้รับ

- มีความรู้ความเข้าใจเกี่ยวกับทฤษฎีและหลักการต่างๆ เกี่ยวกับการควบคุมรถยนต์อย่างอัตโนมัติ
- มีความรู้ความเข้าใจเกี่ยวกับวิธีการประมวลผลและวิเคราะห์ข้อมูลภาพที่เกี่ยวข้องกับการขับเคลื่อนของรถยนต์
- สามารถสร้างระบบควบคุมและขับเคลื่อนรถยนต์อย่างอัตโนมัติ

1.4 ขอบเขตของโครงงาน

รถสามารถขับเคลื่อนบนถนนได้อย่างอัตโนมัติ สามารถหลบหลีกสิ่งกีดขวางได้ โดยมีข้อจำกัดต่างๆดังนี้

- มีสภาพอากาศที่ดี และเป็นเวลากลางวันหรือตอนเย็นที่ไม่มีแดด
- สีของถนนมีความแตกต่างกับขอบถนน
- ต้องเป็นถนนที่ไม่มีน้ำขังหรือมีสิ่งแปลกปลอมบนถนน
- สภาพถนนค่อนข้างเรียบ ไม่เป็นหลุมเป็นบ่อ
- ด้านข้างถนนเป็นที่โล่ง (ไม่มีสิ่งปลูกสร้างและต้นไม้)
- สิ่งกีดขวางบนถนนต้องไม่มีการเคลื่อนที่
- ป้ายจราจรมีลักษณะที่สมบูรณ์ (ป้ายไม่เอียงหรือกลับด้าน)
- ป้ายจราจรไม่ถูกบังจากสิ่งแวดล้อม
- มีแผนที่ที่ถูกต้องบรรจุอยู่ภายในตัวรถ

1.5 ขั้นตอนการดำเนินงาน

ในการทำโครงงานนี้ มีขั้นตอนการดำเนินงานดังนี้

- ศึกษางานวิจัยที่เกี่ยวข้อง
- กำหนดขอบเขตของโครงงาน
- ศึกษาทฤษฎีต่างๆที่เกี่ยวข้อง
- ออกแบบระบบทั้งหมด ได้แก่ส่วนฮาร์ดแวร์ ซอร์ฟแวร์และการประมวลผลภาพ
- สร้างระบบตามที่ได้ออกแบบไว้
- ทำการทดสอบและปรับปรุงระบบ
- สรุปผลการดำเนินงาน และจัดทำเอกสาร

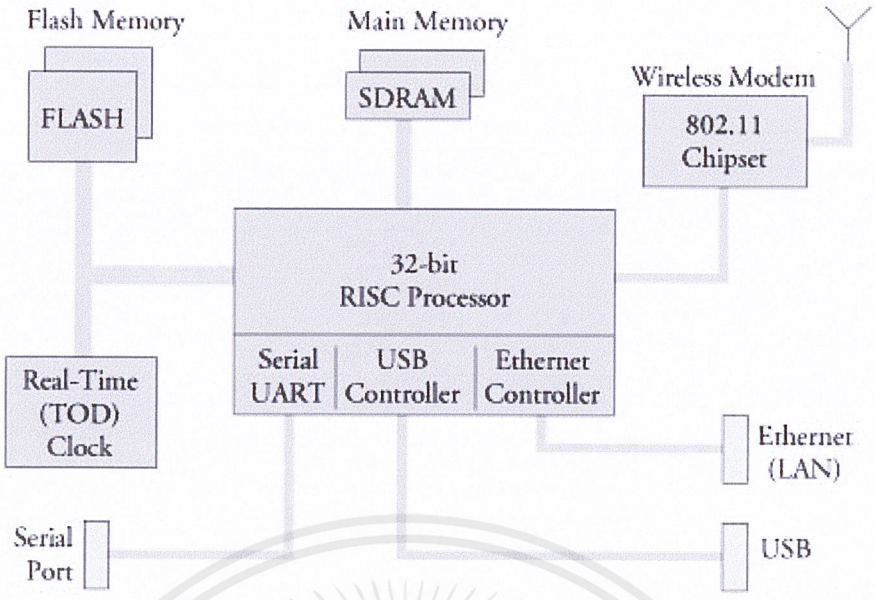
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ระบบสมองกลฝังตัว (Embedded System)

ระบบสมองกลฝังตัวเป็นระบบอิเล็กทรอนิกส์ที่ใช้สำหรับงานควบคุมรวมถึงการแสดงผลการทำงานต่างๆ โดยที่ระบบเหล่านี้ถูกใช้เป็นส่วนหนึ่งของระบบและอุปกรณ์ควบคุมเครื่องมือเครื่องจักรต่างๆ การที่ใช้คำว่า "ระบบสมองกลฝังตัว" เนื่องจากระบบเหล่านี้เป็นส่วนหนึ่งของระบบใหญ่ ในหลายกรณีที่ผู้ใช้ทั่วไปอาจไม่ทราบว่าอุปกรณ์ควบคุม เครื่องมือ เครื่องจักรรวมถึงระบบอื่นๆที่ใช้งานเป็นประจำเหล่านั้นเป็นระบบสมองกลฝังตัว ระบบสมองกลฝังตัวนี้แม้ไม่ใช่เครื่องคอมพิวเตอร์ แต่ก็มีระบบคอมพิวเตอร์อยู่ภายใน อาจจะเป็นเพียงไมโครโพรเซสเซอร์ (Microprocessor) หรือโพรเซสเซอร์ (Processor) ที่ประกอบด้วย ชิป (Chip) ที่มีวงจรซับซ้อน โดยจะมีหลักการทำงานคือ มีสัญญาณข้อมูลเข้า (Input) จากอุปกรณ์ เซ็นเซอร์ (Sensor) เข้าสู่ระบบและมีสัญญาณผลลัพธ์ (Output) ของระบบไปควบคุมบังคับอุปกรณ์ต่างๆภายในระบบ เช่นมอเตอร์ต่างๆ ระบบฝังตัวนั้นมีรูปร่าง ขนาดที่หลากหลาย จากขนาดที่ใหญ่สุดเช่น มัลติเพิล-แรค ดาต้าสตอเรจ (multiple-rack data storage) ไปจนถึงขนาดเล็กๆ อาทิเช่น เอ็มพีสาม (MP3) ส่วนบุคคล คุณสมบัติของระบบฝังตัวนั้นมีดังต่อไปนี้

- มีเครื่องประมวลผลอยู่ภายใน
- ระบบฝังตัวส่วนใหญ่ต้องทำงานทันต่อการใช้งาน
- ถูกออกแบบเฉพาะเจาะจงใน แอปพลิเคชั่น หรือวัตถุประสงค์
- มีขนาดทรัพยากรที่จำกัด อาทิเช่น หน่วยความจำ(ROM, RAM) อินพุทเอาต์พุท
- อาจจะมีพลังงานที่จำกัด
- มักจะมุ่งที่งานนั้นๆ โดยที่ปราศจากมนุษย์เข้ามาเกี่ยวข้อง

ตัวอย่างเช่นวายเลสแอ็คเซสพอยท์ (wireless access point) ระบบนี้มี 32 บิท ริสโพรเซสเซอร์ (RISC processor) เป็นหัวใจของระบบ, มีหน่วยความจำแฟลซ (flash memory) ใช้ในการเก็บข้อมูล มีหน่วยความจำหลักเป็นเอสดีแรม (SDRAM),โมดูล เรียว ไทม์ ค็อก (real-time clock), อีเทอร์เน็ท (Ethernet) , วายเลส โมเด็ม (Wirless modem), พอร์ตอนุกรม (Serial port) และ ยูเอสบี อินเทอร์เฟส (USB interface)



รูปที่ 2.1 ตัวอย่างระบบสมองกลฝังตัว

2.2 เอ็มเบ็ดเด็ดลินุกซ์ (Embedded Linux)

เอ็มเบ็ดเด็ดลินุกซ์เป็นการนำระบบปฏิบัติการลินุกซ์มาใช้งานในระบบสมองกลฝังตัว ลินุกซ์นั้นปัจจุบันเป็นส่วนหนึ่งของผลิตภัณฑ์ต่างๆทั่วโลก เช่นโทรศัพท์เคลื่อนที่ ปริ้นเตอร์ สวิตซ์ (switch) และเร้าเตอร์ (router) และผลิตภัณฑ์อื่นๆอีกหลายตัว ซึ่งสาเหตุที่ลินุกซ์ได้รับความนิยมในระบบสมองกลฝังตัวก็เพราะเหตุผลต่างๆดังนี้

- ลินุกซ์มีประสิทธิภาพสูง เสถียร ซึ่งเป็นคุณสมบัติที่จะนำมาเป็นระบบปฏิบัติการ
- ลินุกซ์เป็นฟรีแวร์
- ลินุกซ์สนับสนุนแอปพลิเคชั่น และโปโตรคอลเน็ทเวิร์คที่หลากหลาย
- มีโอเพ็นซอสซอฟต์แวร์ (open source software) มากมายในปัจจุบัน ที่ทำงานอยู่บนระบบปฏิบัติการลินุกซ์
- สามารถหาไดร์เวอร์ (driver) ของอุปกรณ์ต่างๆสำหรับระบบปฏิบัติการลินุกซ์ ได้ง่าย
- มีผู้ใช้งานจำนวนมาก ทำให้การพัฒนาและการแก้ไขปัญหาสามารถทำได้ง่าย

2.2.1 โครงสร้างทางสถาปัตยกรรมเคอร์เนลของลินุกซ์ (Linux kernel architecture)

สถาปัตยกรรมเคอร์เนลของลินุกซ์สามารถถูกแบ่งออกเป็นส่วนๆดังนี้

2.2.1.1 ชั้นฮาร์ดแวร์นามธรรม (Hardware abstraction layer)

ฮาร์ดแวร์นามธรรมเป็นส่วนที่ซ่อนความแตกต่างกันของฮาร์ดแวร์ ซึ่งทำโดยซอฟต์แวร์ ฮาร์ดแวร์นามธรรมจะทำให้ระบบปฏิบัติการและซอฟแวร์สามารถทำงานบนฮาร์ดแวร์ที่แตกต่างกันได้โดยที่ไม่ต้องเปลี่ยนแปลงใดๆทั้งสิ้น

2.2.1.2 หน่วยจัดการหน่วยความจำ (Memory manager)

หน่วยจัดการหน่วยความจำในลินุกซ์จะมีหน้าที่รับผิดชอบสำหรับการควบคุมการเข้าถึงทรัพยากรหน่วยความจำ หน่วยจัดการหน่วยความจำจะรับผิดชอบเตรียมการจัดหาหน่วยความจำที่เป็นไดนามิกให้กับเคอร์เนลซัปซิสเต็ม อาทิเช่นไดร์เวอร์ ไฟล์ซิสเต็ม และชั้นของเน็ทเวิร์ค หน่วยจัดการหน่วยความจำจะมีซอร์ฟแวร์ที่จำเป็นเพื่อจัดการหน่วยความจำเสมือนให้กับโปรแกรมแอปพลิเคชั่น

2.2.1.3 หน่วยจัดการตารางเวลา (Scheduler)

มีหน้าที่ในการจัดลำดับการทำงานและระยะเวลาในการทำงานของเธรด (thread) ทั้งหมดที่มีอยู่ ซึ่งจะจัดลำดับการทำงานและระยะเวลาการทำงานตามลำดับความสำคัญของเธรดนั้นๆ

2.2.1.4 ไฟล์ซิสเต็ม (File system)

บนลินุกซ์ ไฟล์ซิสเต็มทั้งหลายถูกจัดการโดยวีเอฟเอส (VFS : Virtual File System) ซึ่งวีเอฟเอส จะจัดการมุมมองข้อมูลที่ถูกเก็บไว้ในแต่ละอุปกรณ์บนระบบ สิ่งที่ทำคือทำการแยกมุมมองของผู้ใช้งานโดยใช้ซิสเต็มคอลล์ (system call) พื้นฐาน ลินุกซ์สนับสนุนไฟล์ซิสเต็มหลายชนิดเช่นแฟลซและรอมสำหรับระบบฝังตัว ที่สำคัญลินุกซ์สนับสนุนไฟล์ซิสเต็มเอ็นเอฟเอส (NFS) อีกด้วย ซึ่งทำให้สามารถเม้าท์ไฟล์ซิสเต็มที่อยู่บนเครื่องอื่นเข้ามาใช้งานได้ นอกจากนั้นลินุกซ์สนับสนุนเมโมรี่เบสไฟล์ซิสเต็ม (memory-base file system) ซึ่งใช้ในระบบฝังตัวอีกเช่นกัน

2.2.1.5 ระบบอินพุทเอาต์พุท (IO subsystem)

ระบบอินพุทเอาต์พุทบนลินุกซ์แบ่งเป็น 3 ประเภทดังต่อไปนี้

- ชาเล็กเตอร์ดีไวซ์ (Character devices) สนับสนุนดีไวซ์แบบลำดับ
- บล๊อกดีไวซ์ (Block devices) สนับสนุนดีไวซ์แบบแรนดอม บล๊อกดีไวซ์เป็นส่วนสำคัญในการสร้างไฟล์ซิสเต็ม
- เน็ทเวิร์คดีไวซ์ (Network devices) สนับสนุนการเชื่อมต่อผ่านเน็ตเวิร์ค

2.2.1.6 ระบบเครือข่าย (Networking subsystem)

ลินุกซ์มีการสนับสนุนโปรโตคอลระบบเครือข่ายที่หลากหลาย ซึ่งแสดงดังตารางที่ 2.1 โปรโตคอลระบบเครือข่ายที่ลินุกซ์รองรับ

ตารางที่ 2.1 โปรโตคอลระบบเครือข่ายที่ลินุกซ์รองรับ

	Kernel Availability		
Feature	*2.2*	*2.4*	*2.6*
Layer 2			
Support for bridging	Yes	Yes	Yes
X.25	Yes	Yes	Yes
LAPB	Experimental	Yes	Yes
PPP	Yes	Yes	Yes
SLIP	Yes	Yes	Yes
Ethernet	Yes	Yes	Yes
ATM	No	Yes	Yes
Bluetooth	No	Yes	Yes
Layer 3			
IPV4	Yes	Yes	Yes
IPV6	No	Yes	Yes
IP forwarding	Yes	Yes	Yes
IP multicasting	Yes	Yes	Yes
IP firewalling	Yes	Yes	Yes
IP tunneling	Yes	Yes	Yes
ICMP	Yes	Yes	Yes
ARP	Yes	Yes	Yes
NAT	Yes	Yes	Yes
IPSEC	No	No	Yes
Layer 4 (and above)			
UDP and TCP	Yes	Yes	Yes
BOOTP/RARP/DHCP	Yes	Yes	Yes

2.2.1.7 ไอพีซี (IPC)

เป็นกระบวนการการติดต่อสื่อสารระหว่างโปรเซส (process) ด้วยกัน โดยโปรเซสนั้นจะอยู่บนเครื่องเดียวกันหรือคนละเครื่องก็ได้ ไอพีซีภายในลินุกซ์ประกอบด้วยซิกเนล (signal) ไปป์ (pipe) ซ็อกเก็ต (socket) การแชร์หน่วยความจำ แมสเซสคิว (message queues) เซมาฟอร์

(semaphores) และอื่นๆ ซึ่งเคอร์เนล 2.6 ได้เพิ่มสนับสนุนแมสเซสคิวแบบโพสซิกส์ (POSIX) เข้าไป

2.2.2 ยูสเซอร์สเปซ (User Space)

ยูสเซอร์สเปซเป็นส่วนที่ยูสเซอร์แอปพลิเคชั่นจะใช้ทำงาน แอปพลิเคชั่นโปรแกรมจะเก็บเป็นอิมเมจอยู่บนไฟล์ซิสเต็ม เมื่อแอปพลิเคชั่นต้องการขึ้นมาทำงาน อิมเมจจะถูกโหลดลงสู่หน่วยความจำ จากนั้นเคอเนลจะสร้างโปรเซส และสร้างหน่วยความจำเสมือนให้กับโปรเซสนั้นๆ หน่วยความจำเสมือนจะทำให้แต่ละโพรเซสมีพื้นที่ตำแหน่ง นอกจากนี้เคอเนลยังอนุญาติคุณสมบัติพิเศษอาทิเช่น แชร์ไลบรารี่ (shared library) ยูสเซอร์แอปพลิเคชั่นจะติดต่อกับเคอเนลผ่านทางซิสเต็มคอลล์ ซึ่งซิสเต็มคอลล์เป็นการส่งให้เคอร์เนลทำงาน ดังนั้นเคอร์เนลสามารถประมวลผลบริการต่างๆ ในนามของแอปพลิเคชั่นที่เรียกใช้

2.2.3 ระบบไฟล์ (File System)

ระบบไฟล์ที่เป็นที่นิยมในเอ็มเบ็ดเด็ดซิสเต็ม (Embedded system) มีดังนี้

2.2.3.1 แรมดิสค์ (Ramdisk)

แรมดิสค์เป็นทางเลือกหนึ่งของระบบไฟล์ของลินุกซ์ โดยการจำลองให้มีดิสค์โดยใช้หน่วยความจำ โดยส่วนใหญ่แล้วจะใช้แรมดิสค์เมื่อไม่มีอุปกรณ์ความจำอยู่ในระบบ อาทิเช่น ฮาร์ดดิสค์ หรือ แฟลซ สำหรับเก็บรูทไฟล์ซิสเต็ม แต่ในบางครั้งก็ใช้แรมดิสค์แทนแฟลซเพื่อยืดอายุการใช้งานหน่วยความจำแฟลซให้นานขึ้น แรมดิสค์จะทำงานโดยการโหลดไฟล์ซิสเต็มลงสู่หน่วยความจำและใช้งานรูทไฟล์ซิสเต็มที่อยู่ในหน่วยความจำนั้น

2.2.3.2 แรมเอ็ฟเอส (RAMFS)

แรมเอ็ฟเอสเป็นระบบไฟล์ที่คล้ายกับแรมดิสค์ ต่างกันตรงที่แรมเอ็ฟเอสจะเป็นไฟล์ซิสเต็มจริงๆเลย ส่วนแรมดิสค์จะเป็นบล๊อกดีไวซ์ ทำให้แรมดิสค์จะถูกจำกัดขนาด ไม่สามารถเพิ่มขยายได้และหากว่าใช้พื้นที่ในแรมดิสค์ไม่หมด หน่วยความจำที่เหลือไม่สามารถนำไปใช้งานอื่นได้ สำหรับแรมเอ็ฟเอสนั้นจะไม่ถูกจำกัดขนาด ถ้าหากว่าต้องการขยายพื้นที่ ก็จะจองหน่วยความจำเพิ่ม ถ้าหากว่ามีพื้นที่ไม่ได้ใช้งานก็จะคืนหน่วยความจำให้กับระบบ ที่สำคัญแรมเอ็ฟเอสจะไม่ถูกแคช (cache) เพราะว่าแรมเอ็ฟเอสไม่ใช่บล๊อกดีไวซ์

2.2.3.3 คอมเพรซแรมเอ็ฟเอส (CRAMFS(Compressed RAM File System))

เป็นระบบไฟล์ซิสเต็มที่ใช้งานสำหรับการอ่านอย่างเดียว ส่วนใหญ่จะใช้งานในเอ็มเบ็ดเด็ดซิสเต็ม คอมเพรซแรมเอ็ฟเอสจะมีการบีบอัดข้อมูลด้วย การบีบอัดจะทำเป็นบล๊อคๆ เพื่อให้สามารถเข้าถึงข้อมูลแบบสุ่มได้

2.2.3.4 เจเอ็ฟเอ็ฟเอส (Journaling Flash File System-JFFS และ JFFS2)

เจเอ็ฟเอ็ฟเอส เป็นระบบไฟล์ซิสเต็มแบบล็อก (log-structure file system) สำหรับใช้งานบนหน่วยความจำประเภทนอแฟลช (NOR flash memory) สาเหตุที่เป็นระบบไฟล์ซิสเต็มแบบล็อก ก็เพราะหน่วยความจำแบบแฟลช สามารถลบบล๊อกได้เป็นจำนวนครั้งที่จำกัด หากเราแก้ไขแต่บล๊อกเดิมๆอยู่บ่อยๆ ก็จะทำให้อายุการใช้งานสั้น สำหรับเจเอ็ฟเอ็ฟเอส 2 ได้เพิ่มคุณสมบัติการบีบอัดข้อมูล และสามารถใช้งานบนหน่วยความจำประเภทแนนแฟลช (NAND flash memory) ได้

2.2.3.5 เอ็นเอ็ฟเอส (NFS-Network File System)

เอ็นเอ็ฟเอส เป็นระบบไฟล์ซิสเต็มผ่านระบบเครือข่าย ในช่วงของการพัฒนานักพัฒนามักจะใช้เอ็นเอ็ฟเอสเนื่องจาก สะดวกในการพัฒนาและแก้ไขไฟล์ภายในไฟล์ซิสเต็ม ขนาดของเอ็นเอ็ฟเอสจะขึ้นอยู่กับ สตอเรจ (storage) ที่อยู่บนรีโมท (remote) เซิร์ฟเวอร์ ทำให้สามารถมีขนาดไฟล์ซิสเต็มที่ใหญ่ในช่วงของการพัฒนาได้ ที่สำคัญคือ เอ็นเอ็ฟเอสสามารถถูกนำมาใช้งานเป็นระบบไฟล์รูท (root file system) ของระบบได้อีกด้วย

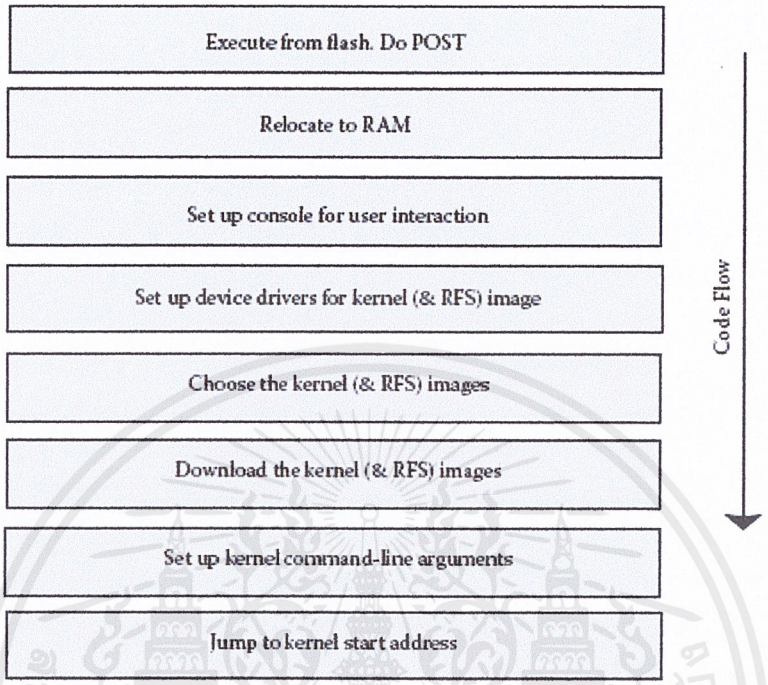
2.2.4 การเริ่มต้นการทำงานของ เอ็มเบดเดด ลินุกซ์ (Embedded Linux startup process)

หลังจากที่เปิดให้บอร์ดเอ็มเบดเด็ดลินุกซ์ทำงาน บูทโหลดเดอร์จะเป็นตัวที่ควบคุมการทำงานของโพรเซสเซอร์ ซึ่งบูทโหลดเดอร์จะกำหนดค่าเริ่มต้นและตรวจสอบฮาร์ดแวร์ ซึ่งประกอบไปด้วยโพรเซสเซอร์ หน่วยความจำ ยูอาร์ท (UART) และอีเทอร์เน็ทคอนโทรเลอร์ ผลลัพธ์ของการกำหนดค่าต่างๆจะถูกแสดงออกมาทางพอร์ตอนุกรม โดยจะมีขั้นตอนการทำงานดังนี้

- ตรวจสอบฮาร์ดแวร์ (POST (Power ON Self-Test))
- หาขนาดหน่วยความจำของบอร์ด
- เซ็ตพอร์ตอนุกรมสำหรับบูทคอนโซล์

หลังจากนั้นก็จะทำการเลือกเคอร์เนลอิมเมจ (kernel image) ตามที่ผู้ใช้งานตั้งค่าไว้ ซึ่งอาจจะอยู่บนหน่วยความจำแฟลช (flash memory) หรืออยู่บนระบบเครือข่าย ถ้าหากว่าเคอร์เนลอิมเมจอยู่บนระบบเครือข่ายก็จะต้องทำการดาวน์โหลดมาก่อน และอาจจะมีการคลายการบีบอัดด้วยถ้า

หากว่าเคอร์เนลอิมเมจนั้นถูกบีบอัดมา หลังจากขั้นตอนนี้จะได้เคอร์เนลอิมเมจมาอยู่บนหน่วยความจำ และจะส่งการทำงานให้กับเคอร์เนลเพื่อให้เคอร์เนลเริ่มต้นระบบต่อไป



รูปที่ 2.2 การเริ่มต้นการทำงานของเอ็มเบดเดดลินุกซ์

เมื่อลินุกซ์เคอร์เนลทำงานลินุกซ์เคอร์เนลจะทำการกำหนดค่าเริ่มต้น ทำการเม้าท์รูทไฟล์ซิสเต็ม และจะเรียกโปรแกรมแอปพลิเคชั่นขึ้นมาทำงาน ในที่นี้ก็คืออินนิท (init) เคอร์เนลล์จะรันอินนิทอยู่ในส่วนของยูซเซอร์สเปซ หรือยูซเซอร์คอนเท็กท์ จากนั้นอินนิทก็จะทำการเรียกโปรแกรมอื่นๆขึ้นมาทำงานตามที่ผู้ใช้ได้กำหนดไว้

2.3 เพลเยอร์โปรเจค (Player Project)

เพลเยอร์โปรเจคเป็นโครงงานที่สร้างซอฟต์แวร์ที่ฟรีสำหรับการทำงานและวิจัยเกี่ยวกับหุ่นยนต์และระบบเซ็นเซอร์ (sensor) ซึ่งภายในโปรเจคประกอบไปด้วยเพลเยอร์เน็ตเวิร์กเซอร์เวอร์ (Player network server), สเตจ (Stage) และกาซีโบ้ (Gazebo) โดยสเตจและกาซีโบ้ เป็นซอฟต์แวร์สำหรับจำลองการทำงานของหุ่นยนต์ และสภาพแวดล้อมภายนอก

เพลเยอร์เน็ตเวิร์กเซอร์เวอร์ สเตจ และกาซีโบ้ นั้นทำงานบนระบบปฏิบัติการที่รองรับมาตรฐานโพสซิกส์ (POSIX) เช่น ลินุกซ์ (Linux), แมคโอเอสเอ็กซ์ (Mac OS X), โซลาลิส (Solaris)

และระบบปฏิบัติการประเภท บีเอสดี (BSD (Berkeley Software Distribution)) เช่น ฟรีบีเอสดี (FreeBSD), โอเพนบีเอสดี (OpenBSD), เน็ตบีเอสดี (NetBSD) โปรเจคนี้มีลิขสิทธิ์แบบ "GNU General Public License" ส่วนเอกสารทั้งหมดของโปรเจ็คนี้มีลิขสิทธิ์แบบ "GNU Free Documentation License"

สำหรับรายละเอียดของซอฟต์แวร์ของเพลเยอร์โปรเจคมีดังนี้

2.3.1 เพลเยอร์ (Player)

เพลเยอร์จะทำหน้าที่เป็นเซอร์เวอร์ บนระบบเครือข่ายเพื่อรองรับการควบคุมหุ่นยนต์จากไคลเอนต์ (client) ผ่านทางทีซีพีซ๊อกเก็ต (TCP socket) โดยเพลเยอร์จะทำงานบนตัวหุ่นยนต์ ซึ่งเพลเยอร์ได้ทำการกำหนดมาตราฐานสำหรับการติดต่อเพื่อควบคุมตัวกระทำ (actuator) และเซ็นเซอร์ (sensor) ประเภทต่างๆผ่านทางเครือข่าย ทำให้ไคลเอนต์ที่เขียนขึ้นไม่ถูกผูกติดอยู่กับหุ่นยนต์ตัวใดตัวหนึ่ง

เพลเยอร์รองรับอุปกรณ์ประเภทต่างๆบนหุ่นยนต์มากมาย เช่นกล้องวีดีโอ, เครื่องรับสัญญาณจีพีเอส, เครื่องวัดระยะห่างโดยใช้คลื่นอัลตร้าโซนิค, ระบบนำทางในระนาบ 2 มิติ, ลำโพง สำหรับอุปกรณ์แต่ละประเภทนั้นเราสามารถพัฒนาไดรเวอร์ เพื่อให้ทำงานร่วมกับอุปกรณ์ประเภทนั้นๆที่เราใช้งานอยู่ได้ นอกจากนี้เรายังสามารถเพิ่มการรองรับอุปกรณ์ประเภทใหม่ๆลงในเพลเยอร์ได้ การกำหนดประเภทของอุปกรณ์และรูปแบบการเชื่อมต่อกับอุปกรณ์ประเภทนั้นๆไว้ ทำให้เราสามารถเปลี่ยนแปลงอุปกรณ์ภายในประเภทเดียวกันได้โดยที่ไม่ต้องแก้ไขส่วนไคลเอนต์เลย อีกนัยหนึ่งคือไคนเอนต์รู้แต่เพียงว่าหุ่นยนต์มีอุปกรณ์ชนิดนี้อยู่และสามารถใช้งานอุปกรณ์ชนิดนั้นได้อย่างไร โดยไม่จำเป็นต้องรู้ว่าจริงๆแล้วอุปกรณ์ชนิดนั้นเป็นอุปกรณ์ยี่ห้อใด ของผู้ผลิตรายใดและมีวิธีการติดต่อสื่อสารเพื่อควบคุมอย่างไร

ไคลเอนต์โปรแกรม (client program) ที่ทำการติดต่อกับเพลเยอร์นั้นสามารถจะเขียนด้วยภาษาอะไรก็ได้ที่รองรับทีซีพีซ๊อกเก็ต (TCP socket) และสามารถนำไปทำงานบนเครื่องไหนก็ได้ ที่มีการเชื่อมต่อกับหุ่นยนต์ผ่านระบบเครือข่าย ซึ่งอาจจะเป็นเครื่องเดียวกันกับเพลเยอร์ก็ได้ นอกจากนี้เพลเยอร์ได้เตรียมไคลเอนต์ไลบรารี่ (client library) สำหรับการพัฒนาไคลเอนต์บนภาษาซี (C), ซีพลัสพลัส (C++) และไพธอน (python) มาไว้ให้แล้ว

เพลเยอร์สามารถรองรับการเชื่อมต่อจากไคลเอนต์หลายๆตัวพร้อมกัน ตัวอย่างเช่น ไคลเอนต์ตัวแรกทำการอ่านค่าจากเซ็นเซอร์และควบคุมหุ่นยนต์ พร้อมๆกันกับไคลเอนต์ตัวที่สองที่ทำการติดตามตรวจสอบสถานะการทำงานของหุ่นยนต์

เพลเยอร์มีส่วนประกอบหลักๆ 5 ส่วนดังนี้

2.3.1.1 ลิบเพลเยอร์คอ (libplayercore)

ลิบเพลเยอร์คอประกอบไปด้วย ส่วนประกาศเอพีไอ (API) สำหรับการพัฒนาดีไวซ์ไดร์เวอร์ (device driver), ระบบเมสเซสคิว (message queue) สำหรับส่งข้อมูลระหว่างอุปกรณ์, ส่วนช่วยเหลือในการอ่านและแปลไฟล์คอนฟิกกูเรชั่น (configuration file) และส่วนของการโหลด (load) และเริ่มต้นการทำงานของไดร์เวอร์ (driver) ส่วนนี้เขียนด้วยภาษาซีพลัสพลัส (C++)

2.3.1.2 ลิบเพลเยอร์ไดร์เวอร์ (libplayerdrivers)

ลิบเพลเยอร์ไดร์เวอร์ประกอบไปด้วยไดร์เวอร์ที่มากับเพลเยอร์ซึ่งไดร์เวอร์ที่อยู่ในส่วนนี้สามารถเลือกนำมาใส่หรือเอาออกได้โดยการแก้ไขพารามิเตอร์ (parameter) ของคอนฟิกกูเรชั่นสคริป (configure script) ก่อนที่จะทำการบิ้ว (build) เพลเยอร์

2.3.1.3 ลิบเพลเยอร์เออเรอร์ (libplayererror)

ลิบเพลเยอร์เออเรอร์เป็นส่วนที่ไว้จัดการกับการรายงานความผิดพลาด (error) และดีบักล็อก (debug log) เพื่อให้สามารถควบคุมการแสดงผลของข้อความประเภทต่างๆในระดับต่างๆได้จากศูนย์กลาง

2.3.1.4 ลิบเพลเยอร์ทีซีพี (libplayertcp)

เป็นไลบรารี่ (library) สำหรับการรับส่งเมสเซส (message) จากเมสเซสคิว (message queue) ผ่านทางทีซีพีซ๊อกเก็ต (TCP socket)

2.3.1.5 ลิบเพลเยอร์เอ็กดีอาร์ (libplayerxdr)

เป็นไลบรารี่ (library) สำหรับการแปลงเมสเซส (message) ให้เป็นไปตามมาตรฐานเอ็กดีอา (XDR (eXternal Data Representation)) ก่อนการส่งออกไปยังเครือข่าย และแปลงเอ็กดีอา (XDR) กลับเป็นเมสเซส (message) เหมือนเดิมหลังจากรับข้อมูลผ่านทางเครือข่ายเข้ามา ซึ่งเอ็กดีอา (XDR) นั้นเป็นมาตรฐานสำหรับการเข้ารหัสข้อมูลประเภทต่างๆ เช่น int, float, char โดยรูปแบบมาตรฐานนี้จะเป็นอิสระจากสถาปัตยกรรมใดๆทั้งหมด เช่นลำดับของแต่ละไบท์ (byte) ในข้อมูล

2.3.2 สเตจ (Stage)

สเตจเป็นซอฟต์แวร์สำหรับการจำลองหุ่นยนต์ และวัตถุในสภาพแวดล้อมในลักษณะ 2 มิติ โดยส่วนใหญ่แล้ว สเตจจะถูกใช้เป็นปลั๊กอินมอดูล (plugin module) ของเพลเยอร์เพื่อทำการ

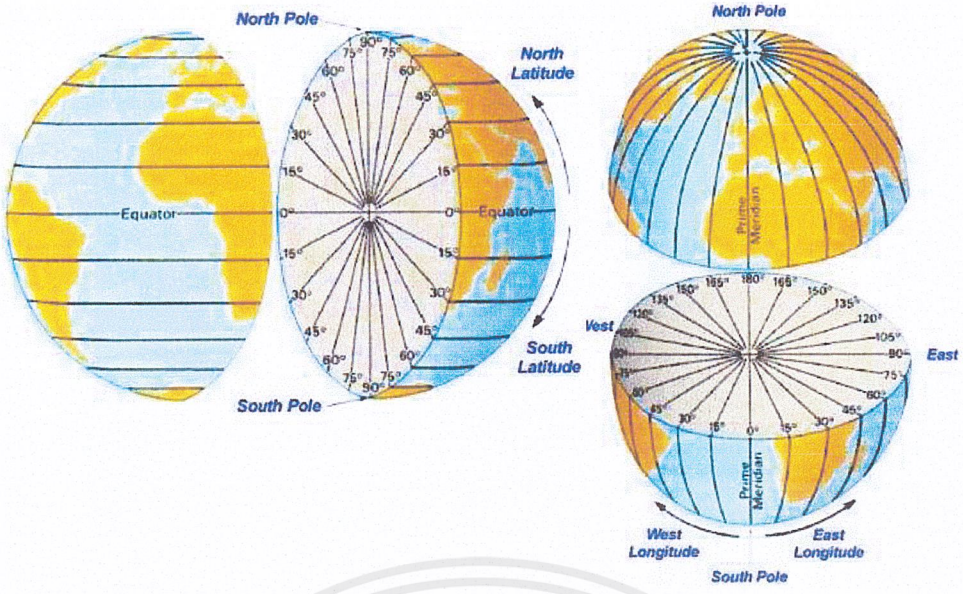
ทดลองไคลเอนต์โดยไม่ต้องติดต่อกับหุ่นยนต์จริงๆ ไคนเอนต์ที่ทดสอบกับ สเตจนั้นจะสามารถนำไปใช้งานกับหุ่นยนต์จริงๆ โดยแทบจะไม่ต้องเปลี่ยนแปลงแก้ไขอะไรเลย

2.3.3 กาซีโบ้ (Gazebo)

กาซีโบ้เป็นซอฟต์แวร์สำหรับจำลองหุ่นยนต์ และวัตถุต่างๆในสภาพแวดล้อมเช่นเดียวกันกับสเตจแต่จะจำลองในลักษณะ 3 มิติ โดยกาซีโบ้สามารถจำลองการป้อนกลับของเซ็นเซอร์ (sensor feed back) ที่สมจริงและสามารถจำลองการเปลี่ยนแปลงต่างๆของวัตถุโดยใช้การคำนวณทางฟิสิกส์

2.4 ระบบพิกัดแบบทรงกลม

ระบบพิกัดแบบทรงกลม (Spherical Coordinate System) เป็นระบบค่าพิกัดที่อ้างอิงเส้นรุ้ง (Latitude) และเส้นแวง (Longitude) โดยตั้งอยู่บนพื้นฐานว่าโลกมีลักษณะกลม ซึ่งเป็นภาพ 3 มิติ เส้นแวง (longitude or meridians) จะลากจากขั้วโลกเหนือมายังขั้วโลกใต้ เส้นแวง 0 องศา (Prime Meridian) จะลากผ่านเมือง Greenwich ประเทศอังกฤษ ทางทิศตะวันออก จะมีค่า 0-180 องศา ส่วนทางทิศตะวันตกของเส้น Prime Meridian จะมีค่า 0-(-180) องศา เส้นรุ้งบางครั้งเรียกว่า Parallels เนื่องจากจะมีระยะห่างที่เท่ากันตลอด เส้นรุ้งที่ลากผ่านเส้นศูนย์สูตร (equator) จะมีค่า 0 องศา เส้นรุ้งที่อยู่ทางทิศเหนือของเส้นศูนย์สูตร จะมีค่าจาก 0-90 องศา ถึงขั้วโลกเหนือ และเส้นรุ้งที่อยู่ทางทิศใต้ของเส้นศูนย์สูตร จะมีค่าจาก 0-(-90) องศา ถึงขั้วโลกใต้ ดังนั้น ค่าพิกัดหนึ่งๆ ของระบบเส้นรุ้ง เส้นแวงจะมีเพียงตำแหน่งเดียวบนพื้นโลก ดังรูปที่ 2.3



รูปที่ 2.3 ระบบพิกัดแบบทรงกลม

2.4.1 ละติจูด

- เส้นรอยตัดบนพื้นผิวพิภพที่เกิดจากการสมมติใช้พื้นราบตัดพิภพโดยให้พื้นราบนั้นตั้งได้ฉากกับแกนหมุนของพิภพเสมอ เส้นรอยตัดดังกล่าวนั้นคือเส้นละติจูด นิยมเรียกสั้นๆ ว่า " เส้นขนาน"
- ละติจูดศูนย์องศา คือ เส้นรอยตัดบนพื้นผิวพิภพ ที่เกิดจากพื้นราบที่ตั้งได้ฉากกับแกนหมุนตัดผ่านจุดศูนย์กลางของพิภพ เส้นรอยตัดเส้นนี้มีชื่อเรียกอีกอย่างหนึ่งว่า "เส้นศูนย์สูตร" (Equator) ซึ่งเป็นวงขนานละติจูดวงใหญ่ที่สุด
- ค่าละติจูดของวงละติจูดใด คือ ค่ามุมที่จุดศูนย์กลางของพิภพนับไปตามพื้นราบที่บรรจุแกนหมุนของพิภพ เริ่มจากพื้นศูนย์สูตรถึงแนวเส้นตรงที่ลากจากจุดศูนย์กลางพิภพไปยังวงละติจู ดนั้น
- ที่จุดขั้วเหนือของพิภพมีค่าละติจูดเท่ากับ 90 องศาเหนือ และที่จุดขั้วใต้ของพิภพมีค่าละติจูดเท่ากับ 90 องศาใต้
- เนื่องจากพื้นของวงละติจูดศูนย์องศา หรือพื้นศูนย์สูตร เป็นพื้นที่ตัดผ่านจุดศูนย์กลางของพิภพ วงศูนย์สูตรจึงถูกเรียกว่า "วงกลมใหญ่" ส่วนละติจูดอื่นๆ เป็นวงกลมเล็ก วงละติจูดจะมีขนาดเล็กลงๆ เมื่อห่างวงศูนย์สูตรออกไปจนกระทั่งกลายเป็นจุดที่ขั้วโลกเหนือและขั้วโลก ใต้

- ระยะห่างระหว่างเส้นละติจูด 1 องศา คิดเป็นระยะทางบนผิวพิภพประมาณ 111 กิโลเมตร (69 ไมล์) และ 1 พิลิปดา มีระยะห่างประมาณ 30.48 เมตร (100 ฟุต)

2.4.2 ลองจิจูด

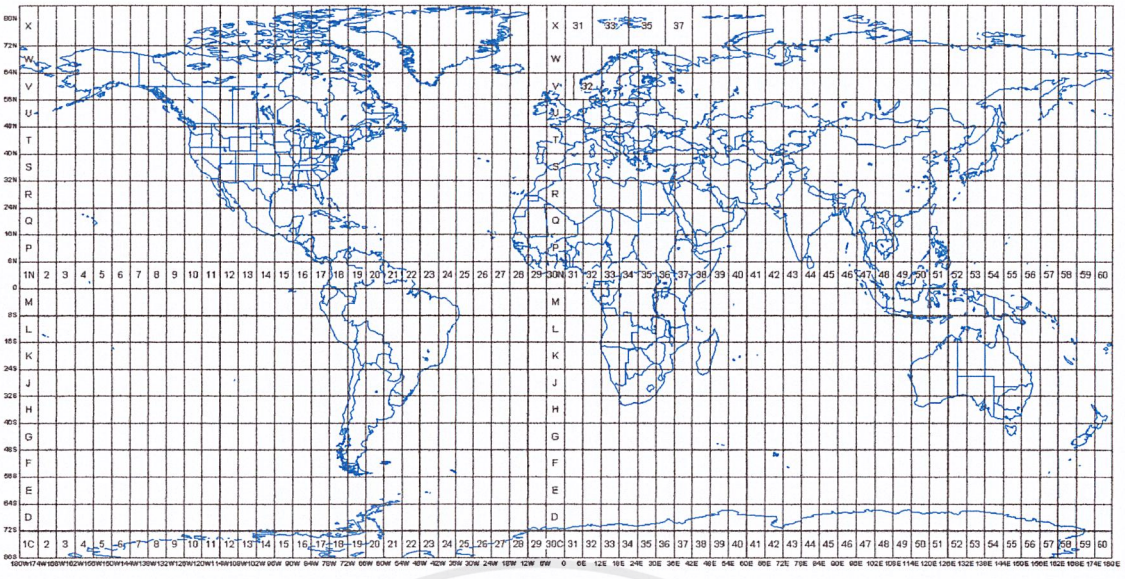
- เส้น รอยตัดบนพื้นผิวพิภพที่เกิดจากการสมมติใช้พื้นราบตัดพิภพ โดยให้พื้นราบผ่านแนวแกนหมุนของพิภพ เส้นรอยตัดบนพื้นผิวพิภพดังกล่าวเรียกว่า เส้นลองจิจูดหรือเส้นเมริเดียน (Meridian)
- ลองจิจูดศูนย์องศา คือเส้นลองจิจูดที่ผ่านหอส่งดาว ณ เมืองกรีนิส (Greenwich) ในประเทศอังกฤษ มีชื่อเรียกอีกชื่อหนึ่งว่า เมริเดียนหลัก (Prime Meridian)
- การกำหนดค่าลองจิจูด คือค่าง่ามมุมที่จุดศูนย์กลางพิภพบนพื้นศูนย์สูตรโดยใช้แนวเส้นตรงที่ลากจาก จุดศูนย์กลางพิภพมายังเมริเดียนหลักเป็นแนวเริ่มนับค่าง่ามมุมไปทางตะวันออก 180 องศา เรียกว่า ลองจิจูดตะวันออก และนับค่าง่ามมุมไปทางตะวันตก 180 องศา เรียกว่า ลองจิจูดตะวันตก เส้นลองจิจูดที่ 180 องศาตะวันออกและตะวันตกเป็นเส้นเดียวกัน
- ลองจิจูดทุกเส้นเป็นส่วนโค้งของวงกลมใหญ่ (Great Circle)
- ระยะห่างระหว่างเส้นลองจิจูด 1 องศา ตามเส้นศูนย์สูตร คิดเป็นระยะทางประมาณ 111 กิโลเมตร (69 ไมล์) และ 1 พิลิปดา มีระยะห่างประมาณ 30.48 เมตร (100 ฟุต) แต่เนื่องจากเส้นลองจิจูตทุกเส้นจะไปบรรจบกันที่จุดขั้วเหนือและขั้วใต้ของ พิภพ ดังนั้น ระยะห่างระหว่างเส้นลองจิจูดจึงน้อยลงๆ เมื่อยิ่งห่างจากเส้นศูนย์สูตรออกไป

2.5 ระบบพิกัดแบบ UTM

ระบบพิกัดกริด UTM (Universal Transvers Mercator co-ordinate System) เป็นระบบตารางกริดที่ใช้ช่วยในการกำหนดตำแหน่งและใช้อ้างอิง ในการบอกตำแหน่ง ที่นิยมใช้กับแผนที่ในกิจการทหารของประเทศต่างๆ เกือบทั่วโลกในปัจจุบัน เพราะเป็นระบบตารางกริดที่มีขนาดรูปร่างเท่ากันทุกตาราง และมีวิธีการกำหนดบอกค่าพิกัดที่ง่ายและถูกต้องเป็นระบบกริดที่นำเอาเส้น โครงแผนที่แบบ Universal Transvers Mercator Projection ของ Gauss Krugger มาใช้ดัดแปลงการถ่ายทอดรายละเอียดของพื้นผิวโลกให้รูปทรงกระบอก Mercator Projection อยู่ในตำแหน่ง

Mercator Projection (แกนของรูปทรงกระบอกจะทับกับแนวเส้นอิเควเตอร์ และตั้งฉากกับแนวแกนของขั้วโลก) ประเทศไทยเราได้นำเอาเส้นโครงแผนที่แบบ UTM นี้มาใช้กับการทำแผนที่กิจการทหารภายในประเทศจากรูปถ่ายทางอากาศในปี 1953 ร่วมกับสหรัฐอเมริกา เป็นแผนที่มาตราส่วน 1:50,000 ชุด 708 และปรับปรุงใหม่เป็นชุด L 7017 ที่ใช้ในปัจจุบัน

แผนที่ ระบบพิกัดกริด ที่ใช้เส้นโครงแผนที่แบบ UTM เป็นระบบเส้นโครงชนิดหนึ่งที่ใช้ผิวรูปทรงกระบอกเป็นผิวแสดงเส้นเมริเดียน (หรือเส้นลองกิจูด) และเส้นละติจูดของโลก โดยใช้ทรงกระบอกตัดโลกระหว่างละติจูด 84 องศาเหนือ และ 80 องศาใต้ในลักษณะแกนรูปทรงกระบอก ทำมุมกับแกนโลก 90 องศารอบโลก แบ่งออกเป็น 60 โซนๆ ละ 6 องศา โซนที่ 1 อยู่ระหว่าง 180 องศา กับ 174 องศาตะวันตก และมีลองกิจูด 177 องศาตะวันตก เป็นเมริเดียนย่านกลาง (Central Meridian) มีเลขกำกับแต่ละโซนจาก 1 ถึง 60 โดยนับจากซ้าย ไปทางขวาระหว่างละติจูด 84 องศาเหนือ 80 องศาใต้ แบ่งออกเป็น 2 ช่อง ช่องละ 8 องศา ยกเว้นช่องสุดท้ายเป็น 12 องศา โดยเริ่มนับตั้งแต่ละติจูด 80 องศาใต้ ขึ้นไปทางเหนือ ให้ช่องแรกเป็นอักษร C และช่องสุดท้ายเป็นอักษร X (ยกเว้น I และ O) จากการแบ่งตามที่กล่าวแล้วจะเห็นพื้นที่ในเขตลองจิจูด 180 องศาตะวันตก ถึง 180 องศาตะวันออก และละติจูด 80 องศาใต้ ถึง 84 องศาเหนือ จะถูกแบ่งออกเป็นรูปสี่เหลี่ยมผืนผ้า 1,200 รูป แต่ละรูปมีขนาดกว้างยาว 6 องศา x 8 องศา จำนวน 1,140 รูป และกว้างยาว 6 องศา x 12 องศา จำนวน 60 รูป รูปสี่เหลี่ยมนี้เรียกว่า Grid Zone Designation (GZD) การเรียกชื่อ Grid Zone Designation ประเทศไทยมีพื้นที่อยู่ ระหว่างละติจูด 5 องศา 30 ลิปดา เหนือ ถึง 20 องศา 30 ลิปดา เหนือ และลองกิจูดประมาณ 97 องศา 30 ลิปดา ตะวันออก ถึง 105 องศา 30 ลิปดา ตะวันออก ดังนั้น ประเทศไทยจึงตกอยู่ใน GZD 47N 47P 47Q 48N 48P และ 48Q



รูปที่ 2.4 การแบ่งกริดโซนบนระบบพิกัดแบบ UTM

2.6 อัลกอริทึม A*

การค้นหาเส้นทางแบบ A* เป็นการค้นหาแบบการหาทางที่ดีที่สุดก่อน (best-first) บนโครงสร้างข้อมูลแบบกราฟ โดยจะหาเส้นทางที่มีค่าใช้จ่ายน้อยที่สุดจากโหนดเริ่มต้นไปยังโหนดปลายทาง

โดยการค้นหาแบบ A* จะใช้ฮิวริสติกฟังก์ชัน (heuristic function) แบบระยะทางบวกกับค่าใช้จ่าย (distance-plus-cost) เพื่อหาลำดับของโหนดในกราฟที่จะเข้าถึง โดยค่าระทางบวกกับค่าใช้จ่าย (distance-plus-cost) นั้นคือผลรวมของ 2 ฟังก์ชัน ได้แก่ค่าใช้จ่ายของระยะทางที่ผ่านมาก่อนหน้านี้ ซึ่งอาจจะเป็นฮิวริสติกฟังก์ชัน หรือไม่เป็นฮิวริสติกฟังก์ชันก็ได้ บวกกับค่าใช้จ่ายแบบฮิวริสติก (heuristic cost) ไปยังปลายทาง

โดยค่าใช้จ่ายแบบฮิวริสติก (heuristic cost) ไปยังปลายทางจะต้องเป็นค่าที่ยอมรับได้ (ไม่มีการประมาณค่าเกินจริง) ตัวอย่างของฮิวริสติกฟังก์ชันที่ยอมรับได้ ได้แก่ระยะทางการกระจัดไปยังปลายทางซึ่งไม่มีทางที่จะมากกว่าระยะทางไปยังปลายทางโดยผ่านโหนดอื่นๆ แน่นอน

การค้นหาแบบ A* จะค่อยๆค้นหาเส้นทางจากโหนดเริ่มต้นจนกระทั่งค้นพบเส้นทางที่สั่นที่สุดไปยังปลายทาง โดยจะค้นหาไปยังโหนดที่คาดว่าจะเป็นเส้นทางที่ดีที่สุดไปยังปลายทางก่อน แต่ละโหนดที่เข้าถึงนั้นจะเก็บค่าต่างๆไว้ดังนี้

g(x) เป็นค่าใช้จ่ายที่น้อยที่สุดที่ใช้เดินทางจากโหนด เริ่มต้นมายังโหนดนั้นๆ

h(x) เป็นค่าใช้จ่ายแบบฮิวริสติก สำหรับการเดินทางจากโหนดนี้ไปยังปลายทาง

f(x) เป็นผลรวมของ g(x) และ h(x)

การค้นหาแบบ A* จะมีคิวที่มีลำดับความสำคัญ (priority queue) เพื่อเก็บโหนดที่จะทำการค้นหาในอนาคต เรียกว่าโอเพนเซ็ต (open set) โดยโหนดที่มีค่า f(x) น้อยที่สุดจะมีความสำคัญสูงสุด ทุกครั้งของการค้นหาโหนดถัดไป โหนดที่มีความสำคัญสูงที่สุดจะถูกนำออกจากคิว จากนั้นค่า g(x) h(x) และ f(x) ของโหนดข้างเคียงจะถูกแก้ไขและโหนดข้างเคียงจะถูกเพิ่มลงในคิว ถ้าหากว่ายังไม่ได้อยู่ในคิวการค้นหาจะทำซ้ำขั้นตอนนี้ไปเรื่อยๆจะกระทั่งโหนดปลายทางถูกนำออกจากคิวหรือคิวว่าง ค่า f(x) ของโหนดปลายทางนั้นจะเป็นระยะทางที่สั้นที่สุด เพราะค่า h(x) จะของโหนดปลายทางจะเป็น 0 สำหรับการประมาณค่าฮิวริสติกที่ยอมรับได้

```
function A*(start,goal)
    closedset := the empty set
        % The set of nodes already evaluated.
    openset := set containing the initial node
        % The set of tentative nodes to be evaluated.
    g_score[start] := 0
         %Distance from start along optimal path.
    h_score[start] := heuristic_estimate_of_distance(start, goal)
    f_score[start] := h_score[start]
        % Estimated total distance from start to goal through y.
    while openset is not empty
        x := the node in openset having the lowest f_score[] value
        if x = goal
            return reconstruct_path(came_from,goal)
        remove x from openset
        add x to closedset
        foreach y in neighbor_nodes(x)
            if y in closedset
                continue
            tentative_g_score := g_score[x] + dist_between(x,y)
            tentative_is_better := false
            if y not in openset
                add y to openset
                h_score[y] := heuristic_estimate_of_distance(y, goal)
                tentative_is_better := true
            elseif tentative_g_score
tentative_is_better := true
            if tentative_is_better = true
                came_from[y] := x
                g_score[y] := tentative_g_score
                f_score[y] := g_score[y] + h_score[y]
    return failure

function reconstruct_path(came_from,current_node)
    if came_from[current_node] is set
        p = reconstruct_path(came_from,came_from[current_node])
        return (p + current_node)
    else
        return the empty path
```

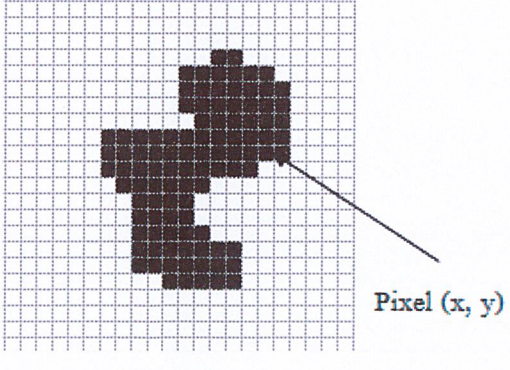
2.7 ความรู้พื้นฐานสำหรับการประมวลผลภาพดิจิตอล

ภาพดิจิตอลคือภาพที่เก็บอยู่ในรูปแบบของดิจิตอล ภาพที่เรามองเห็นด้วยสายตาทั่วไปนั้นเป็นภาพในลักษณะสามมิติ คือ มีมิติของความกว้าง ความยาว และความลึกหรือความสูง ส่วนภาพถ่ายที่เห็นกันอยู่ในโทรทัศน์หรือเครื่องคอมพิวเตอร์นั้น เป็นการแปลงภาพจากสามมิติ มาเป็นสองมิติ โดยการแปลงสัญญาณไฟฟ้าในรูปแบบอนาล็อก ยกตัวอย่างเช่น ในกล้องวิดีโอ เซ็นเซอร์ที่อยู่ในกล้องจะทำการสแกนหรือวัดผลรวมความเข้มแสงที่จุดต่างๆ ไปตามแนวสแกนที่เรียกว่า ราสเตอร์สแกน (Raster Scan) การสแกนแบบนี้จะมีทิศทางจากบนลงล่าง และจากซ้ายไปขวา ภาพที่ได้จากการสแกนนั้นจะเป็นภาพแบบต่อเนื่อง (Continuous) ด้วยความเร็วทั่วไปที่ 24 ภาพต่อวินาที เช่นเดียวกันนี้ในเครื่องรับภาพวิดีโอก็จะรับภาพที่ได้มาจากเครื่องถ่ายวิดีโอ และแสดงผลโดยเริ่มจากบนลงล่างและจากซ้ายไปขวาเช่นเดียวกัน

แต่ภาพที่ได้มาจากระบบอนาล็อกนั้นยังเป็นภาพแบบต่อเนื่อง ที่ยังไม่สามารถนำมาใช้ในการประมวลผลได้ ต้องมาทำการแปลงให้เป็นภาพเชิงตัวเลขเสียก่อนด้วยวิธีการ ดิจิไทเซชั่น (Digitization) ซึ่งเป็นการแปลงฟังก์ชันต่อเนื่อง $f(x,y)$ ให้เป็นฟังก์ชันไม่ต่อเนื่อง $g(x,y)$ เพื่อให้นำมาประมวลผลด้วยคอมพิวเตอร์ได้

2.7.1 การแทนภาพด้วยข้อมูลแบบดิจิตอล

จากที่ได้กล่าวไปแล้วในตอนต้น ข้อมูลภาพแบบดิจิตอลเป็นภาพที่ถูกดัดแปลงมาจากภาพแบบต่อเนื่อง ให้อยู่ในรูปตัวเลข ด้วยวิธีการดิจิไทเซชั่น โดยภาพอนาล็อกจะถูกแบ่งให้เป็นพื้นที่สี่เหลี่ยมเล็กๆ ที่เรียกว่า พิกเซล (Pixel) โดยแต่ละพิกเซลจะใช้ (x,y) ในการระบุตำแหน่ง การแสดงข้อมูลภาพดิจิตอลสามารถอธิบายได้ด้วย เมทริกซ์ $(M \times N)$ และให้จุดต่างๆ ที่อยู่ในเมทริกซ์เป็นจุดที่พิกัด (x,y) ใดๆ เป็นส่วนประกอบของภาพ



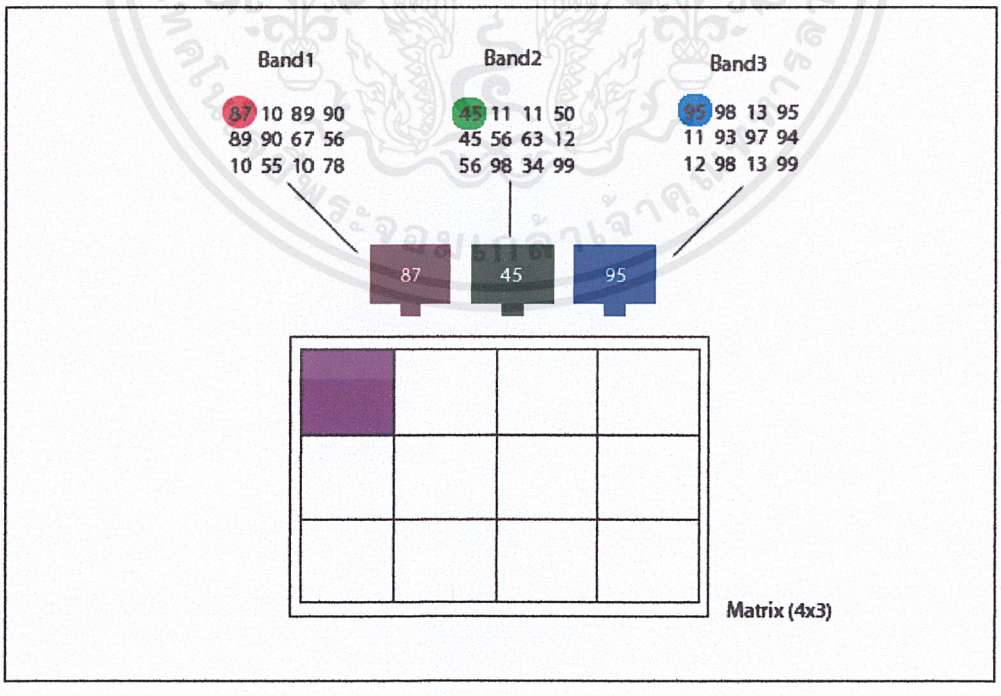
รูปที่ 2.5 ตัวอย่างพิกัดของจุดในรูป

ค่าของพิกเซล หรือ ฟังก์ชัน (x,y) ณ จุดใดๆ จะแสดงได้ด้วยค่าของความเข้มแสงซึ่งอาจแบ่งได้หลายระดับ ถ้ามี 2 ระดับ ก็จะเป็นแค่ 0 กับ 1 จากรูป จุดต่างๆ ที่แสดงอยู่ในพิกเซลดังนี้ก็คือพิกเซล ซึ่งก็คือ ความสว่างหรือค่า ลูมิแนนซ์ (Luminance (L)) ของภาพ ถ้าภาพนั้นเป็นภาพขาวดำขนาด 8 บิต จะมีค่า L เท่ากับ 28 หรือเท่ากับ 256 คือตั้งแต่ระดับ 0 จนถึง 255 บางครั้งค่าความสว่าง (L) อาจหมายถึงระดับความละเอียดของภาพ (Image Resolution) ถ้าพิกเซล เป็นภาพขาวดำจะอ่านค่าภาพดิจิตอลในรูปแบบเมทริกซ์ 2 มิติ ขนาด $(M \times N)$ ได้ดังนี้

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & . & f(0,n-1) \\ f(1,0) & f(1,1) & . & f(1,n-1) \\ . & . & . & . \\ f(m-1,0) & f(m-1,1) & . & f(m-1,n-1) \end{bmatrix}_{(M*N)}$$

รูปที่ 2.6 ตำแหน่งของแต่ละพิกเซลในรูปของเมทริกซ์

โดยที่ค่า $f(x,y)$ จะอยู่ในช่วง 0 ถึง 255 สมมติว่าอ่านค่าพิกเซล จากภาพหนึ่งได้ $f(x,y)$ เท่ากับ 10 แสดงว่าจุดพิกเซลนั้นมีความสว่างน้อยหรือค่อนข้างจะดำ ถ้าอ่านได้เป็น 255 แสดงว่าจุดพิกเซลนั้นมีความสว่างมาก หรือเป็นสีขาว



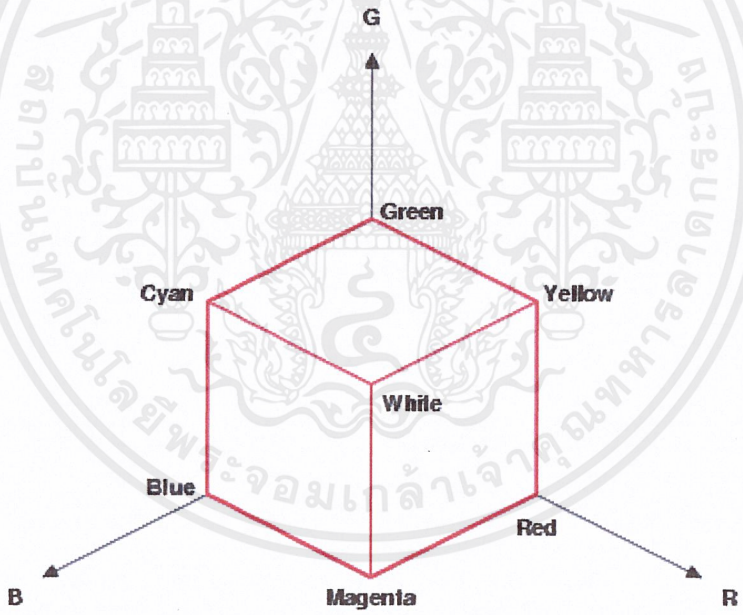
รูปที่ 2.7 แสดงค่าของพิกเซลที่มาจากการผสมสีRGB

จากรูปด้านบนจะช่วยทำให้เข้าใจการแสดงค่า พิกเซลในเมทริกซ์มากขึ้น เริ่มต้นด้วยพิกัด $f(x,y) = f(0,0)$ ค่าของพิกเซลที่ได้จะเป็นการผสมสีกันระหว่างค่าของแม่สีทั้งสาม ซึ่งได้แก่ แดง เขียว น้ำเงิน

2.7.2 ระบบสีอาจีบี (RGB)

ระบบสีอาจีบี (RGB) เป็นระบบสีที่เกิดจากการรวมกันของแสงสีแดง เขียวและน้ำเงิน โดยปกติจะใช้ในจอภาพแบบ ซีอาที (CRT) และเนื่องจากระบบสีอาจีบี (RGB) เป็นระบบสีของแสง จึงทำให้ภาพที่ได้ออกมานั้นมีความสมจริงและยังดูสวยงาม

โมเดลสี (Color Space) ประกอบด้วย 3 แม่สีหลัก ได้แก่ สีแดง สีเขียว และสีน้ำเงิน ถ้านำแต่ละแม่สีมาพล็อตกราฟในระดับพิกัด โดยแต่ละสีมีค่า 0 ถึง 1 (0 แสดงถึงค่าความมืด และ 1 แสดงถึงความสว่าง) จะได้ภาพการผสมสีทางแสงหรือการบวกแม่สีเข้าด้วยกัน (Additive Primary Color) ดังรูป

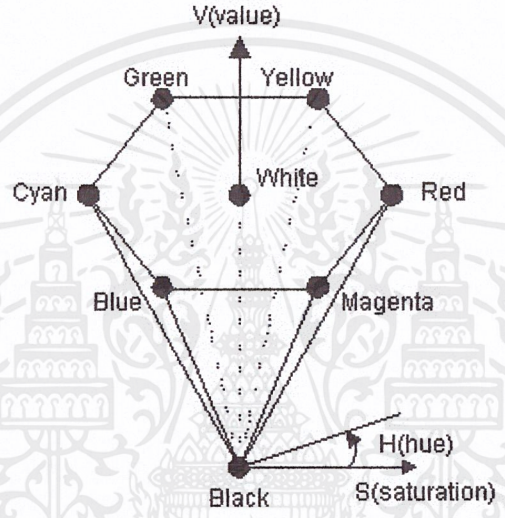


รูปที่ 2.8 โมเดลสี อาจีบี(RGB)

2.7.3 ระบบสีเอชเอสวี (HSV)

สีโหมดเอชเอสวี (HSV) ย่อมาจากฮิว (Hue) แซตเอชัน (Saturation) วาลู (Value) ถูกเสนอโดย A.R. Smith (1978) มีวัตถุประสงค์เพื่อทำให้สะดวกในการใช้สีต่างๆ มากกว่าที่ใช้เฉพาะแม่สีทั้งสาม (Red, Green, Blue) โดยสีโหมดนี้จะใช้ค่าทินเชรด (tint shade) และโทน (tone) ของนักวาดเขียน สีเอชเอสวี-โมเดล (HSV Model) จัดเป็นสีแบบ User Oriented ซึ่งต่างจากสีในโหมด RGB

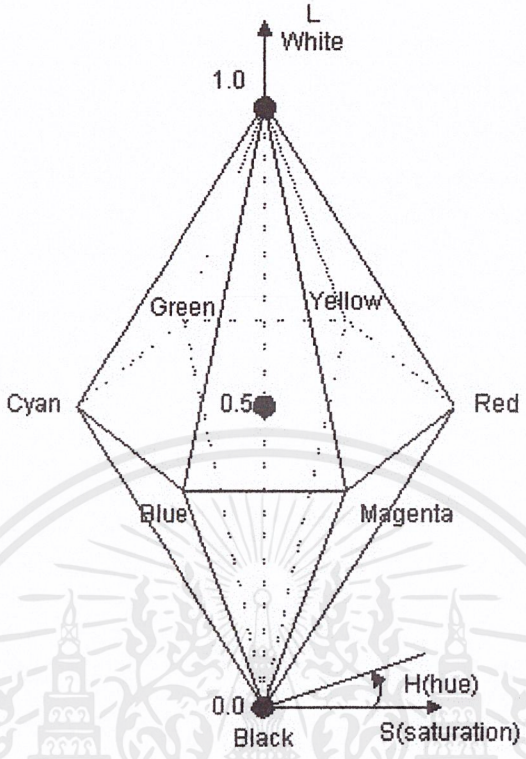
CMY หรือ YIQ ที่เป็น Hardware Oriented Color Space ของ HSV มีลักษณะเป็นพีระมิดฐานหกเหลี่ยม โดยพีรามิดของ HSV นี้เป็นการแปลงมาจาก โมเดลอาจีบีแบบลูกบาศ์ก (RGB cube) ซึ่งเป็นแบบ นอน-ลิเนียร์ (non-linear) สีโหมด HSV จะใช้ค่าในพิกัดเชิงมุม โดยค่า H เป็นค่ามุมรอบแกนตั้งจะระบุเป็นองศามีค่าระหว่าง 0-360 ค่า S เป็นค่าอัตราส่วนมีค่าตั้งแต่ 0 ถึง 1 วัดจากแกนตั้ง (V-axis) ไปยังพื้นผิวของพีรามิด และค่า V เป็นค่าความสูงของพีระมิด โดยระบบสีที่ใช้พิกัดเชิงมุมระบบแรกได้แก่ระบบสีของ Munsell (1946) สีระบบนี้จะมีค่าพารามิเตอร์ต่างๆ ได้แก่ ฮิว (Hue), ชโลมา (Chroma) และวาลู (Value)



รูปที่ 2.9 ระบบสี เอชเอสวี(HSV)

จากรูปที่ 2.9 ส่วนยอดของพีระมิดมีค่า V=1 ซึ่งบรรจุสีต่างๆ ที่มีความเข้มสูงสุด สีคอมพรีเมนต์ทารี (Complementary Color) คือสีที่อยู่ตรงกันข้ามกันและมีค่า H ห่างกัน 180 องศา ซึ่งค่ามุมนี้จะวัดรอบแกนตั้ง โดยสีแดงจะอยู่ที่ 0 องศา ค่า S เป็นค่าอัตราส่วนมีค่าตั้งแต่ 0 ถึง 1 วัดจากแกนตั้ง (V-axis) ไปยังพื้นผิวของพีรามิด โดยบน V-axis ค่า S=0 และบนพื้นผิวค่า S=1 รูปพีระมิดดังกล่าวมีค่าความสูง V และมียอดอยู่ที่จุด ออริจิน(Origin Point) โดยที่จุดออริจินนี้จะมีค่า V=0 และเป็นสีดำ ค่าต่างๆ ของ S ระหว่าง 0 ถึง 1 จะสัมพันธ์กับจุดที่ V=0 จุดที่ S=0, V=1 จะเป็นสีขาว สีแดงจะตรงกับจุดที่ค่า H=0, S=1, V=1 โดยจุดที่ค่า S=1, V=1 จะเป็นสีตั้งต้น (Pure Pigment) ในการผสมสีของนักวาดเขียน การเพิ่มสีขาวเท่ากับเป็นการลดค่า S (โดยค่า V ไม่เปลี่ยน) การเพิ่มสีดำเท่ากับเป็นการลดค่า V (โดยค่า S ไม่เปลี่ยน) โทนสีสร้างได้โดยการลดทั้งค่า S และค่า V และการเปลี่ยนค่า H จะเป็นการเลือกสีตั้งต้น (Pure Pigment) ในการผสมสี

2.7.4 ระบบสี เอชเอลเอส(HLS)



รูปที่ 2.10 ระบบสี เอชเอลเอส(HLS)

ระบบสีเอชเอสบี (HSB) หรือฮิว (Hue) แซตเอชัน (Saturation) และความสว่าง (Brightness) ค่า Hue คือค่าที่แสดงความแตกต่างของสีหรือความแตกต่างของคลื่นแสง (wave length , nm) ค่า saturation บ่งถึงความบริสุทธิ์ของสี (purity) และค่า Brightness บ่งถึงความสว่างและมืด ค่า Hue จะมีค่าตั้งแต่ 0 องศา ถึง 360 องศาโมเดลสีแบบเอชเอลเอส(HLS Color Model) มีลักษณะคล้ายกับ โมเดลสีเอชเอสวี (HSV Color Model) ซึ่งมีใช้ในบริษัท Tektronix เป็นโมเดลที่ใช้ระบบสีของ Ostwald Color Space ของ HLS Color Model จะคล้ายแบบ HSV แต่จะเป็น พีรามิคฐานหกเหลี่ยมคู่ (Double Hexcone) ค่าของ Hue เป็นค่ามุมรอบแกนตั้งของ Double Hexcone ระบุเป็นองศามีค่าระหว่าง 0-360โดยที่มุม 0 องศาเป็นสีแดง สีต่างๆ ที่อยู่รอบฐานพีระมิดจะเรียงลำดับเช่นเดียวกับใน ซีไออีไดอาแกรม (CIE Diagram) ในทิศทวนเข็มนาฬิกาตามลำดับดังนี้คือ สีแดง เหลือง เขียว สีระหว่างน้ำเงินกับเขียว น้ำเงิน และ ม่วงแดง ซึ่งก็เป็นลำดับเหมือนใน HSV ด้วย ค่า S เป็นค่าอัตราส่วนมีค่าตั้งแต่ 0 ถึง 1 วัดจากแกนตั้ง (L-axis) ไปยังพื้นผิวของพีรามิด และค่า L เป็นค่าความสูงของพีระมิด สีขาวและสีดำจะอยู่ที่ยอดพีรามิดทั้งสอง

2.7.4.1 การแปลงจากระบบสี RGB เป็นระบบสี HSL

ให้ r, g, b $\in$ [0,1] เป็นแดง, เขียว, และฟ้าตามลำดับ ให้ max เป็นค่ามากสุดของ r, g, และ b และ min เป็นค่าน้อยสุด

การหามุมของสี h $\in$ [0, 360] สำหรับ HSL , คำนวณจากสมการที่ 2.1

$$H = \begin{cases} 0, & \text{if } \max = \min \\ \left(60° \times \frac{g-b}{\max-\min} + 360°\right) \bmod 360°, & \text{if max=r} \\ 60° \times \frac{b-r}{\max-\min} + 120°, & \text{if max=g} \\ 60° \times \frac{r-g}{\max-\min} + 240°, & \text{if } \max = \text{b} \end{cases} \quad (2.1)$$

การหาเฉดสีและความสว่าง s, l $\in$ [0,1] สำหรับ HSL คำนวณจากสมการที่ 2.2 และ 2.3

$$l = \frac{1}{2}(\max + \min) \quad (2.2)$$

$$s = \begin{cases} 0 \text{ if } \max = \min \\ \frac{\max-\min}{\max+\min} = \frac{\max-\min}{2l}, \text{if } l \leq \frac{1}{2} \\ \frac{\max-\min}{2-(\max-\min)} = \frac{\max-\min}{2-2l}, \text{ if } l > \frac{1}{2} \end{cases} \quad (2.3)$$

ค่าของ h โดยปกติทั่วๆไปจะอยู่ระหว่าง 0 ถึง 360°, และ h = 0 จะใช้เมื่อ max = min

2.7.5 ระดับเทา (Gray Level)

ระดับเทาเป็นค่าซึ่งระบุความสว่างหรือความเข้ม ซึ่งมีค่าตั้งแต่ 0-255 (0 คือระดับเข้ม 255 คือระดับสว่าง) รวมทั้งพิกัดแนวนอนและแนวตั้ง ซึ่งใช้ระบุตำแหน่งในแถวลำดับภาพ (Image Array)

วิธีการหาค่าระดับเทา (Gray Level)

$$Gray\ Level = \frac{R+G+B}{3} \quad (2.4)$$

วิธีการหาค่าระดับเทาที่กล่าวมาข้างต้นเป็นการเฉลี่ยค่าของแม่สีทั้งสาม ซึ่งเป็นวิธีการที่ง่ายที่สุด แต่ก็อาจจะมีความเพี้ยนของสีได้ จึงมีวิธีอีกอย่างหนึ่งซึ่งจะคิดตามความสว่างของแต่ละแม่สี โดยมีรูปแบบดังสมการ

$$R_R = \frac{(R_S + G_S + B_S)}{3} \quad \text{หรือ} \quad R_R = ((0.299 \times R_S) + (0.587 \times G_S) + (0.114 \times B_S)) \quad (2.5)$$

$$G_R = \frac{(R_S + G_S + B_S)}{3} \quad \text{หรือ} \quad G_R = ((0.299 \times R_S) + (0.587 \times G_S) + (0.114 \times B_S)) \quad (2.6)$$

$$B_R = \frac{(R_S + G_S + B_S)}{3} \quad \text{หรือ} \quad B_R = ((0.299 \times R_S) + (0.587 \times G_S) + (0.114 \times B_S)) \quad (2.7)$$

โดยที่ R_R หมายถึง ค่าเอาต์พุตพิกเซลสีแดง

G_R หมายถึง ค่าเอาต์พุตพิกเซลสีเขียว

B_R หมายถึง ค่าเอาต์พุตพิกเซลสีน้ำเงิน

R_S หมายถึง ค่าอินพุตพิกเซลสีแดง

G_S หมายถึง ค่าอินพุตพิกเซลสีเขียว

B_S หมายถึง ค่าอินพุตพิกเซลสีน้ำเงิน

2.7.6 การแปลงภาพสีให้เป็นภาพขาว-ดำ (Thresholding)

เป็นกระบวนการแปลงภาพสีให้มีการแสดงผลได้แค่ 2 ระดับ คือ ขาว และดำ โดยจะแปลงข้อมูลภาพให้เป็นภาพไบนารี (Binary Image) มีกระบวนการแปลงภาพที่มีความเข้มหลายระดับ (Multilevel Image) ให้เป็นภาพที่มีความเข้มเพียง 2 ระดับ หรือ 1 บิต (bit) คือ 0 และ 1 โดย 0 แทนด้วยจุดที่มีภาพสีขาว และ 1 แทนด้วยจุดที่มีภาพสีดำ

เทรดโชดิ้งเทกนิค (Thresholding Technique) คือการพิจารณาจุดพิกเซลในภาพว่าจุดใดควรจะเป็นจุดขาว หรือจุดใดควรจะเป็นจุดที่มีค่าเท่ากับ 1 (จุดดำ) โดยจะทำการเปรียบเทียบค่าของแต่ละพิกเซล f(x,y) กับค่าคงที่ที่เรียกว่า เทรดโช (Threshold (Threshold Value)) เทคนิคนี้นิยมใช้กันมากในกรณีที่ความแตกต่างระหว่างวัตถุ (Object) และพื้นหลัง (Background) ค่าพิกเซลในภาพที่มีค่าน้อยกว่าค่าเทรดโชจะถูกกำหนดเป็น 1 (จุดดำ) และถ้าค่าของพิกเซลใด ๆ ในภาพมีค่ามากกว่าหรือเท่ากับค่าเทรดโชจะถูกกำหนดให้เป็น 0 (จุดขาว)

ในการทำภาพ ไบนารี(Binary) โดยการทำเทรดโชดิ้ง (Thresholding) ให้ได้ภาพดีและคมชัด ต้องเกิดจากการเลือกค่าเทรดโชที่ถูกต้องและเหมาะสม ถ้าเลือกค่าเทรดโชไม่เหมาะสม เช่น ค่าเทรดโชที่มากหรือน้อยจนเกินไป ภาพที่ได้จะขาดความคมชัดหรืออาจทำให้รายละเอียดของภาพขาดหายไป หรือภาพที่ได้อาจจะมืดเกินไป หรือสว่างเกินไป หรืออาจจะเป็นภาพที่มีสิ่งรบกวน (Noise) เกิดขึ้น ทำให้ภาพผลลัพธ์ที่ได้ไม่ชัดเจน

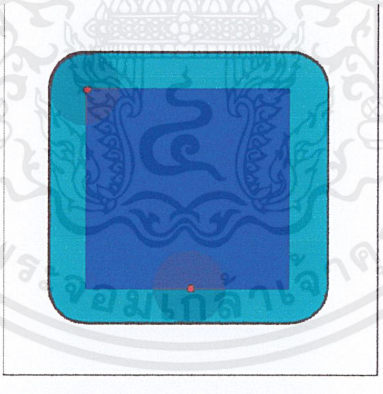
2.7.7 การประมวลผลภาพกับรูปร่างและโครงร่างของภาพ (Morphological Image Processing)

การประมวลผลภาพกับรูปร่างและโครงร่างของภาพเป็นการประมวลผลภาพโดยการเปลี่ยนแปลงลักษณะรูปร่างหรือโครงสร้างของภาพ โอเปอเรชั่นพื้นฐานโดยทั่วไปได้แก่ การทำไดเรชัน (Dilation) และอีโลชัน (Erosion) โดยการทำไดเรชันคือการขยายภาพโดยมีสัดส่วนเท่ากันทั่วทั้งภาพ (Uniform) การทำอีโลชันคือการย่อภาพ

นอกจากโอเปอเรชั่นพื้นฐานดังที่ได้กล่าวข้างต้นแล้วยังมีโอเปอเรชั่นอื่นๆ อีกที่ได้กล่าวไว้ในบทนี้ได้แก่การทำแบบโอเพน (Opening) และแบบโคลด (Closing) เป็นต้น

2.7.7.1 การขยายภาพ (Dilation)

การขยายภาพในที่นี้จะพิจารณาสำหรับข้อมูลภาพที่เป็นแบบไบนารี ซึ่งการขยายภาพจะทำได้โดยกำหนดรูปแบบ (ซึ่งสามารถสร้างได้จาก * และ 1 โดยมีจุดเริ่มต้นที่กำหนดโดยวงกลม) และนำรูปแบบนี้สแกนไปบนข้อมูลภาพตามลำดับตลอดทั้งภาพซึ่งในขณะที่จุดเริ่ม (Origin) ของรูปแบบที่ตรงกับตำแหน่งข้อมูลภาพที่พิกเซลมีค่าเท่ากับ 1 นั้นก็จะทำการยูเนียรูปแบบนี้เข้ากับข้อมูลภาพดังตัวอย่าง

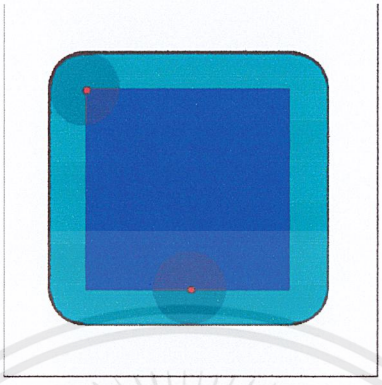


รูปที่ 2.11 เปรียบเทียบภาพก่อนและหลังการขยายภาพ

2.7.7.2 การย่อภาพ (Erosion)

การย่อภาพเป็นลักษณะของการลบข้อมูลภาพบริเวณขอบของภาพ การย่อภาพสามารถทำได้มีลักษณะคล้ายกับการขยายภาพโดยการสร้างรูปแบบขึ้นแล้วนำรูปแบบนี้ไปสแกนตามข้อมูลภาพ

สำหรับทุกตำแหน่งที่เลื่อนรูปแบบไปบนภาพก็จะมีการเปรียบเทียบกับข้อมูลภาพ ถ้าข้อมูลภาพมีค่าเหมือนกับรูปแบบที่กำหนดจะทำการกำหนดค่าข้อมูลภาพในตำแหน่งที่ตรงกับจุดเริ่มต้น (Origin) ของเทมเพลต (Template) ถูกกำหนดให้มีค่าเท่ากับ 1



รูปที่ 2.12 เปรียบเทียบภาพก่อนและหลังการย่อภาพ

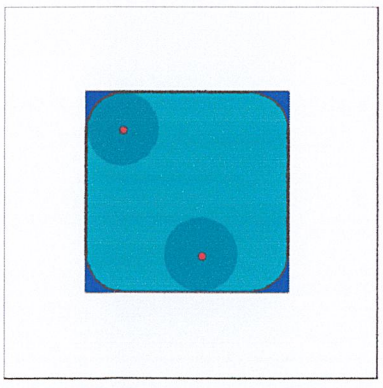
2.7.7.3 โอเปอเรชั่น โคลด (Closing) และโอเพน (Opening)

- **โอเปอเรชั่นการ โอเพน (Open)**

กำหนดให้ OPEN (I, T) เป็นการกระทำโอเพน (Opening) ของภาพ I โดยใช้รูปแบบ T ซึ่งมีลักษณะดังสมการต่อไปนี้

$$OPEN\ (I, T) = D(E(I)) \qquad (2.8)$$

จากสมการจะได้ว่าการทำโอเปอเรชั่น OPEN คือการนำข้อมูลภาพ I ผ่านการทำขยายภาพ (Erosion) แล้วตามด้วยการย่อภาพ (Dilation) โดยใช้เทมเพลต (Template) ชุดเดียวกันคือ T



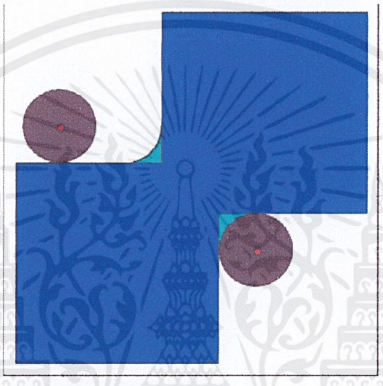
รูปที่ 2.13 เปรียบเทียบภาพก่อนและหลังการโอเพน

- **โอเปอเรชั่นการ โคลส (Closing)**

กำหนดให้ CLOSE (I, T) เป็นการกระทำแบบโคลส (Closing) ของภาพ I โดยใช้รูปแบบ T ซึ่งมีลักษณะดังสมการต่อไปนี้

$$CLOSE (I, T) = E(D(I)) \quad (2.9)$$

จากสมการจะได้ว่าการทำโอเปอเรชั่น CLOSE คือการนำข้อมูลภาพ I ผ่านการทำการย่อภาพ (Dilation) แล้วตามด้วยการขยายภาพ (Erosion) โดยใช้รูปแบบชุดเดียวกันคือ T



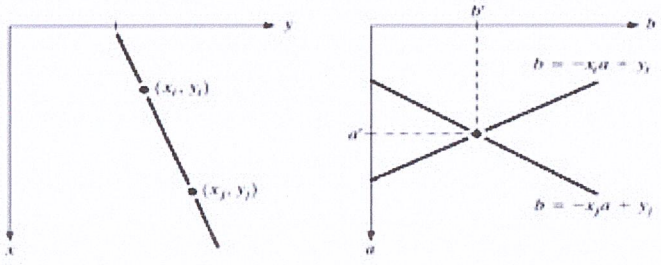
รูปที่ 2.14 เปรียบเทียบภาพก่อนและหลังการโคลส

2.7.8 ฮัฟทรานสฟอร์ม (Hough Transform)

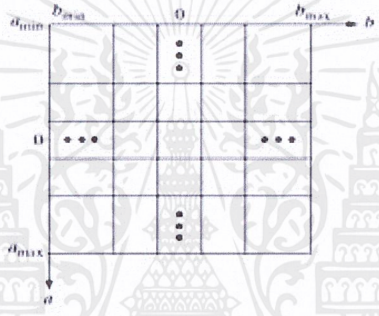
ให้ภาพภาพหนึ่งมีทั้งหมด n จุดโดยที่เราจะหาเส้นตรงทุกเส้นที่อยู่ในภาพ ซึ่งมีอาจจะมีบางคำตอบที่เป็นไปได้โดยการหาทุกบรรทัดที่เป็นไปได้จากจุดที่ เป็นคู่แล้วนำมาหา ซับเซต (subset) ของจุดเหล่านั้นเพื่อดูว่าต่อกันเป็นเส้นขึ้นมาหรือไม่ ซึ่งในวิธีขั้นต้นก็มีปัญหาอยู่ที่วุ่นวายในการหามากเพราะเราอาจต้องหาถึง n(n-1)/2 ~n2 เส้นหรือในบางครั้งที่เราต้องการให้มีคุณภาพดีขึ้นก็อาจจะต้องหาถึง n(n(n-1))/2~n3 เส้นเพื่อนำมาเปรียบเทียบทุกจุดในบรรทัดนั้นที่เดียว

Hough [1962] ได้เสนอวิธีการของตนโดยใช้ชื่อว่า ฮัฟทรานสฟอร์ม (Hough Transform) โดยจะประกอบไปด้วยจุดพิกัด (xi , yi) และสมการของเส้นตรง yi = axi + b สำหรับซึ่งจะมีเส้นที่ลากผ่านจุดนี้ได้มากมายและจะทำให้มีค่าของ a และ b เป็นอย่างมาก อย่างไรก็ตามเราก็สามารถเขียนสมการใหม่ได้เป็น b = -xia + yi โดยจะพิจารณาจากส่วน ab-plane (ในที่นี้จะเรียกว่า parameter space) ซึ่งเราจะทำการแทนสมการแค่เส้นเดียวเพื่อจะฟิก (xi , yi) นอกจากนี้ในจุดที่สอง (xj , yj) ก็กำหนดค่าของ พารามิเตอร์สเปซ (parameter space) ให้ซึ่งจะกำหนดค่าเพื่อให้สมการทั้ง

สองมีจุดตัดที่ (a' , b') โดยที่ a' เป็นค่าของความชันและ b' เป็นค่าของระยะห่างแกนตั้ง แล้วจึงใส่ทั้งสองสมการลงใน xy-plane ในความเป็นจริงทุกจุดจะอยู่ในส่วนที่มีการตัดกันของสมการแล้วมาอินเตอร์เซก (intersect) กันที่จุด (a' , b') ดังรูปที่ 2.15



รูปที่ 2.15 ระนาบ xy ของฮัฟทรานสฟอร์ม



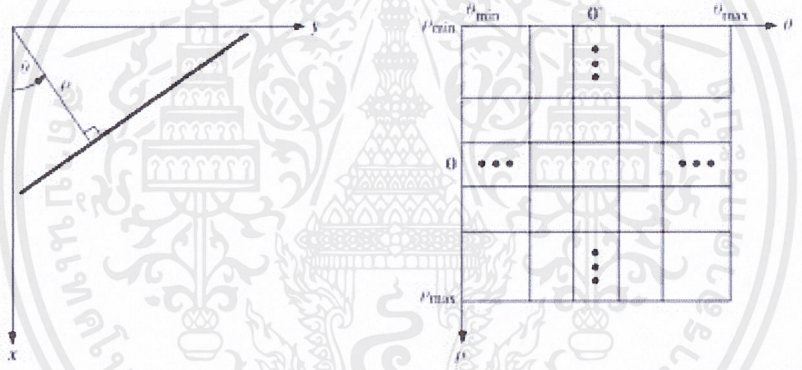
รูปที่ 2.16 ระนาบส่วนย่อยสำหรับใช้ในฮัฟทรานสฟอร์ม

ในการคำนวณของฮัฟทรานสฟอร์มมีความน่าสนใจเนื่องจากรายละเอียดของส่วนพารามิเตอร์สเปซ ซึ่งในต่อมาเรียกว่า แอคคิวมูเรตเซล (accumulator cells) ดังรูปที่ 10.18 โดย (amin , amax) และ (bmin , bmax) คือค่าที่กำหนดช่วงของค่าความชันและค่าของส่วนที่มีการขัดแย้งกัน cell ที่พิกัด (i , j) กับค่า accumulator A(i , j) ก็มีความเกี่ยวข้องกันเหมือนกับค่าพารามิเตอร์สเปซกับพิกัด (ai , bj) ในเริ่มต้นค่าของ cell จะมีค่าเป็น 0 ดังนั้นสำหรับทุกจุด (xk , yk) ในภาพเราสามารถกำหนดค่า a ให้มีค่าอยู่ในแกน a และหาค่า b ได้จากสมการ b = -xka + yk ซึ่งถ้าผลลัพธ์ของไม่ได้อยู่ในช่วงของแกน b เราก็กำหนดให้ค่านั้นมีค่าใกล้เคียงที่สุดที่อยู่บนแกน b โดยถ้าผลลัพธ์ที่ได้จาก ap เป็น bq แล้วเราก็จะให้ A(p , q) = A(p , q) + 1 เมื่อถึงที่สุดแล้วนั้นค่าของ Q อยู่ใน A(i , j) ให้ทำการแปลงค่า Q ที่สอดคล้องกันไปอยู่บน xy-plane ในสมการ y = aix + bj และส่วนของค่าที่อยู่บน ab-plane ก็ถือเป็นค่าที่แม่นยำในการหาจุดเชื่อมต่อของจุดที่เป็นเส้น

ปัญหาในการใช้สมการ y = ax + b ในการอธิบายเส้น และความชันทำให้เกิดเส้นที่มากมายในแนวตั้ง ซึ่งมีวิธีที่ทำให้เรื่องที่กล่าวมาง่ายขึ้นด้วยการอธิบายเป็นดังสมการที่ 2.10

$$x\cos\theta + y\sin\theta = \rho \qquad (2.10)$$

โดยในรูปที่ 2.17 เป็นตัวอย่างในการภาพเลขาคณิตที่อธิบายเกี่ยวกับตัว แปรในสมการด้านบนซึ่งภาพนี้ได้อธิบายถึงส่วนประกอบภายในตาราง แอคคิวมูเรเตอร์ (accumulators) ที่ระบุถึงค่าความชันผกผัน (slope-intercept) และส่วนที่เป็นเส้นตรงก็จะเป็นส่วนปลายของเส้นโค้งใน ρθ-plane โดยก่อนหน้านี้ค่าของ Q ที่เป็นจุดเชื่อมของเส้นก็ถูกเปลี่ยนไปเป็น xcosθ_j+ysinθ_j=ρ_i เมื่อ Q เป็นปลายของส่วนโค้งและอินเตอร์เซก(intersect) ที่ (ρ_i , θ_j) ในส่วนของพารามิเตอร์สเปซเมื่อเพิ่มค่าของ θ และหาค่าของ ρ จากค่าของ Q ที่มาจาก accumulator A(i , j) ที่เกี่ยวกับ cell ที่กำหนดโดย (ρ_i ,θ_j) ในภาพ (b) เป็นภาพของส่วนของค่าพารามิเตอร์สเปซ



รูปที่ 2.17 เส้นแบบปกติและส่วนของระนาบ p-θ ในเซลล์

ช่วงของมุมของ θ คือ ±90 องศา ซึ่งจะเทียบกับแกน x ดังนั้นเมื่อเทียบกับภาพ ด้านบนซ้ายเส้นในแนวแกนตั้งก็จะมีค่า θ = 0 องศา และค่า ρ ก็จะมีค่าเท่ากับค่าความยาวในแนวแกน x แต่ถ้าเส้นตรงเป็นเส้นในแนวแกนนอนค่าของ θ = 90 องศาและค่าของ ρ ก็จะมีค่าเท่ากับค่าในแกน y

ในรูปที่ 2.18 เป็นรูปของฮัฟทรานสฟอร์มบนพื้นฐานของสมการด้านบนโดยรูป บนซ้ายเป็นภาพที่แสดงให้เห็นถึง 5 labels point โดยแต่ละจุดจะนำไปเปรียบเทียบลงบนระนาบ ρθ-plane ดังในรูปบนขวาซึ่งช่วงของค่า θ คือ ±90 องศา และช่วงของค่า ρ-axis คือ ±√2D เมื่อ D คือค่าระยะห่างระหว่างมุมของภาพ ซึ่งจะแตกต่างกับทรานฟอร์ม (Transform) ที่ทำบนพื้นฐานของการใช้ค่าความ-ชันผกผัน (slope-intercept) โดยแต่ละส่วนโค้งจะมีลักษณะที่แตกต่างกัน ซึ่งผลลัพธ์

ของเส้นในแนวตั้งจากการเปรียบเทียบของจุดที่ 1 จะทำให้ได้ผลลัพธ์ที่มีค่าแอมพลิจูด (amplitude) เป็น 0

การหาโคลิเนียลิตี (colinearity) ของฮัฟทรานสฟอร์มในภาพล่างซ้าย จุด A (อย่าเพิ่งสับสนตัว A กับค่าของ accumulator) ซึ่งแสดงถึงการอินเตอร์เซกกันของเส้น 1,3,5 บนแกน xy-plane จากจุด A ที่ได้มาจะพบว่าเส้นตรงที่ได้มานั้นได้ลากผ่านจุดเริ่มต้นจึงทำให้มีค่า $\rho = 0$ และมีค่า $\theta = -45$ องศาและเช่นเดียวกันที่จุด B ก็เป็นจุดที่เกิดจากการอินเตอร์เซกกันระหว่างเส้น 2,3,4 ทำให้พบเส้นที่มีค่า $\theta = 45$ องศาและมีระยะห่างเป็นครึ่งนึงของความยาวของเส้นทแยงมุมระหว่างจุดเริ่มต้นกับมุมตรงข้ามของภาพ

สุดท้ายในส่วนของภาพล่างขวาแสดงให้เห็นถึงการกลับด้านกันของเส้น A, B, C ที่แสดงในภาพล่างขวาเมื่อมีการเปลี่ยนค่าของ θ ก็จะทำให้ค่าของ ρ เปลี่ยนไปด้วยเช่นกัน

รูปที่ 2.18 ตัวอย่างภาพของฮัฟทรานสฟอร์ม

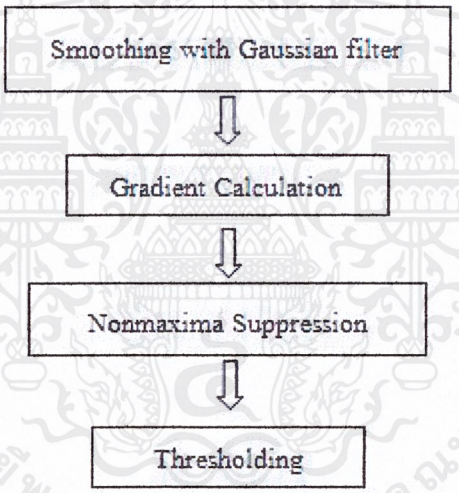
ถึงแม้ว่าจุดที่เราสนใจจะยังห่างไกลจากส่วนที่เป็นเส้นตรง แต่ฮัฟทรานสฟอร์มก็ยังสามารถปรับใช้กับฟังก์ชันอื่นๆได้อีกเช่นกันเช่น $g(v , c) = 0$ เมื่อ v เป็นเวกเตอร์ของค่าพิกัดและ c เป็นค่าเวกเตอร์ของสัมประสิทธิ์ สำหรับตัวอย่างสมการดังนี้

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2 \qquad (2.11)$$

สามารถหาได้โดยใช้ขั้นตอนที่ได้กล่าวไปแล้ว แต่มีความแตกต่างอยู่ที่ค่าของพารามิเตอร์นั้นจะมีอยู่ 3 ค่าคือ c1 , c2 , c3 ซึ่งจะทำให้ได้ผลลัพธ์มาเป็น 3-D พารามิเตอร์สเปซที่มีลักษณะเป็นลูกบาศก์ และตัวแอคคิวมูเรเตอร์คือ A(i , j , k) โดยในการคำนวณจะค่อยๆเพิ่มค่าของ c1 , c2 แล้วจึงหาค่า c3 จากสมการที่ด้านบนและอัพเดทค่าแอคคิวมูเรเตอร์ให้มีความสอดคล้องกับเซลล์ที่เกี่ยวข้องกับค่า triplet (c1 , c2 , c3) เพื่อขจัดความซับซ้อนของฮัฟทรานสฟอร์มเพื่อจะให้ได้ค่าของพิกัดและค่าของสัมประสิทธิ์ในฟังก์ชันที่เราได้อธิบายไปในที่นี้ นอกจากนี้ลักษณะทั่วไปของฮัฟทรานสฟอร์มยังหาเส้นโค้งกับการวิเคราะห์ที่ไม่สามารถอธิบายได้โดยง่ายให้เป็นไปได้ ซึ่งทำให้เป็นแอปพลิเคชันของทรานฟอร์มโดยเป็นภาพระดับเทา

2.7.9 วิธีการหาขอบภาพแบบแคนนี่ (Canny Edge Detection Algorithm)

ขั้นตอนการหาขอบโดยวิธีของแคนนี่ประกอบด้วย 4 ขั้นตอนดังรูปที่ 2.19



รูปที่ 2.19 ขั้นตอนการหาขอบโดยวิธีของแคนนี่

การทำงานของ แคนนี-เอด-ดีเทคชัน (Canny edge detection) นั้นเริ่มต้นจากการปรับภาพให้เรียบ (Smoothing) ด้วยตัวกรองเกาเซียน (Gaussian filter) เพื่อกำจัดสัญญาณรบกวน หลังจากนั้น คำนวณค่าขนาด (magnitude) และทิศทาง (orientation) ของ การ์เดียน (gradient) โดยใช้การหาอนุพันธ์อันดับหนึ่ง ในถัดมาจึงใช้ นอนแมกซีมาซุบเพรซชัน (nonmaxima suppression) กับการ์เดียน-แมกนีจูด (gradient magnitude) เพื่อทำให้ได้ขอบที่บางลงและในขั้นตอนสุดท้ายใช้ดับเบิล-เทรดโชอัลกอริทึม(double thresholding algorithm) เพื่อระบุพิกเซลที่เป็นขอบและช่วยเชื่อมต่อขอบโดยในแต่ละขั้นตอนมีรายละเอียดดังต่อไปนี้

2.7.9.1 การปรับภาพให้เรียบ (Smoothing)

ในขั้นตอนแรกของการหาขอบโดยอัลกอริทึมนี้จะต้องกำจัดสัญญาณรบกวนออกก่อน โดย ใช้เกาเชียนฟิวเตอร์ (Gaussian filter) ซึ่งสามารถคำนวณได้จากการใช้กรอบ (mask) ขนาดเล็ก ขนาดของกรอบเกาเชียน (Gaussian mask) นี้หากมีขนาดกว้างจะมีผลทำให้ลดสัญญาณรบกวนได้มาก แต่ถ้ากว้างมากเกินไปจะมีผลทำให้ขอบย่อยๆ ที่เป็นส่วนรายละเอียดนั้นหายไป สำหรับการคำนวณหาภาพที่ได้จากการใช้ เกาส์เชียนฟิวเตอร์ (Gaussian filter) เป็นดังสมการดังนี้

$$S[i, j] = G[i, j,\sigma]* I[i,j] \qquad (2.12)$$

โดยกำหนดให้

I[i, j] เป็นภาพที่ต้องการหาขอบ

G[i, j, σ] เป็น Gaussian smoothing filter

σ เป็น spread of the Gaussian (ควบคุมระดับของการ smoothing)

2.7.9.2 การคำนวนการ์เดียน (Gradient Calculation)

ในขั้นแรกนำภาพที่ผ่านการทำให้เรียบมาแล้ว S[i, j] มาสร้าง x, y partial derivatives P[i, j] และ Q[i, j] ตามลำดับ ดังสมการที่ 2.13 และ 2.14

$$P [i , j] \approx (S[i, j+1] - S[i, j] + S[i+1, j+1] - S[i+1, j]) / 2 \qquad (2.13)$$

$$Q [i , j] \approx (S[i, j] - S[i+1, j] + S[i, j+1] - S[i+1, j+1]) / 2 \qquad (2.14)$$

หลังจากนั้นนำค่า x, y partial derivatives มาคำนวณด้วยสูตรมาตรฐานสำหรับการแปลงรูปแบบจาก rectangular ไปเป็น polar (rectangular-to-polar conversion) เพื่อหาขนาดและทิศทางของการ์เดียนตามสมการที่ 2.15 และ 2.16

$$M[i, j] = \sqrt{P[i, j]^2 + Q[i, j]^2} \qquad (2.15)$$

$$\theta[i, j] = \arctan(Q[i, j], P[i, j]) \quad (3.21) \qquad (2.16)$$

จากสมการข้างต้นจะสามารถหาค่ามุม θ ออกมาได้เมื่อแทนค่าตัวแปรในฟังก์ชั่น arctan(x , y)

2.7.9.3 นอนแมกซีมา ซุปเพรตชัน (Nonmaxima Suppression)

สำหรับการหาขอบโดย แคนนีเมททอต(Canny method) จุดที่ถือเป็นเส้นขอบได้นั้นต้องเป็นจุดที่ให้ค่าสูงสุดเฉพาะที่และเป็นทิศ ทางเดียวกับการ์เดียนด้วย ซึ่งด้วยวิธีดังกล่าวนี้ทำให้

ได้ขอบที่บางเพียง 1 พิกเซล ภาพที่ได้หลังการทำนอนแมกซีมา ซุปเพรตชันจะให้ค่าเป็นศูนย์ในทุกจุดยกเว้นจุดที่เป็น จุดโลคอลแมกซีมา (local maxima points) ซึ่งจะยังคงค่าเดิมไว้

2.7.9.4 การทำเทรดโช (Thresholding)

แม้ว่าภาพจะผ่านการทำให้เรียบในขั้นตอนแรกแล้วก็ตาม ภาพที่ได้อาจยังมีเส้นขอบที่ไม่ใช่ขอบที่แท้จริงปรากฏอยู่อันเนื่องจาก สัญญาณรบกวนหรือลักษณะของวัตถุในภาพเป็นพื้นผิวที่มีลวดลายหรือมีรายละเอียด ภายในมาก ดังนั้นเพื่อลดปัญหาดังกล่าวจึงได้มีการกำหนดค่าเทรดโชขึ้นมา 2 ค่า คือค่าเทรดโชสูง (T1) และ ค่าเทรดโชต่ำ (T2) โดยพิกเซลที่มีค่ามากกว่า T1 จะถูกปรับเป็น 1 (เป็นพิกเซลที่เป็นขอบ) แต่ถ้าน้อยกว่า T2 จะถูกปรับเป็น 0 ส่วนค่าที่อยู่ระหว่างค่าเทรดโชทั้งสอง การปรับเป็นค่า 0 หรือ 1 นั้นขึ้นอยู่กับพิกเซลที่อยู่รอบข้าง หากพบว่าพิกเซลที่อยู่รอบข้างของพิกเซลที่เป็นขอบ (ค่า >T1) มีค่ามากกว่า T2 แล้ว จะปรับค่าพิกเซลดังกล่าวให้มีค่าเป็น 1 และถือเป็นสมาชิกหนึ่งในภาพขอบด้วยเช่นกัน

2.7.10 โปรแกรม โอเพนซีวี (OpenCV)

โอเพนซีวี (OpenCV) สร้างขึ้นเพื่อให้การพัฒนาโปรแกรมทางด้านการมองเห็นของคอมพิวเตอร์ (Computer Vision) เป็นไปได้อย่างสะดวก โอเพนซีวี (OpenCV) นี้พัฒนาขึ้นโดยได้รับการสนับสนุนจากอินเทล คอเปอร์เรชัน จำกัด ให้เป็นซอฟต์แวร์แบบเปิดเผยรหัส (Open Source) เพื่อให้สามารถนำไปต่อยอดพัฒนาโปรแกรมต่างๆ ได้ง่าย ใช้ได้บนระบบปฏิบัติการที่เป็นลินุกซ์และวินโดวส์ โอเพนซีวีประกอบด้วยโครงสร้างข้อมูล และอัลกอริทึม

- โครงสร้างข้อมูลใช้เก็บข้อมูลต่างๆ อาทิ เช่น รูปภาพ เมทริกซ์ พิกัด
- อัลกอริทึม เพื่อการประมวลผลต่าง โดยเฉพาะการประมวลผลทางรูปภาพ

ข้อจำกัดของโอเพนซีวีคือ สามารถใช้งานได้เฉพาะเครื่องคอมพิวเตอร์ที่ใช้หน่วยประมวลผล (CPU) ในตระกูลของ x86 แต่ข้อจำกัดนี้ทำให้เกิดจุดเด่นเช่นกัน กล่าวคือ การประมวลผลต่างๆ จะใช้ความสามารถของหน่วยประมวลผลอย่างเต็มที่ ทำให้โปรแกรมที่พัฒนาโดยการใช้โอเพนซีวีนี้มีประสทธิภาพในการประมวลผลที่สูงมาก

2.8 แคนบัส (CAN Bus)

2.8.1 ความเป็นมา

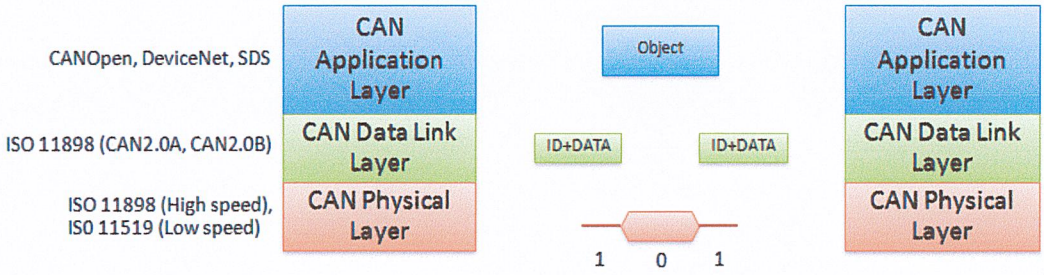
แคน ย่อมาจากคอนโทรลเลอร์แอเรียเนทเวิร์ค (Controller Area Network) เป็นบัสสื่อสารข้อมูลแบบอนุกรมที่ออกแบบมาสำหรับใช้ในงานควบคุมแบบเวลาจริง (realtime) ที่มีจุดเด่นอยู่ที่ความสามารถในการสื่อสารข้อมูลเป็นเครือข่าย (Network) โดยไม่ต้องมีหมายเลขที่อยู่ของโหนด (Node Addressing) อุปกรณ์ทุกตัวสามารถเรียกใช้งานบัสได้ (Multi-master) และร้องขอใช้งานบัสได้พร้อมกันหลายตัว ด้วยความเร็วในการสื่อสารข้อมูลสูงสุด 1 เมกะบิตต่อวินาที มีระบบป้องกันและตรวจสอบความผิดพาดที่มีประสิทธิภาพสูง

จุดเริ่มต้นของแคนนั้นถูกพัฒนาขึ้นโดยบริษัท โรเบริ์ต บอช (Robert Bosh) ในประเทศเยอรมันเมื่อประมาณ 20 ปีที่ผ่านมาสำหรับใช้ในระบบอิเล็กทรอนิกส์ภายในรถยนต์ เนื่องจากปัญหาในการพัฒนารถยนต์รุ่นใหม่ที่ต้องการสิ่งอำนวยความสะดวกและระบบรักษาความปลอดภัยเพิ่มขึ้น ส่งผลให้ระบบอิเล็กทรอนิกส์ภายในรถยนต์มีความซับซ้อนมากขึ้น มีมอดูลของวงจรต่างๆ เป็นจำนวนมากที่ต้องมีการสื่อสารข้อมูลระหว่างกัน

ด้วยเหตุผลนี้จึงมีความต้องการบัสสื่อสารข้อมูลที่มีประสิทธิภาพและความน่าเชื่อถือสูงในราคาที่เหมาะสม แทนที่ระบบบัสเดิมที่มีความยุ่งยากซับซ้อนและไม่สะดวกที่จะนำมาใช้และมีค่าใช้จ่ายสูง ซึ่งภายหลังนอกจากจะใช้ในอุตสาหกรรมรถยนต์แล้วยังได้มีการนำแคนไปประยุกต์ใช้ในงานอุตสาหกรรมอื่นๆอีกมากมายเนื่องจากคุณสมบัติที่โดดเด่นของบัสนี้ในด้านความน่าเชื่อถือและความยืดหยุ่น รวมทั้งความง่ายต่อการใช้งานและยังมีค่าใช้จ่ายต่ำอีกด้วย

เนื่องจากความแพร่หลายในการใช้งานบัสนี้ภายหลังจึงได้มีการกำหนดให้แคนบัสเป็นมาตรฐานระหว่างประเทศขึ้นมา นั่นคือมาตรฐาน ไอเอสโอ 11898 (ISO 11898) (ความเร็วสูง) และไอเอสโอ 11519 (ความเร็วต่ำ) โดยมาตรฐานนี้ได้ครอบคลุมข้อกำหนดการทำงานของแคนถึงระดับชั้นที่สอง (Layer 2) ตามแบบจำลอง โอเอสไอ (OSI) คือชั้นกายภาพ (Physical Layer) และชั้นเชื่อมโยงข้อมูล (Data Link Layer)

2.8.2 โครงสร้างการทำงาน



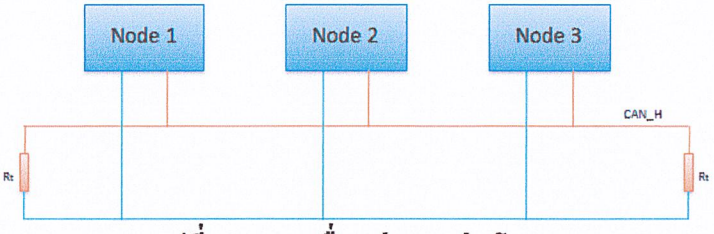
รูปที่ 2.20 ชั้นต่างๆในแคนเมื่อเทียบกับโอเอสไอโมเดล

โครงสร้างการทำงานของแคนเมื่อเทียบกับแบบจำลองโอเอสไอ ประกอบไปด้วยสามชั้น คือ ชั้นกายภาพ, ชั้นเชื่อมโยงข้อมูล และชั้นประยุกต์ใช้งาน (Application Layer) ดังโครงสร้างในรูปที่ 2.20

2.8.2.1 ชั้นที่ 1 ชั้นกายภาพ

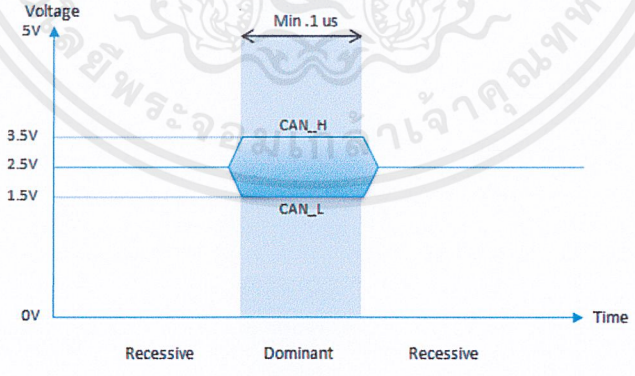
โครงสร้างการทำงานในชั้นนี้จะเป็นส่วนที่เกี่ยวข้องกับกายภาพของบัส ได้แก่ ตัวกลางที่ใช้ส่งผ่านสัญญาณข้อมูล ตัวเชื่อมต่อ (Connector) และการเชื่อมต่อสายสัญญาณ รวมทั้งคุณสมบัติทางไฟฟ้าของสัญญาณข้อมูล มาตรฐานที่เกี่ยวข้องกับชั้นนี้มี 2 มาตรฐาน คือ ไอเอสโอ 11519 สำหรับการสื่อสารข้อมูลความเร็วต่ำ มีความเร็วอยู่ที่ 5 ถึง 125 กิโลบิตต่อวินาที และไอเอสโอ 11898 สำหรับการสื่อสารข้อมูลความเร็วสูง มีความเร็วสูงสุดที่ 1 เมกะบิตต่อวินาที

ตัวกลางที่ใช้ในการส่งผ่านข้อมูลในแคนนั้นมีด้วยกันหลายรูปแบบ เช่นการใช้สายสัญญาณ 2 เส้น การใช้สายสัญญาณเส้นเดียว การใช้สายไฟเบอร์ออปติก หรืออาจไม่ใช่สายเลยก็เป็นได้ แต่ที่นิยมใช้มากที่สุดคือการใช้สายสัญญาณ 2 เส้นแบบตีเกลียว (Twisted-pair cable) ซึ่งจะขอใช้ในการอ้างอิงการเชื่อมต่อในที่นี้



รูปที่ 2.21 การเชื่อมต่อระหว่างโหนด

การส่งผ่านข้อมูลในแคนมีรูปแบบของการส่งข้อมูลเป็นแบบอนุกรมโดยใช้สาย 2 เส้น มีลักษณการเชื่อมต่อสายสัญญาณ (Topology) ดังรูปที่ 2.21 โดยอุปกรณ์ทุกตัวจะต่ออยู่บนสายสัญญาณคู่เดียวกัน และปิดปลายของคู่สายทั้งสองข้างด้วยตัวต้านทาน (Termination Impedance) โดยสัญญาณข้อมูลที่ส่งจะใช้วิธีการส่งแบบสัญญาณแรงดันผลต่าง (Differential Voltage Signal) ค่าข้อมูลในสายส่งได้จากการเปรียบเทียบระดับแรงดันหรือศักย์ไฟฟ้าระหว่างสายสัญญาณ CAN_H และ CAN_L สถานะของสัญญาณข้อมูลในสายส่งจะมีอยู่สองสถานะคือรีเซสสีฟ (Recessive) และโดมิแนนต์ (Dominant) โดยถ้าผลต่างของสัญญาณแรงดันระหว่าง CAN_H และ CAN_L น้อยกว่าหรือเท่ากับ 0.5 โวลต์ จะเป็นสถานะรีเซสซีฟซึ่งจะแทนข้อมูลที่มีลอจิกเป็น '1' ถ้าผลต่างแรงดันระหว่าง CAN_H และ CAN_L มากกว่าหรือเท่า 0.9 โวลต์ จะมีสถานะเป็นโดมิแนนต์ ซึ่งจะแทนข้อมูลที่มีลอจิกเป็น '0'

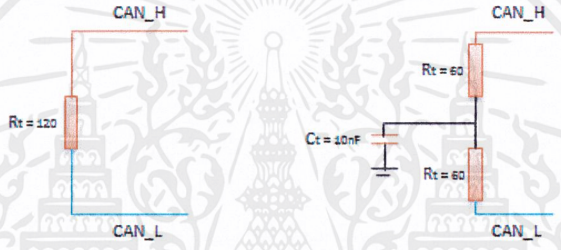


รูปที่ 2.22 ระดับสัญญาณของสถานะเด่นและด้อย

การส่งผ่านข้อมูลโดยใช้ผลต่างแรงดันของคู่สายมีข้อดีคือ จะช่วยลดการรบกวนอันเนื่องมากจากการรบกวนทางแม่เหล็กไฟฟ้า (EMI, Electromagnetic interference) ลงได้เป็นอย่าง

มาก ทำให้สามารถส่งสัญญาณได้ในอัตราเร็วที่สูงขึ้น และได้ระยะทางที่ไกลขึ้นอีกด้วย ทั้งนี้ความเร็วยังขึ้นอยู่กับความยาวของสายสัญญาณอีกด้วย ซึ่งการที่จะส่งข้อมูลได้ถึง 1 เมกะบิตต่อวินาทีนั้นจะต้องใช้สายส่งที่มีความยาวไม่เกิน 40 เมตร

การเดินสายของแคนบัสมีข้อควรระวัง คือ ในกรณีพ่วงสายจากบัสกลาง สายที่พ่วงไม่ควรมีความยาวเกิน 2 เมตร ถ้าใช้ความเร็วไม่เกิน 250 กิโลบิตต่อวินาที และ 30 เซนติเมตรเมื่อใช้ความเร็วสูงขึ้น ทั้งนี้ความยาวของสายพ่วงรวมกันทั้งหมดไม่ควรเกิน 30 เมตร และต้องมีการปิดปลายสายทั้งสองข้างด้วยตัวต้านทานดังรูปที่ 2.23 ซึ่งการต่อตัวต้านทานที่ปลายมีหลายวิธี แต่วิธีที่ได้รับความนิยมมีสองแบบดังรูป ซึ่งการใช้ตัวต้านทานค่า 60 โอห์ม สองตัวต่ออนุกรมกันและมีตัวเก็บประจุผ่านสัญญาณลงกราวด์ทางด้านขวาจะสามารถลดสัญญาณรบกวนได้ดีกว่า



รูปที่ 2.23 การปิดปลายสายด้วยวิธีต่างๆ

2.8.2.2 ชั้นที่ 2 ชั้นเชื่อมโยงข้อมูล

ในชั้นนี้จะเกี่ยวข้องกับโปรโตคอลและรูปแบบของข้อมูลในเชิงตรรกะ รวมถึงการป้องกันความผิดพลาดที่จะเกิดขึ้นกับข้อมูลเพื่อรับประกันว่าข้อมูลที่ได้มีความถูกต้องโดยข้อกำหนดของการทำงานในชั้นนี้ได้รวมอยู่ในมาตรฐาน ไอเอสโอ 11898 ด้วย และยังได้มีการแก้ไขและขยายข้อกำหนดของมาตรฐานในชั้นนี้เพิ่มเติม โดยจัดทำเป็นสองรุ่น คือ แคน 2.0เอ (CAN 2.0A) ซึ่งเป็นของเดิมและแคน 2.0บี (CAN 2.0B) เป็นรุ่นขยายเพิ่มเติม

รูปแบบการสื่อสารข้อมูลของแคนเป็นแบบอนุกรมแบบไม่ประสานเวลา (Asynchronous) เนื่องจากไม่มีการส่งสัญญาณนาฬิกาเพื่อใช้ในการเข้าจังหวะ แต่อาศัยการเข้าจังหวะจากจุดเริ่มต้นของแต่ละข้อความ โดยหลักการที่สำคัญที่ทำให้แคนเป็นบัสข้อมูลที่ทำงานได้อย่างมีประสิทธิภาพได้แก่ การสื่อสารแบบกระจายข้อมูล (Broadcast), การใช้บัสจากหลายโหนด (Multiple Bus Access), การสื่อสารที่อิงจากข้อความ (Message-based Communication) และการป้องกันความผิดพลาด (Error Protection)

การสื่อสารแบบกระจายข้อมูล คือ ทุกโหนดที่อยู่บนบัสสามารถรับข้อมูลที่ส่งมาจากผู้ส่งได้ หลังจากที่ได้รับข้อมูลแล้วเป็นหน้าที่ของแต่ละโหลดที่จะต้องตรวจสอบเองว่าเป็นข้อมูลที่ต้องการหรือไม่ ซึ่งการตรวจสอบตรงจุดนี้จะอาศัยตัวกรอง (Acceptance Filter) ทำหน้าที่กันข้อมูลที่ไม่ต้องการทิ้ง และรับเฉพาะข้อมูลที่ต้องการเท่านั้น โดยดูจากค่ารหัสประจำตัว (Identifier) ของข้อมูลหรือข้อความนั้นๆ นอกจากนั้นยังสามารถร้องขอข้อมูล (Remote Request) จากโหลดที่มีข้อมูลดังกล่าวได้อีกด้วย โดยการส่งค่ารหัสประจำตัวของข้อมูลที่ต้องการทราบออกไป หากโหนดใดมีข้อมูลดังกล่าวก็จะส่งข้อมูลนั้นตอบกลับมาซึ่งนอกจากโหนดที่ร้องขอข้อมูลนี้ไปแล้ว โหนดอื่นๆยังสามารถรับข้อมูลดังกล่าวได้อีกด้วยเนื่องจากเป็นส่งข้อมูลแบบกระจายข้อมูลนั่นเอง

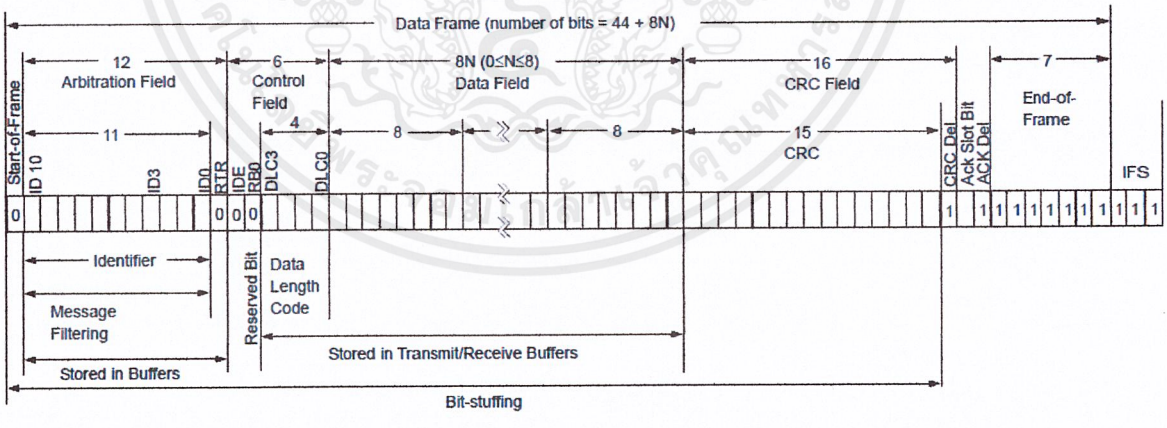
การเข้าใช้บัสจากหลายโหนด ถือได้ว่าเป็นจุดเด่นอย่างหนึ่งของแคน โดยในกรณีที่มีโหนดมากกว่าหนึ่งโหนดต้องการส่งข้อมูลพร้อมกัน จะมีวิธีที่ตัดสินเด็ดขาดว่าโหนดใดได้สิทธิในการส่งนั้น ในขณะที่โหนดอื่นจะต้องหยุดทำการส่งและรอที่จะส่งข้อมูลนั้นในภายหลังเมื่อบัสว่างแล้ว การใช้บัสในแคนจะใช้วิธีที่เรียนว่า ซีเอสเอ็มเอ ซีดี (CSMA/CD, Carrier Sense Multiple Access with Collision Detection) กล่าวคือ เมื่อโหนดใดต้องการใช้บัส จะต้องทำการตรวจสอบจนพบว่าบัสว่างไม่ได้ถูกใช้งานอยู่เป็นระยะเวลาหนึ่งจึงเริ่มทำการส่งข้อมูลออกไป ซึ่งขณะที่บัสว่างอยู่นี้แต่ละโหนดมีสิทธิที่จะส่งข้อมูลออกมาพร้อมกัน ซึ่งถ้ามีการส่งข้อมูลออกมาพร้อมกันโหนดจะทราบได้ว่ามีการชนกันเกิดขึ้นและดำเนินการแก้ไขการชนกันของข้อมูลต่อไป ในแคนจะใช้ค่าลำดับความสำคัญของข้อมูลเป็นตัวตัดสิน (AMP, Arbitration on Message Priority) ซึ่งค่าลำดับความสำคัญนั้นจะถูกระบุอยู่ในค่ารหัสประจำตัว (Identifier) ของข้อมูลนั้น โดยการตัดสินชี้ขาดจะเกิดขึ้นโดยไม่ทำลายโอกาสในการส่งข้อมูลของข้อมูลที่มีค่าลำดับความสำคัญสูงกว่า (Non-destructive Arbitration) และการตัดสินจะเกิดขึ้นในระดับบิตของข้อมูล (Bitwise Arbitration) ซึ่งกระบวนการเช่นนี้ ต้องอาศัยคุณสมบัติสำคัญสองประการ คือ สถานะของสัญญาณข้อมูลที่เป็นแบบเด่น (Dominant) และสัญญาณข้อมูลที่เป็นแบบด้อย (Recessive) โดยแคนกำหนดให้สถานะเด่นแทนลอจิก ‘0’ และด้อยแทนลอจิก ‘1’ ในการใช้งานเมื่อสัญญาณเด่นและด้อยถูกส่งออกมาพร้อมกัน บัสจะมีสถานะเป็นเด่นเสมอ กล่าวคือถ้าส่งลอจิก ‘0’ พร้อมกับลอจิก ‘1’ ค่าที่ได้ออกมาจะเป็น ‘0’ เสมอ และอีกประการหนึ่ง คือ โหนดผู้ส่งสามารถตรวจสอบสถานะของบัสตลอดเวลาว่าข้อมูลที่อยู่ในบัสตรงกับข้อมูลที่ส่งออกไปหรือไม่หากไม่ตรงกันแสดงว่ามีการชนกันของข้อมูลหรือมีความผิดพลาดเกิดขึ้น โดยการส่งข้อมูลจะเริ่มต้นด้วยการส่งข้อมูลรหัสประจำตัวของข้อมูลออกมาก่อน ในขณะเดียวกันก็จะคอยตรวจสอบข้อมูลที่อยู่บนบัสว่ามีค่าตรงกับค่าที่ส่งออกไป

หรือไม่ ถ้ามีหลายโหนดเริ่มส่งข้อมูลพร้อมๆกัน ก็จะทำการส่งตั้งแต่บิตแรกไปเรื่อยๆจนกระทั่งถึงบิตที่ไม่เหมือนกัน เช่นโหนดที่หนึ่งส่ง '0' ในขณะที่โหนดที่สองส่ง '1' สถานะของบัสจะมีค่าเป็น '0' ดังนั้นเมื่อโหนดที่สองตรวจได้ว่าค่าข้อมูลในบัสไม่เป็นไปตามที่ต้องการก็จะทำการหยุดส่ง ดังนั้นโหนดที่หนึ่งจึงสามารถส่งข้อมูลต่อได้ตามปกติ จากตัวอย่างที่กล่าวไปจะเห็นได้ว่ายิ่งรหัสประจำตัวมีค่าน้อยยิ่งมีโอกาสในการส่งข้อมูลสูง หรือยิ่งมีระดับความสำคัญสูงนั่นเอง

แคนจะแบ่งข้อมูลที่ส่งออกเป็นแพ็กเก็จ (Packets) หรือเฟรม (Frames) ซึ่งมีทั้งหมด 4 ชนิดได้แก่ เฟรมข้อมูล (Data Frame) ใช้ในการส่งข้อมูลตามปกติ, เฟรมร้องขอข้อมูล (Remote Frame) ใช้ในการร้องขอข้อมูลจากโหนดอื่น, เฟรมแสดงความผิดพลาด (Error Frame) ใช้แสดงว่ามีความผิดพลาดเกิดขึ้นและเฟรมโอเวอร์โหลด (Overload Frame) ใช้เพื่อบอกว่าต้องการเวลาในการประมวลผลข้อมูลที่ได้รับเพิ่มขึ้น

ในเฟรมข้อมูลนั้นจะประกอบไปด้วยส่วนย่อยๆที่เรียกว่าฟิลด์ (Fields) ดังในรูปที่ 2.24 และรูปที่ 2.25 ซึ่งแสดงส่วนประกอบของเฟรมข้อมูลมาตรฐาน (Standard Frame) และเฟรมข้อมูลขยาย (Extended Data Frame) ตามลำดับเฟรมข้อมูลทั้งสองแบบจะมีข้อแตกต่างกันที่ขนาดของรหัสประจำตัวของข้อมูล

- **เฟรมข้อมูลมาตรฐาน แคน2.0เอ**



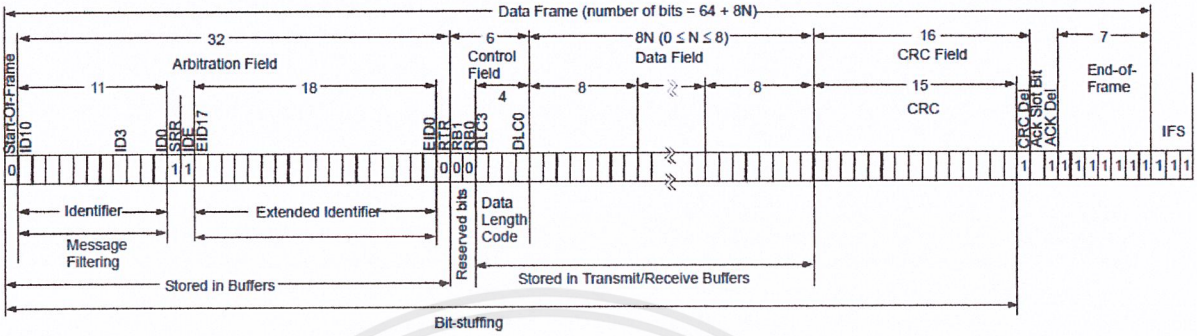
รูปที่ 2.24 เฟรมข้อมูลมาตรฐาน

ส่วนประกอบต่างๆในเฟรมข้อมูลมีดังนี้

- บิตเริ่มต้นเฟรม (SOF, Start Of Frame) มีสถานะเป็นสถานะเด่น หรือลอจิก '0' ซึ่งจะเป็นจุดที่ทุกโหนดในบัสใช้ในการเริ่มต้นเข้าจังหวะสำหรับการรับ-ส่งข้อมูล
- ฟิลด์แสดงรหัสข้อมูล (Arbitration Field) มีขนาด 12 บิต ประกอบด้วยหมายเลขรหัสประจำข้อความ ทำให้มีหมายเลขได้ทั้งหมด 2048 หมายเลข โดยมีการสำรองเอาไว้ 16 หมายเลขสำหรับใช้งานฟังก์ชั่นพิเศษเฉพาะอย่าง จึงเหลือให้ใช้งานได้ 2032 หมายเลข และมีบิตร้องขอการตอบกลับ (RTR, Remote Transmission Request) อีก 1 บิต
- ฟิลด์ควบคุม (Control Field) มีขนาด 6 บิต ประกอบด้วยบิตแสดงส่วนขยาย (IDE, Identifier Extension) 1 บิต, บิตสงวน0 (RB0, Reserved Bit 0) ซึ่งถูกสำรองไว้ใช้ในอนาคต และส่วนบอกความยาวข้อมูล DLC (Data Length Code) จำนวน 4 บิตกำหนดได้ตั้งแต่ 0-8 ไบต์
- ฟิลด์ข้อมูล (Data Field) เป็นส่วนของข้อมูลที่ต้องการส่งมีขนาดไม่เกิน 8 ไบต์
- ฟิลด์ตรวจสอบ (CRC Field) มีขนาด 16 บิต เป็นส่วนที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่งในเฟรมนั้น ประกอบด้วยค่าซีอาร์ซี (CRC, Cyclic Redundancy Check) ขนาด 15 บิต และอักขระคั่นซีอาร์ซี (CRC Delimiter) อีก 1 บิต การส่งข้อมูลในแต่ละเฟรมจะมีการคำนวณค่าซีอาร์ซีเริ่มตั้งแต่บิตแรกไปจนถึงฟิลด์ข้อมูล โดยสามารถตรวจพบความผิดพลาดของข้อมูลที่เกิดขึ้นในช่วงดังกล่าวได้หากมีความผิดพลาดไม่เกิด 5 บิต
- ฟิลด์ตอบรับ (Acknowledge Field) มีขนาด 2 บิต เป็นส่วนที่โหนดผู้รับตอบกลับไปยังผู้ส่งว่าข้อมูลที่ส่งมาได้รับถูกต้องหรือไม่ ประกอบด้วยช่วงตอบรับ (ACK Slot) 1 บิต, และอักขระคั่นการตอบรับ(ACK Delimiter) อีก 1 บิต เมื่อโหนดผู้ส่งอ่านค่าช่วงตอบรับกลับมาได้เป็นสถานะเด่น แสดงว่ามีอย่างน้อย 1 โหนดที่ได้รับข้อมูลถูกต้อง
- ฟิลด์สิ้นสุดเฟรม (EOF, End Of Frame) ประกอบด้วยบิตด้อย 7 บิตติดต่อกัน เป็นส่วนที่แสดงว่าได้สิ้นสุดการส่งเฟรมนี้แล้วและเพื่อเป็นการให้เวลาโหนด

ผู้รับสำหรับทำการประมวลผล หรือจัดการกับข้อมูลที่ได้รับมาให้เป็นที่เรียบร้อยเสียก่อน

- **เฟรมข้อมูลขยาย แคน2.0บี**

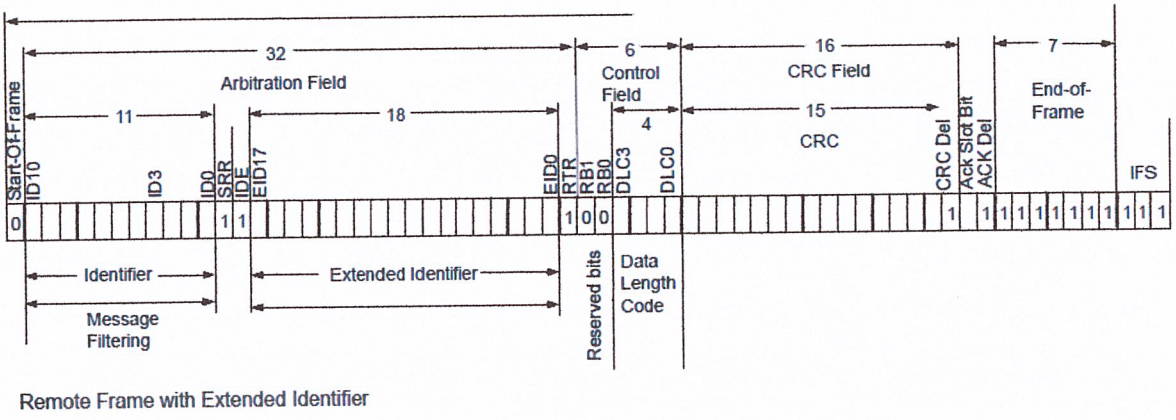


รูปที่ 2.25 เฟรมข้อมูลขยาย

ส่วนประกอบต่างๆที่แตกต่างกันกับเฟรมข้อมูลมาตรฐาน มีดังนี้

- ฟิลด์แสดงรหัสข้อมูล จะมีขนาด 32 บิตแทนที่จะเป็น 12 บิต โดยแบ่งเป็นหมายเลขรหัสประจำข้อความ 29 บิต ซึ่งจะถูกแบ่งออกเป็นสองส่วนคือ 11 บิต และ 18 บิต เพื่อให้สอดคล้องกับเฟรมข้อมูลมาตรฐาน, บิตทดแทนบิตร้องขอการตอบกลับ (SRR, Substitute Remote Request) 1 บิต จะอยู่ในตำแหน่งเดียวกับบิตร้องขอการตอบกลับในเฟรมแบบมาตรฐานโดยจะมีค่าเป็นสถานะด้อย, บิตแสดงส่วนขยาย 1 บิตเช่นเดียวกับเฟรมข้อมูลมาตรฐาน ส่วนบิตร้องของการตอบกลับจะอยู่ต่อจากหมายเลขรหัสประจำข้อความส่วนที่สอง
- ฟิลด์ควบคุม มีขนาด 6 บิตเช่นกัน ส่วนที่ต่างออกไปจากเฟรมข้อมูลมาตรฐาน คือ บิตสงวน1 (RB1, Reserved Bit 1) ซึ่งอยู่ในตำแหน่งเดียวกับบิตแสดงส่วนขยายในฟิลด์ควบคุมของเฟรมข้อมูลมาตรฐานเนื่องจากบิตแสดงส่วนขยายได้ถูกย้ายไปอยู่ในส่วนของฟิลด์แสดงรหัสข้อมูลแล้ว ดังนั้นจึงกำหนดให้เป็นบิตสงวน1 สำรองไว้ใช้ในอนาคต

- **เฟรมร้องขอข้อมูล**



รูปที่ 2.26 เฟรมร้องขอข้อมูล

โหนดผู้รับสามารถร้องขอข้อมูลจากโหนดที่มีข้อมูลนั้นอยู่ เพื่อให้โหนดนั้นส่งข้อมูลที่ต้องการมาให้ โดยการส่งเฟรมร้องขอข้อมูลที่มีหมายเลขรหัสประจำข้อความที่ต้องการนั้นออกไป หากโหนดใดมีข้อมูลที่มีหมายเลข ID ตรงกับที่ถูกร้องขอมาก็จะทำการส่งข้อมูลนั้นตอบกลับไป ลักษณะของเฟรมร้องขอข้อมูล จะคล้ายกับเฟรมข้อมูล แต่จะต่างกันที่บิตร้องขอการตอบกลับ จะมีค่าเป็นสถานะเด่น และในเฟรมร้องขอข้อมูลจะไม่มีข้อมูล

- **การเติมบิตแทรก (Stuff Bit)**

เนื่องจากแคนส่งสัญญาณแบบไม่กลับสู่ศูนย์ (NRZ, Non Return to Zero) ที่ระดับของสัญญาณแต่ละบิตมีค่าคงที่ตลอดช่วงเวลา และเป็นการส่งข้อมูลแบบไม่ประสานเวลาซึ่งอาจทำให้เกิดปัญหาการเข้าจังหวะเวลาในกรณีที่ส่งข้อมูลค่าเดียวติดต่อกันหลายบิตได้ ดังนั้นแคนจึงมีบิตแทรกเพื่อใช้คั่นข้อมูล โดยถ้ามีการส่งบิตข้อมูลเดียวกัน 5 บิต จะต้องคั่นด้วยบิตแทรก 1 บิต

- **การตรวจสอบและแก้ไขข้อผิดพลาด**

ความผิดพลาดที่สามารถตรวจสอบได้มีอยู่ทั้งหมดด้วยกัน 5 อย่าง คือ

- ซีอาร์ซีผิดพลาด (CRC Error) ในขณะที่โหนดผู้ส่งทำการส่งข้อมูลจะทำการคำนวณค่าซีอาร์ซีของข้อมูลที่ส่งไปด้วยและส่งไปในฟิลด์ตรวจสอบของเฟรมนั้นๆ ที่โหนดผู้รับก็มีการคำนวณค่าซีอาร์ซีเช่นกัน แล้วนำมาเปรียบเทียบว่าตรงกันหรือไม่ หากไม่ตรงกันจะทำการส่งเฟรมแสดงความ

ผิดพลาดแจ้งกลับไปเพื่อบอกว่าข้อมูลที่ได้รับไม่ถูกต้อง เพื่อให้ส่งข้อมูลมาใหม่อีกครั้ง

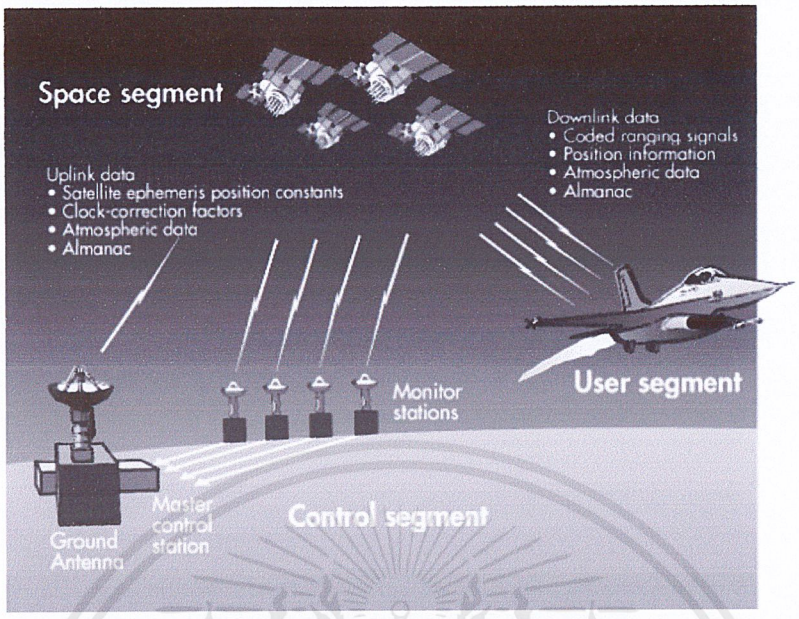
- การตอบรับผิดพลาด (Acknowledge Error) โหนดผู้ส่งจะทำการตรวจสอบที่บิตตอบรับว่ามีโหนดใดตอบรับหรือไม่ ถ้าไม่มีก็จะทำการส่งข้อมูลใหม่
- รูปแบบผิดพลาด (Form Error) ถ้าพบว่ามีสถานะเด่นเกิดขึ้นในช่วงต่อไปนี้ อักขระคั่นซีอาร์ซี, อักขระคั่นการตอบรับ และฟิลด์สิ้นสุดเฟรม ซึ่งช่วงเหล่านี้ควรเป็นสถานะด้อยตลอดเวลา แสดงว่ามีการส่งข้อมูลผิดพลาดเกิดขึ้น
- บิตผิดพลาด (Bit Error) จะเกิดขึ้นเมื่อโหนดผู้ส่งอ่านค่าจากบัสกลับมาได้ค่าไม่ตรงกับค่าที่ส่งออกไป ซึ่งสำหรับกรณีที่ส่งสถานะด้อยออกไปแต่อ่านได้เป็นสถานะเด่นกลับมานั้นไม่ถือว่าเป็นความผิดพลาดเนื่องจากสามารถเกิดขึ้นได้
- การแทรกผิดพลาด (Stuff Error) เป็นข้อผิดพลาดที่เกิดขึ้นเมื่อพบข้อมูลค่าเดียวกันติดเกิน 5 บิต เนื่องจากปกติแล้วเมื่อมีการส่งข้อมูลค่าเดียวกัน 5 บิต จะต้องมีบิตแทรกเข้ามา 1 บิตเสมอ

นอกจากนี้ยังมีการกำหนดสถานะความผิดพลาดของแต่ละโหนดในบัสขึ้นเพื่อควบคุมให้การใช้งานบัสมีประสิทธิภาพมากขึ้น กล่าวคือ โหนดใดมีความผิดพลาดมากกว่าเกณฑ์ที่กำหนดก็จะไม่อนุญาตให้ใช้งานบัสได้ เพื่อไม่ให้เสียเวลาของระบบโดยรวม

2.9 จีพีเอส

จีพีเอส หรือ ระบบหาพิกัดบนพื้นโลก (GPS, Global Positioning System) เป็นระบบหาพิกัดบนพื้นโลกระบบเดียวในปัจจุบัน ที่สามารถแสดงที่อยู่ที่แน่นอนว่าอยู่ ณ ตำแหน่งใดบนพื้นโลก ระบบนี้ได้พัฒนาขึ้นโดยกระทรวงกลาโหมประเทศสหรัฐอเมริกาซึ่งจัดทำโครงการมาตั้งแต่ปี พ.ศ.2521 โดยอาศัยการรับสัญญาณจากดาวเทียมที่มีความแม่นยำสูงและระบบวิทยุนำร่องเป็นพื้นฐานในการกำหนดตำแหน่งค่าพิกัดของเครื่องรับ ซึ่งเมื่อเสร็จสิ้นโครงการจะมีจำนวนดาวเทียมทั้งหมดถึง 24 ดวงมีชื่อว่า NAVSTAR ที่โคจรอยู่รอบโลกวันละ 2 รอบพร้อมด้วยสถานีความคุมภาคพื้นดินเพื่อให้ระบบสามารถที่จะทำงานได้ทุกสภาวะตลอด 24 ชั่วโมง

2.9.1 องค์ประกอบของระบบจีพีเอส



รูปที่ 2.27 องค์ประกอบของระบบจีพีเอส

องค์ประกอบทั่วไปของระบบจีพีเอสประกอบด้วยส่วนประกอบสำคัญ 3 ส่วน ได้แก่

2.9.1.1 ส่วนอวกาศ

ประกอบด้วยดาวเทียมทั้งหมด 24 ดวง โดยดาวเทียมจำนวน 21 ดวงจะใช้ในการบอกค่าพิกัด ส่วนที่เหลือ 3 ดวงจะสำรองเอาไว้ ดาวเทียมทั้ง 24 ดวงนี้จะมีวงโคจรอยู่ 6 วงโคจรด้วยกัน โดยแบ่งจำนวนดาวเทียมวงโคจรละ 4 ดวง และมีรัศมีวงโคจรสูงจากพื้นโลกประมาณ 20,200 กิโลเมตร (12,600 ไมล์) วงโคจรทั้ง 6 จะเอียงทำมุมกับเส้นศูนย์สูตร (Equator) เป็นมุม 55 องศา ห่างกัน 60 องศา โดยดาวเทียมหนึ่งดวงจะสามารถโคจรรอบโลกได้ 1 รอบใน 12 ชั่วโมง(ประมาณ 1.8 ไมล์ต่อวินาที) ในระหว่างการโคจรรอบโลกนั้น ดาวเทียมจะมีการส่งสัญญาณสู่พื้นโลกผ่านเสาส่งสัญญาณที่ติดตั้งจากดาวเทียมมายังโลก และมีการนำพลังงานแสงอาทิตย์มาใช้ในการขับเคลื่อน การที่จีพีเอส จะทำงานได้อย่างน้อยต้องรับสัญญาณจากดาวเทียม 3 ดวงขึ้นไป โดยถ้ารับสัญญาณได้ 4 ดวงขึ้นไปก็จะสามารถบ่งบอกพิกัดความสูงได้ด้วย ดาวเทียมชุดแรกเรียกจีพีเอสบลอค 1 มีทั้งหมด 10 ดวงแต่ละดวงจะมีนาฬิกาที่มีความแม่นยำสูง ซึ่งเป็นชุดของนาฬิกาอะตอมมิค แบ่งออกเป็นแหล่งกำเนิดความถี่รูบิเดียม 2 เรือน และซีเซียม 2 เรือนทำให้เวลามาตรฐานของดาวเทียมมีความถูกต้องสูงมาก นาฬิกาดังกล่าวยังช่วยในการคำนวณระยะทางระหว่างดาวเทียมกับเครื่องรับสัญญาณเพื่อที่จะคำนวณค่าพิกัดตำแหน่งได้

คลื่นสัญญาณวิทยุที่ส่งออกมาจากดาวเทียมจะมีการเข้ารหัสสัญญาณ 2 รหัสด้วยกัน คือ รหัสซีเอ C/A (Coarse/Acquistion) และรหัสพี P (Precision) คือ รหัสซีเอ มีความถี่เป็น 1 ใน 10 ของความถี่พื้นฐาน คือ 1.023 MHz ความยาวคลื่น 300 เมตร มีคาบเป็น 1 ใน 1000 วินาที นั่นคือในช่วงเวลา 1 วินาที จะสร้างรหัสซีเอที่มีรูปแบบเหมือนกันซ้ำถึง 1000 ครั้ง การตรวจสอบรูปแบบของรหัสซีเอ จึงทำได้ง่ายและรวดเร็วมาก รหัสซีเอเปิดให้ทุกคนใช้ได้อย่างอิสระ ส่วนรหัสพีมีความถี่เท่ากับความถี่พื้นฐาน คือ 10.23 MHz ความยาวคลื่นเป็น 30 เมตร และมีคาบเป็น 267 วัน นั่นคือช่วง 267 วัน รหัสพีที่ส่งออกมาจะมีรูปแบบที่ไม่ซ้ำกัน จึงเป็นการยากที่จะตรวจสอบว่ารหัสพีที่ดาวเทียมใช้ในแต่ละวันเป็นส่วนไหนของรหัส ผู้ที่ไม่มีส่วนเกี่ยวกับการสร้างรหัสพีของดาวเทียมจึงไม่อาจใช้ประโยชน์จากรหัสพีเพื่อหาตำแหน่งได้ รหัสพีจะถูกสงวนไว้ใช้เฉพาะวงการทหาร หน่วยงาน และทั้งพันธมิตรทางการทหาร ของรัฐบาลสหรัฐอเมริกาเท่านั้น ซึ่งคลื่นส่งแอล1 (L1) ถูกรวมสัญญาณ (Modulate) ด้วยรหัสทั้งสองชนิด ส่วนคลื่นส่งแอล2 (L2) มีเพียงรหัสพี และรหัสพีจะถูกเปลี่ยนเป็นรหัสวาย (Y) ในกรณีที่ต้องการป้องกันการใช้ประโยชน์จากรหัสพี

นอกจากคลื่นวิทยุแล้วยังมีคลื่นดาวเทียมอื่นๆ ที่ใช้สำหรับการคำนวณตำแหน่งถูกรวมสัญญาณมาพร้อมคลื่นส่งด้วย ได้แก่ อีเฟเมอริสดาวเทียม (Satellite Ephemeris) ซึ่งเป็นข้อมูลจำเพาะ ประกอบไปด้วยข้อมูลวงโคจร สถานภาพของดาวเทียม เวลามาตรฐานดาวเทียม พฤติกรรมของนาฬิกาดาวเทียม

2.9.1.2 สถานีควบคุม

ประกอบด้วยสถานีภาคพื้นดินที่ควบคุมระบบ (OCS, Operational Control System) ที่กระจายอยู่ตามส่วนต่างๆ บนโลกมีหน้าที่ปรับปรุงให้ข้อมูลดาวเทียมมีความถูกต้องทันสมัยอยู่ตลอดเวลา โดยแบ่งออกเป็นสถานีควบคุมหลักและสถานีภาคพื้นดิน

สถานีควบคุมหลัก ตั้งอยู่ที่ฐานทัพอากาศในเมืองโคโลราโดสปริงส์ มลรัฐโคโรลาโดของประเทศสหรัฐอเมริกา สถานีติดดาวเทียม 5 แห่ง ทำหน้าที่คอยติดต่อสื่อสาร (Tracking) กับดาวเทียมโดยใช้เรดาร์ส่งสัญญาณไปยังดาวเทียม เพื่อให้ดาวเทียมอยู่ในวงโคจร ในความสูง ความเร็ว และตำแหน่งที่ถูกต้อง ทำการคำนวณผล (Computation) เพื่อบอกตำแหน่งของดาวเทียมแต่ละดวง และส่งข้อมูลที่ได้ไปยังดาวเทียมอยู่ตลอดเวลา ทำให้ข้อมูลที่ได้เป็นข้อมูลที่ทันสมัยอยู่เสมอ และในทางกลับกัน สถานทีเหล่านี้ยังทำหน้าที่รับสัญญาณจากดาวเทียมและส่งข้อมูลไปยังเครื่องลูกข่ายจีพีเอส เพื่อบอกตำแหน่งและข้อมูลของเครื่องลูกข่ายนั้นๆ อย่างถูกต้องด้วย ซึ่งสถานี

ที่ทำการควบคุมดาวเทียมจะมีอยู่ 5 แห่ง คือ สถานีหลักที่ Colorado, สถานีบนเกาะ Ascension, สถานี Diego Garcia (มหาสมุทรอินเดีย), Kwajalein และ Hawaii

ในช่วงเดือนสิงหาคมและกันยายน ปี 2548 เอ็นจีไอเอ (NGIA, National Geospatial-Intelligence Agency) ได้เพิ่มสถานีติดตามดาวเทียมเพิ่มอีก 6 แห่ง ดังนั้นในปัจจุบันมีสถานีควบคุมหลักและสถานีติดตามดาวเทียมรวมทั้งสิ้น 11 แห่ง ทำให้สถานติดตามดาวเทียมอย่างน้อย 2 สถานีมองเห็นดาวเทียมทั้งหมด เพื่อทำการประมวลผลหาวงโคจรของดาวเทียมดวงต่างๆ เมื่อได้พยากรณ์ตำแหน่งดาวเทียมล่วงหน้าแล้ว ก็จะจัดส่งข้อมูลที่ได้ปรับปรุงแล้ว พร้อมกับข้อมูลเวลาและข้อมูลอุตุนิยมวิทยาไปยังสถานีรับส่งสัญญาณ 3 แห่ง เพื่อส่งไปเก็บบันทึกไว้ในดาวเทียมต่อไป

สถานีภาคพื้นดินที่ควบคุมระบบ จะเฝ้าระวังติดตามดาวเทียม และข้อมูลที่ได้จากการเฝ้าระวังติดตามดาวเทียม สามารถที่จะบอกวงโคจรล่วงหน้าได้อย่างถูกต้องแม่นยำ และจะส่งสัญญาณข้อมูลวงโคจรจากสถานีสู่ดาวเทียมวันละ 3 ครั้งซึ่งจะทำให้เครื่องรับสัญญาณจีพีเอสคำนวณตำแหน่งในเวลาจริงได้ สำหรับการสำรวจศึกษาทางด้านสัณฐานของโลกนั้นอาจจะไม่ละเอียดเพียงพอ

2.9.1.3 ส่วนผู้ใช้

ประกอบด้วยเครื่องรับสัญญาณมีหลายขนาด ซึ่งอาจพกพาได้ ติดบนรถ เรือ เครื่องบิน เครื่องรับสัญญาณจะทำหน้าที่ในการเปลี่ยนสัญญาณจากดาวเทียมเป็นตำแหน่ง ความเร็ว และเวลาโดยประมาณ ถ้าต้องการทราบความสูงด้วยต้องใช้ดาวเทียม 4 ดวงในการประมวลผล ความถูกต้องของตำแหน่งขึ้นอยู่กับนาฬิกาและตัวจีพีเอสซึ่งอาจจะหาตำแหน่งที่มีความผิดพลาดได้น้อยกว่า 3 ฟุต นาฬิกาที่ใช้จะมีความถูกต้องสามารถวัดได้ในเวลา 3x10-9 วินาที ซึ่งเวลาที่ใช้ในการอ้างอิงสำหรับระบบดาวเทียมจีพีเอส เรียกว่าเวลาจีพีเอส

2.9.2 ความคลาดเคลื่อนในการงานจีพีเอส

ในการใช้งานจีพีเอส ความคลาดเคลื่อนที่เกิดขึ้น อาจพิจารณาแยกได้เป็น 3 กลุ่มคือ กลุ่มเกี่ยวข้องกับดาวเทียมได้แก่ ความคลาดเคลื่อนวงโคจรและความคลาดเคลื่อนนาฬิกาดาวเทียม กลุ่มเกี่ยวข้องกับการแพร่กระจายของสัญญาณได้แก่ ความคลาดเคลื่อนของการหักเหในชั้นบรรยากาศและการเกิดคลื่นสะท้อน กลุ่มสุดท้ายเกี่ยวข้องกับเครื่องรับสัญญาณ เช่นนาฬิกาเครื่องรับ

2.9.3 มาตรฐานในการสื่อสารข้อมูลจีพีเอส

ในการที่เครื่องรับสัญญาณดาวเทียมจีพีเอสจะสามารถส่งข้อมูลไปยังอุปกรณ์อื่นได้นั้นต้องอาศัยมาตรฐานในการส่งข้อมูล โดยหน่วยงานที่ทำหน้าที่กำหนดมาตรฐานในการสื่อสารข้อมูลจีพี

เอส คือหน่วยงานเอ็นเอ็มอีเอ (NMEA, National Marine Electronics Association) ซึ่งได้มีการกำหนดให้โปรโตรคอล เอ็นเอ็มอีเอ-0183 (NMEA-0183) เป็นโปรโตคอลมาตรฐานในการสื่อสารข้อมูลจากเครื่องรับจีพีเอสไปสู่อุปกรณ์เชื่อมต่อภายนอก ซึ่งข้อมูลเหล่านี้อยู่ในรูปของรหัสแอสกี้ (ASCII Code) โดยกำหนดให้อัตราเร็วในการส่งข้อมูลอยู่ที่ 4800 บิตต่อวินาที โดยมีบิตข้อมูล 8 บิต ไม่มีพาริตี้ และไม่มีบิตสิ้นสุด

ตามโปรโตคอลลักษณะข้อมูลที่ใช้ในการสื่อสารอยู่ในรูปแบบประโยคโดยจะขึ้นต้นประโยคด้วยตัวอักษร '$' และลงท้ายด้วยอักษรพิเศษ "<CR><LF>" ซึ่งรูปแบบประโยคจะมีดังนี้

$ttsss,d1,d2,….<CR><LF>

tt	หมายถึง รหัสประจำตัวผู้ส่ง (Talker Identifier) มีค่าเป็น "GP" สำหรับจีพีเอส
sss	หมายถึง รหัสประจำประโยค (Sentence Identifier)
d1,d2,..	หมายถึง ข้อมูลที่ต้องการส่ง

ตัวอย่างรูปแบบประโยคที่ได้รับจากจีพีเอส

$--RMC,hhmmss.ss,A,llll,ll,a,yyyyy,yy,a,x.x,x.x,xxxxxx,x.x,a*hh

hhmmss.ss	หมายถึง เวลา (UTC)
A	หมายถึง สถานะถ้าเป็น V แปลว่าข้อมูลที่ได้ยังไม่แม่นยำเพียงพอแก่การใช้งาน
llll.ll	หมายถึง ละติจูด
a	มีค่าเป็น N หรือ S
yyyy.yy	หมายถึง ลองจิจูด
a	มีค่าเป็น E หรือ W
x.x	หมายถึง ความเร็วบนพื้นดิน มีหน่วยเป็นนอต (knot)
x.x	หมายถึง ทิศทางการเคลื่อนที่ มีหน่วยเป็นองศาจริง (degree true)
xxxxxx	หมายถึงวันที่อยู่ในรูป ddmmyy
x.x	หมายถึงค่าความแตกต่างระหว่างทิศเหนือของเข็มทิศและทิศเหนือจริงๆ (Magnetic variation) มีหน่วยเป็นองศา
a	มีค่าเป็น E หรือ W

hh	เป็นค่าผลรวมของข้อมูล (check sum)

บทที่ 3

การออกแบบ

3.1 ระบบโดยรวม

โครงงานนี้ได้มุ่งเน้นให้ระบบที่สร้างขึ้นนั้นมีลักษณะการทำงานเป็นแบบฝังตัว หรือเป็นเอ็มเบ็ดเด็ดซิสเต็ม (Embedded System) นอกจากนี้เรายังได้ออกแบบระบบเพื่อให้สามารถนำมาพัฒนาต่อได้ง่าย โดยการแบ่งเป็น 3 ลำดับชั้นได้แก่ชั้นของฮาร์ดแวร์และเซ็นเซอร์ (sensor) ชั้นฮาร์ดแวร์นามธรรม (hardware abstraction) ของรถ และชั้นของระบบตัดสินใจ แต่ละส่วนนั้นจะมีความเกี่ยวข้องหรือขึ้นต่อกันให้น้อยที่สุด

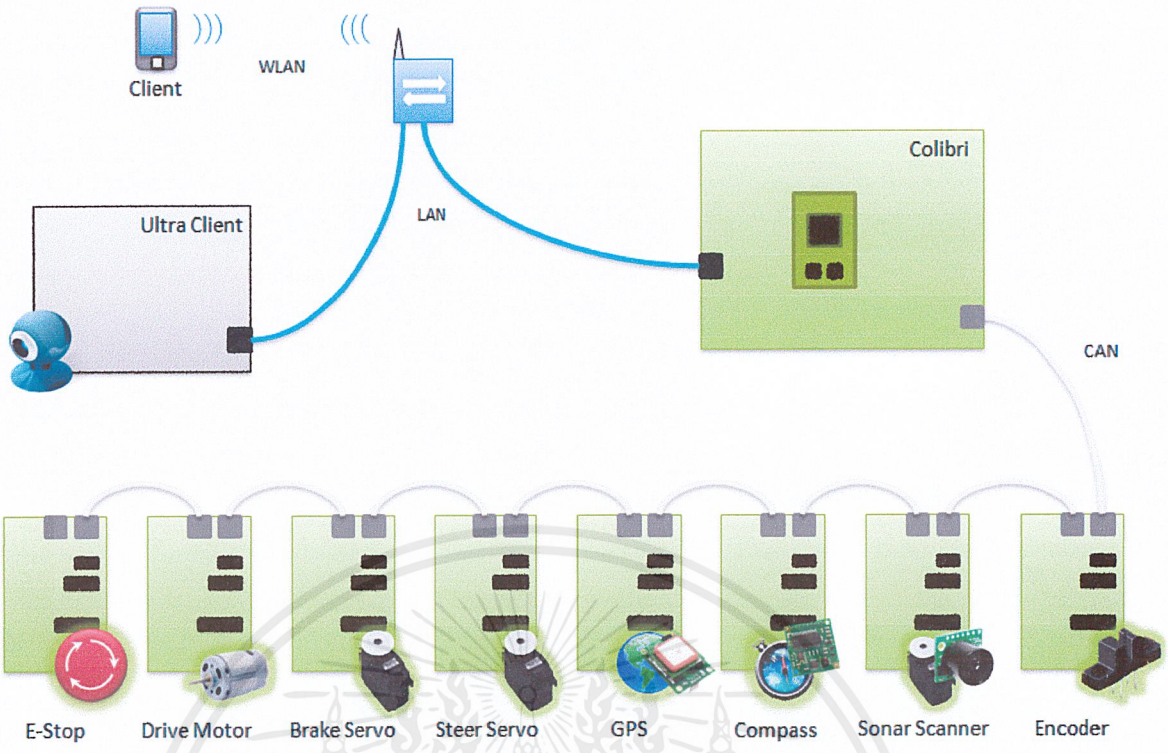
รูปที่ 3.1 ลำดับชั้นของระบบภายในรถ

การติดต่อระหว่างชั้นฮาร์แวร์และเซ็นเซอร์กับชั้นฮาร์ดแวร์นามธรรมจะใช้การรับส่ง เมสเซส (message) กัน การเปลี่ยนชนิดหรือลักษณะการทำงานฮาร์ดแวร์แต่ละชิ้นจะไม่ทำให้เกิดการเปลี่ยนแปลงในส่วนของชั้นฮาร์ดแวร์นามธรรม

ชั้นฮาร์ดแวร์นามธรรมนั้นเราได้นำเพลเยอร์ ซึ่งเป็นโอเพนซอสโปรเจค (open source project) มาใช้ โดยเพลเยอร์นั้นได้กำหนดรูปแบบการติดต่อสื่อสารสำหรับควบคุมและรับข้อมูลจากดีไวซ์ (device) ประเภทต่างๆไว้มากมาย ซึ่งมีข้อดีดังนี้

- ชั้นฮาร์ดแวร์นามธรรมกับชั้นของระบบตัดสินใจมีการขึ้นต่อกันน้อยมาก ระบบตัดสินใจที่เขียนขึ้นสามารถนำไปควบคุมรถคันอื่นๆที่มีรูปแบบเดียวกันได้ หรือแม้กระทั่งควบคุมรถที่จำลองขึ้นมาในระบบคอมพิวเตอร์ รถสามารถถูกควบคุมโดยระบบตัดสินใจอื่นๆได้ โดยไม่จำเป็นต้องแก้ไขชั้นฮาร์ดแวร์นามธรรมเลย
- ระบบตัดสินใจไม่จำเป็นต้องอยู่บนแพลตฟอร์ม (platform) หรือฮาร์ดแวร์ เดียวกันกับชั้นฮาร์ดแวร์นามธรรมเนื่องจากการติดต่อระหว่าง 2 ชั้นนี้ มีการติดต่อกันผ่านทางทีซีพีไอพี (TCP/IP) ทำให้ระบบตัดสินใจสามารถย้ายไปที่ใดก็ได้ และพัฒนาบนภาษาใดก็ได้ที่สามารถเขียนโปรแกรมติดต่อกับเครือข่ายทีซีพีไอพีได้
- ชั้นฮาร์ดแวร์นามธรรมจะมีลักษณะมุมมองแบบกลุ่มของดีไวซ์ ทำให้การเพิ่มดีไวซ์ประเภทใหม่ๆเข้าไปในชั้นของฮาร์ดแวร์และเซ็นเซอร์ ไม่มีผลกระทบต่อระบบตัดสินใจตัวเดิมซึ่งยังไม่รองรับดีไวซ์ที่เพิ่มขึ้นมาใหม่ ซึ่งส่งผลให้เราสามารถพัฒนาชั้นของฮาร์ดแวร์และเซ็นเซอร์และชั้นฮาร์ดแวร์นามธรรมไปได้เรื่อยๆ โดยที่ไม่มีผลกระทบต่อการทำงานของระบบโดยรวม

ระบบจะถูกแบ่งออกเป็นทั้งหมด 4 ส่วนหลัก คือ มอดูลประมวลผลกลาง, มอดูลประมวลผลภาพ, มอดูลฮาร์ดแวร์ควบคุมรถ และส่วนไคลอัล ดังรูปที่ 3.2 โดยมอดูลประมวลผลกลางอยู่ในชั้นของระบบตัดสินใจและชั้นของฮาร์ดแวร์นามธรรม มอดูลประมวลผลภาพและมอดูลฮาร์ดแวร์ควบคุมรถอยู่ในชั้นของฮาร์ดแวร์และเซ็นเซอร์ สำหรับส่วนไคลอัลนั้นเป็นเพียงแค่ส่วนที่ไว้ตรวจสอบสถานะและสั่งงานระบบตัดสินใจ จึงไม่ได้จัดอยู่ในชั้นใดๆ



รูปที่ 3.2 ระบบโดยรวม

มอดูลประมวลผลกลาง เป็นระบบสมองกลฝังตัวซึ่งใช้บอร์ดคอลิบริ ซึ่งใช้ซีพียูรุ่นพีเอ็กซ์เอ270 (PXA270) ในการประมวลผล มีหน้าที่รับข้อมูลจากมอดูลอินพุตนำมาประมวลผลและตัดสินใจ และสั่งงานไปยังมอดูลเอาต์พุตเพื่อควบคุมรถต่อไป

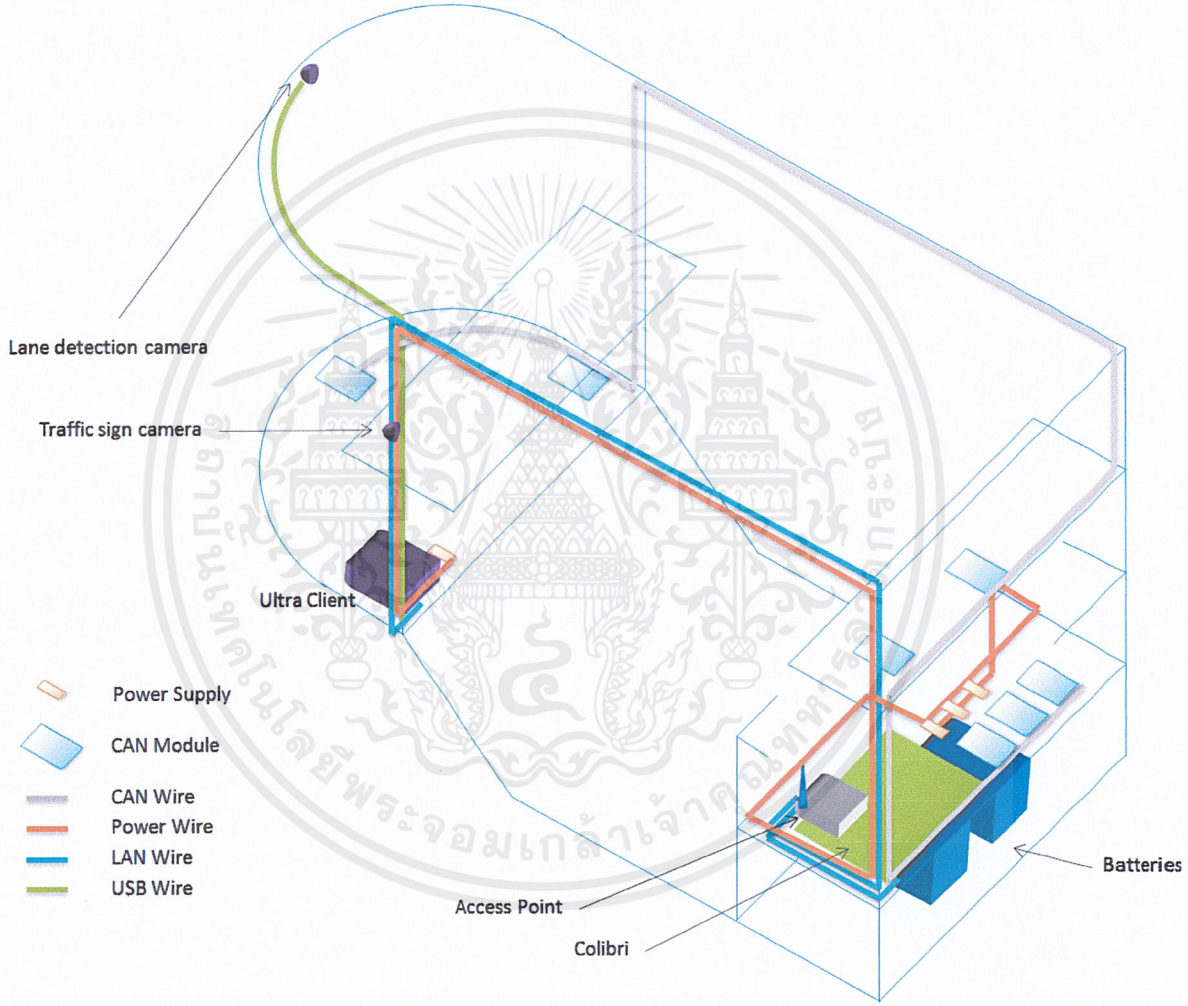
มอดูลประมวลผลภาพ ประกอบไปด้วยพีซีฝังตัว (Embedded PC) รุ่น อัลตร้าไคลอัล (Ultra Client) ของบริษัทนอร์เทค (Norhtec) และกล้องเวปแคม (Web camera) มีหน้าที่ในการนำข้อมูลภาพมาประมวลผลหาขอบทาง และตรวจจับป้ายจราจร และนำข้อมูลที่ได้ส่งต่อให้มอดูลประมวลผลกลางผ่านทางโปรโตคอลอีเทอร์เน็ต (Ethernet) ทั้งนี้เพื่อลดผลกระทบในการปรับเปลี่ยนฮาร์ดแวร์ของโมดูล

มอดูลฮาร์ดแวร์ควบคุมรถ ประกอบด้วยมอดูลอินพุต/เอาต์พุต ดังนี้ มอดูลระบบหยุดรถฉุกเฉิน, มอดูลขับมอเตอร์ขับเคลื่อน, มอดูลขับมอเตอร์เบรก, มอดูลขับมอเตอร์ควบคุมทิศทาง, มอดูลรับข้อมูลจีพีเอส, มอดูลเข็มทิศดิจิตอล, มอดูลโซนาร์ (SONAR, SOund NAvigation Ranging) และมอดูลวัดระยะทาง (Encoder) ทั้งหมดนี้จะถูกเชื่อมต่อเข้ากับมอดูลประมวลผลกลางด้วยแคนบัส ซึ่งในแต่ละมอดูลจะมีไมโครคอนโทรลเลอร์ (Microcontroller) จัดการห่อหุ้มการ

เชื่อมต่อกับอุปกรณ์ต่างๆในระดับล่าง และจัดรูปแบบข้อมูลให้เป็นไปตามโพรโทคอล (Protocol) ที่ตกลงไว้

มอดูลไคลอัล ใช้ในการสั่งการกำหนดพิกัดเริ่มต้น พิกัดปลายทางและแสดงสถานะต่างๆ ของระบบ

3.2 ระบบฮาร์ดแวร์ (Hardware)



รูปที่ 3.3 ระบบฮาร์ดแวร์

3.2.1 ระบบบัส

การออกแบบทางด้านมอดูลฮาร์ดแวร์นั้น เนื่องจากมีโหนดที่ต้องติดต่อด้วยเป็นจำนวนมาก มีความยาวสายรวมไม่ต่ำกว่า 3 เมตร การแลกเปลี่ยนข้อมูลต้องมีเสถียรภาพ แคนบัสจึงเป็นระบบบัสที่ถูกเลือกให้เป็นระบบบัสหลักของรถ และเนื่องจากแคนบัสถูกออกแบบให้เป็นการสื่อสาร

แบบกระจายยิ่งเป็นผลให้การพัฒนาระบบทำได้ง่ายขึ้น ยกตัวอย่างเช่น การพัฒนาระบบควบคุมความเร็ว โดยให้ส่วนกลางสั่งเพียงว่าต้องการความเร็วเป็นเท่าไหร่ ระบบเบรกและระบบขับเคลื่อนก็สามารถนำข้อมูลจากเซนเซอร์วัดระยะทางที่ถูกส่งออกมาตลอดเวลาอยู่แล้ว มาคำนวณเป็นความเร็ว หรือความหน่วงเพื่อที่จะปรับแรงขับมอเตอร์ และแรงเบรกได้อย่างถูกต้อง โดยถ้าไม่ใช่ระบบแบบกระจายเช่นนี้ อาจทำให้ต้องมีการร้องขอข้อมูลจากเซ็นเซอร์วัดความเร็วเพิ่มขึ้นอีกสองครั้งจากมอดูลเบรก และมอดูลขับเคลื่อน ทำให้สิ้นเปลืองเวลาในการใช้งานบัสโดยเปล่าประโยชน์ สายที่นำมาใช้เป็นสายคู่ตีเกลียวแบบไม่ชิลด์ (UTP, Unshielded Twisted Pair) ประเภทเดียวกับสายแลน เนื่องจากเป็นสายตีเกลียวซึ่งสามารถช่วยให้เกิดการหักล้างสัญญาณรบกวนได้ดีขึ้น และยังนำสายอีกสองคู่ภายในมาใช้ในการจ่ายไฟเลี้ยง 12 โวลต์ เช่นเดียวกับในมาตรฐานพีโออี (POE, Power On Ethernet) อีกด้วย

สำหรับส่วนประมวลผลภาพ ส่วนประมวลผลกลางและไคลอันนั้นติดต่อกันผ่านทางอีเธอร์เน็ตนั้นเนื่องจากมอดูลเหล่านี้มักมีมาให้อยู่แล้ว อีกทั้งยังเป็นการสะดวกในการปรับเปลี่ยนโมดูลอีกด้วย

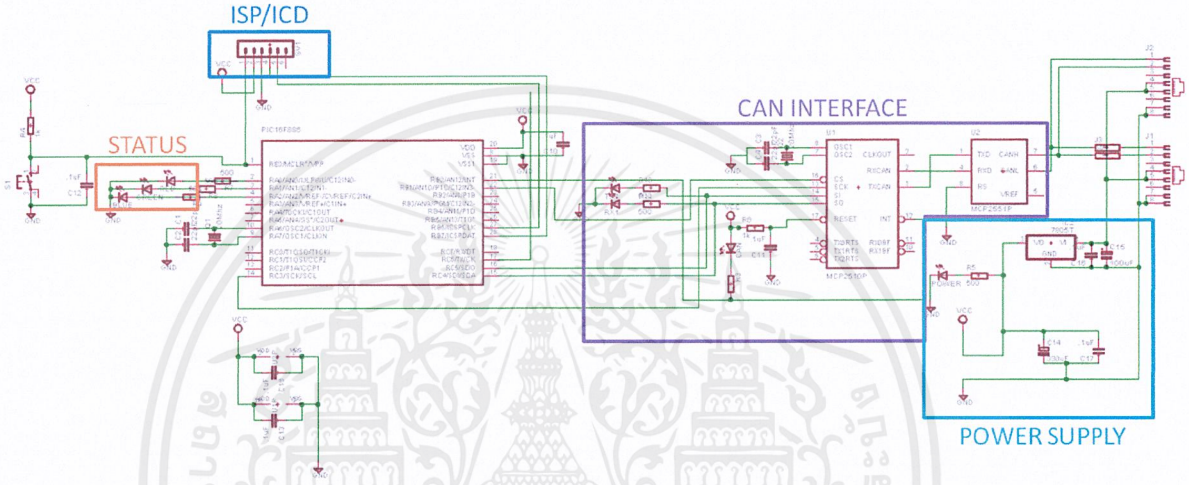
3.2.2 โครงสร้างหลักมอดูลฮาร์ดแวร์

การติดต่อกับแคนบัสโดยใช้ไมโครคอนโทรลเลอร์นั้นมีสองวิธีหลักๆ คือ ใช้ไมโครคอนโทรลเลอร์ที่มีมอดูลแคนในตัว หรือให้ไมโครคอนโทรลเลอร์ติดต่อกับแคนคอนโทรลเลอร์ภายนอก ซึ่งในที่นี้ผู้จัดทำเลือกใช้วิธีหลังเนื่องจาก ไมโครคอนโทรลเลอร์ที่มีมอดูลแคนในตัวนั้นหาซื้อได้ยากในประเทศไทย มีราคาค่อนข้างสูง อีกทั้งเมื่อเทียบคุณสมบัติในราคารวมที่เท่ากันจะมีความคุ้มค่าที่ต่ำกว่า และความยืดหยุ่นก็ต่ำกว่าเนื่องจากจะถูกจำกัดด้วยจำนวนน้อยรุ่นที่มีมอดูลแคนในตัว ซึ่งแคนคอนโทรลเลอร์นั้นได้เลือกใช้ของบริษัทไมโครชิพเบอร์ เอ็มซีพี2515 (MCP2515) ซึ่งสามารถหาซื้อได้ง่ายในประเทศไทย ซึ่งมีการเชื่อมต่อกับคอนโทรลเลอร์ตัวหลักด้วยบัสเอสพีไอ (SPI, Serial Peripheral Interface)

การเลือกไมโครคอนโทรลเลอร์นั้นมีความต้องการขั้นต่ำมีเพียงแค่การมีมอดูลเอสพีไอในตัว และมีหน่วยความจำโปรแกรมที่เพียงพอ และสามารถโปรแกรมและดีบักขณะอยู่บนวงจร (ISP, In Circuit Programming) (ICD, In Circuit Debugging) โดยเครื่องมือพัฒนาพิคคิททู (PICKIT2) ได้ทำให้มีรุ่นให้เลือกเป็นจำนวนมาก ผู้จัดทำจึงเน้นไปที่ความคุ้มค่าต่อราคา และการหาซื้อได้เป็นหลัก ทำให้ได้ออกมาเป็นไมโครคอนโทรลเลอร์ของบริษัทไมโครชิพ รุ่น พีไอซี16เอฟ883/886 (PIC16F883/886) ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต สามารถโปรแกรมได้หลายครั้ง มี

จำนวนขา 40 ขา มีหน่วยความจำโปรแกรม 4096 และ 8192 คำ (words) และมีเอสแรม (SRAM) จำนวน 256 และ 368 ไบท์ สำหรับรุ่น 883 และ 886 ตามลำดับ ซึ่งหาซื้อได้ในราคา 70 บาท จึงนับได้ว่ามีความคุ้มค่าเป็นอย่างมาก

โครงสร้างหลักของมอดูลฮาร์ดแวร์ประกอบไปด้วย ไมโครคอนโทรลเลอร์, แคนคอนโทรลเลอร์, ภาคจ่ายไฟ และส่วนแสดงสถานะ นอกจากนั้นจะแตกต่างกันไปตามหน้าที่ของมอดูลนั้นๆ ซึ่งมีแผนภาพการทำงานดังนี้



รูปที่ 3.4 โครงสร้างของมอดูลอาร์ดแวร์

3.2.2.1 ส่วนติดต่อแคน

ส่วนติดต่อแคนประกอบด้วยแคนคอนโทรลเลอร์เอ็มซีพี2515 และแคนไดรว์เวอร์เอ็มซีพี2551 ซึ่งแคนเอ็มซีพี2515 ติดต่อกับไมโครคอนโทรลเลอร์ผ่านทางบัสเอสพีไอ ถูกตั้งให้เกิดอินเทอร์รัพท์เมื่อมีข้อความสอดคล้องกับตัวกรองที่ตั้งค่าไว้, เมื่อเกิดข้อผิดพลาด, เมื่อส่งข้อมูลสำเร็จ (ขึ้นอยู่กับความต้องการของแต่ละโมดูล) เมื่อต้องการส่งข้อความคอนโทรลเลอร์จะส่งคำสั่งเขียนข้อมูลลงในหน่วยความจำของเอ็มซีพี2515 ประกอบไปด้วยรหัสประจำข้อความ และความยาวที่ต้องการส่ง จากนั้นจึงส่งคำสั่งร้องขอให้ส่งข้อความในหน่วยความจำนั้นออกไป โดยที่เอ็มซีพี2515 จะเป็นผู้รับผิดชอบในการส่งที่เหลือทั้งหมด ไม่ว่าจะเป็นการตรวจจับบิตตอบสนอง หรือการส่งใหม่อีกครั้งหากมีความผิดพลาด ข้อมูลที่ออกจากเอ็มซีพี2515 จะอยู่ในรูปแบบของแคนส่งและแคนรับ (CAN_TX, CAN_RX) ซึ่งต้องนำสัญญาณที่ได้นี้มาเข้าแคนไดรว์เวอร์เอ็มซีพี2551 เพื่อเปลี่ยนให้อยู่ในรูปแบบสัญญาณแบบอาศัยความแตกต่าง (Differential) เป็นแคนสูงและแคนต่ำ (CAN_H,

CAN_L) ต่อไป ซึ่งไดรว์เวอร์นี่เองจะมีส่วนที่ทำหน้าที่ตรวจสอบด้วยว่าแคนคอนโทรลเลอร์ส่งสถานะเด่นออกมานานเกินที่จะเป็นไปได้หรือไม่ ถ้านานเกินเอ็มซีพี2551 จะทำเปลี่ยนไปอยู่ในสถานะบัสออฟทันทีเพื่อให้การสื่อสารในบัสหลักดำเนินต่อไปได้

3.2.2.2 ภาคจ่ายไฟ

แต่ละโหนดจะมีภาคจ่ายไฟเป็นของตนเองสำหรับลดแรงดันจาก 12 โวลต์ เหลือ 5 โวลต์ทำให้มั่นใจได้ว่าแรงดันที่ไมโครคอนโทรลเลอร์ได้รับจะมีค่าที่คงที่ไม่ต่ำจนเกินไป และมีความเสถียรเพียงพอ ภาคจ่ายไฟประกอบด้วยไอซีรักษาระดับแรงดัน แอลเอ็ม7805เอซีที ซึ่งมีค่าคลาดเคลื่อนเพียง 2-4% มีวงจรจำกัดกระแส และอุณหภูมิเกินภายในตัว

3.2.2.3 ส่วนไอเอสพี/ไอซีดี

ส่วนไอเอสพี/ไอซีดี เป็นพอร์ตที่ใช้ในการโปรแกรม หรือดีบักไมโครคอนโทรลเลอร์ ซึ่งสามารถกระทำได้แม้ในขณะที่อยู่ในวงจรเพื่อความสะดวกในการปรับปรุงและพัฒนา

3.2.2.4 ส่วนแสดงผล

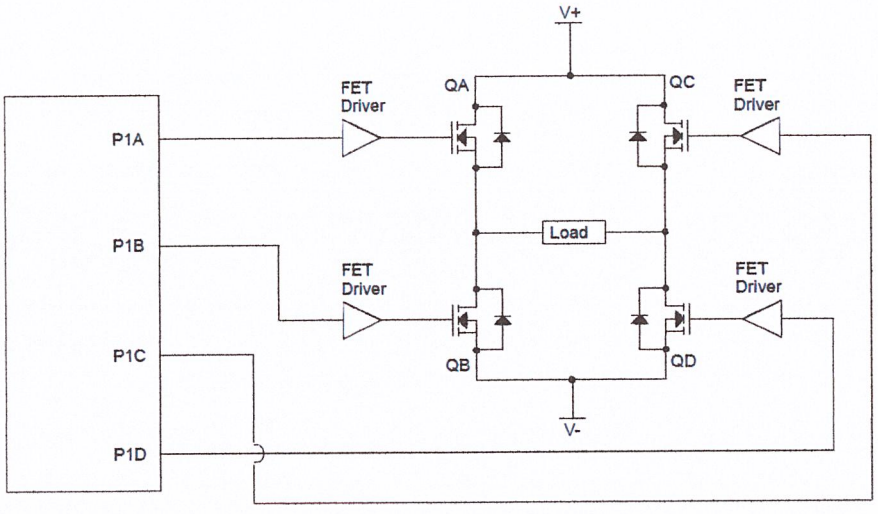
ส่วนแสดงผล มีใช้ในการแสดงสถานการณ์ทำงานเพื่อให้ใช้เป็นที่สังเกตความผิดปกติของมอดูลได้โดยง่าย

3.2.3 มอดูลระบบหยุดรถฉุกเฉิน

มอดูลหยุดรถฉุกเฉินประกอบด้วยส่วนแปลงสัญญาณแอนะล็อกเป็นดิจิทัลเพื่อใช้ในการวัดค่าแรงดันแบตเตอรี่ และแรงดันไฟเลี้ยงในระบบแคน, ส่วนรับอินพุตหยุดฉุกเฉินจากปุ่มหยุดฉุกเฉิน, รีเลย์เอาต์พุต ในมอดูลนี้ภาคจ่ายไฟจะถูกเปลี่ยนไปใช้แบบสวิตซ์ชิ่งเนื่องจากออกแบบให้ต่อเข้ากับแรงดันแบตเตอรี่โดยตรง ทำให้มีแรงดันตกคร่อมที่ไอซีรักษาระดับแรงดันมาก ถ้าใช้ไอซีแบบลิเนียร์จะทำให้พลังงานตกคร่อม ถูกแปลงเป็นพลังงานทำให้มีอุณหภูมิสูงตามไปด้วย และนอกจากนั้นยังมีส่วนการแสดงผลเพิ่มขึ้นซึ่งเมื่ออยู่ในสภาวะปกติจะเป็นการแสดงแรงดันแบตเตอรี่ และบอกถึงสาเหตุเมื่อเกิดเหตุการณ์ที่ทำให้หยุดฉุกเฉินขึ้น เช่นแรงดันแบตเตอรี่ต่ำ ปุ่มหยุดฉุกเฉินถูกกด และอื่นๆ

3.2.4 มอดูลขับมอเตอร์ขับเคลื่อน

มอดูลมอเตอร์ขับเคลื่อนใช้ความสามารถของมอดูลพีดับเบิ้ลยูเอ็มในโหมดปรับปรุง (Enhanced Mode PWM) ในไมโครคอนโทรลเลอร์ ซึ่งสามารถต่อวงจรได้ดังรูปที่ 3.5 ซึ่งในที่นี้จะใช้ออฟติคอลไอโซเลเตอร์เป็นตัวขับเฟท (FET, Field Effect Transistor)



รูปที่ 3.5 วงจรมอดูลขับมอเตอร์ขับเคลื่อน

ซึ่งสิ่งที่ควรต้องระวังคือช่วงตาย (Dead band) ซึ่งเกิดขึ้นจากเวลาใช้ในการปิดเฟทมากกว่าเปิดเฟททำให้การกลับข้างการหมุนอาจทำให้เกิดการลัดวงจรได้ ยกตัวอย่างเช่นในขณะที่ไมโครคอนโทรลเลอร์หยุดการนำกระแสที่พอร์ตพี1เอ และเริ่มนำกระแสทันทีที่พอร์ตพี1บี ในขณะที่เฟทเอยังไม่หยุดนำกระแสจะทำให้เกิดการลัดวงจรขึ้น ซึ่งการที่จะเกิดเหตุการณ์เช่นนี้ได้ คือ การที่เฟทหยุดการนำกระแสช้ากว่าเริ่มการนำกระแส และมีดิวตี้ไซเกิ้ลใกล้ 100% ซึ่งวิธีการหลีกเลี่ยง คือ หาเฟทที่เริ่มนำกระแสเร็วกว่าหยุดนำกระแส หรือ ลดความกว้างพัลส์ก่อนที่จะกลับด้านการหมุน

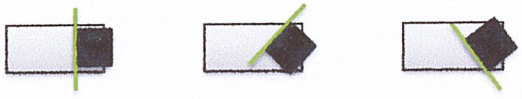
3.2.5 มอดูลขับมอเตอร์เบรก และมอดูลขับมอเตอร์ควบคุมทิศทาง

มอเตอร์เบรกและมอเตอร์ควบคุมทิศทางเป็นเซอร์โวมอเตอร์ ดังนั้นการใช้งานจึงเป็นเพียงการกำหนดความกว้างพัลส์ให้เหมาะสมกับแรงเบรก/ทิศที่ต้องการ การทำงานของมอดูลนี้จึงเป็นเพียงการรับค่าที่ต้องการจากแคนบัสแล้วนำไปคำควณหาความกว้างพัลส์ที่เหมาะสมต่อไป ซึ่งวงจรขับมอเตอร์นั้นเป็นวงจรที่ดัดแปลงมาจากเซอร์โวมอเตอร์ขนาดเล็ก นำเอาเอาต์พุตที่ได้ไปขับเฟทขนาดใหญ่อีกทอดหนึ่งทำให้ได้กำลังขับที่สูงขึ้น

3.2.6 มอดูลโซนาร์

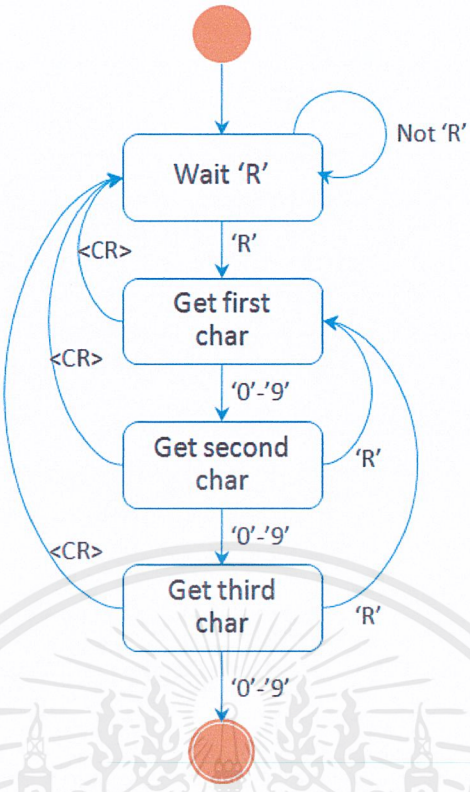
มอดูลโซน่าร์ประกอบด้วยเซอร์โวมอเตอร์สำหรับหมุนมอดูลโซนาร์ให้กวาดในมุมต่างๆ และมอดูลแอลวี-แม็กโซนาร์-อีซี3 (LV-MaxSonar-EZ3) ซึ่งเป็นมอดูลโซนาร์สำเร็จรูปขนาดเล็กให้

เอาท์พุตแบบอนุกรมอาร์เอส-232 (RS-232) ซึ่งสามารถวัดระยะได้ตั้งแต่ 6 นิ้วถึง 254 นิ้ว หรือประมาณ 6.45 เมตร



รูปที่ 3.6 มอดูลโซนาร์

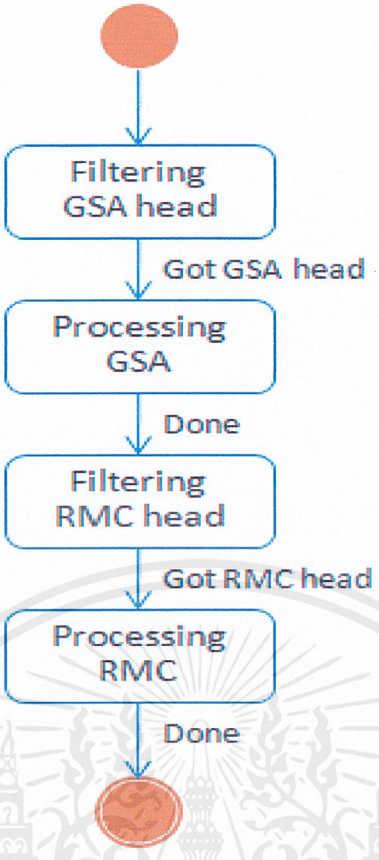
มอดูลโซนาร์อีซี3 ถ้าปล่อยให้ขารับข้อมูลลอยไว้หรือต่อเข้ากับลอจิก '0' จะทำให้มอดูลส่งข้อมูลระยะทางที่ได้ออกมาขึ้นต้นด้วยตัวอักษร 'R' ตามด้วยอักขระแอสกี้สามตัวแทนระยะห่างในหน่วยนิ้ว ปิดท้ายด้วย <CR> แต่ถ้าขารับข้อมูลต่อเข้ากับลอจิก '1' จะทำให้ไม่มีการส่งข้อมูลออกมา ดังนั้นการทำงานจึงเริ่มต้นด้วยการหมุนมอเตอร์ไปยังมุมที่ต้องการแล้วทำการหน่วงเวลาเพื่อให้มอเตอร์หมุนไปถึงมุมนั้นๆ แล้วจึงให้ลอจิก '0' กับขารับข้อมูลของโซนาร์ จากนั้นจึงเริ่มรับข้อมูล เมื่อมีตัวอักษรเข้ามาจะทำให้เกิดอินเทอร์รัพท์ขึ้น ซึ่งการเขียนโปรแกรมในอินเทอร์รัพท์เซอร์วิสรูทีน (ISR, Interrupt Service Routine) ต้องมีการเก็บสถานะไว้ว่ารับถึงตัวอักษรใดแล้ว จึงได้ออกแบบการทำงานด้วยสเตทไดอะแกรมในรูปที่ 3.7 ทำทั้งหมดสามครั้งแล้วนำข้อมูลมาโหวตกัน โดยตัดค่าที่ไกลค่าเฉลี่ยที่สุดทิ้งแล้วนำทั้งสองค่าที่เหลือมาทำการเฉลี่ยกัน ที่ต้องทำเช่นนี้เพราะมีในบางกรณีที่โซนาร์อ่านค่าผิดพลาดทำให้ค่าที่ออกมาผิดต่างจากความเป็นจริงมาก



รูปที่ 3.7 กระบวนการรับข้อมูลของมอดูลโซนาร์

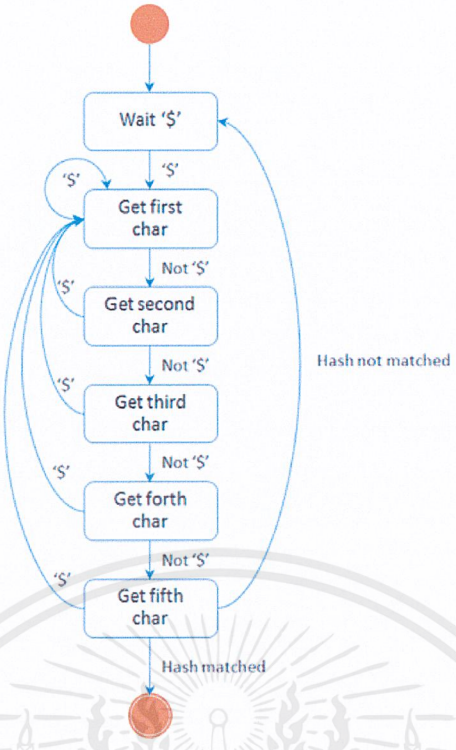
3.2.7 มอดูลรับข้อมูลจีพีเอส

การรับข้อมูลจากมอดูลจีพีเอสจะมีลักษณะคล้ายกับการรับข้อมูลจากมอดูลโซนาร์กล่าวคือ ในอินเทอร์รัพท์เซอร์วิสรูทีนจะต้องจำสถานะขณะนั้นได้ว่ารับถึงส่วนใดของข้อมูลแล้ว เพื่อที่จะได้ประมวลผลไปพร้อมๆกันกับการรับข้อมูล เพื่อเป็นการประหยัดเวลาและหน่วยความจำ โดยมีสเตทไดอะแกรมภาพรวมดังนี้



รูปที่ 3.8 สเตทไดอะแกรมของมอดูลจีพีเอส

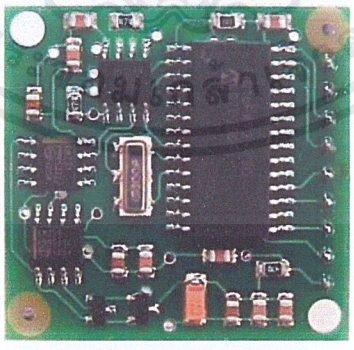
เริ่มต้นด้วยการรอรับข้อความที่ต้องการซึ่งสามารถตรวจสอบได้จากตัวอักษร 5 ตัวแรกของประโยค(ถัดจาก '$') ซึ่งการเปรียบเทียบจะใช้วิธีนำมาคำนวณเป็นแฮชโค้ด(Hash) และนำมาเปรียบเทียบกับแฮชโค้ดของประโยคที่ต้องการที่ได้เตรียมไว้ล่วงหน้า ซึ่งเป็นไปตามสเตทไดอะแกรมในรูปที่ 3.9 โดยปกติแล้วจะมาเป็นลำดับคือเริ่มด้วย "GPGSA" เมื่อตรวจสอบได้ว่าเป็นต้นประโยคที่ต้องการ จึงมาสู่สถานะต่อไปคือการคัดแยกเฉพาะข้อมูลที่ต้องการในประโยคแล้วนำไปเก็บไว้ในหน่วยความจำ โดยใช้วิธีการนับอักษร ',' ว่าถึงชุดข้อมูลที่ต้องการหรือไม่



รูปที่ 3.9 สเตทไดอะแกรมของการอ่านข้อมูลจีพีเอส

อย่างไรก็ตามในเบื้องต้นคอนโทรลเลอร์จะทำการตรวจสอบว่าสถานะของข้อมูลพิกัดเป็นอย่างไร แม่นยำพอที่จะใช้นำทางได้หรือไม่ ถ้าไม่ก็จะไม่ทำการถอดรหัสข้อความที่เหลือ แต่จะส่งข้อความไปบอกส่วนตัดสินใจว่ายังไม่สามารถระบุพิกัดได้

3.2.8 มอดูลเข็มทิศดิจิทัล

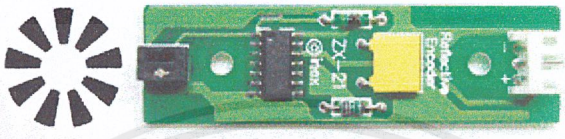


รูปที่ 3.10 มอดูลเข็มทิศดิจิทัล

เข็มทิศดิจิทัลที่ใช้เป็นรุ่น ซีเอ็มพีเอส-03 (CMPS-03) ซึ่งถูกออกแบบมาเพื่อช่วยในการกำหนดทิศทางการเคลื่อนที่ของหุ่นยนต์อัตโนมัติ โดยใช้ไอซีตรวจจับสนามแม่เหล็กเบอร์ เคเอ็มซี 51 (KMZ51) จำนวนสองตัวเพื่อให้มีความเร็วและความแม่นยำในการทำงานเพียงพอในการ

ตรวจจับสนามแม่เหล็กโลก มีการเชื่อมต่อให้เลือก คือ แบบความกว้างของคลื่นสัญญาณ และแบบบัสไอสแควซี ซึ่งผู้จัดทำเลือกที่จะใช้บัสไอสแควซีเนื่องจากทำให้มีการอ่านค่าความแม่นยำมากกว่าการใช้งานเพียงแต่ส่งคำสั่งอ่านค่ารีจิสเตอร์ให้ถูกตำแหน่งไปยังโมดูล ก็จะได้ค่าองศากลับมา ซึ่งมีรีจิสเตอร์ที่น่าสนใจคือ รีจิสเตอร์ 2,3 มีค่าตั้งแต่ 0-3599 ซึ่งสามารถเปลี่ยนไปเป็น 0-359.9 องศาได้โดยตรง

3.2.9 มอดูลวัดระยะทาง



รูปที่ 3.11 มอดูลวัดระยะทาง

มอดูลวัดระยะทางจะใช้มอดูลเอ็นโค้ดเดอร์ (Encoder) ของไอเน็กซ์ (i-nex) ซึ่งมีตัวเซ็นเซอร์เป็นโฟโต้ทรานซิสเตอร์ (Photo transistor) และไดโอดอินฟาเรด (Infrared diode) และไอซีขยายสัญญาณหนึ่งตัวเพื่อให้เอาต์พุตออกมาในระดับสัญญาณทีทีแอล ซึ่งสามารถต่อเข้ากับไมโครคอนโทรลเลอร์ได้ทันทีโดยไม่ต้องผ่านมอดูลแปลงแอนะล็อกเป็นดิจิทัล ซึ่งโมดูลนี้จะถูกวางใกล้จานที่ยึดติดกับล้อมีลายสลับระหว่างสีขาวและสีดำ

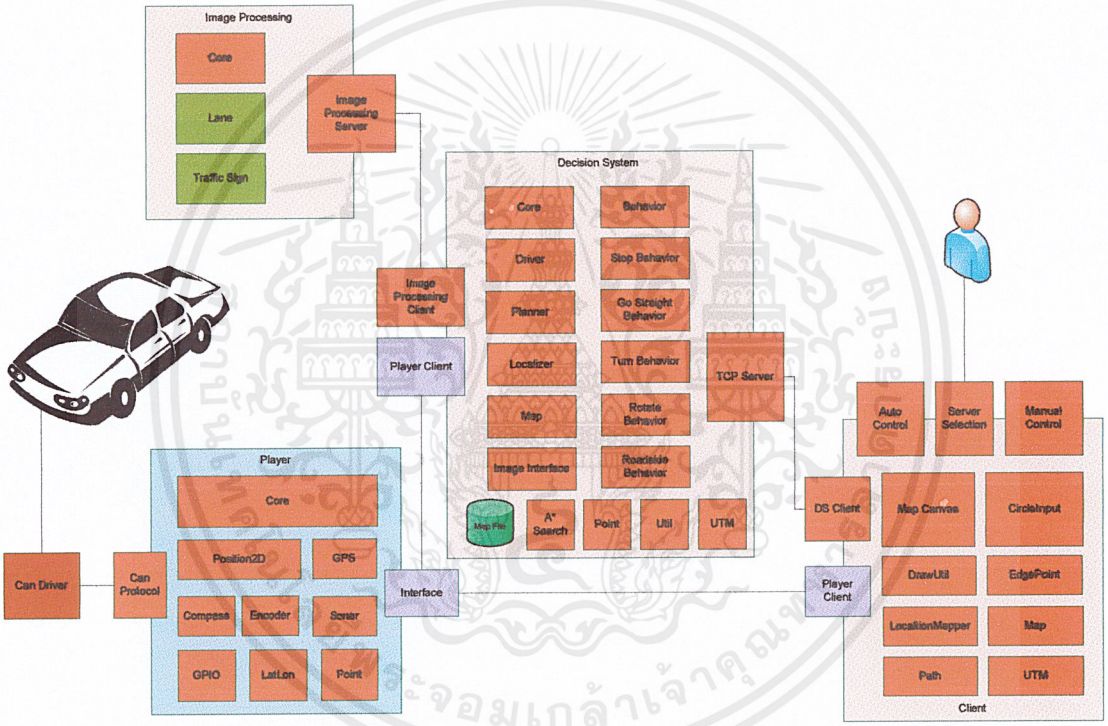
การนับจำนวนการตัดของสีขาวกับดำนั้นจะใช้มอดูลตัวนับ(Counter) ของไมโครคอนโทรลเลอร์ ซึ่งถูกกำหนดให้รับสัญญาณนาฬิกาจากภายนอก แล้วนำสัญญาณการตัดกันของสีขาวและสีดำนี้เองมาป้อนเป็นสัญญาณอินพุต การทำเช่นนี้ทำให้สามารถนับได้ด้วยความถี่สูงมากเนื่องจากการนับทั้งหมดทำโดยมอดูลฮาร์ดแวร์ เมื่อครบกำหนดเวลาที่ตั้งไว้อ่านค่าในตัวนับแล้วส่งไปยังหน่วยประมวลผลกลาง

3.2.10 มอดูลจ่ายไฟ (Power supply)

มอดูลจ่ายไฟนั้นมีหน้าที่ในการลดแรงดันไฟฟ้าจากแบตเตอรี่ซึ่งมีค่า 20-27 โวลต์ ให้เหลือตามที่แต่ละมอดูลต้อการ โดยจะมีทั้งหมดด้วยกัน 4 มอดูล คือ มอดูลภาคจ่ายไฟสำหรับมอดูลฮาร์ดแวร์ต่างๆ ต้องการกระแส 1 แอมแปร์แรงดัน 12 โวลต์ มอดูลจ่ายไฟสำหรับบอร์ดคอลิบริต้องการกระแส 1 แอมแปร์แรงดัน 12 โวลต์ มอดูลจ่ายไฟสำหรับส่วนประมวลผลภาพต้องการกระแส 2.5 แอมแปร์แรงดัน 5 โวลต์ มอดูลจ่ายไฟสำหรับแอคเซสพอยน์ ต้องการกระแส 1.2 แอมแปร์แรงดัน 5 โวลต์ ซึ่งผู้จำกัดเลือกที่จะใช้ภาคจ่ายไฟแบบสวิตซ์ชิ่งแบบเดียวกันทั้งหมด โดย

จะใช้ไอซีรักษาแรงดันเบอร์ แอลเอ็ม2576 (LM2576) ซึ่งทนกระแสได้สูงสุด 3 แอมแปร์ มีรุ่นสำหรับแรงดันเอาต์พุตให้เลือกใช้งานตามความต้องการ ซึ่งทำให้ใช้แผ่นวงจรลายเดียวกันในทุกมอดูลจ่ายไฟได้ทันที การที่ออกแบบให้ส่วนประมวลผลทุกตัวต้องต่อผ่านมอดูลจ่ายไฟนั้น นอกจากจะทำให้ได้รับไฟเลี้ยงที่เรียบมีแรงดันที่เหมาะสมแล้ว ยังสามารถช่วยป้องกันความเสียหายที่อาจเกิดจากการลัดวงจร และการต่อสายไฟอินพุตกลับขั้วได้อีกด้วย เนื่องจากในวงจรได้มีส่วนป้องกันไว้ในระดับหนึ่งอยู่แล้ว

3.3 การออกแบบระบบซอฟต์แวร์



รูปที่ 3.12 สถาปัตยกรรมของระบบซอฟต์แวร์

ระบบซอฟต์แวร์ ของโครงงานนี้จะแบ่ง 3 ส่วนหลักๆ ดังนี้

3.3.1 เพลเยอร์ (Player) และไดร์เวอร์ (driver) ของเพลเยอร์

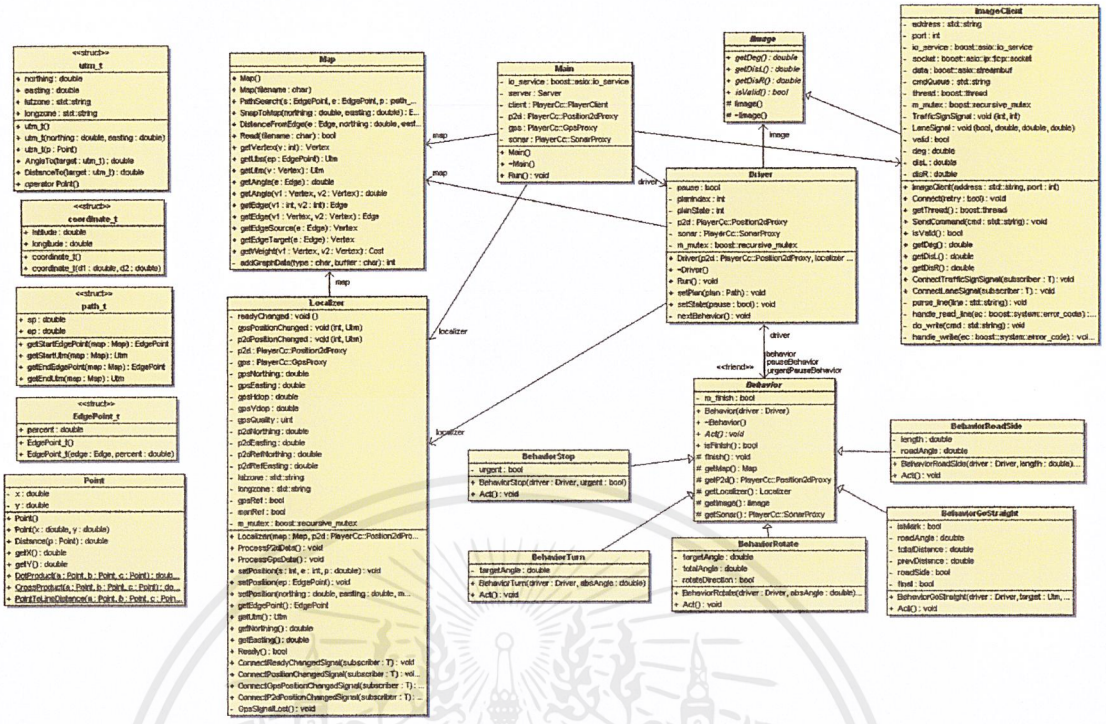
ส่วนนี้จะประกอบไปด้วยเพลเยอร์ (Player) และไดร์เวอร์ (driver) ของเพลเยอร์ ซึ่งทั้งหมดนั้นจะทำงานบนตัวรถโดยจะรอรับการเชื่อมต่อผ่านทางทีซีพีซ๊อกเก็ต (TCP socket) เพื่อควบคุมรถ โดยเพลเยอร์จะเป็นตัวประสานระหว่างทีซีพีซ๊อกเก็ต และไดร์เวอร์ที่เราเขียนขึ้นมา

ไดร์เวอร์ที่เขียนขึ้นนั้นจะเป็นการอิมพลีเม้นอินเตอร์เฟส (implement interface) ของอุปกรณ์ประเภทต่างๆที่เพลเยอร์ประกาศไว้ซึ่งได้แก่ ระบบควบคุมการเคลื่อนที่และระบุตำแหน่งในระนาบ 2 มิติ (position2d) ระบบการวัดระยะห่างโดยใช้คลื่นความถี่อัลตราโซนิกส์ (sonar) ระบบระบุพิกัดโดยใช้จีพีเอส (Global Positioning System) ไดร์เวอร์ที่เขียนขึ้นทั้งหมดในส่วนนี้จะติดต่อกับไมโครคอนโทรเลอร์ที่เชื่อมต่ออยู่กับตัวอุปกรณ์จริงๆ ผ่านทางแคนบัส (CAN bus) สำหรับระบบควบคุมการเคลื่อนที่และระบุตำแหน่งในระนาบ 2 มิตินั้นจะอาศัย มอเตอร์ต่างๆเพื่อขับเคลื่อนรถ หยุดรถและควบคุมทิศทางของล้อ อุปกรณ์ตรวจจับความเร็วเพื่อควบคุมความเร็วในการเคลื่อนที่และเข็มทิศ อิเล็กทรอนิกส์เพื่อความคุมทิศทางการเคลื่อนที่ สำหรับการระบุตำแหน่งแบบสัมพัทธ์นั้นจะใช้ข้อมูลจากอุปกรณ์ตรวจจับความเร็ว และเข็มทิศในการคำนวณหาตำแหน่ง ไดร์เวอร์ทั้งหมดจะทำงานเป็นเธรด (thread) ในโปรเซส (process) ของเพลเยอร์

3.3.2 ระบบการตัดสินใจและควบคุมรถ

ระบบนี้จะรอรับคำสั่งจากผู้ใช้และควบคุมรถให้ปฏิบัติตามคำสั่งนั้น โดยจะมีแผนที่อยู่ภายในระบบด้วยเพื่อใช้สำหรับการหาเส้นทางสำหรับเคลื่อนที่ไปยังเป้าหมายที่ผู้ใช้งานระบุ ซึ่งระบบนี้จะติดต่อกับ ระบบติดต่อกับผู้ใช้ เพลเยอร์และระบบประมวลผลภาพผ่านทางทีซีพีซ็อกเก็ต

ระบบนี้จะทำการพัฒนาโดยใช้ภาษาซีพลัสพลัส (C++) โดยใช้ไลบรารี่มาตรฐาน (standart library) และบูสซีพลัสพลัสไลบรารี่ (Boost C++ library) เท่านั้น เพื่อให้สามารถนำไปทำงานบนหลายๆระบบปฏิบัติการได้ ระบบนี้มีคลาส (class) หลักๆตามรูปที่ 3.13 โดยแต่ละคลาสจะมีรายละเอียดดังนี้



รูปที่ 3.13 คลาสไดอะแกรมของระบบตัดสินใจ

- **ส่วน Main**

ส่วนนี้จะเป็นส่วนที่ตั้งแต่และเริ่มต้นการทำงาน (Initialize) ส่วนอื่นๆ เชื่อมต่อซิกเนล (signal) ต่างๆ เข้ากับสล็อต (slot) ที่จำเป็น และเริ่มต้นเธรดของทีซีพีเซอร์เวอร์ (tcp server) และเธรดของเพลเยอร์ไคลแอนต์ (player client)

- **ส่วน Image Processing Client**

ส่วนนี้จะทำหน้าที่ เชื่อมต่อกับระบบประมวลผลภาพ โดยจะทำการเปิดการเชื่อมต่อทีซีพี (tcp connection) และรับข้อมูลมาเก็บไว้เพื่อรอให้ส่วนอื่นเรียกใช้งาน สำหรับข้อมูลที่รับมานั้นจะมีการประมวลผลเบื้องต้นก่อนเพื่อกรองข้อมูลที่ ผิดพลาดออกไป

- **ส่วน Player Client**

ส่วนนี้จะเป็นส่วนที่เชื่อมต่อกับเพลเยอร์ผ่านทางทีซีพี โดยส่วนนี้เราได้นำซีพลัสพลัสไคลแอนต์ไลบรารี่ (C++ Client library) ของเพลเยอร์มาใช้

- **ส่วน Map**

คลาสนี้จะทำหน้าที่ในการโหลดแผนที่ขึ้นมาจากไฟล์ที่เก็บข้อมูลแผนที่และหาเส้นทางในแผนที่ โดยข้อมูลของถนนจะเก็บโดยใช้กราฟแบบมีทิศทาง (directed graph) ที่มี

โครงสร้างข้อมูลแบบรายการ (adjacency list) สาเหตุที่ใช้กราฟแบบรายการ (adjacency list) ในการเก็บก็เพราะว่ากราฟที่แทนถนนนั้นจะมีปริมาณขอบ (edge) ต่อจุด (vertex) น้อย ถ้าหากเก็บแบบเมตริกซ์ (adjacency matrix) จะเปลืองพื้นที่มาก แต่ละจุดของกราฟนั้นจะมี คุณสมบัติเป็นพิกัดทั้งแบบทรงกลม (Latitude Longitude) และพิกัดแบบยูทีเอ็ม (UTM) สำหรับแต่ละขอบ (edge) ก็จะมีความยาวของถนนเป็นคุณสมบัติสำหรับการคำนวณต่างๆทั้งหมด จะคำนวณโดยใช้พิกัดแบบยูทีเอ็ม เนื่องจากเป็นระบบพิกัดที่สะดวกต่อการคำนวณ

คลาส Map จะสามารถหาเส้นทางระหว่างจุดบนขอบ edge สองจุดได้ โดยการใช้การค้นหาแบบ A* ที่ได้ดังแปลงจากการค้นหาระหว่างจุดเป็นการค้นหาระหว่างจุดบนขอบแทน นอกจากนี้ คลาส Map ยังสามารถแนบพิกัดต่างๆ เข้ากับขอบได้โดยจะแนบเข้ากับจุดบนขอบที่ใกล้ที่สุด

- **ส่วน Point**

คลาสนี้แทนจุด 1 จุด และสามารถคำนวณดอทโปรดักส์ (dot product), คลอสโปรดักส์ (cross product) และระยะห่างระหว่าจุดไปยังเส้นได้

- **ส่วน Utm**

คลาสนี้แทนพิกัดแบบยูทีเอ็ม (UTM) และสามารถคำนวณหาทิศทางไปยังพิกัดอื่นจากพิกัดนี้ได้ และคำนวณระยะห่างระหว่างพิกัด 2 พิกัดได้

- **ส่วน Localizer**

คลาสนี้รับผิดชอบในการระบุพิกัดของรถทั้งแบบยูทีเอ็ม และตำแหน่งบนแผนที่ โดยจะใช้ข้อมูลจากอินเตอร์เฟส Position2d ของเพลเยอร์และข้อมูลจากจีพีเอสนอกจากนี้ยังใช้ข้อมูลที่ได้จากพฤติกรรมต่างๆมาปรับแก้ตำแหน่งที่ผิดพลาดด้วย

- **ส่วน Planner**

คลาสนี้รับผิดชอบในการวางแผนการเคลื่อนที่ของรถเพื่อไปยังปลายทาง โดยจะรับเป้าหมายมาจากผู้ใช้ผ่านทางทีซีพีซ็อกเก็ตและทำการหาเส้นทางจากตำแหน่งปัจจุบันไปยังปลายทางโดยเรียกใช้งานคลาส Map หลังจากได้เส้นทางแล้วก็จะส่งต่อไปยังส่วนของไดร์เวอร์

- **ส่วน Driver**

คลาสนี้จะทำหน้าที่เลือกและสร้างอินสแตน (instance) ของคลาส Behavior ที่เหมาะสมต่างๆ มาทำการควบคุมรถตามลำดับที่เหมาะสม เพื่อเดินตามเส้นทางที่ได้รับมาจากส่วน Planner

- **ส่วน Behavior**

เป็นคลาสนามธรรม (abstract class) ของพฤติกรรมทั้งหมด

- **ส่วน Stop Behavior**

เป็นพฤติกรรมที่จะทำการหยุดรถ โดยสามารถเลือกหยุดทั้งแบบปกติและแบบฉุกเฉินได้

- **ส่วน Go Straight Behavior**

เป็นพฤติกรรมที่จะทำการขับ เคลื่อนรถไปตามถนน และหลบหลีกสิ่งกีดขวาง โดยจะพยายามขับเคลื่อนรถให้เกาะขอบถนนด้านซ้ายอยู่ตลอดเวลา ซึ่งพฤติกรรมนี้จะต้องอาศัยข้อมูลตำแหน่งจากคลาส Localizer ข้อมูลทิศทางของรถจากเข็มทิศดิจิตอล ข้อมูลของถนนที่ได้จากการทำการประมวลภาพและข้อมูลจากอัลตร้าโซนิค

- **ส่วน Turn Behavior**

เป็นพฤติกรรมที่จะเลี้ยวรถไปยังองศาที่กำหนด โดยอาศัยข้อมูลจากเข็มทิศดิจิตอล ซึ่งพฤติกรรมนี้จะใช้สำหรับการเลี้ยวระหว่างจุดแต่ละจุดในแผนที่

- **ส่วน Rotate Bahavior**

เป็นพฤติกรรมที่จะกลับรถไปยังทิศทางที่กำหนด โดยจะอาศัยข้อมูลจากเข็มทิศดิจิตอลและข้อมูลจากการทำการประมวลผลาพ

- **ส่วน Road Side Behavior**

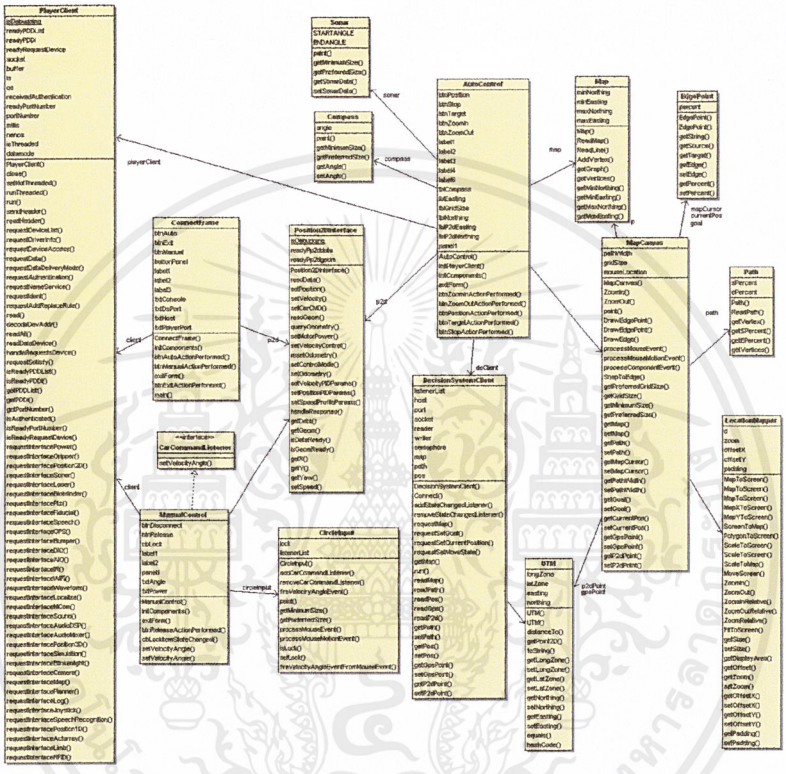
เป็นพฤติกรรมที่จะทำการขับเคลื่อนรถให้ชิดขอบทางด้านซ้าย โดยจะอาศัยข้อมูลที่ได้จากการประมวลผลภาพเป็นหลัก

3.3.3 ระบบติดต่อกับผู้ใช้

ส่วนนี้จะเป็นโปรแกรมแบบจียูไอ (GUI (Graphic User Interface)) ที่ทำงานอยู่บนเครื่องของผู้ใช้โดยสามารถทำงานบนเครื่องคอมพิวเตอร์ส่วนบุคคล และพ็อกเก็ตพีซี (Pocket PC) ส่วนนี้

จะใช้สำหรับติดต่อสั่งงานและตรวจสอบสถานะของการทำงานของหุ่นยนต์ผ่าน ทางระบบเครือข่าย

ระบบในส่วนนี้จะพัฒนาโดยใช้ภาษาจาวา (Java) เพื่อให้สามารถทำงานทั้งบนวินโดวส์ (Windows) และบนวินโดวส์โมบาย (Windows Mobile) ได้โดยไม่ต้องเปลี่ยนแปลงใดๆทั้งสิ้น สำหรับการนำไปใช้งานบนวินโดวส์โมบายจะต้องลงเจวีเอ็ม (JVM) สำหรับวินโดวส์โมบายโดยเฉพาะ ซึ่งเราได้เลือกใช้มายไซฟุเจวีเอ็ม (Mysaifu JVM)



รูปที่ 3.14 คลาสไดอะแกรมของระบบติดต่อกับผู้ใช้

ระบบนี้ประกอบไปด้วยคลาส ต่างๆดังนี้

- **คลาส control.DecisionSystemClient**

เป็นไคนแอนต์ของระบบตัดสินใจ มีการทำงานเป็นเธรดจำนวน 1 เธรด เพื่อรับส่งข้อมูล คลาสนี้อิมพลีเม้น (implement) อินเตอร์เฟส java.Runnable

- **คลาส form.AutoControl**

สืบทอดจากคลาส java.awt.Frame เป็นฟอร์มที่ใช้สำหรับควบคุมรถแบบอัตโนมัติ

- **คลาส form.ManualControl**

สืบทอดจากคลาส java.awt.Frame เป็นฟอร์มที่ใช้สำหรับควบคุมรถด้วยตนเอง

- **คลาส form.ConnectFrame**

สืบทอดจากคลาส java.awt.Frame เป็นฟอร์มเริ่มต้นที่ใช้สำหรับระบุแอ็ดเดส (address) และพอร์ต (port) ของรถที่ต้องการควบคุม และเลือกว่าต้องการควบคุมรถแบบใด

- **คลาส form.canvas.CircleInput**

สืบทอดจากคลาส java.awt.Panel เป็นส่วนที่ไว้สำหรับให้ผู้ใช้ระบุความเร็วและองศาล้อรถ สำหรับการควบคุมรถด้วยตนเอง

- **คลาส form.canvas.Compass**

สืบทอดจากคลาส java.awt.Panel เป็นส่วนที่ไว้สำหรับแสดงค่าองศาของรถ ที่อ่านได้จาก เข็มทิศผ่านทางอินเตอร์เฟส position2d ของเพลเยอร์

- **คลาส form.canvas.MapCanvas**

สืบทอดจากคลาส java.awt.Panel เป็นส่วนที่ไว้สำหรับแสดงแผนที่ ตำแหน่งจีพีเอส ตำแหน่งรถปัจจุบันที่ได้จากคลาส localizer ของระบบตัดสินใจ นอกจากนี้ยังสำหรับให้ผู้ใช้ระบุตำแหน่งรถด้วยตนเองและกำหนดปลายทางที่ต้องการไป

- **คลาส form.canvas.Sonar**

สืบทอดจากคลาส java.awt.Panel เป็นส่วนที่ไว้สำหรับแสดงค่าระยะห่างที่มอดูลอัลตร้าโซนิคอ่านได้ผ่านทางอินเตอร์เฟสโซนาร์ของเพลเยอร์

- **คลาส map.CoordinateConversion**

เป็นคลาสช่วยเหลือ (utility class) สำหรับแปลงค่าพิกัดระหว่างพิกัดแบบทรงกลมกับพิกัดแบบ UTM

- **คลาส map.DrawUtil**

เป็นคลาสช่วยเหลือ (utitliy class) สำหรับการคำนวณต่างๆ สำหรับการวาดแผนที่ เช่น การขยายจากเส้นไปเป็นขอบเขตการหาดอทโปรดักส์ (Dot Product), คลอสโปรดักส์ (Cross Product), หาระยะห่างระหว่างจุดกับเส้น

- **คลาส map.EdgePoint**

เป็นคลาสที่แทนจุดบนขอบสำหรับการอ้างอิงตำแหน่งต่างๆบนแผนที่

- **คลาส map.LocationMapper**

เป็นคลาสสำหรับแปลงระหว่างพิกัดบนแผนที่แบบยูทีเอ็มเป็นพิกัดบนจอสำหรับให้คลาส map.Map เรียกใช้งาน โดยจะมีการคำนวณอ๊อฟเซต (offset) และระยะการย่อขยายด้วย

- **คลาส map.Map**

เป็นคลาสที่แทนแผนที่ เก็บแผนที่โดยใช้โครงสร้างข้อมูลแบบกราฟ คลาสนี้สามารถโหลดแผนที่จากอินพุตสตรีม (InputStream) ต่างๆได้ เช่นแบบไฟล์หรือเครือข่าย

- **คลาส map.Path**

เป็นคลาสที่ใช้แทนเส้นทาง 1 เส้น โดยภายในจะประกอบด้วยจุดต่างๆบนเส้นทาง รวมทั้งจุดเริ่มต้นและจุดสุดท้าย ส่วนของขอบที่เริ่มต้นและปลายทาง

- **คลาส map.UTM**

เป็นคลาสที่ใช้แทนพิกัดแบบยูทีเอ็ม

สำหรับคลาสที่อยู่ใน package javaclient2 จะเป็นไคลแอนต์ภาษาจาวา (java) สำหรับเพลเยอร์เวอร์ชัน 2.0 ซึ่งต้องนำมาแก้ไขบางส่วนเพื่อให้สามารถใช้กับเพลเยอร์เวอร์ชัน 2.1 ได้

3.4 ระบบประมวลผลภาพ

ในการทำงานของส่วนนี้เราได้ออกแบบให้มีการทำงานแบ่งออกเป็น 2 ส่วนด้วยกันคือ ส่วนที่ทำหน้าที่หาพื้นที่ที่เป็นส่วนของถนน และส่วนที่ทำหน้าที่หาและวิเคราะห์ป้ายจราจรซึ่งแต่ละส่วนจะทำการประมวลผลเมื่อได้ผลลัพธ์ก็จะส่งผลลัพธ์ที่ได้ไปให้กับส่วนตัดสินใจต่อไปโดยใช้โปรโตคอลทีซีพี (TCP Protocol) ในการส่งข้อมูล ซึ่งในการทำงานทั้งหมดนี้ได้ทำการพัฒนาบนไลบรารี่โอเพนซีวี (OpenCV library) บนระบบปฏิบัติการวินโดวส์ (Microsoft Windows)

ซึ่งในการออกแบบการทำงานนั้นจะแบ่งการทำงานออกเป็น 2 โปรเซส (Process) โดยโปรเซสแรกจะทำหน้าที่ในการหาขอบของถนน และอีกโปรเซสหนึ่งจะทำหน้าที่หาและวิเคราะห์ป้ายจราจร โดยจะมีหลักการทำงานคือ ให้โปรเซสที่ทำหน้าที่หาส่วนของถนนทำงานเป็น 3 เท่าของโปรเซสที่ทำหน้าที่หาป้ายจราจรและส่วนหาถนนจะมีความสำคัญ (Priority) มากกว่าส่วนที่ทำหน้าที่หาป้ายจราจรเช่น เมื่อโปรเซสที่ทำหน้าที่หาถนนต้องการจะรัน (Run) ในขณะที่โปรเซสที่หาป้ายทำงานอยู่ก็ให้โปรเซสที่หาถนนทำงานก่อน เป็นต้น

3.4.1 การออกแบบการหาส่วนของถนน

กระบวนการของการหาส่วนของถนน นั้นจะทำการจับภาพมาจากกล้องเวปแคม เพื่อเอามาหาเส้นขอบของถนน เพื่อใช้ข้อมูลของเส้นขอบถนนนั้นมาอธิบายสิ่งแวดล้อมรอบๆรถ เช่น ระยะห่างจากขอบถนนทั้งทางด้านซ้ายและขวา หากข้อมูลที่ได้เมื่อทำการเช็คแล้วมีข้อผิดพลาดก็จะทำการส่งค่าเพื่อบอกความ ถูกต้องหรือมีการกรองข้อมูลก่อนที่จะทำการส่งออกไปด้วย โดยผลลัพธ์ของข้อมูลที่ส่งออกไปนั้นจะส่งออกผ่านทางโปรโตคอลทีซีพีที่ได้ เปิดรอรับการเชื่อมต่อจากระบบอื่นๆ ไว้ก่อนหน้านี้แล้ว

3.4.2 การออกแบบการวิเคราะห์ป้ายจราจร

ในการหาป้ายจราจรนั้นเริ่มต้นจะต้องหาให้ได้ก่อนว่าป้ายที่เราต้องการอยู่ที่ส่วนไหนของภาพและในขอบเขตของเรานั้นได้กำหนดป้ายที่เป็นป้ายเตือน (เป็นป้ายที่มีสีเหลืองทั้งหมด) ซึ่งโดยปกติแล้วนั้นป้ายจราจรจะอยู่ทางฝั่งซ้ายของถนน เราจึงจะทำการประมวลผลเฉพาะฝั่งซ้ายของถนนเท่านั้นโดยการทำงานในส่วนของการหาป้ายจราจรโดยจะหาจุดที่มีความเป็นได้ของป้ายจากจุดที่มีสีที่ตรงกับสีของป้ายที่ต้องการ

3.5 การสื่อสารระหว่างระบบต่างๆ

3.5.1 การสื่อสารระหว่างเพลเยอร์กับไมโครคอนโทลเลอร์

การสื่อสารระหว่างเพลเยอร์กับไมโครคอนโทลเลอร์ จะทำการสื่อสานผ่านแคนบัส (CAN Bus) โดยจะมีเมสเซส (message) ประเภทต่างๆดังนี้

ตารางที่ 3.1 รายละเอียดโปรโตคอลของการติดต่อสื่อสารระหว่างเพลเยอร์กับมอดูลต่างๆ

ประเภท	ความสำคัญ	Id 1	Id 2	Id 3	ความยาว	ประเภท	ต้นทาง	ปลายทาง	ความดี	รายละเอียด
คำสั่ง	0	00	0000	00000	0	-	*	*	-	จะส่งเมื่อเกิดเหตุการณ์ฉุกเฉินขึ้น ซึ่งจะทำให้ทุกมอดูลหยุดทำงานและระบบเบรคทำการเบรคเต็มที่
คำสั่ง	1	00	0001	00100	1	int8	ส่วนกลาง	ระบบควบคุมเบรค	-	กำหนดความแรงของการเบรค โดย 0 หมายถึงไม่เบรค และ 100 หมายถึงเบรคเต็มที่
คำสั่ง	2	00	0001	01000	1	int8	ส่วนกลาง	ระบบควบคุมล้อหน้า	-	กำหนดองศาล้อหน้า โดย -90 หมายถึง 90 องศาทางขวา และ 90 หมายถึง 90 องศาทางซ้าย
คำสั่ง	3	00	0001	01100	1	int8	ส่วนกลาง	ระบบควบคุมการขับเคลื่อน	-	กำหนดความแรงของการขับเคลื่อนไปด้านหน้า โดย 0 หมายถึงไม่ขับเคลื่อนไปด้านหน้า 100 หมายถึงขับเคลื่อนไปด้านหน้าเต็มกำลัง
ข้อมูล	4	01	1100	00011	2	int8, uint8	ระบบตรวจหาสิ่งกีดขวาง	ส่วนกลาง	?	ส่งระยะที่ระบบตรวจหาสิ่งกีดขวางตรวจจับได้ โดยจะส่งหมายเลขขององศา ตามด้วยระยะที่ตรวจได้ หน่วยเป็นนิ้ว

ตารางที่ 3.1 (ต่อ) รายละเอียดโปรโตคอลของการติดต่อสื่อสารระหว่าง Player กับมอดูลต่างๆ

ประเภท	ความสำคัญ	Id 1	Id 2	Id 3	ความยาว	ประเภท	ต้นทาง	ปลายทาง	ความถี่	รายละเอียด
คำสั่ง	6	10	1000	00000	0	-	ส่วนกลาง	ส่วนวัดระยะทาง	-	สั่งให้ส่วนวัดระยะทางเริ่มนับจาก 0 ใหม่
ข้อมูล	7	11	1100	01000	4	int32	ส่วนวัดระยะทาง	ส่วนกลาง	64 ms	ส่งจำนวนครั้งของการนับของมอดูลวัดระยะทาง
ข้อมูล	8	11	1100	01100	2	int16	เข็มทิศ	ส่วนกลาง	64 ms	ส่ง องศาของรถ มีค่าตั้งแต่ 0 ถึง 3599 โดย 0 หมายถึงเข็มทิศหันไปทางทิศเหนือ 900 หมายถึงเข็มทิศหันไปทางทิศตะวันออก 1800 หมายถึงเข็มทิศหันไปทางทิศใต้ 2700 หมายถึงเข็มทิศหันไปทางทิศตะวันตก
ข้อมูล	9	11	1100	10000	8	int32, int32	จีพีเอส	ส่วนกลาง	1 s	ส่งค่าละติจูดของรถในรูปแบบของ DD MM.mmmm โดย int32 ชุดแรกจะเป็นค่าของ DDMM ส่วน int32 ชุดหลังจะเป็นค่าของ mmmm

ตารางที่ 3.1 (ต่อ) รายละเอียดโปรโตคอลของการติดต่อสื่อสารระหว่าง Player กับมอดูลต่างๆ

ประเภท	ความสำคัญ	Id 1	Id 2	Id 3	ความยาว	ประเภท	ต้นทาง	ปลายทาง	ความถี่	รายละเอียด
ข้อมูล	9	11	1100	10001	8	int32, int32	จีพีเอส	ส่วนกลาง	1 s	ส่งค่าลองจิจูดของรถในรูปแบบของ DD MM.mmmm โดย int32 ชุดแรกจะเป็นค่าของ DDMM ส่วน int32 ชุดหลังจะเป็นค่าของ mmmm
ข้อมูล	9	11	1100	10010	5	char, int16, int16	จีพีเอส	ส่วนกลาง	1 s	ส่งค่าสถานะของจีพีเอส ค่า HDOP และ VDOP ของรถ โดย char ตัวแรกจะเป็นรหัสแอสกี้ (ASCII) ถ้าเป็นตัว A หมายถึง available ส่วน V จะหมายถึง invalid int16 ตัวถัดไปจะเป็นค่าของ HDOP คูณ 10 และ int16 ตัวสุดท้ายจะเป็นค่าของ VDOP คูณ 10

ซึ่งเมสเซส (message) ได้มีการตั้ง Id ตามลำดับความสำคัญ เพื่อให้เมสเซสที่มีความสำคัญสูงกว่าถูกส่งก่อน ถ้าหากว่ามีเมสเซสหลายๆเมสเซสต้องการส่งพร้อมกัน นอกจากนี้ยังได้ตั้ง Id ตามประเภทของอุปกรณ์ที่รับเมสเซสนั้นๆ เพื่อให้อุปกรณ์ต่างๆที่รับเมสเซส ตั้งตัวกรองได้สะดวก

ความเร็วในการรับส่ง เราได้เลือกใช้ความเร็ว 125 kbps เนื่องจากเป็นความเร็วที่เพียงพอและสามารถใช้สายสัญญาณได้ยาวเพียงพอด้วย โดยตารางด้านล่างจะแสดงถึงปริมาณข้อมูลมากที่สุดที่เป็นไปได้ภายในระยะเวลา 1 วินาที

ตารางที่ 3.2 ปริมาณข้อมูลที่รับส่งบนแคนบัส

ประเภทข้อมูล	เฮดเดอร์ของแคน (ไบท์)	ปริมาณข้อมูล (ไบต์)	ปริมาณข้อมูลทั้งหมด (ไบต์)	ความถี่สูงสุดใน 1 วินาที	ปริมาณบิตต่อวินาที
คำสั่งควบคุมเบรก	5.5	1	6.5	16	832
คำสั่งควบคุมองศาล้อหน้า	5.5	1	6.5	16	832
คำสั่งควบคุมกำลังขับเคลื่อน	5.5	1	6.5	16	832
ข้อมูลจากส่วนตรวจสอบสิ่งกีดขวาง	5.5	2	7.5	32	1920
ข้อมูลจากส่วนวัดระยะทาง	5.5	4	9.5	16	1216
ข้อมูลจากเข็มทิศ	5.5	2	7.5	16	960
ข้อมูลละติจูด จากจีพีเอส	5.5	8	13.5	2	216
ข้อมูลลองจิจูด จากจีพีเอส	5.5	8	13.5	2	216
ข้อมูลสถานะ, hdop, vdop จากจีพีเอส	5.5	5	10.5	2	168
รวมทั้งหมด					7192

3.5.2 การสื่อสารระหว่างระบบตัดสินใจกับเพลเยอร์

การเชื่อมต่อระหว่างระบบตัดสินใจกับเพลเยอร์จะเป็นไปตามโปรโตคอล (Protocol) ของเพลเยอร์ผ่านทางทีซีพีไอพี (TCP/IP) โดยเราได้ใช้ซีพลัสพลัสไคลแอนต์ไลบรารี่ (C++ Client library) ของเพลเยอร์ในการเชื่อมต่อ สำหรับอินเตอร์เฟสต่างๆของเพลเยอร์ที่เลือกใช้งานมี Position2d Sonar และ GPS

3.5.3 การสื่อสารระหว่างระบบตัดสินใจกับระบบการประมวลผลภาพ

การสื่อสาร ระหว่าง 2 ระบบนี้จะสื่อสารผ่านทางทีซีพีซ็อกเก็ต (Tcp Socket) เนื่องจากว่าสองระบบนี้จำเป็นต้องทำงานบนฮาร์แวร์คนละตัวกัน เพราะว่าเอ็มเบ็ดเด็ดบอร์ด (embedded board) มีข้อจำกัดทางด้านความสามารถในการประมวลผล โดยโปรโตคอล (protocol) ที่ใช้จะเป็นแบบเชิงข้อความ (text based) ทั้งหมดเพื่อให้สามารถตรวจสอบการทำงานได้สะดวก

โดยการสื่อสารจะมีเมสเซส (message) 2 ประเภทดังนี้

3.5.3.1 ข้อมูลขอบถนน

ข้อมูลขอบถนน จะส่งมาทุกๆครั้งหลังจากส่วนประมวลผลภาพทำงานเสร็จ มีความถี่ในการส่งประมาณ 3 ครั้งต่อวินาที ซึ่งจะมีข้อมูลต่างๆดังนี้

- เฟลก (flag) ไว้ระบุว่าขณะนี้สามารถประมวลผลได้ข้อมูลที่ถูกต้องหรือไม่
- องศาที่รถทำมุมกับถนน หน่วยเป็นเรเดียน (radian)
- ระยะห่างจากขอบถนนด้านซ้าย หน่วยเป็นเมตร ถ้าหากว่าหาขอบด้านซ้ายไม่เจอจะเป็น -1
- ระยะห่างจากขอบถนนด้านขวา หน่วยเป็นเมตร ถ้าหากว่าหาขอบด้านขวาไม่เจอจะเป็น -1

โดย จะส่งมาในรูปแบบของ "L valid deg disL disR" เช่น "L 1 -0.349066 1.352 2.77467" จะหมายความว่าขณะนี้สามารถประมวลผลข้อมูลได้ โดยรถทำมุม -0.349066 เรเดียนกับถนน อยู่ห่างจากขอบถนนด้านซ้าย 1.352 เมตร อยู่ห่างจากขอบถนนด้านขวา 2.77467 เมตร

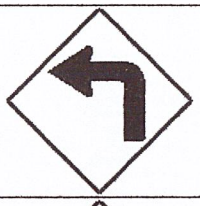
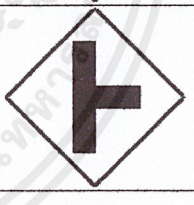
3.5.3.2 ข้อมูลป้ายจราจร

ข้อมูลป้ายจราจรที่ตรวจสอบได้ จะส่งมาเมื่อตรวจพบป้ายจราจร โดยจะส่ง 1 ครั้งต่อป้ายจราจร 1 ป้ายเท่านั้น ซึ่งจะมีข้อมูลต่างๆดังนี้

- หมายเลขของป้าย ตามตารางที่ 3.3
- ระยะห่างระหว่างป้ายกับถนน หน่วยเป็นเซนติเมตร

โดยจะส่งมาในรูปแบบของ "T index length" เช่น "T 1 837" หมายความว่าตรวจเจอป้ายเลี้ยวขวาที่ระยะห่างจากรถ 837 เซนติเมตร

ตารางที่ 3.3 หมายเลขของป้ายจราจร

หมายเลข	ป้าย	รูป
0	เลี้ยวซ้าย	
1	เลี้ยวขวา	
2	สามแยกตัวที	
3	สี่แยก	
4	สามแยกด้านซ้าย	
5	สามแยกด้านขวา	

3.5.4 การสื่อสารระหว่างระบบตัดสินใจกับระบบติดต่อกับผู้ใช้

การสื่อสารระหว่าง 2 ระบบนี้จะสื่อสารผ่านทางทีซีพีซ็อกเก็ต (Tcp Socket) เพื่อให้ระบบสองระบบมีอิสระต่อกัน โดยโปรโตคอล (protocol) ที่ใช้จะเป็นแบบเชิงข้อความ (text based) ทั้งหมดเพื่อให้สามารถตรวจสอบการทำงานได้สะดวก โดย 1 บรรทัดจะเท่ากับ 1 คำสั่งที่รับส่งกัน

ข้อมูลที่ส่งจากระบบติดต่อกับผู้ใช้ไปสู่ระบบตัดสินใจ จะมีดังตารางที่ 3.4

ตารางที่ 3.4 ข้อมูลที่ส่งจากระบบติดต่อกับผู้ใช้ไปสู่ระบบตัดสินใจ

คำสั่ง	รายละเอียด
Getmap	ร้องขอให้ระบบตัดสินใจส่งข้อมูลแผนที่มาให้
Setgoal[EdgePoint]	ตั้งเป้าหมายใหม่ให้กับระบบตัดสินใจ
Setpos[EdgePoint]	ตั้งตำแหน่งปัจจุบันของรถด้วยตนเอง
State pause	หยุดรถชั่วคราว
State resume	ขับเคลื่อนรถต่อ

ส่วนข้อมูลที่ระบบตัดสินใจส่งให้กับระบบติดต่อกับผู้ใช้ จะมีดังตารางที่ 3.5

ตารางที่ 3.5 ข้อมูลที่ส่งจากระบบตัดสินใจไปสู่ระบบติดต่อกับผู้ใช้

คำสั่ง	รายละเอียด
Map[line]	ส่งแผนที่ให้กับระบบติดต่อผู้ใช้ โดย line หมายถึงจำนวนบรรทัดของแผนที่ที่จะส่งมาให้ ไม่รวมกับบรรทัดคำสั่งนี้ ข้อมูลหลังจากคำสั่งนี้จำนวนบรรทัดเท่ากับ line จะเป็นข้อมูลแผนที่ที่เซอร์เวอร์ส่งมาให้
Path[path]	ส่งทางที่รถจะเคลื่อนที่
Pos[EdgePoint]	ส่งตำแหน่งปัจจุบันของรถ
Gps[Quality][UTM]	ส่งตำแหน่งที่อ่านได้จากจีพีเอส
P2d[Quality][UTM]	ส่งตำแหน่งของรถที่คำนวณโดยอาศัยเข็มทิศและส่วนวัดระยะทาง

โดยข้อมูลที่อยู่ใน [] จะหมายถึงสิ่งต่างๆดังตารางที่ 3.6

ตารางที่ 3.6 ประเภทข้อมูลที่รับส่งระหว่างระบบตัดสินใจกับระบบติดต่อกับผู้ใช้

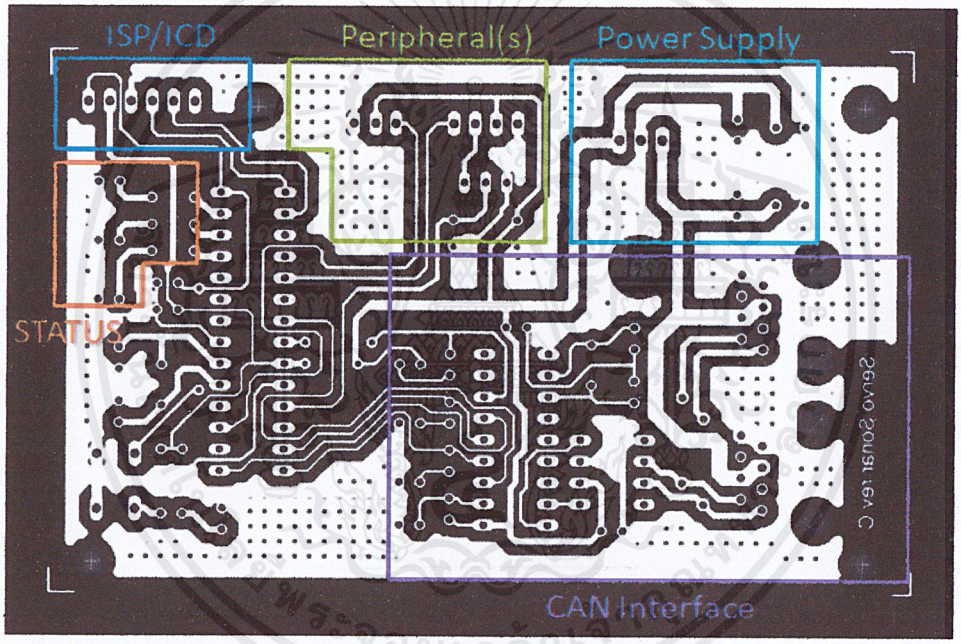
ประเภท	รายละเอียด
Line	เป็นข้อความที่แทนตัวเลข เช่น "124"
Path	เป็นข้อมูลที่แทนเส้นทาง 1 เส้นทาง ภายในจะประกอบด้วยเปอร์เซนต์ของขอบเริ่มต้น เปอร์เซนต์ของขอบปลายทาง และกลุ่มของจุดทั้งหมดมาประกอบเป็นเส้นทาง ตัวอย่างเช่น "12.3 83.4 1 2 12 3 8 9" จะหมายถึงเส้นทางที่เชื่อมต่อโดยจุดหมายเลข 1 2 12 3 8 9 ตามลำดับ โดยมีจุดเริ่มต้นอยู่ที่ 12.3 เปอเซนต์ของระยะทางของขอบที่เชื่อมระหว่างจุด 1 ไปยังจุดที่ 2 และมีปลายทางอยู่ที่ 83.4 เปอเซนต์ของระยะทางของขอบที่เชื่อมระหว่างจุดที่ 8 ไปยังจุดที่ 9
EdgePoint	เป็นข้อมูลที่แทนจุดบนขอบจะประกอบไปด้วยจุดเริ่มต้น, จุดปลายทางและเปอร์เซนต์ ตัวอย่างเช่น "1 2 12.3" จะหมายถึงจุดที่อยู่ห่างจากจุดที่ 1 ไปยังจุดที่ 2 เป็นระยะทาง 12.3 เปอเซนต์ของระยะทางจากจุดที่ 1 ไปยังจุดที่ 2
UTM	เป็นข้อมูลที่แทนพิกัดแบบยูทีเอ็ม จะประกอบไปด้วยค่าทางเหนือ (northing) และทางตะวันออก (easting) ตัวอย่างเช่น "1518427.418 692135.327" หมายถึงพิกัดยูทีเอ็มที่มีค่าทางเหนือ (northing) 1518427.418 และค่าทางตะวันออก (easting) 692135.327
Quality	เป็นค่าที่บ่งบองถึงคุณภาพของข้อมูล โดย 0 หมายถึงไม่ถูกต้อง และ 1 หมายถึงถูกต้อง

บทที่ 4
การพัฒนา

4.1 การพัฒนาระบบฮาร์ดแวร์

4.1.1 แผ่นวงจรแม่แบบ

เนื่องจากมอดูลฮาร์ดแวร์ต่างๆมีส่วนที่คล้ายคลึงกันอยู่มาก จะแตกต่างออกไปเพียงส่วนที่เชื่อมต่อเข้ากันกับอุปกรณ์ต่อพ่วงอื่นเท่านั้น การออกแบบแผ่นวงจรจึงได้ออกแบบแบ่งเป็นภาคที่คงที่ไว้เพื่อให้การพัฒนามอดูลอื่นทำได้รวดเร็วขึ้น



รูปที่ 4.1 แผ่นวงจรแม่แบบ

ส่วนที่แตกต่างกันในมอดูลต่างๆจะมีเพียงแค่ส่วนสีเขียวที่ติดต่อกับอุปกรณ์ต่อพ่วงอื่นเท่านั้น เช่น ต่อเข้ากับเข็มทิศดิจิทัล จีพีเอส เป็นต้น ยกเว้นในมอดูลหยุดรถฉุกเฉินที่จะมีส่วนของสีฟ้าอ่อนคือแหล่งจ่ายไฟที่เปลี่ยนไปเป็นแบบสวิตช์ชิ่งเนื่องจากจะใช้แรงดัน 24 โวลต์จากแบตเตอรี่โดยตรง (แต่ก็จะมีราคาสูงขึ้นตามมาด้วย)

4.1.2 การตั้งค่าบัสเอสพีไอเพื่อเชื่อมต่อเอ็มซีพี 2515

ในไมโครคอนโทรลเลอร์จะมีโมดูลเอ็มเอสเอสพี (MSSP, Master Synchronous Serial Port) ในการจัดการการเชื่อมต่อบัสแบบเอสพีไอ ซึ่งในที่นี้คอมไพเลอร์ที่ใช้ คือ ซีซีเอส ซี (CCS C)

ซึ่งมีตัวช่วยในการตั้งค่าการใช้งานมอดูล ซึ่งในที่นี้จะใช้ติดต่อกับแคนคอนโทรลเลอร์ โดยจะมีการกำหนดค่าดังนี้

```
setup_spi(SPI_MASTER | SPI_H_TO_L | SPI_XMIT_L_TO_H | SPI_CLK_DIV_(4;
```

SPI_MASTER เป็นการให้คอนโทรลเลอร์ทำงานในโหมดมาสเตอร์

SPI_H_TO_L | SPI_XMIT_L_TO_H เป็นการตั้งค่าให้ใช้งานเอสพีไอในโหมด 1,1 กล่าวคือ ปกติขาสัญญาณนาฬิกาจะเป็น '1' และสัญญาณนาฬิกาจะเริ่มด้วยขอบขาลง

SPI_CLK_DIV_4 เพื่อลดความถี่ในการสื่อสาร ซึ่งใช้สัญญาณนาฬิกาหลักหารด้วย 4 (มอดูลส่วนใหญ่จะใช้สัญญาณนาฬิกาหลักที่ความถี่ 20 เมกะเฮิรตซ์) ทำให้สัญญาณนาฬิกาที่ใช้ในการสื่อสารในบัสเอสพีไอมีความถี่ 5 เมกะเฮิรตซ์ ซึ่งเอ็มซีพี2515 สามารถรองรับได้สูงสุดถึง 10 เมกะเฮิรตซ์

ซึ่งก่อนที่จะใช้งานรับส่งข้อมูลแคนได้นั้นต้องทำการกำหนดค่าเริ่มต้นให้กับเอ็มซีพี 2515 ซึ่งขั้นตอนในการกำหนดค่าเริ่มต้นมีดังนี้

รูปที่ 4.2 ขั้นตอนการตั้งค่าการทำงานของ MCP 2515

เริ่มด้วยการส่งคำสั่งรีเซตเพื่อเป็นการให้เอ็มซีพี2515 ลบค่าที่ตั้งไว้เดิมทิ้ง เมื่อสั่งให้รีเซตแล้วต้องรอเป็นช่วงระยะเวลาหนึ่งเพื่อให้เอ็มซีพี2515 เริ่มทำงาน จากนั้นจึงส่งคำสั่งตั้งค่าแคน เช่น

ความกว้างของสัญญาณ จุดชักสัญญาณ ความถี่ที่ต้องการใช้งานเป็นต้น แล้วจึงส่งคำสั่งกำหนดเหตุการณ์สนใจที่จะทำให้เกิดสัญญาณขัดจังหวะ ซึ่งในการใช้งานในที่นี้จะมีเหตุการณ์ที่สนใจดังนี้ ข้อมูลที่ผ่านตัวกรอง เกิดข้อผิดพลาดในการส่งข้อมูล และการส่งข้อมูลสำเร็จ จากนั้นจึงทำการตั้งค่าขาอินพุต/เอาต์พุต เนื่องจากเอ็มซีพี2515 สามารถกำหนดบางขาให้เป็นอินพุต/เอาต์พุตได้ แล้วจึงทำการตั้งค่าตัวกรองข้อมูลซึ่งประกอบด้วยส่วนที่เป็นหน้ากาก และส่วนที่เป็นตัวกรอง ซึ่งการกรองจะสนใจเฉพาะบิตที่ตรงกับหน้ากากที่เป็น '1' เท่านั้น ขั้นสุดท้ายจึงร้องขอให้เข้าสู่โหมดทำงานปกติ จึงจะเริ่มทำการรับ/ส่งข้อมูลได้

4.1.3 การใช้งานอุปกรณ์ต่อพ่วงแบบไอสแควซีเพื่อติดต่อกับมอดูลเข็มทิศดิจิทัล

การใช้งานบัสไอสแควซีสามารถใช้มอดูลเอ็มเอสเอสพีในไมโครคอนโทรลเลอร์ได้ทันที ซึ่งเป็นมอดูลเดียวกันกับเอสพีไอ ซึ่งมีเพียงมอดูลเดียวเท่านั้น จึงจำเป็นต้องสร้างการเชื่อมต่อขึ้นโดยใช้ซอร์ฟแวร์ ซึ่งมีข้อเสียคือในขณะที่ส่งจะต้องใช้เวลาของไมโครคอนโทรลเลอร์ในการส่งข้อมูล ต่างกับการใช้มอดูลฮาร์ดแวร์ที่แค่ย้ายค่าที่ต้องการใส่ในริจิสเตอร์ที่กำหนดเท่านั้น แต่อย่างไรก็ตามซีซีเอส ซี มีฟังก์ชั่นในการส่งตรงนี้ไว้ให้อยู่แล้ว สิ่งที่ต้องทำคือตั้งค่าการใช้งานให้ถูกต้องเท่านั้น

```
#use i2c(master, scl=PIN_B4, sda=PIN_B5, fast=100000, stream=CMPS)
```

master เป็นการกำหนดค่าให้ไมโครคอนโทรลเลอร์ทำงานเป็นตัวมาสเตอร์

scl=PIN_B4 เป็นการกำหนดขาที่ทำหน้าที่ส่งสัญญาณนาฬิกา (SCL)

sda=PIN_B5 เป็นการกำหนดขาที่ทำหน้าที่ส่งข้อมูลออก หรือรับข้อมูลเข้า (SDA)

fast=100000 ใช้งานบัสตามข้อกำหนดความเร็วสูง ด้วยความถี่ 0.1 เมกะเฮิรตซ์

stream=CMPS เป็นการตั้งชื่อให้กับสตรีมที่สร้างขึ้น

การใช้งานก็สามารถสั่งใช้ i2c_start();, i2c_stop();, i2c_read();, i2c_write();, ได้ทันที

ตัวอย่างการอ่านข้อมูลจากมอดูลเข็มทิศดิจิทัล ซีเอ็มพีเอส-03

```
i2c_start(CMPS);
i2c_write(CMPS, 0xC0);          //ส่งที่อยู่โหนดที่ต้องการ
i2c_write(CMPS, 0x02);          //ส่งริจิสเตอร์ที่ต้องการอ่านข้อมูล

i2c_start(CMPS);                //เริ่มการติดต่อใหม่เพื่อที่จะทำการอ่านข้อมูล
i2c_write(CMPS, 0xC1);          //ส่งคำสั่งอ่านข้อมูล

can_data[1] = i2c_read(CMPS);   //อ่านข้อมูลขึ้นจากบัส
can_data[0] = i2c_read(CMPS,0); //อ่านข้อมูลขึ้นจากบัสโดยไม่รอการตอบกลับ
```

```
i2c_stop(CMPS);                    //สิ้นสุดการส่งข้อมูล
```

4.1.4 การใช้งานอุปกรณ์ต่อพ่วงแบบอาร์เอส-232

การรับส่งข้อมูลผ่านทางอาร์เอส-232 จะใช้มอดูลอียูเอสเออาร์ที (EUSART, Enhanced Universal Synchronous Asynchronous Receiver Transmitter) ซึ่งซีซีเอส ซี ก็ได้มีการพัฒนาส่วนที่ช่วยในการตั้งค่าการใช้งานเช่นเดียวกัน โดยการตั้งค่าที่ใช้มีดังนี้

```
#use rs232(uart1, baud=9600, stream=DEBUG, ERRORS)
```

uart1 เป็นการบอกว่าต้องการตั้งค่าสำหรับฮาร์ดแวร์มอดูลที่ 1

baud=9600 เป็นการกำหนดค่าบอดเรตของการสื่อสาร

stream=DEBUG เป็นการตั้งชื่อสตรีมที่ใช้เชื่อมต่อ

ERRORS เป็นการกำหนดให้เมื่อมีการผิดพลาดให้เก็บข้อผิดพลาดในตัวแปร RS232_ERRORS และทำการรีเซตข้อผิดพลาดนั้น

4.1.5 การใช้งานอุปกรณ์ต่อพ่วงที่ควบคุมด้วยความกว้างพัลส์เพื่อควบคุมเซอร์โวมอเตอร์

การที่ต้องการให้ไมโครคอนโทรลเลอร์สร้างสัญญาณพีดับเบิ้ลยูเอ็มออกมานั้น สามารถทำได้โดยใช้มอดูล ซีซีพี (CCP, Capture/Compare/Pulse width modulate) ซึ่งในพีไอซี16เอฟ883/886 มีให้ใช้งานสองตัว ซึ่งมีโหมดการทำงานที่แตกต่างกันเล็กน้อย มอดูลจะให้สัญญาณนาฬิกาจากสัญญาณนาฬิกาหลักมาผ่านตัวหาร แต่เนื่องจากตัวหารที่มีให้มีค่าน้อยเกินไปทำให้ความถี่ที่ได้ออกมานั้นมากกว่าความถี่ที่เหมาะสมที่จะควบคุมเซอร์โวมอเตอร์ได้ จึงจำเป็นต้องลดสัญญาณนาฬิกาหลักลงจาก 20 เมกะเฮิรตซ์เหลือเพียงแค่ 4 เมกะเฮิรตซ์

การสั่งงานเซอร์โวมอเตอร์ที่เหมาะสมจะขึ้นอยู่กับประเภทของเซอร์โวด้วย ซึ่งแบ่งได้ออกเป็นสองประเภท(ตามข้อมูลของบริษัทฟุตาบะ) คือ ดิจิทัลเซอร์โว และแอนะล็อกเซอร์โว ซึ่งดิจิทัลเซอร์โวหมายถึงภายในจะใช้ไมโครคอนโทรลเลอร์ในการควบคุม แต่ยังคงมีส่วนการทำงานที่เหลือเหมือนเดิม ซึ่งมีข้อดีคือสามารถส่งงานได้ด้วยความถี่ที่สูงขึ้นถึง 6 เท่า (จากทุกๆ 20 มิลลิวินาทีเป็น 3.33 มิลลิวินาที) และเมื่อมีคำสั่งที่ผิดพลาดเช่นสั่งให้หมุนไปในมุมที่ไม่สามารถหมุนไปได้ ดิจิทัลเซอร์โวจะสามารถตรวจจับได้และไม่สั่งงานให้มอเตอร์หมุนเพื่อไม่ให้เกิดความเสียหายต่อตัวมอเตอร์และชุดเฟือง

การตั้งค่ามอดูลสำหรับดิจิทัลเซอร์โวสามารถคำนวณได้จากสูตร

$$\text{Pwm period} = [(PR2)+1]*4*Tosc*(\text{TMR2 Prescale Value}) \qquad (4.1)$$

ซึ่งสามารถตั้งค่าได้ดังนี้

```
setup_timer_)2T2_DIV_BY_16, 208, (1;
setup_ccp)2CCP_PWM);
```

T2_DIV_BY_16 คือการหารความถี่สัญญาณนาฬิกาหลักด้วย 16

208 คือค่าที่ต้องการให้ไทม์เมอร์ทำการรีเซตเมื่อนับถึง

1 เป็นค่าโพสสเกลเลอร์ซึ่งไม่มีผลต่อการใช้งานมอดูลซีซีพี 2

CCP_PWM เป็นการกำหนดให้มอดูลซีซีพี 2 ทำงานในรูปแบบพีดับเบิ้ลยูเอ็ม

ความกว้างพัลส์สามารถกำหนดได้โดยใช้ฟังก์ชั่น set_pwm2_duty(); ซึ่งสามารถรับพารามิเตอร์เป็น int16 เพื่อนำไปใส่ในริจิสเตอร์ CCPRxL:CCPxCON<5:4> เป็นความกว้างพัลส์ซึ่งสามารถคำนวณได้จากสูตร

Pulse width = (CCPRxL:CCPxCON<5:4>) * (Tosc*TMR2 Prescale Value) (4.2)

ซึ่งค่าที่สามารถใส่ได้จะอยู่ระหว่าง 222 ถึง 528 ซึ่งเป็นมุม -60 องศาถึง 60 องศา

4.2 การพัฒนาระบบซอฟต์แวร์

4.2.1 การติดตั้งเอ็มเบ็ดเด็ดบอร์ด (Embedded Board)

เอ็มเบ็ดเด็ดบอร์ดที่เราได้นำมาใช้คือ คอริบลิอีโวลูชั่นแคเรียบอร์ด (Colibri Evaluation Carrier Board) ซึ่งภายในประกอบไปด้วยอินเตอร์เฟสที่จำเป็นต่อโปรเจค ดังนี้

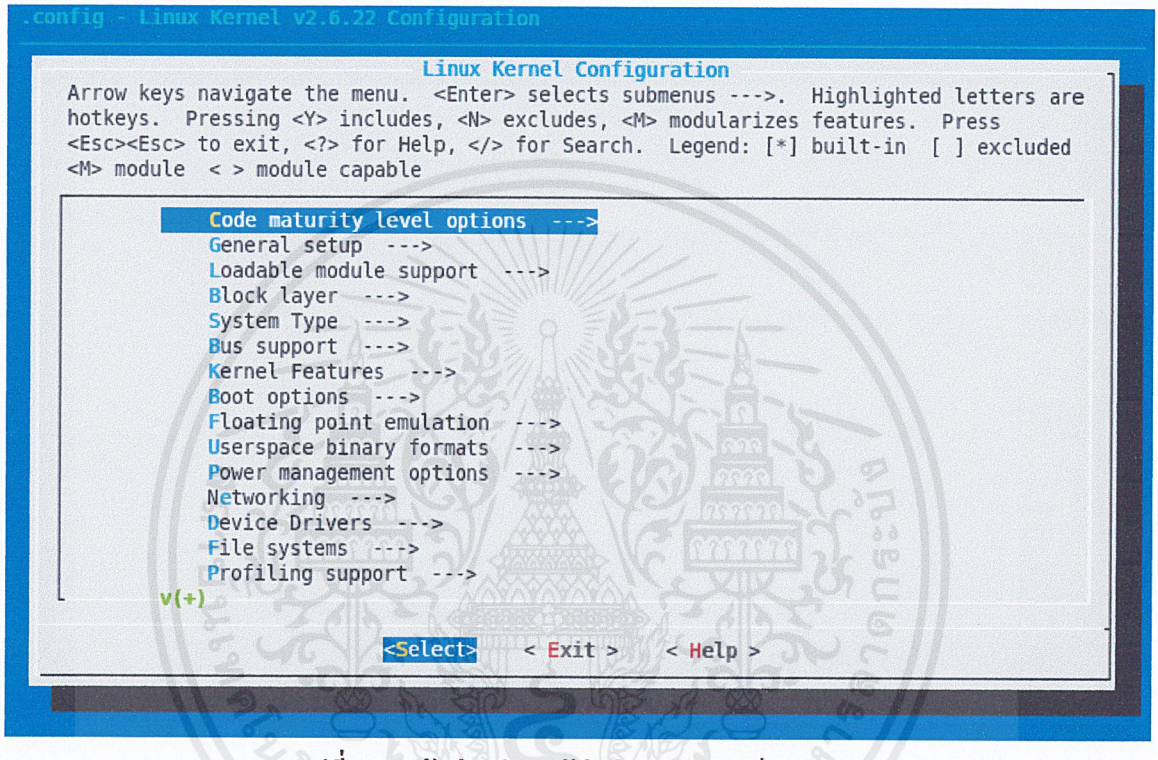
- 10/100MBit อีเทอเน็ต (Ethernet) สำหรับการเชื่อมต่อบอร์ดนี้เข้ากับระบบเครือข่ายอีเทอเน็ตบนรถ
- แคน (CAN) ฟิลิปส์ เอสเจเอ 1000 (Philips SJA1000) สำหรับการเชื่อมต่อบอร์ดนี้เข้ากับแคนบัส (CAN Bus) บนรถ
- เอสดีการ์ด (SDCard) สำหรับเชื่อมต่อ (mount) เข้ากับแฟ้มข้อมูล (directory) /var บนระบบปฏิบัติการลินุกส์ (Linux)

โดยเราได้เลือกลินุกส์เวอร์ชัน 2.6.22 เป็นระบบปฏิบัติการ ซึ่งสามารถดาวน์โหลดได้จาก http://www.emcraft.com/technology.php

ขั้นตอนการติดตั้งเอ็มเบ็ดเด็ดบอร์ดมีดังนี้

4.2.1.1 แก้ไขและตั้งค่าเคอร์เนล (Configure Kenel)

เราสามารถแก้ไขและตั้งค่าเคอร์เนลโดยเรียกใช้คำสั่ง "make menuconfig" เพื่อแก้ไขค่าต่างๆ เช่นแก้ไขให้รองรับระบบไฟล์รูท (root file system) แบบเอ็นเอ็ฟเอส (NFS) ได้เพิ่มไดร์เวอร์ (driver) ต่างๆที่จำเป็นเข้าไป ตั้งค่าให้เคอร์เนลสามารถร้องขอไอพีเอสเดรส (ip address) ผ่านทาง ดีเอชซีพีโปรโตคอล (dhcp protocol) ขณะเริ่มทำงานได้



รูปที่ 4.3 หน้าต่างปรับแก้ค่าต่างๆของเคอร์เนล

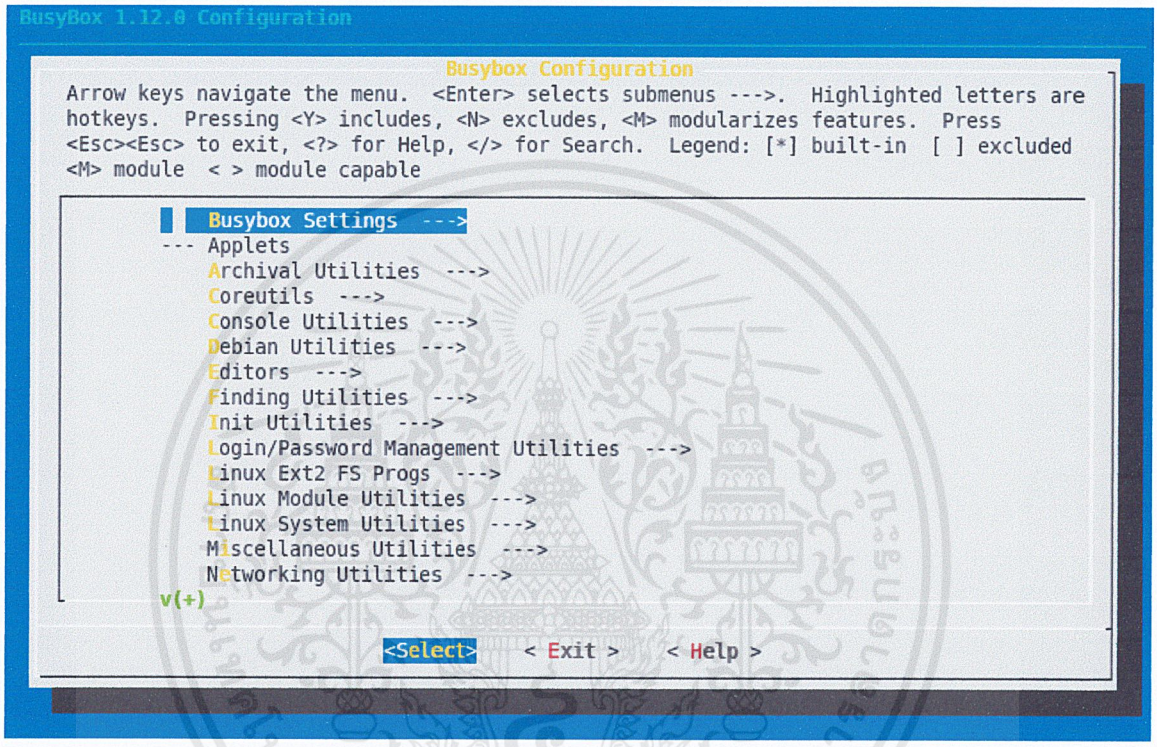
4.2.1.2 การสร้างเคอร์เนล (Build kernel)

หลังจากแก้ไขและตั้งค่าเคอร์เนลแล้วก็จะสามารถสร้างเคอร์เนล (build kernel) โดยใช้คำสั่ง “make” ผลลัพธ์สุดท้ายก็จะได้ลินุกส์เคอร์เนลอิมเมจ (Linux kernel image) ที่สามารถนำไปใช้งานต่อได้

4.2.1.3 จัดเตรียมยูทิลิตี้ (utility) ต่างๆ ที่จำเป็นลงบนระบบไฟล์รูท (root file system)

บีซี่บ๊อค (BusyBox) เป็นโปรแกรมที่ได้รวบรวมคำสั่งต่างๆ ของจีเอ็นยู (GNU) มาไว้ในไฟล์ที่สามารถทำงานได้ (executable file) เดียวกัน ซึ่งบีซี่บ๊อค (BusyBox) จะใช้แทนที่ยูทิลิตี้ต่างๆของจีเอ็นยู (GNU) บีซี่บ๊อค (BusyBox) ถูกเขียนโดยคำนึงถึงขนาดและทรัพยากรที่ใช้ บีซี่บ๊อค (BusyBox) จึงเหมาะสมที่จะนำมาใช้ในเอ็มเบ็ดเด็ดซิสเต็ม (embedded system)

คำสั่งต่างๆที่อยู่ในบีซี่บ๊อค (BusyBox) นั้นสามารถถูกเลือกและปรับแต่งได้ระหว่างการปรับแก้บีซี่บ๊อค (configure BusyBox) โดยใช้คำสั่ง “make menuconfig” หลังจากนั้นก็ทำการสร้างและติดตั้ง โดยใช้คำสั่ง “make” และ “make CONFIG_PREFIX=/work/rootfs install” ตามลำดับ สำหรับ “/work/rootfs” นั้นคือที่อยู่ (path) ของระบบไฟล์รูท (root file system) ที่เราได้จัดเตรียมไว้



รูปที่ 4.4 หน้าต่างปรับแก้ค่าต่างๆของบีซี่บ๊อค (BusyBox)

4.2.2 จัดเตรียมไลบรารี่ (library) ต่างๆที่จำเป็น

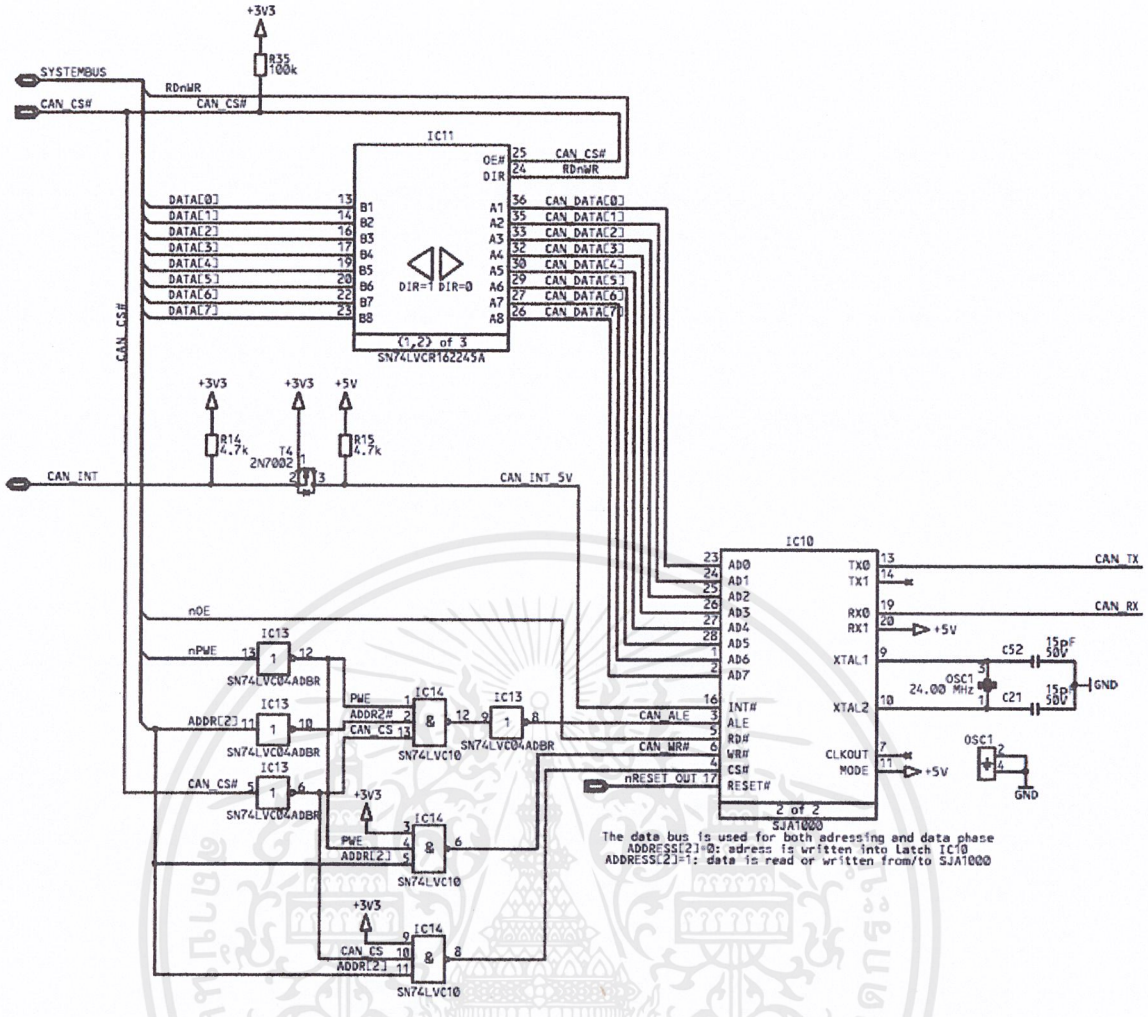
ไลบรารี่ที่จำเป็นได้แก่

- ลิบแอลทีดีแอล (libltdl) http://www.gnu.org/software/libtool/ เพื่อ ให้เพลเยอร์รองรับปลั๊กอินไดว์เวอร์ (plugin driver) ขั้นตอนของการจัดเตรียมได้แก่การดาวน์โหลดซอสโค้ด (download source code), ปรับแต่งและแก้ไข (configure), สร้าง (build) และติดตั้ง (install) โดยจะต้องปรับแต่งและแก้ไขเพื่อให้สร้างแบบคลอสคอมไพล์ (cross compile) สำหรับทำงานบนสถาปัตยกรรมแบบอาร์ม (arm architecture)

- บูส (Boost) http://www.boost.org/ เพื่อใช้ในระบบตัดสินใจ สำหรับส่วนของทีซีพีเซอร์เวอร์ (Tcp Server), ทีซีพีไคลแอนต์ (Tcp Client), ซิกแนล (Signal) และ เธรด (Thread)
- ล็อคโฟซีพลัสพลัส (log4c++) http://sourceforge.net/projects/log4cpp/ เพื่อใช้เป็นระบบล็อค (log) ของระบบตัดสินใจ

4.2.3 การพัฒนาแคนไดว์เวอร์ (CAN Driver) ฟิลิปส์ เอสเจเอ 1000 (Philips SJA1000)

สำหรับในส่วนนี้เราได้นำไดว์เวอร์ (driver) แคนโฟลินุกส์ (can4linux) ของ http://www.port.de มาปรับปรุง ซึ่งโดยปกติแล้วแคนโฟลินุกส์จะรองรับฟิลิปส์ เอสเจเอ 1000 ที่ต่ออยู่บนอุปกรณ์อื่นๆ เช่น AIM104-CAN, VCMA9 แต่ไม่รองรับ ฟิลิปส์ เอสเจเอ 1000 ที่ต่ออยู่บนคอริบลิอีโวลูชั่นแคเรียบอร์ด (Colibri Evaluation Carrier Board) ซึ่งคอริบลิอีโวลูชั่นแคเรียบอร์ดมีการต่อกับเอสเจเอ 1000 ดังรูปที่ 4.5



รูปที่ 4.5 การเชื่อมต่อของฟิลิปส์ เอสเจเอ 1000 บนคอริบลิอีโวลูชั่นแคเรียบอร์ด

โดยที่ CAN_CS# ต่ออยู่กับ GPIO15/nCS1 ส่วน CAN_INT ต่ออยู่กับ GPIO52 ซึ่งการต่อเอสเจเอ 1000 ลักษณะดังกล่าวสามารถอ่านเขียนรีจิสเตอร์ (register) ของเอสเจเอ 1000 ผ่านทางวาริเอเบิล ลาเต็นซี่ไอโอ (Variable-Latency I/O) อินเตอร์เฟสของพีเอ็กเอ 27x (PXA27x) ได้ โดยการเพิ่มโค้ดด้านล่าง

```
static struct resource sja1000_resources[] = {
   [0] = {
        .start     = 0x04000000,
        .end     = 0x04000000 + 0x100,
        .flags     = IORESOURCE_MEM,
   },
   [1] = {
        .start     = IRQ_GPIO(52),
        .end     = IRQ_GPIO(52),
        .flags     = IORESOURCE_IRQ,
   }
};
static struct platform_device sja1000_device = {
```

```
        .name           = "colibrican",
        .id         = 0,
        .num_resources      = ARRAY_SIZE(sja1000_resources),
        .resource       = sja1000_resources,
    };
```

เข้าไปในไฟล์ /arch/arm/mach-pxa/colibri.c และทำการตั้งค่ารีจิสเตอร์ (register) ที่เกี่ยวข้องเพิ่มโดยโค้ดด้านล่าง

```
    pxa_gpio_mode(GPIO18_RDY_MD);
    pxa_gpio_mode(GPIO15_nCS_1_MD);
    pxa_gpio_mode(GPIO49_nPWE_MD);
    MSC0 = 0x7FF47FF0;
```

ในไฟล์ /arch/arm/mach-pxa/colibri.c สำหรับในส่วนการเริ่มต้น (init) ของไดร์เวอร์นั้นจะใช้ฟังก์ชันด้านล่าง

```
int CAN_VendorInit (int minor)
{
   DBGin("CAN_VendorInit");
   can_range[minor] = 0x100;

   /* Request the controllers address space */
   if(NULL == request_mem_region(Base[minor], can_range[minor], "CAN-
IO")) {
   DBGprint(DBG_DATA,("Request CAN-IO failed at 0x%x\n",
Base[minor]));
   return -EBUSY;
   }
   can_base[minor] = ioremap_nocache(Base[minor], can_range[minor]);
   /* now the virtual address can be used for the register address
macros */
   if( IRQ[minor] > 0 || IRQ[minor] > MAX_IRQNUMBER ){
       int err;
       set_irq_type(IRQ[minor], IRQT_FALLING);
   err = request_irq( IRQ[minor], CAN_Interrupt, 0 , "Can",
&Can_minors[minor]);
       if( !err ){
       DBGprint(DBG_BRANCH,("Requested IRQ: %d @ 0x%lx",
                   IRQ[minor], (unsigned long)CAN_Interrupt));
       IRQ_requested[minor] = 1;
   } else {
       release_mem_region(Base[minor], can_range[minor]);
       DBGout(); return -EBUSY;
   }
   } else {
   /* Invalid IRQ number in /proc/.../IRQ */
   release_mem_region(Base[minor], can_range[minor]);
   DBGout(); return -EBUSY;
   }
   DBGout(); return 0;
}
```

ฟังก์ชันสำหรับการเขียนและอ่านรีจิสเตอร์ของ เอสเจเอ 1000 จะใช้ฟังก์ชันด้านล่าง

```
void _ColibriCANout(void __iomem *bd, u8 adr, u8 v) {
   iowrite8(adr, bd);
   iowrite8(v, (bd + 4));
}
u8 _ColibriCANin(void __iomem *bd, u8 adr) {
   u8 v;
   iowrite8(adr, bd);
   v = ioread8((bd + 4));
   return v;
}
```

หลังจากเพิ่มโค้ดเข้าไปในเคอร์เนลและไดว์เวอร์ของ เอสเจเอ 1000 แล้วก็ทำการคอมไพล์เคอร์เนล (compile kernel) และไดว์เวอร์ใหม่

4.2.4 การตั้งค่า (Configure) และสร้างเพลเยอร์ (Build Player)

การตั้งค่าเพลเยอร์นั้นจะใช้คำสั่งต่อไปนี้

./configure --prefix=/work/rootfs --build=x86-linux --host=arm-linux --disable-alldrivers --disable-shared --enable-dummy

โดยตัวเลือกต่างๆมีความหมายดังนี้

--prefix=/work/rootfs เป็นการกำหนดตำแหน่งที่จะติดตั้งเพลเยอร์

--build=x86-linux ระบุสถาปัตยกรรมของเครื่องที่ใช้สร้างเพลเยอร์

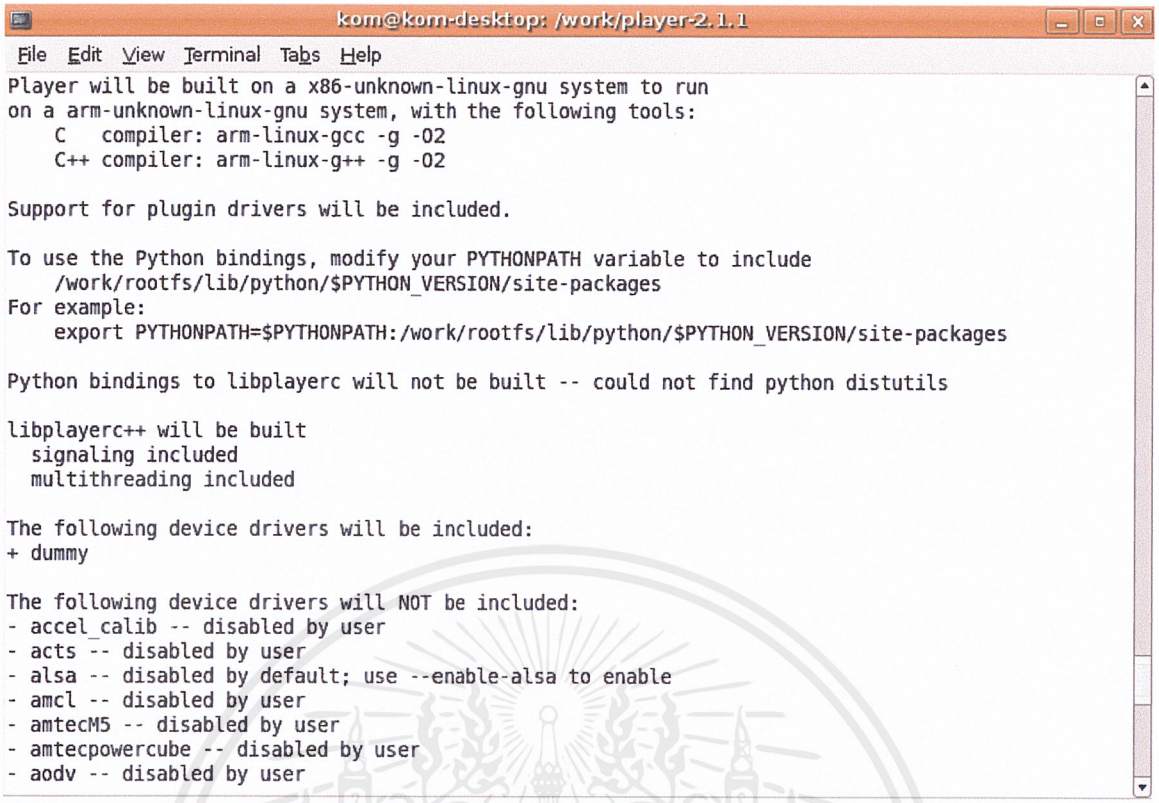
--host=arm-linux ระบุสถาปัตยกรรมของเครื่องที่ต้องการให้เพลเยอร์ทำงาน

--disable-alldrivers เป็นการระบุให้ไม่ต้องสร้างไดว์เวอร์ที่มากับเพลเยอร์ทั้งหมด

--disable-shared เพื่อไม่ให้ใช้งานแชร์ไลบรารี่ (shared library) หากเป็นไปได้ เพื่อจะได้ไม่ต้องคัดลอกไลบรารี่หลายๆตัว ไปพร้อมกับไฟล์ที่ทำงานได้ (executable file) ของเพลเยอร์

--enable-dummy เป็นการระบุให้สร้างไดว์เวอร์ dummy ของเพลเยอร์สำหรับทดสอบการทำงานเบื้องต้น

หลังจากที่ตั้งค่าแล้วก็จะทำการสร้างและติดตั้งโดยใช้คำสั่ง “make” และ “make install” ตามลำดับ



รูปที่ 4.6 ผลลัพธ์ของการตั้งค่าเพลเยอร์

4.2.5 การพัฒนาไดว์เวอร์ของเพลเยอร์

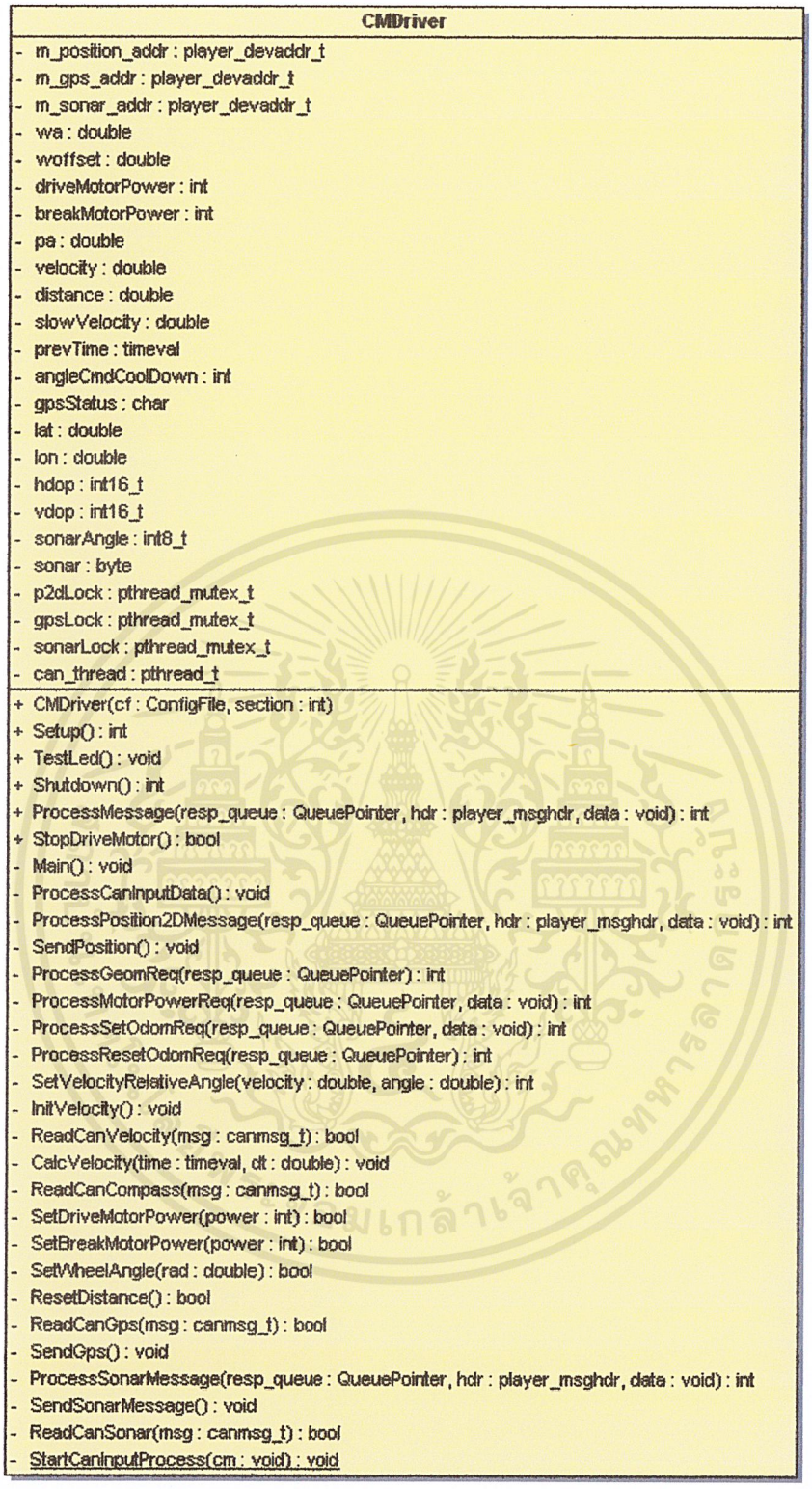
ภายในไดว์เวอร์จะแบ่งเป็นส่วนต่างๆดังนี้

- ส่วนหลัก (core) ซึ่งส่วนนี้จะเป็นส่วนที่จะเริ่มต้นการทำงาน โดยจะอ่านค่าที่เซตต่างๆ เช่น can port, baud rate, wheel offset มาจากไฟล์ตั้งค่า (config file) จากนั้นจะกำหนดค่าเริ่มต้นให้กับตัวแปรต่างๆ เปิดการใช้งานแคนบัส (CAN Bus) สร้างเธรด (thread) สำหรับรับข้อมูลจากแคนบัส และสร้างเธรดหลักของไดว์เวอร์ โดยเธรดหลักของไดว์เวอร์จะทำการอ่านและส่งต่อคำสั่งที่รับมาผ่านทางเพลเยอร์ให้กับส่วนอื่นๆที่รับผิดชอบต่อคำสั่งนั้นๆ และทำการส่งข้อมูลตำแหน่งและความเร็วของรถออกไปให้ภายนอกผ่านทางเพลเยอร์ส่วนเธรดที่รับข้อมูลจากแคนบัสนั้นจะรับข้อมูลเข้ามาตรวจสอบว่าเป็นเมสเซสประเภทใด และส่งต่อให้กับส่วนที่รับผิดชอบต่อเมสเซสประเภทนั้นๆ
- ส่วนควบคุมรถ ส่วนนี้จะทำการรับเมสเซสผ่านทางเพลเยอร์แล้วตีความหมายเป็นคำสั่งควบคุมรถแล้วส่งต่อไปยังฮาร์ดแวร์ผ่านทางแคนบัส (CAN Bus) อีกทีหนึ่ง ประเภทคำสั่งควบคุมรถที่อินเตอร์เฟส (interface) นี้รองรับคือคำสั่งควบคุมรถแบบ

PLAYER_POSITION2D_CMD_CAR ซึ่งจะระบุความเร็วและองศาของล้อหน้า นอกจากจะรับคำสั่งจากส่วนอื่นแล้ว ส่วนนี้จะมีฟังก์ชัน สำหรับส่งข้อมูลตำแหน่งและทิศทางรถในระนาบ 2 มิติและเวกเตอร์ (vector) ของการเคลื่อนที่ของรถในระนาบ 2 มิติออกไปทุกๆ 50 ms เพื่อให้ส่วนหลัก (core) เรียกใช้งาน

- ส่วนรับข้อมูลจากมอดูลวัดระยะทาง ส่วนนี้จะนำค่าที่มอดูลวัดระยะทางส่งมา มาคำนวณความเร็วและปรับตำแหน่งของรถ โดยจะทำงานร่วมกับข้อมูลที่ได้จากเข็มทิศด้วย
- ส่วนรับข้อมูลจากเข็มทิศส่วนนี้จะนำค่าที่ได้จากเข็มทิศมาแปลงให้เป็นองศาที่อ้างอิงจากทิศตะวันออก และปรับแก้ค่าที่ได้ตามการปรับแต่ง (calibrate)
- ส่วนรับข้อมูลจากโซนาร์ส่วนนี้จะรับข้อมูลจากโซนาร์และส่งออกไปให้ส่วนอื่นผ่านทางเพลเยอร์ทุกๆครั้งที่รับข้อมูลเข้ามา โดยจะต้องแปลงจากหน่วยนิ้วเป็นเมตรตามอินเตอร์เฟสของเพลเยอร์
- ส่วนรับข้อมูลจากจีพีเอส ส่วนนี้จะอ่านค่า ละติจูด, ลองจิจูด, สถานะ, hdop, vdop ตามลำดับ แล้วส่งออกไปยังส่วนอื่นผ่านทางเพลเยอร์

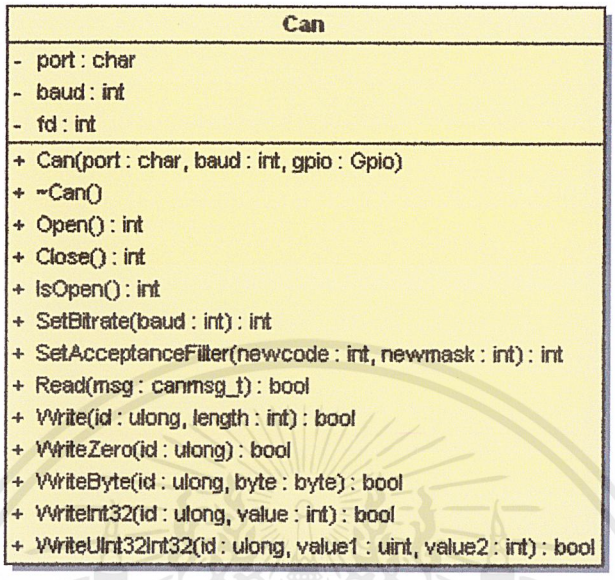
ส่วนด้านบนทั้งหมดจะรวมอยู่ในคลาส CMDriver ตามคลาสไดอะแกรมรูปที่ 4.7



รูปที่ 4.7 คลาสไดอะแกรมของคลาส CMDriver

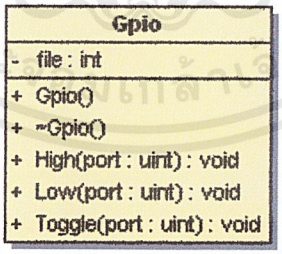
- ส่วนติดต่อกับแคนบัส (CAN Bus) ซึ่งสามารถกำหนดบอร์ดเรท (baud rate) และตัวกรอง (acceptance filter) และสามารถรับอินสแตน (instance) ของคลาส Gpio เพื่อให้

กลับความสว่าง (toggle) หลอดแอลอีดี (LED) สำหรับแสดงการส่งข้อมูลทางแคนบัส ได้ ส่วนนี้มีคลาสไดอะแกรมดังรูปที่ 4.8



รูปที่ 4.8 คลาสไดอะแกรมของคลาส Can

- ส่วนติดต่อกับจีพีไอโอ (Gpio) ส่วนนี้ใช้สำหรับ ติด ดับ และกลับค่า (toggle) แอลอีดี (LED) สำหรับการดูการทำงานของรถแบบคร่าวๆได้ ส่วนนี้มีคลาสไดอะแกรม ดังรูปที่ 4.9



รูปที่ 4.9 คลาสไดอะแกรมของคลาส Gpio

4.2.6 การพัฒนาระบบตัดสินใจ

ระบบตัดสินใจจะมีส่วนหลักๆดังนี้

- **ส่วนแผนที่**

  คลาสนี้สามารถอ่านแผนที่ขึ้นมาจากไฟล์ซึ่งจะมีรูปแบบตามตัวอย่างด้านล่าง

```
#vertex
v
1     13.72863992085863 100.77783465385437
2     13.728629498538623      100.77749133110046
3     13.728598231575793      100.77601075172424
4     13.728535697637628      100.77336072921753
5     13.728504430662293      100.77182650566101
#bidirection edge
b
1     2
2     39
2     7
3     6
```

โดยบรรทัดที่ขึ้นต้นด้วย # จะถือว่าเป็นข้อคิดเห็น (comment)

บรรทัดที่มีตัวอักษร v ตัวเดียวหมายความว่าบรรทัดต่อจากนี้ไปจะหมายถึงจุดของกราฟ

บรรทัดที่มีตัวอักษร b ตัวเดียวหมายความว่าบรรทัดต่อจากนี้ไปจะหมายถึงขอบแบบ 2 ทิศทางของกราฟ

บรรทัดที่มีตัวอักษร u ตัวเดียวหมายความว่าบรรทัดต่อจากนี้ไปจะหมายถึงขอบแบบทิศทางเดียวของกราฟ

สำหรับบรรทัดประเภทจุด (vertex) นั้นจะมีลักษณะดังนี้ “index latitude longitude” ตัวอย่างเช่น

```
3     13.728598231575793      100.77601075172424
```

จะหมายถึงจุด (vertex) หมายเลข 3 อยู่บนตำแหน่งละติจูด 13.728598231575793 ลองจิจูด 100.77601075172424

สำหรับบรรทัดประเภทขอบนั้นจะมีลักษณะดังนี้ “SourceIndex TargetIndex” ตัวอย่างเช่น

```
2     39
```

จะหมายถึงมีขอบ (edge) เชื่อมระหว่างจุด (vertex) หมายเลข 2 กับจุด (vertex) หมายเลข 39

หลังจากที่อ่านแผนที่เสร็จแล้วก็จะทำการคำนวณหาพิกัดแบบยูทีเอ็มของทุกจุด (vertex) และคำนวณระยะทางของแต่ละขอบ (edge) เพื่อให้สะดวกต่อการคำนวณ สำหรับการหาเส้นทางที่สั้นที่สุดไปยังปลายทางนั้นจะใช้การค้นหาแบบ A* ในการค้นหาโดยได้มีการดัดแปลง

จากการค้นหาแบบ A* ปกติเพื่อให้สามารถค้นหาเส้นทางที่สั้นที่สุดจากจุดบนขอบไปยังจุดบนขอบได้ โดยมีการดัดแปลงดังนี้

ก่อนเริ่มต้นทำการค้นหาจะนำจุด (vertex) ที่ปลายทั้งสองด้านของขอบ (edge) เริ่มต้นเข้าไปในคิวที่มีลำดับความสำคัญ (priority queue) โดยแต่ละจุด (vertex) จะมีค่าใช้จ่าย (cost) เริ่มต้นเป็นระยะห่างจากจุด (vertex) นั้นๆไปยังจุดเริ่มต้นบนขอบ (edge)

กำหนดให้

e เป็นขอบ (edge) ที่มีจุดปลายทางอยู่บนขอบ

v1 เป็นต้นทาง (source) ของขอบ (edge) e

v2 เป็นปลายทาง (target) ของขอบ (edge) e

w1 เป็นระยะห่างระหว่างจุด v1 ไปยังจุดปลายทางบนขอบ

w2 เป็นระยะห่างระหว่างจุด v2 ไปยังจุดปลายทางบนขอบ

ค่าใช้จ่ายแบบฮิวริสติก (heuristic cost) จากจุด v ใดๆไปยังปลายทางจะเท่ากับ

min(distance(v, v1) + w1, distance(v, v2) + w2)

โดยที่ distance(x, y) หมายถึงระยะห่างจากจุด x ไปยังจุด y

การค้นหาจะสิ้นสุดเมื่อจุด v1 หรือ v2 ถูกนำออกจากคิวที่มีลำดับความสำคัญ (priority queue) โดยค่าใช้จ่ายที่น้อยที่สุด (optimum cost) จะเท่ากับระยะทางที่ใช้เดินทางจากจุดเริ่มต้นมายังจุดนี้รวมกับระยะทางจากจุดนี้ไปยังจุดปลายทางจริงๆ (w1 ถ้าหากว่าจุดนั้นคือจุด v1 หรือ w2 ถ้าหากว่าจุดนั้นคือจุด v2)

- **ส่วน image processing client**

ส่วนนี้เราได้นำ Boost.Asio ซึ่งเป็นซีพลัสพลัสไลบรารี่ (c++ library) สำหรับการเขียนโปรแกรมเกี่ยวกับระบบเครือข่ายและการจัดการอินพุต เอาต์พุต (input output) ในระดับล่าง มาใช้งาน

- **ส่วนระบุตำแหน่งรถบนแผนที่**

ส่วนนี้จะรับข้อมูลตำแหน่งรถเริ่มต้นจากจีพีเอส หรือไม่ก็จากจุดที่ผู้ใช้กำหนดด้วยตนเอง ถ้าหากว่าเป็นการอ้างอิงจุดเริ่มต้นจากจีพีเอสนั้น ค่า HDOP และ VDOP ของตำแหน่งจีพีเอส ต้องน้อยกว่าหรือเท่ากับ 16 ทั้งคู่ หลังจากที่มีตำแหน่งเริ่มต้นแล้วก็จะทำการติดตามตำแหน่งรถตามค่าที่อ่านได้จากอินเตอร์เฟส position2d ของเพลเยอร์

นอกจากการติดตามตำแหน่งรถโดยใช้ค่าที่ได้จากอินเตอร์เฟส position2d ของเพลเยอร์แล้ว ส่วนนี้ยังรองรับการปรับแก้ค่าตำแหน่งรถโดยส่วนอื่นๆของระบบตัดสินใจ เช่นปรับแก้ตำแหน่งรถโดยการทำงานของพฤติกรรมต่างๆ

- **ส่วนวางแผนเส้นทางเดินรถ**

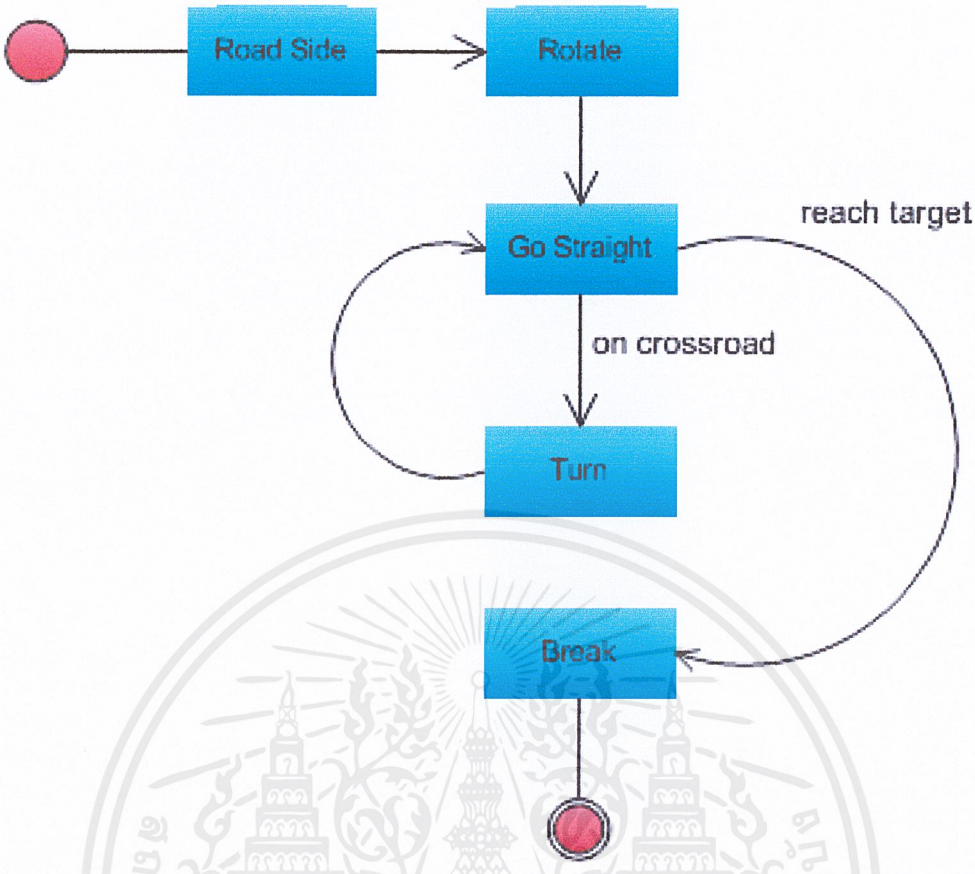
ส่วนนี้จะรับปลายทางจากผู้ใช้งาน และทำการค้นหาเส้นทางโดยเรียกใช้ฟังก์ชันในการหาเส้นทางของคลาส Map จากนั้นจะส่งต่อเส้นทางที่หาได้ให้ส่วนของไดว์เวอร์ทำงานต่อ

- **ส่วน Driver**

คลาสนี้จะรับเส้นทางการขับเคลื่อนเข้ามาและทำการสร้างพฤติกรรมต่างๆ มาทำงานเป็นลำดับขั้นตอนเพื่อให้รถวิ่งไปตามทางที่รับเข้ามาในแต่ละรอบที่คลาสนี้ทำงานจากการตรวจสอบสิ่งกีดขวางก่อน ถ้าหากตรวจพบสิ่งกีดขวางอยู่ในระยะใกล้เกินกว่าที่จะหลบได้ ก็จะสั่งการให้พฤติกรรมการเบรคทำการเบรคแบบฉุกเฉินทันที

หลังจากนั้นก็จะทำการตรวจสอบว่าผู้ใช้ได้สั่งหยุดรถชั่วคราวหรือไม่ ถ้าหากใช่ก็จะให้พฤติกรรมการหยุดรถทำงานแทน แต่ถ้าไม่ใช่ก็จะให้พฤติกรรมอื่นๆทำงาน โดยการขับเคลื่อนรถไปยังปลายทางจะมีขั้นตอนดังนี้

- นำรถชิดขอบทางด้านซ้าย โดยพฤติกรรมการนำรถชิดขอบทาง
- กลับรถโดยพฤติกรรมการกลับรถ ถ้าหากว่ารถอยู่ในองศาที่ถูกต้องอยู่แล้ว พฤติกรรมการกลับรถจะเสร็จสิ้นทันที
- ขับเคลื่อนรถไปตามถนนด้านหน้าจนถึงแยกถัดไป โดยพฤติกรรมการขับเคลื่อนไปตามถนน
- เลี้ยวรถไปยังปลายทางถัดไปถ้าหากว่ายังไม่ถึงจุดหมายปลายทาง แต่ถ้าหากว่าถึงจุดหมายปลายทางแล้วก็จะหยุดรถโดยพฤติกรรมการหยุดรถ
- ถ้าหากว่ายังไม่ถึงจุดหมายปลายทาง หลังจากเลี้ยวรถเสร็จสิ้นก็จะกลับไปสู่การขับเคลื่อนรถไปตามถนนต่อไป



รูปที่ 4.10 ขั้นตอนการทำงานของคลาส Driver

- **พฤติกรรมการนำรถชิดขอบทาง**

ขั้นตอนแรกจะคำนวณหาองศาของถนนก่อน จากนั้นในแต่ละรอบจะมีขั้นตอนการทำงานดังนี้ ถ้าหากว่าข้อมูลจากการประมวลผลภาพไม่ผิดพลาดก็จะถือว่าสิ้นสุดพฤติกรรมนี้ ถ้าหากว่ารถอยู่ห่างจากขอบทางด้านซ้ายน้อยกว่าระยะที่ต้องการบวกกับ 0.3 เมตร ก็จะถือว่าสิ้นสุดพฤติกรรมนี้

ถ้าไม่ตรงตามเงื่อนไขด้านบนทั้ง 2 ข้อก็จะทำการหันล้อรถไปยังจุดที่อยู่ห่างจากรถไปทางด้านหน้า 3 เมตร และห่างจากขอบทางด้านซ้ายตามระยะที่กำหนด จากนั้นจะขับเคลื่อนรถด้วยความเร็ว 1 เมตรต่อวินาที

- **พฤติกรรมการกลับรถ**

ขั้นตอนแรกจะทำการตรวจสอบก่อนว่า จะกลับรถทางซ้ายหรือทางขวา โดยมีเงื่อนไขดังนี้

- ถ้าองศาการกลับรถน้อยกว่า 90 องศา จะกลับรถไปทางที่ใกล้กว่า

- ถ้าหากว่าองศาการกลับรถมากกว่า 90 องศา จะใช้ข้อมูลจากการประมวลผลภาพตรวจสอบว่าห่างจากขอบถนนด้านใดมากกว่า โดยจะกลับรถด้านที่ห่างจากขอบถนนมากกว่า
- ถ้าหากว่าองศาการกลับรถมากกว่า 90 องศา และข้อมูลจากการประมวลผลภาพมีความผิดพลาดก็จะกลับรถทางขวาเสมอ

เมื่อได้ทิศทางการกลับรถแล้ว ในแต่ละรอบของการทำงานจะมีขั้นตอนดังนี้

- คำนวณว่าองศารถปัจจุบันอยู่ห่างจากเป้าหมายกี่องศา
- ถ้าหากว่าองศาที่ยังขาดอยู่น้อยกว่าหรือเท่ากับ 15 องศา ก็จะถือว่าเสร็จสิ้นการกลับรถ
- ถ้าหากว่าองศาที่ยังขาดอยู่มากกว่าองศากลับรถทั้งหมดที่คำนวณได้ตอนแรกมากกว่า 10 องศา ก็จะถือว่าโปรแกรมทำงานผิดพลาด และหยุดพฤติกรรมการกลับรถ กรณีนี้จะเกิดขึ้นเพราะกลับรถด้วยความเร็วสูงและระบบตัดสินใจตอบสนองช้ามากๆ ซึ่งเป็นไปไม่ได้แน่นอน
- ทำการคำนวณองศาล้อรถที่ต้องกลับรถ โดยมีเงื่อนไขว่าถ้าหากว่าองศาการกลับรถมากกว่า 45 องศา จะกำหนดองศาล้อเป็น 45 องศาตามทิศการกลับรถ แต่ถ้าหากว่าน้อยกว่า 45 องศาแต่ยังมากกว่า 15 องศา ก็จะกำหนดองศาล้อเป็น 0.9 เท่าขององศาการกลับรถที่ยังเหลืออยู่ แต่ถ้าองศาการกลับรถน้อยกว่าหรือเท่ากับ 15 องศา ก็จะถือว่าเสร็จสิ้นการกลับรถ

สำหรับความเร็วของการกลับรถจะตั้งไว้ที่ 1 เมตรต่อวินาที

- **พฤติกรรมการขับเคลื่อนไปตามถนน**

ขั้นตอนแรกจะคำนวณระยะทางทั้งหมดไปยังปลายทางก่อน จากนั้นแต่ละรอบของการทำงานจะมีขั้นตอนดังนี้

- คำนวณระยะทางที่ยังเหลืออยู่ และทำการตรวจสอบว่าถึงจุดหมายของพฤติกรรมหรือยัง ถ้าหากว่าถึงแล้วก็จะถือว่าสิ้นสุดพฤติกรรมนี้
- ถ้าหากว่าอยู่ในระยะ 5 เมตรแรกของการเริ่มพฤติกรรม องศาของล้อรถจะตั้งตรงไปยังปลายทางสุดท้ายของพฤติกรรมนี้ แต่ถ้าหากว่าอยู่นอกระยะ 5 เมตรแรกแล้ว องศาของล้อรถจะตั้งให้ขนานกับถนนที่วิ่งอยู่

- หลังจากคำนวณองศาล้อรถเบื้องต้นแล้ว ก็จะคำนวณองศาล้อรถโดยอาศัยข้อมูลจากการประมวลผลภาพถ้าหากว่าข้อมูลนั้นไม่ผิดพลาดและรถอยู่หลังจากระยะ 5 เมตรแรกของการเริ่มพฤติกรรม โดยองศาล้อจะหันไปยังจุดที่อยู่ห่างจากรถไปทางด้านหน้า 5 เมตร และห่างจากขอบทางด้านซ้าย 1.5 เมตร หลังจากคำนวณองศาล้อแล้วจะมาหาความแตกต่างขององศาล้อหน้าที่คำนวณแบบไม่ใช้ข้อมูลจากการประมวลผลภาพและแบบที่ใช้ข้อมูลจากการประมวลผลภาพ ถ้าหากว่าแตกต่างกันมากกว่า 35 องศาก็จะถือว่าองศาที่คำนวณโดยใช้ข้อมูลจากการประมวลผลภาพนั้นมีความผิดพลาดและจะนำองศาล้อรถแบบที่คำนวณโดยไม่ใช้ข้อมูลจากการประมวลผลภาพมาใช้แทน
- ถัดไปจะเป็นการตรวจสอบสิ่งกีดขวางด้านหน้าของรถ โดยถ้าหากพบสิ่งกีดขวางก็จะหลบไปด้านขวาด้วยองศาที่เหมาะสม
- ขั้นตอนสุดท้ายของพฤติกรรมนี้คือการตรวจสอบความถูกต้องขององศาล้อรถ ถ้าหากว่าองศาล้อรถมากกว่า 90 องศาจะถือว่าทำงานผิดพลาดและจะตั้งองศาล้อรถให้ตรง แต่ถ้ามากกว่า 50 องศาแต่น้อยกว่า 90 องศาก็จะยังถือว่าทำงานถูกต้องอยู่ แต่จะกำหนดองศาล้อรถให้เป็น 50 องศาเท่านั้น สำหรับความเร็วของการเคลื่อนที่ไปด้านหน้าตามถนนจะตั้งไว้ที่ 1 เมตรต่อวินาที

- **พฤติกรรมการเลี้ยวรถ**

ในแต่ละรอบของการทำงานจะมีการคำนวณองศาที่ยังต้องทำการเลี้ยว จากนั้นถ้าหากว่าองศาที่ต้องทำการเลี้ยวมากกว่า 35 องศา ก็จะกำหนดองศาล้อเป็น 35 องศาตามทิศทางการเลี้ยว แต่ถ้าน้อยกว่าหรือเท่ากับ 35 องศาแต่ยังมากกว่า 15 องศา ก็จะกำหนดองศาล้อเป็น 0.9 เท่าขององศาที่ยังขาดอยู่ ถ้าหากองศาที่ยังขาดอยู่น้อยกว่าหรือเท่ากับ 15 องศา ก็จะถือว่าเสร็จสิ้นพฤติกรรมการเลี้ยว สำหรับความเร็วของการเลี้ยวนั้นจะตั้งไว้ที่ 1 เมตรต่อวินาที

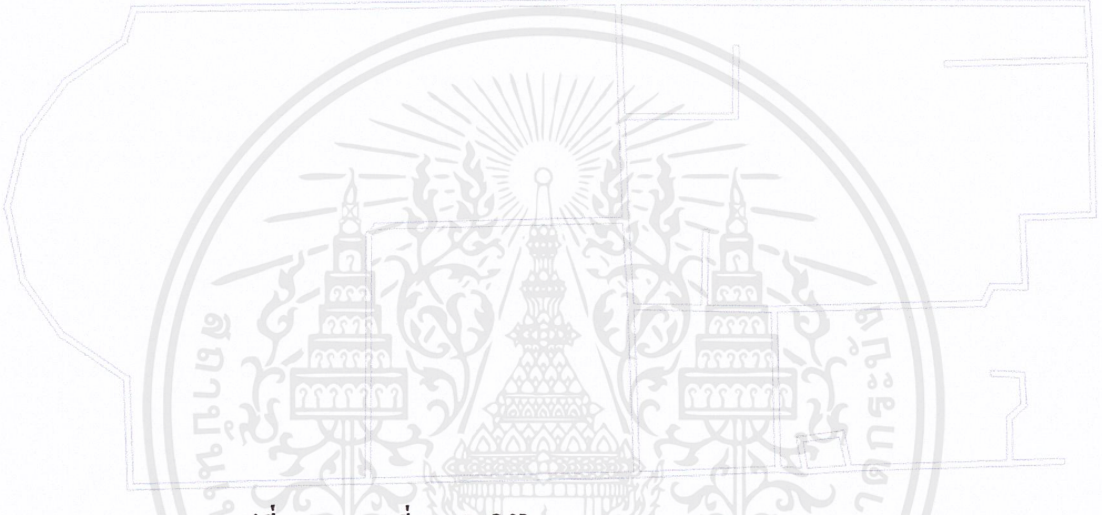
- **พฤติกรรมการหยุดรถ**

แต่ละรอบของการทำงาน จะทำการคำนวณความเร็วจากข้อมูล position2d แล้วนำมากำหนดความแรงของการเบรค โดยถ้าหากว่าเป็นการเบรคฉุกเฉินก็ตั้งความแรงของการเบรคเป็น 40 เท่าของความเร็วรถ แต่ถ้าไม่ฉุกเฉินก็จะเป็น 12 เท่าของความเร็วรถ

4.2.7 การพัฒนาส่วนการจำลองรถเพื่อทดสอบระบบตัดสินใจ

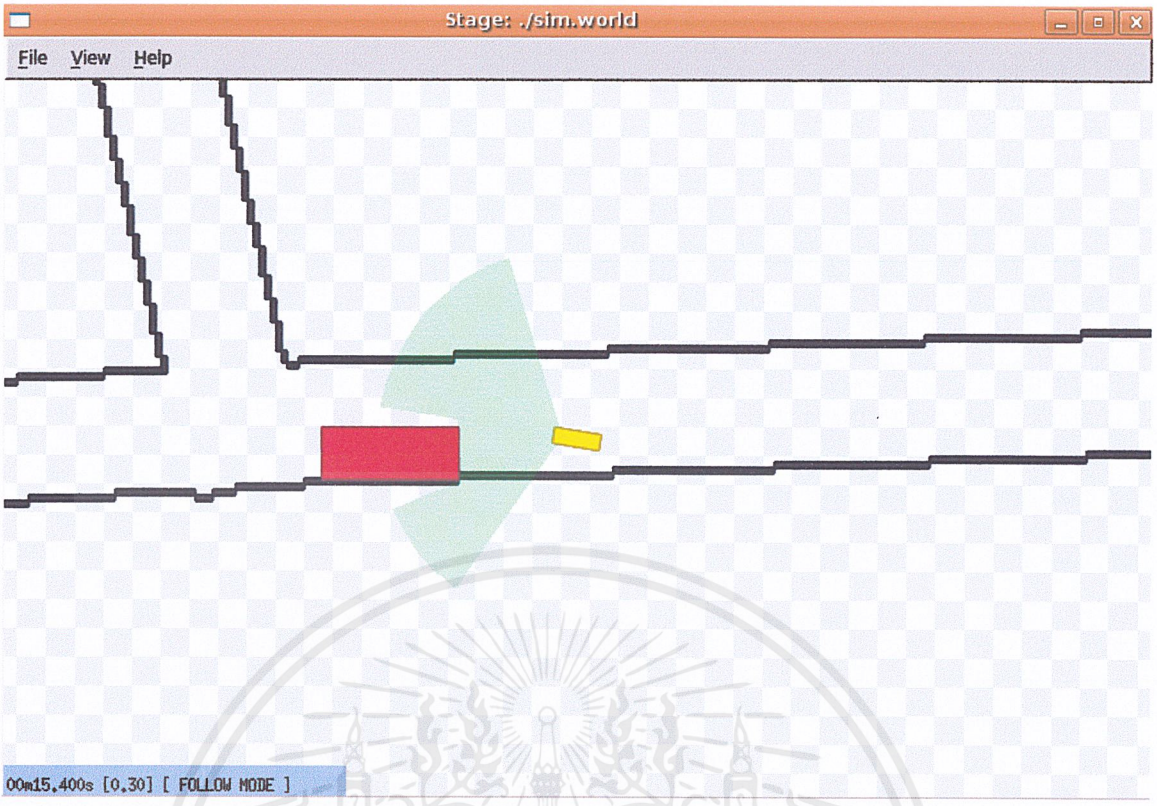
สำหรับการจำลองรถนั้นเราได้ใช้สเตจ (Stage) มาใช้ในการจำลองรถบนระนาบ 2 มิติ อินเตอร์เฟส (interface) ของเพลเยอร์ที่สเตจสามารถจำลองได้ ได้แก่ position2d และ sonar ส่วนข้อมูลจากการประมวลผลภาพ เราได้ทำการเขียนคลาสขึ้นมาทำงานในระบบตัดสินใจเพื่อจำลองข้อมูลที่ได้จากระบบประมวลผลภาพเองเลย

การจะจำลองถนนนั้นเราจำเป็นต้องสร้างภาพของขอบถนนทั้งหมด เราจึงเขียนโปรแกรมที่แปลงจากข้อมูลแผนที่แบบเวกเตอร์ (vector) เป็นภาพพีเอ็นจี (png) โดยได้ผลลัพธ์ดังรูปที่ 4.11



รูปที่ 4.11 แผนที่สำหรับใช้ในการจำลองการทำงานของรถ

จากนั้นก็เขียน world file ตามรูปแบบของสเตจ (stage) เพื่อสร้างแบบจำลองของรถและโซนาร์นอกจากจะจำลองรถและถนนแล้วสเตจยังสามารถจำลองสิ่งกีดขวางได้ด้วย



รูปที่ 4.12 การจำลองการทำงานของรถโดยสเตจ (Stage)

4.2.8 การพัฒนาส่วนติดต่อกับผู้ใช้

ในส่วนนี้เราได้แบ่งโปรแกรมออกเป็น 3 ส่วนใหญ่ๆคือ

- **ส่วนของไลบรารี่ภาษาจาวาสำหรับติดต่อกับเพลเยอร์ (player java client library)**

เราได้ดาวน์โหลดไคลแอนต์ไลบรารี่ จาก http://java-player.sourceforge.net/ และนำมาแก้ไขอินเตอร์เฟส (interface) ต่างๆตามการเปลี่ยนแปลงของเพลเยอร์จากเวอร์ชัน 2.0 เป็นเวอร์ชัน 2.1

- **ส่วนของแผนที่**

ส่วนนี้เราได้สร้างตามคลาสไดอะแกรมที่ออกแบบไว้ โดยแยกออกมาอีก 1 โปรเจคเพื่อให้ใช่ร่วมกับโปรแกรมสำหรับสร้างภาพของถนนจากแผนที่สำหรับการจำลองการทำงานได้

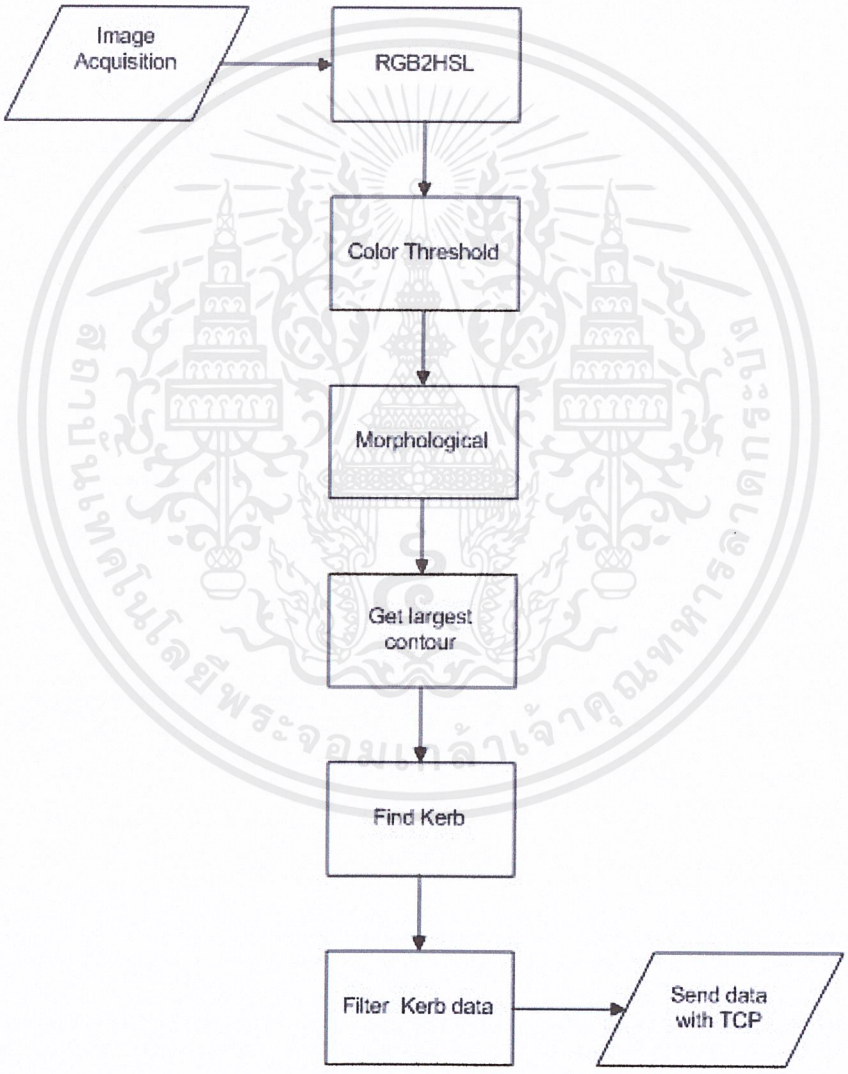
- **ส่วนของไคลแอนต์**

ส่วนนี้เราก็ได้สร้างตามคลาสไดอะแกรมที่ออกแบบไว้ โดยใช้ไลบรารี่จาก 2 ส่วนแรกข้างต้น

4.3 การพัฒนาระบบประมวลผลภาพ

ในการทำงานในส่วนนี้จะใช้กล้องเวปแคมในการรับภาพเข้ามาประมวลผลโดยพีซีที่เป็นแบบระบบสมองกลฝังตัว (Embeded PC) ซึ่งหน้าที่ของอิมเมจเซ็นเซอร์จะรับผิดชอบในการตรวจหาขอบถนน และการตรวจหาป้ายจราจรข้างทาง ซึ่งข้อมูลที่ถูกส่งออกมาจากงานทั้ง 2 นั้นจะส่งไปให้ระบบตัดสินใจนำไปใช้ช่วยในการควบคุมรถอีกทีหนึ่ง โดยข้อมูลต่างๆนั้นจะถูกส่งผ่านทางทีซีพีซ็อคเก็ต (TCP Socket)

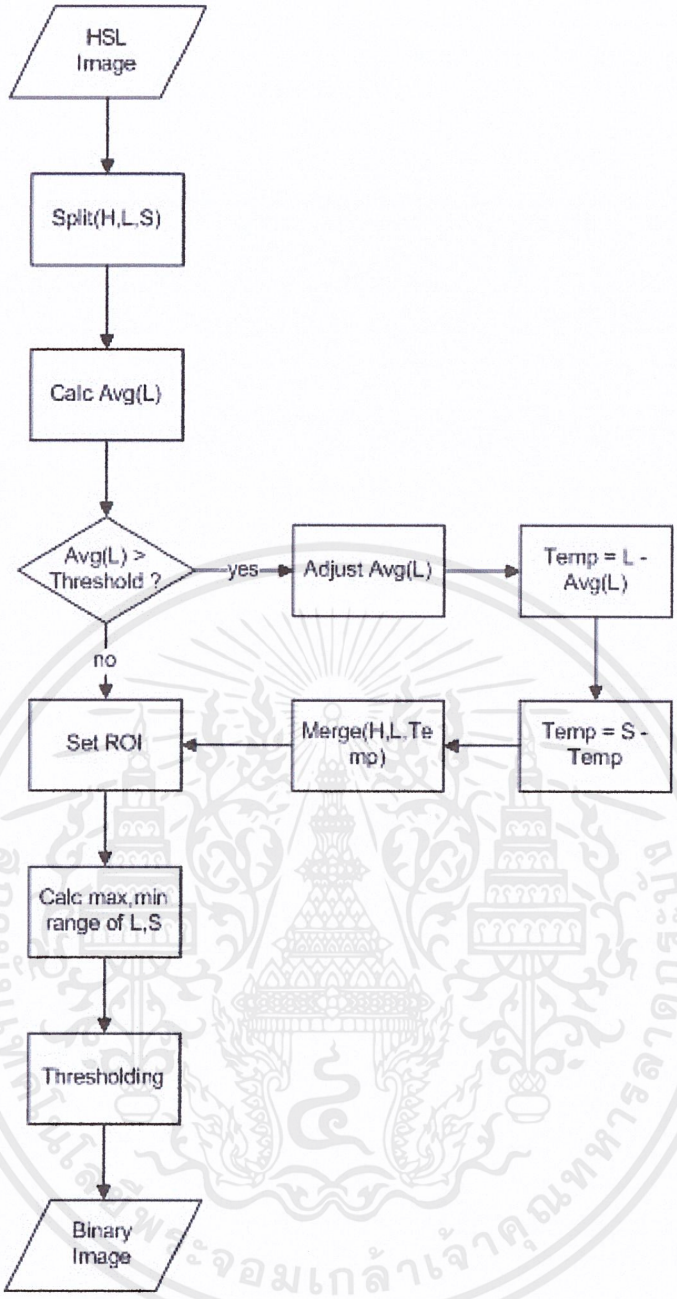
4.3.1 การหาขอบถนน



รูปที่ 4.13 ขั้นตอนการทำงานส่วนการหาขอบถนน

ซึ่งกระบวนการหลักทำงานหลักๆในส่วนนี้ จะมีดังนี้คือ

- เมื่อรับภาพเข้ามาจากเวปแคมแล้วจะทำการแปลงรูปภาพที่ได้นั้นจากรูปอาจีบี (RGB) เป็นรูปเอชเอสแอล (HSL) เนื่องจากโมเดลสีนี้จะยืดหยุ่นต่อสภาพแสงมากกว่า
- ทำการแบ่งช่วงสี (Color Threshold) เพื่อหาส่วนที่คาดว่าจะเป็นถนนออกมา ซึ่งข้อมูลที่ออกมาจากกระบวนการนี้จะเป็นไบนารีอิมเมจโดยในรายละเอียดจะอธิบายอยู่ด้านล่าง
- ทำการปรับโครงสร้าง (Morphological Processing) โดยในตอนแรกจะทำการหดก่อนเพื่อตัดส่วนถนนออกมาและค่อยทำการขยายกลับมาอีกที ซึ่งช่วยลบสิ่งแปลกปลอมที่ไม่ต้องการต่างๆออกไป อีกทั้งยังทำให้ส่วนต่างๆนั้นดูเรียบขึ้น
- ทำการหาเส้นรอบขอบ (contour) ที่ใหญ่ที่สุดออกมาซึ่งก็คือส่วนของถนนที่ต้องการนั่นเอง
- ทำการหาขอบของถนนที่ได้ โดยผ่านฟังก์ชันต่างๆ และการกรองของข้อมูลที่ได้ออกมา โดยในรายละเอียดจะอธิบายในหัวข้อถัดไป
- จากนั้นจะนำข้อมูลที่ได้มาทำเป็นสตริง (string) ชุดหนึ่งก่อน ซึ่งก่อนที่จะนำไปส่งนั้นจะทำการกรองข้อมูลที่ผิดพลาดก่อนในระดับหนึ่งก่อนที่จะทำการส่งค่าออกไปทางทีซีพีซ็อคเก็ต



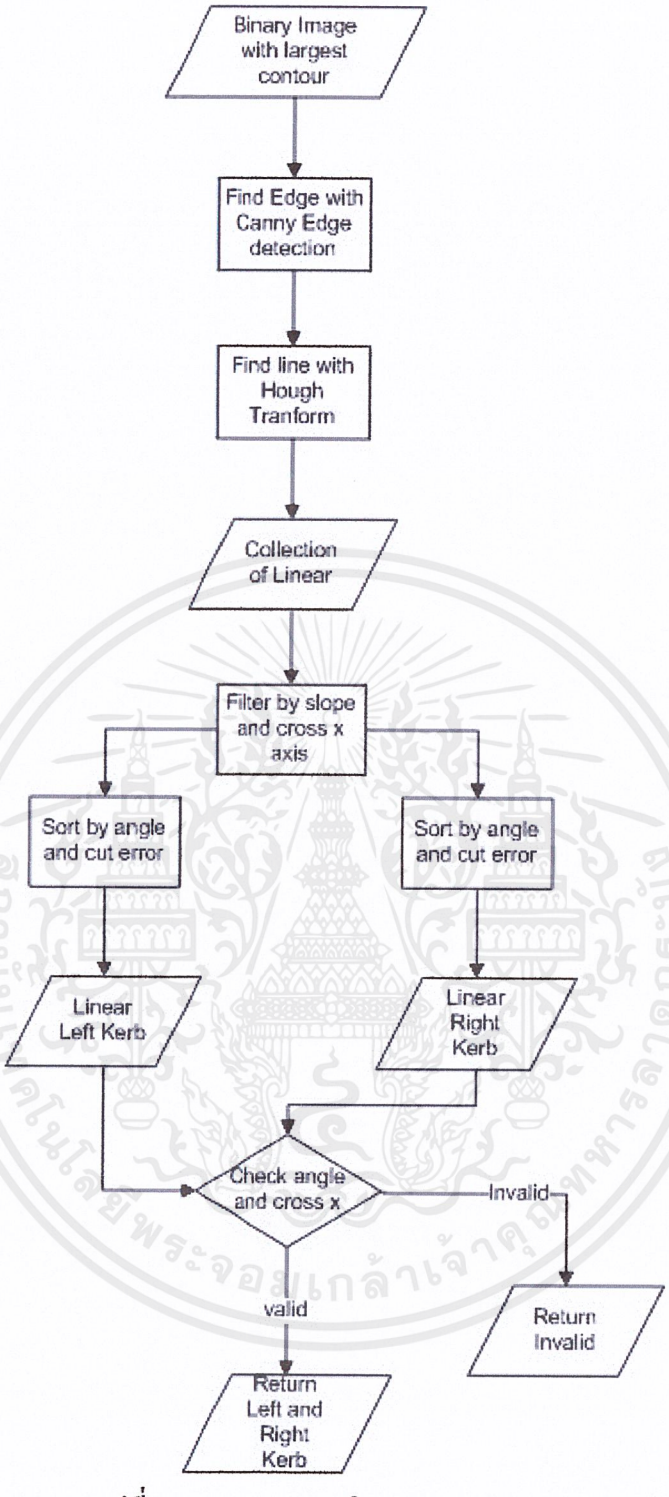
รูปที่ 4.14 กระบวนการในการทำการปรับแบ่งช่วงสี

กระบวนการในการทำการปรับแบ่งช่วงสีมีดังนี้

- ในกระบวนการนี้เริ่มจากนำภาพต้นฉบับที่แปลงจากอาจีบีเป็นเอชเอสแอลแล้วแยกชั้นของทั้ง 3 ชั้นออกจากกันโดนใช้ฟังก์ชัน cvSplit
- ทำการหาค่าเฉลี่ยของค่าสีในชั้นสีแอล (L layer) เพื่อเช็คดูว่ามากกว่าค่าขีดเริ่มเปลี่ยนที่กำหนดไว้หรือไม่หากไม่มากกว่าก็ไปทำขั้นตอนต่อไปได้เลย หรือหากมากกว่าก็แสดงว่าภาพนี้มีโอกาสที่จะเจอแดดก็จะทำการปรับค่าเฉลี่ยที่ ได้มาแล้วเอารูปแอลมาลบกับ

ค่านั้นก็จะได้ภาพที่มีเฉพาะส่วนที่เป็นแสงมากๆออกมา แล้วเอารูปเอส (S layer) มาลบกับภาพนั้นแล้วเอารูปทั้งหมดมารวมกันเหมือนเดิมโดยใช้ cvMerge

- ทำการกำหนดหน้าต่างที่เป็นอาโอไอ (Region of Interest) คือส่วนที่จะใช้เป็นตัวอย่างสีถนนโดยใช้ cvSetImageROI
- ทำการเอารูปอาโอไอนั้นไปหาช่วงสีของชั้นแอลและชั้นเอส
- ทำการนำช่วงสีนั้นมาทำการปรับแบ่งช่วงสีโดยหากค่านั้นอยู่ในช่วงที่กำหนด ให้จุดพิกเซลนั้นเป็น 1 หากไม่อยู่ในช่วงที่กำหนดให้เป็น 0 จะทำให้ได้ผลลัพธ์ของกระบวนการทั้งหมดนั้นเป็นไบนารีอิมเมจ



รูปที่ 4.15 กระบวนการในการหาขอบถนน

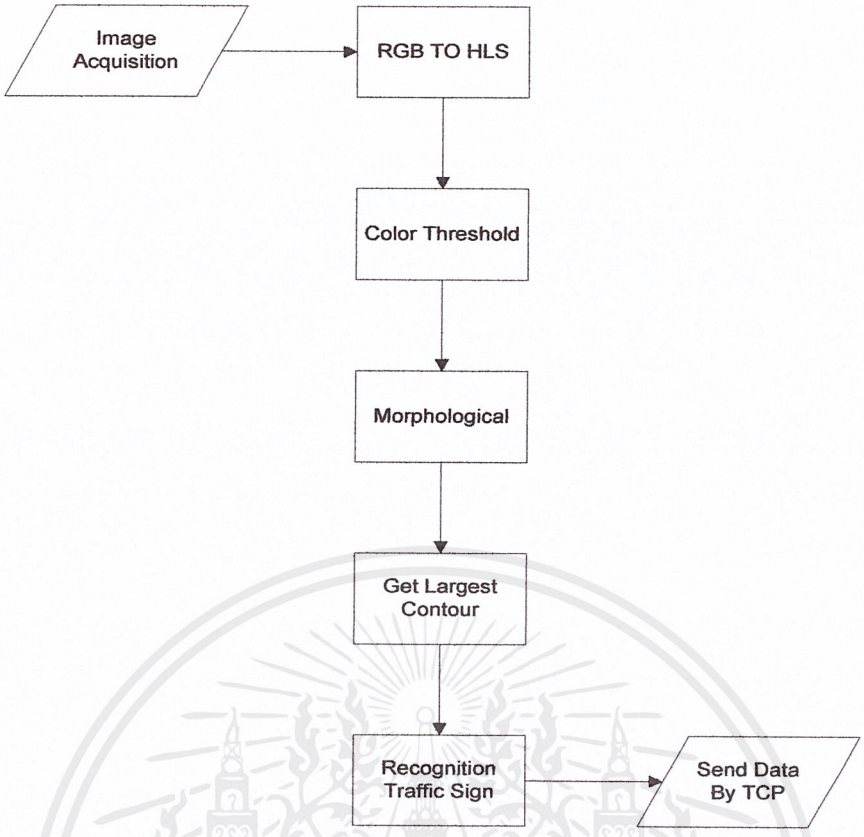
กระบวนการในการหาขอบถนนมีดังนี้

- เริ่มจากนำภาพที่มีเส้นรอบขอบที่ใหญ่ที่สุดมาทำการหาขอบภาพซึ่งใช้ทฤษฎีของแคนนี่ (Canny) โดยใช้ฟังก์ชัน cvCanny

- หาเส้นที่มีโอกาสจะเกิดได้โดยใช้ฟังก์ชัน cvHoughLines2 โดยเซตพารามิเตอร์ต่างๆ ตามที่ต้องการเช่น ความยาวอย่างน้อยที่ต้องการ ก็จะทำให้ได้กลุ่มข้อมูลของจุดต่างแต่ละเส้นออกมา
- ทำการเช็คว่า ข้อมูลของแต่ละเส้นนั้นเป็นของเส้นของขอบซ้ายหรือขอบขวาโดยเช็คจากองศาของแต่ละเส้นนั้น และทำการกรองเส้นที่ไม่ต้องการออกไปด้วยการเช็คจุดตัดแกน x
- ทำการนำกลุ่มข้อมูลของเส้นทั้งซ้ายและขวามาเรียงตามองศาแล้วทำการตัดข้อมูลที่ผิดพลาดไปตามจำนวนของเส้นทั้งซ้ายและขวา
- นำกลุ่มข้อมูลของเส้นทั้งซ้ายและขวามาหาค่าเฉลี่ยโดยถ่วงน้ำหนักตามความยาวเส้นด้วยเพื่อให้ได้เส้นขอบซ้ายและขวาเพียงอย่างละเส้น
- จากนั้นทำการเช็คว่าเส้นขอบซ้ายและขวานั้นถูกต้องหรือไม่โดยดูจากองศาและจุดตัดแกน x ที่ไม่น่าจะเป็นไปได้ โดยถ้าเป็นค่าที่ถูกต้องจะส่งค่าที่ใช้เช็คความถูกต้องเท่ากับ 1 พร้อมกับค่าต่างๆเช่น ระยะห่างทั้งข้างซ้ายและขวา หรือถ้าไม่ถูกต้องจะส่งค่าที่ใช้เช็คความถูกต้องเท่ากับ 0

4.3.2 การหาและวิเคราะห์ป้ายสัญญาณจราจร

ในการหาป้ายจราจรนั้นเริ่มต้นจะต้องหาให้ได้ก่อนว่าป้ายที่เราต้องการอยู่ที่ส่วนไหนของภาพซึ่งโดยปกติแล้วนั้นป้ายจราจรจะอยู่ทางฝั่งซ้ายของถนนเราจึงจะทำการประมวลผลเฉพาะฝั่งซ้ายของถนนเท่านั้นโดยการทำงานในส่วนของการหาป้ายจราจรมีขั้นตอนดังรูปที่ 4.16



รูปที่ 4.16 ขั้นตอนการทำงานในส่วนของการหาป้ายจราจร

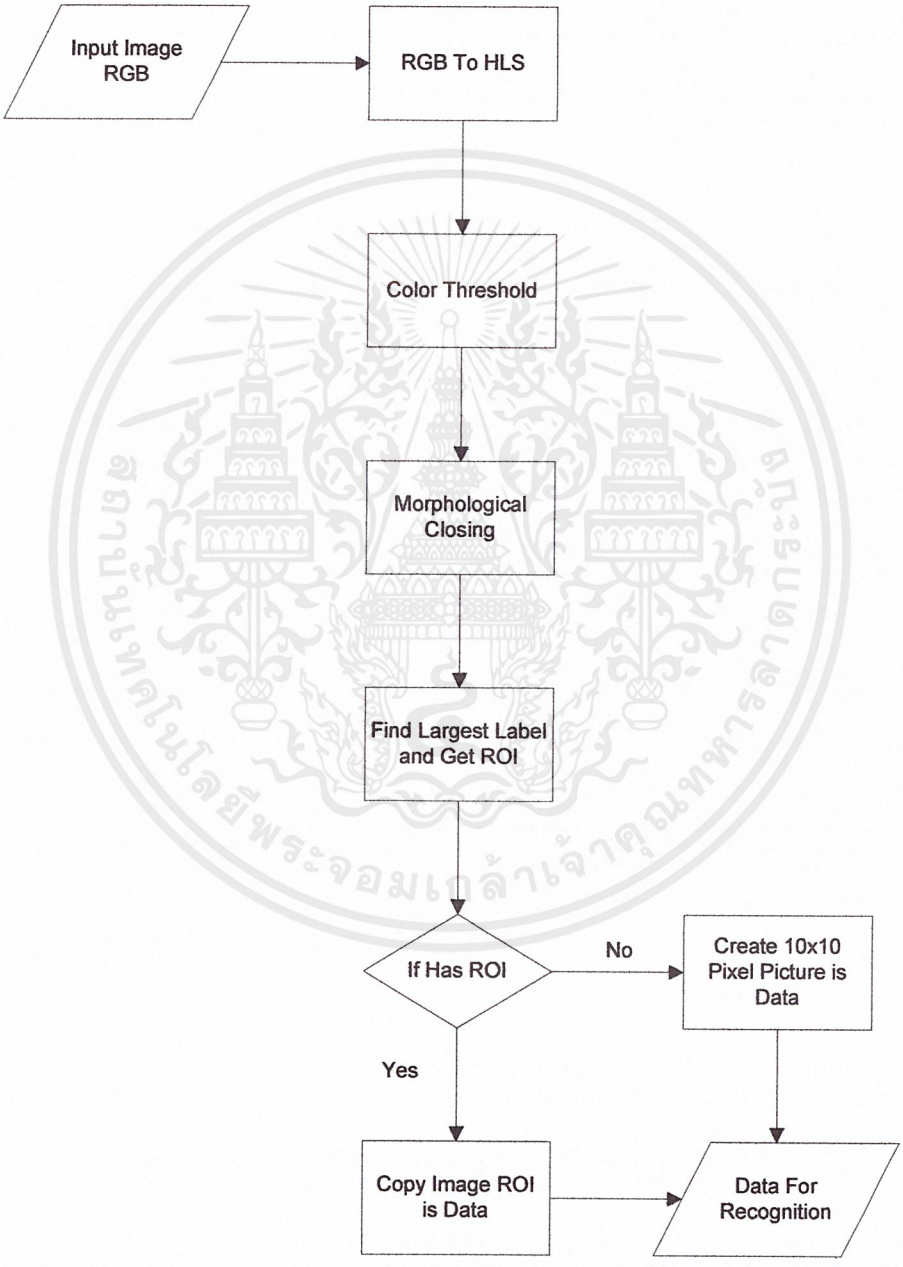
ในการทำงานของส่วนนี้ได้แบ่งขั้นตอนการทำงานหลักดังนี้

- รับภาพจากกล้องแล้วส่งภาพที่รับมาในรูปแบบอาจีบีไปยังโปรแกรมเพื่อประมวลผลต่อไป
- แปลงภาพที่รับมาจากอาจีบีเป็นเอชเอสแอลสำหรับการประมวลผลในขั้นตอนต่อไป
- ทำขั้นตอนการปรับแบ่งช่วงสีเพื่อให้ได้ส่วนที่มีสีตรงกับป้ายจราจรที่ต้องการแล้วทำเป็นภาพแบบไบนารีแล้วจึงนำส่วนที่ได้ไปประมวลผลในขั้นตอนต่อไป
- ในการทำการปรับโครงร่างนั้นคือการปรับคุณภาพ ซึ่งภาพที่ได้เป็นภาพแบบไบนารีมาจากในขั้นตอนที่แล้วให้มีสิ่งรบกวนลดลง
- ในขั้นตอนนี้จะทำการหาพื้นที่ส่วนที่เป็นสีขาวใหญ่ที่สุดในภาพแบบไบนารีแล้วทำการเลือกเฉพาะส่วนนั้นออกมาเก็บไว้เพื่อใช้ในการวิเคราะห์ภาพต่อไป
- ในขั้นตอนนี้จะนำภาพที่ได้มาจากขั้นตอนที่แล้วมาทำการวิเคราะห์กับภาพที่เป็นไปได้แล้วหาเปอร์เซนต์ที่เป็นไปได้มากที่สุด

- เมื่อได้ข้อมูลที่มีความเป็นไปได้มากที่สุดแล้วก็จะทำการส่งค่านั้นไปให้กับส่วนตัดสินใจต่อไปโดยผ่านทางทีซีพีซ็อคเก็ต

4.3.2.1 ขั้นตอนการหาป้ายจราจรจากภาพที่ได้รับมา

ในขั้นตอนนี้จะทำการหาป้ายจราจรที่อยู่ในภาพที่รับเข้ามาโดยจะมีขั้นตอนดังรูปที่ 4.17



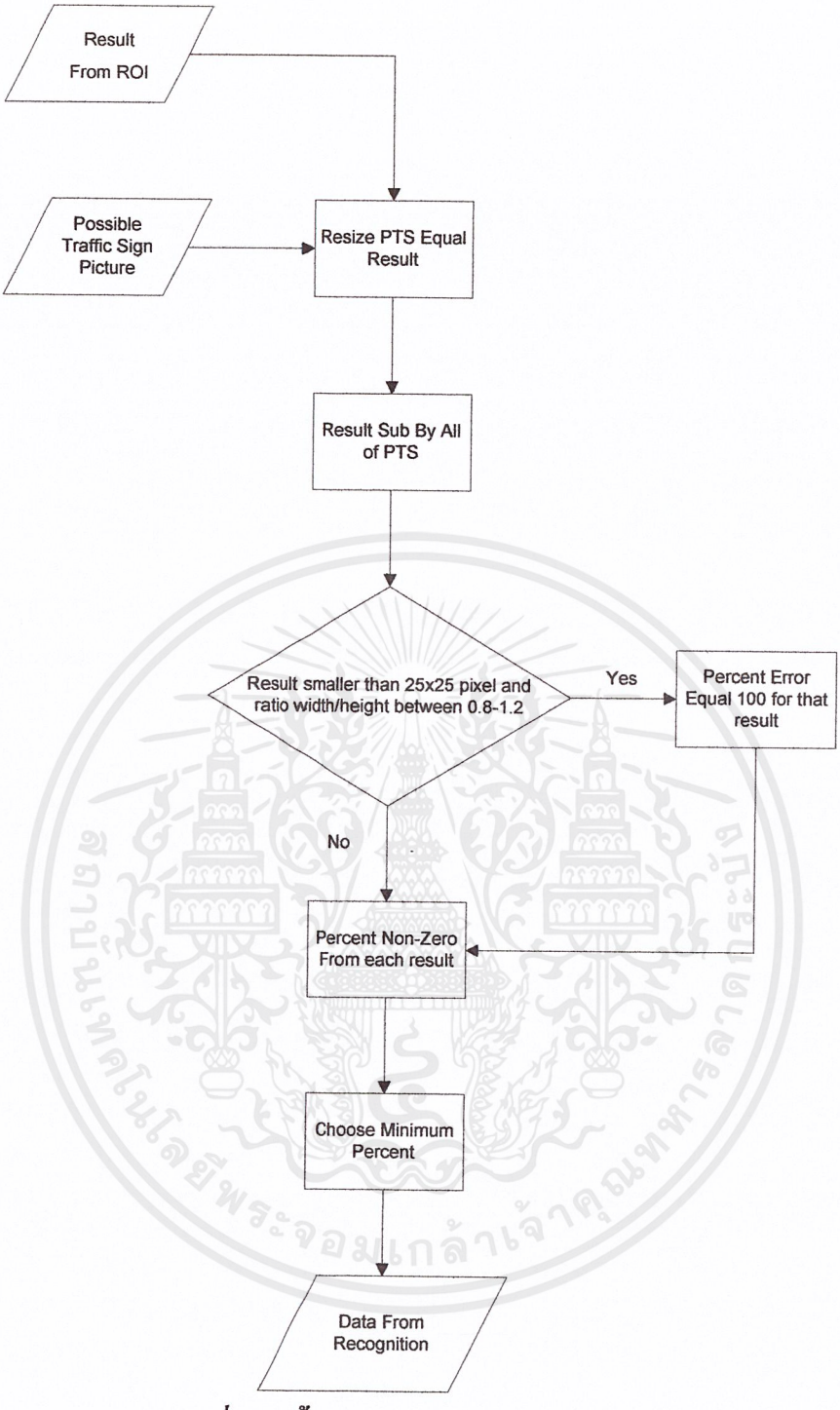
รูปที่ 4.17 ขั้นตอนในการหาป้ายจราจรจากภาพที่รับมา

จากรูปที่ 4.17 มีการหลักทำงานดังนี้

- รับภาพเข้ามาจากกล้องโดยใช้ฟังก์ชันของโอเพนซีวี (OpenCV) เพื่อที่จะนำมาเก็บลงบนตัวแปรที่ชี้ไปยังภาพชนิด IplImage* เนื่องจากป้ายจราจรจะพบในฝั่งซ้ายของถนนซึ่งก็คือฝั่งซ้ายของภาพนั่นเอง จึงจะทำการวิเคราะห์เฉพาะส่วนครึ่งซ้ายของภาพเท่านั้น
- หลังจากรับภาพมาแล้วนั้นก็ทำการนำภาพที่ได้รับมาแบบอาจีบีแปลงให้เป็นภาพแบบเอชเอสแอลโดยใช้ฟังก์ชั่น cvCvtColor ของโอเพนซีวี
- เมื่อได้ภาพที่แปลงมาแล้วก็ทำขั้นตอนของการปรับแบ่งช่วงสีเพื่อเลือกเฉพาะสีที่สนใจออกมาจากภาพโดยสีที่เลือกนั้นเป็นสีของป้ายจราจรที่ต้องการแล้วเปลี่ยนภาพให้เป็นภาพแบบไบนารีหรือก็คือภาพขาว-ดำ
- เมื่อได้ภาพแบบไบนารีมาแล้วนั้นก็จะทำขั้นตอนการปรับโครงร่างแบบปิด (Closing) เพื่อที่จะปรับคุณภาพของภาพให้มีสิ่งรบกวนลดลง
- เมื่อผ่านการปรับคุณภาพของภาพแล้วจะนำภาพที่ได้มาทำขั้นตอนฮิสโตแกรม (Histogram) เพื่อหาว่าส่วนที่ไม่ใช่สีดำในภาพที่ตำแหน่งใดมากที่สุดแล้วจึงทำการดึงภาพที่ตำแหน่งนั้นออกมาจากภาพใหญ่
- ในขณะที่ดึงภาพเฉพาะส่วนที่ผ่านการทำฮิสโตแกรมนั้นถ้าเกิดในภาพใหญ่ไม่มีส่วนที่เป็นสีขาวเลยหรือส่วนที่เป็นสีขาวมีขนาดเล็กกว่า 10x10 พิกเซล ก็จะถือว่าในภาพไม่มีป้ายจราจรให้ทำการสร้างภาพที่มีขนาด 10x10 พิกเซลที่เป็นสีดำทั้งภาพเป็นข้อมูลที่จะส่งไปแต่ถ้ามีภาพและภาพมีขนาดใหญ่กว่า 10x10 พิกเซลก็ให้ทำการดึงภาพนั้นมาเป็นข้อมูลที่จะส่งไปวิเคราะห์ต่อไป

4.3.2.2 ขั้นตอนในการวิเคราะห์ป้ายจราจร

ในขั้นตอนนี้จะนำเอาภาพผลลัพธ์ที่ได้มาจากขั้นตอนที่แล้วมาทำการวิเคราะห์เพื่อหาว่าป้ายจราจรป้ายใดที่มีความเป็นไปได้มากที่สุดที่จะส่งผลที่ได้ไปบอกกับส่วนตัดสินใจ โดยจะมีขั้นตอนการทำงานดังรูปที่ 4.18



รูปที่ 4.18 ขั้นตอนการวิเคราะห์ป้ายจราจร

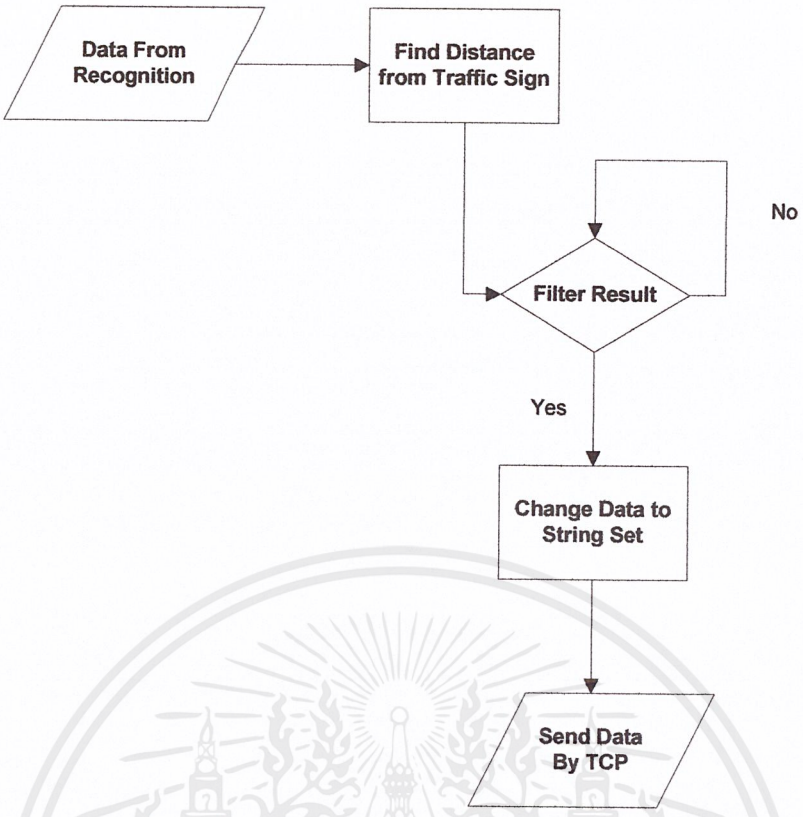
จากรูปในขั้นตอนนี้มีหลักการทำงานดังนี้

- เมื่อได้ภาพผลลัพธ์มาจากขั้นตอนที่แล้วนั้นให้ทำการโหลดภาพที่มีความเป็นไปได้ที่จะเป็นป้ายจราจรที่จะหาเจอเข้ามาแล้วทำการปรับขนาดให้มีขนาดเท่ากับภาพผลลัพธ์ที่ได้มา โดยใช้ฟังก์ชัน cvResize ของโอเพนซีวี

- นำภาพป้ายที่มีความเป็นไปได้ (Possible Traffic Sign picture : PTS) ที่ผ่านการปรับขนาดแล้วทุกภาพไปลบกับภาพผลลัพธ์ที่ได้มาแล้วเก็บภาพที่ผ่านการลบไว้แยกกันของแต่ละภาพ
- เลือกเอาเฉพาะภาพที่มีขนาดใหญ่กว่า 25x25 พิกเซล และมีค่าอัตราส่วนระหว่างความกว้างของภาพกับความสูงของภาพอยู่ในช่วง 0.8-1.2
- ถ้าภาพที่ได้เล็กกว่า 25x25 พิกเซล หรือค่าอัตราส่วนความกว้างต่อความสูงภาพไม่อยู่ในช่วง 0.8-1.2 จะให้เปอร์เซ็นต์ผิดพลาดของภาพนั้นเท่ากับ 100
- ถ้าภาพที่ได้ไม่ได้อยู่ในเงื่อนไขที่กำหนดจะหาเปอร์เซนต์ความผิดพลาดจากค่าของจุดที่ไม่ใช่สีดำในภาพเทียบกับพิกัดที่มีทั้งหมดในภาพ
- เลือกภาพที่เปอร์เซ็นต์ความผิดพลาดต่ำสุดเป็นป้ายจราจรที่วิเคราะห์ได้

4.3.2.3 การหาระยะทางระหว่างพาหนะกับป้ายจราจร และการกรองข้อมูลก่อนส่งไปให้ส่วนตัดสินใจ

ในขั้นตอนนี้จะทำการหาระยะทางระหว่างพาหนะกับป้ายจราจรว่ามีระยะห่างเท่าใดแล้วจึงส่งไปบอกส่วนตัดสินใจ โดยก่อนที่จะส่งไปบอกก็จะมีการกรองข้อมูลก่อนส่งว่าเป็นข้อมูลที่ถูกต้องจริงและไม่ส่งข้อมูลไปให้ซ้ำถ้าข้อมูลยังไม่เปลี่ยนแปลงหรือหายไป โดยจะมีขั้นตอนการทำงานดังรูปที่ 4.19



รูปที่ 4.19 ขั้นตอนการหาระยะห่างและการกรองข้อมูล

จากรูปในขั้นตอนนี้มีหลักการทำงานดังนี้

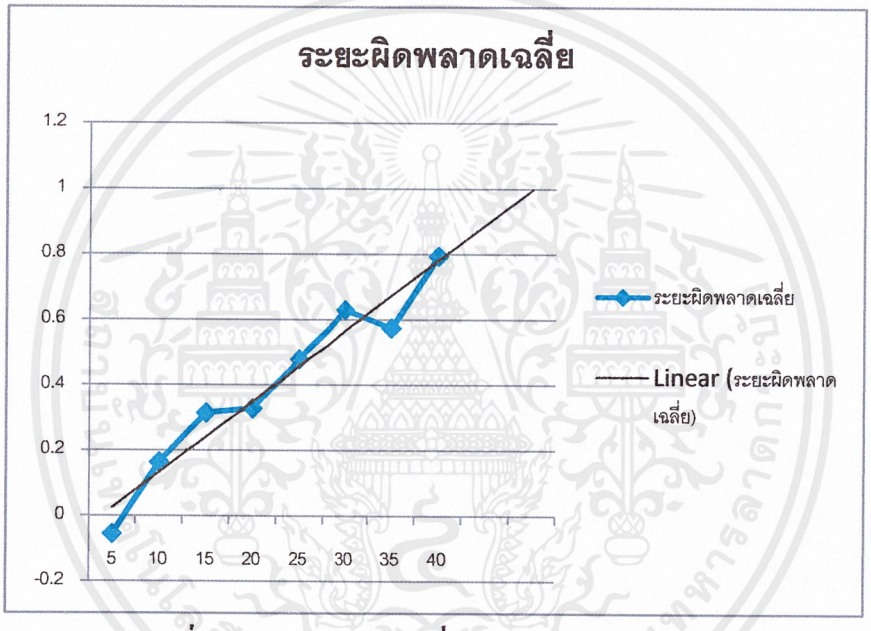
- หาระยะห่างจากป้ายจราจรด้วยการคำนวณจากขนาดของภาพที่ดึงออกมาเทียบอัตราส่วนกับระยะจริงที่วัดระยะมาได้
- กรองข้อมูลโดยจะทำการตรวจสอบข้อมูลที่วิเคราะห์ได้ว่าเหมือนเดิมตลอดหรือไม่ ข้อมูลไม่เปลี่ยนไปมา และข้อมูลที่ได้ไม่ซ้ำกับข้อที่ส่งไปก่อนหน้านี้ในระยะเวลาหนึ่ง
- เมื่อกรองข้อมูลเรียบร้อยแล้วจึงทำการสร้างข้อมูลนั้นให้เป็นข้อมูลชนิดสตริงแล้วค่อยทำการส่งข้อมูลผ่านทางทีซีพี ซ็อคเก็ต ไปให้ยังส่วนตัดสินใจต่อไป

บทที่ 5

การทดลองและผลการทดลอง

5.1 การทดลองมอดูลวัดระยะทาง

จากการทดสอบโดยการนำรถไปวิ่งที่ระยะต่างๆพบว่า มอดูลวัดระยะทางสามารถบอกระยะทางได้ดีในระดับหนึ่ง โดยมีความผิดพลาดมากขึ้นเรื่อยๆตามระยะทางเป็นเชิงเส้น ซึ่งเกิดจากการวัดขนาดของล้อหน้าที่ผิดพลาดและขนาดของวงล้อหน้าไม่คงที่

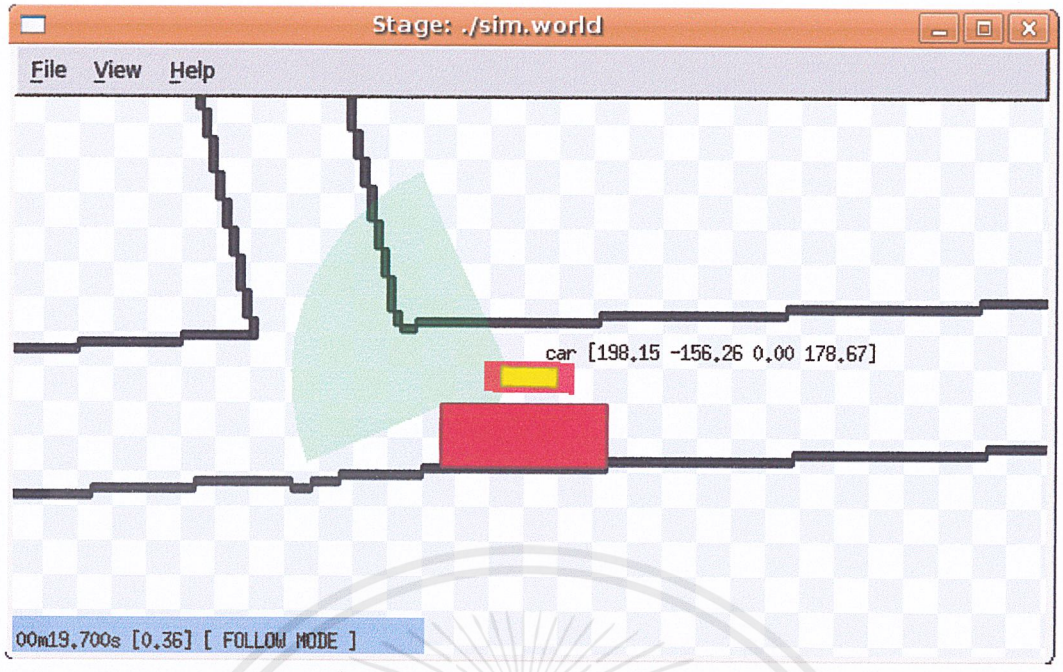


รูปที่ 5.1 ระยะผิดพลาดเฉลี่ยของมอดูลวัดระยะทาง

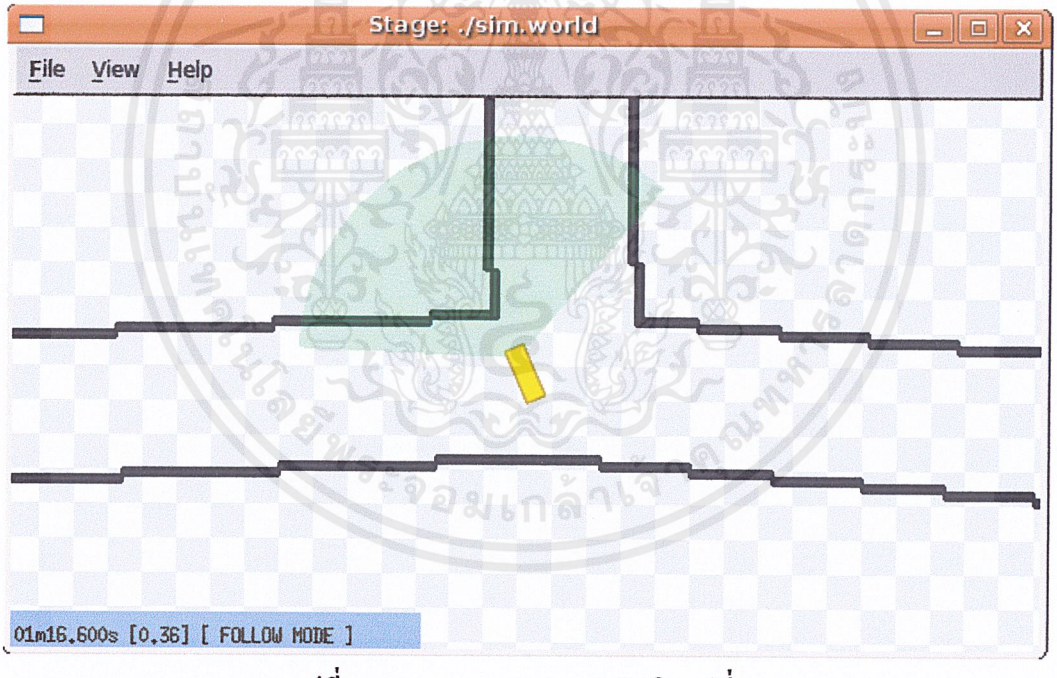
5.2 การทดลองระบบตัดสินใจ

การทดลองระบบตัดสินใจ เราได้ทำการทดสอบบนระบบจำลองรถ 2 แบบคือแบบที่ใช้ข้อมูลจากระบบประมวลผลภาพที่เราได้จำลองขึ้นมา และแบบที่ไม่ใช้ข้อมูลจากระบบประมวลผลภาพ

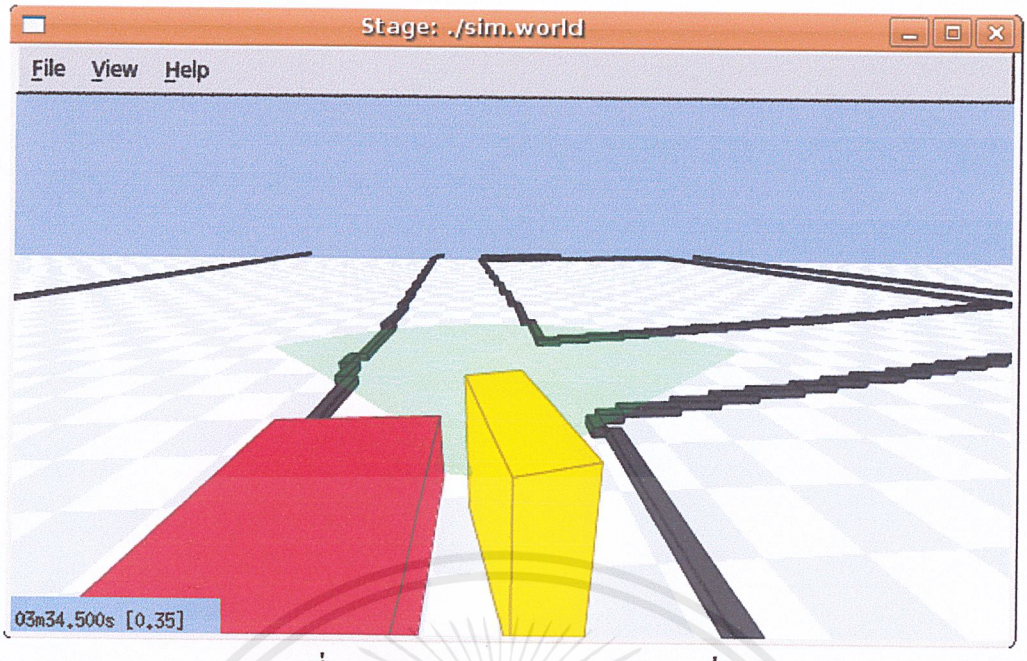
โดยการทดลองทั้ง 2 แบบพบว่า ระบบตัดสินใจมีพฤติกรรมตรงตามที่ออกแบบไว้ สามารถนำรถหลบหลีกสิ่งกีดขวางได้อย่างถูกต้อง และขับเคลื่อนรถไปยังจุดหมายปลายทางได้โดยใช้เส้นทางที่เหมาะสม



รูปที่ 5.2 การทดลองระบบตัดสินใจรูปที่ 1



รูปที่ 5.3 การทดลองระบบตัดสินใจรูปที่ 2



รูปที่ 5.4 การทดลองระบบตัดสินใจรูปที่ 3

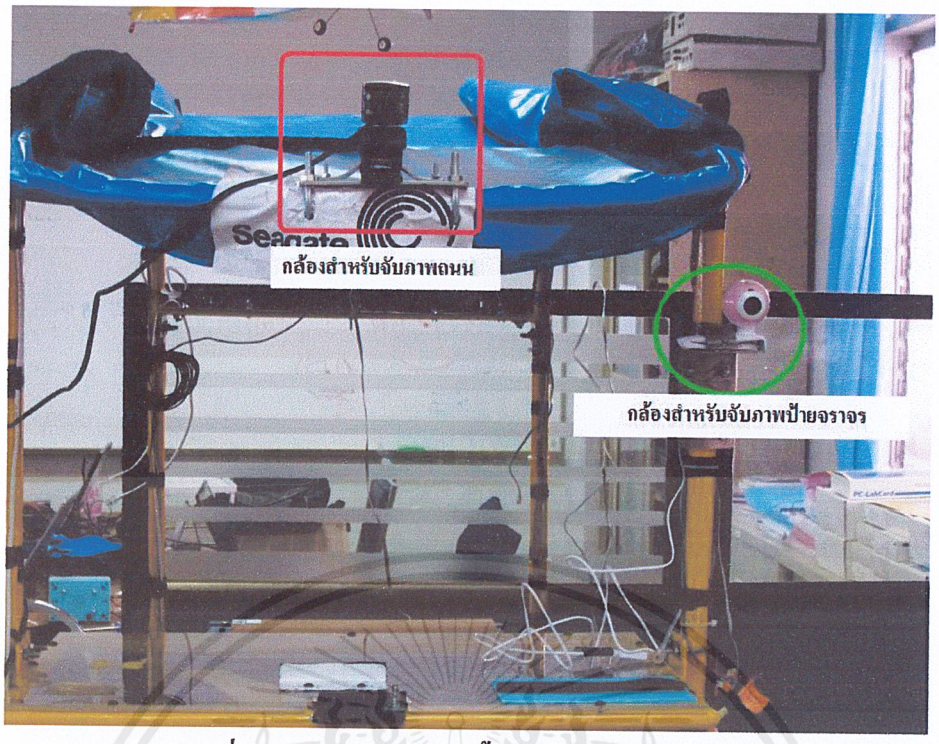
5.3 การทดลองและผลการทดลองของส่วนการประมวลผลภาพ

ในการทดลองการทำงานของส่วนอิมเมทเซ็นต์เซอร์นี้เราจะใช้ตัวบอร์ด อัตราไคแอน (UltraClient) ของบริษัทนอร์เทค (Norhtec) ซึ่งมีคุณสมบัติดังนี้

- CPU - Onboard 1.2 Ghz VIA EDEN Esther processor
- 2 x USB 2.0 ports
- 1 x VGA out port with hardware
- 1 RJ45 for 10/100 MBit Lan
- Single 5 Volts DC @ 5 A (max) support
- 512 MB RAM
- 160 GB 2.5" hard disk

5.3.1 การติดตั้งกล้องบนยานพาหนะที่ใช้

การติดตั้งกล้องนั้นเราจะทำการติดตั้งกล้องทั้งสองตัวไว้บนพาหนะดังรูปที่ 5.5



รูปที่ 5.5 ตำแหน่งของการตั้งกล้องบนยานพาหนะ

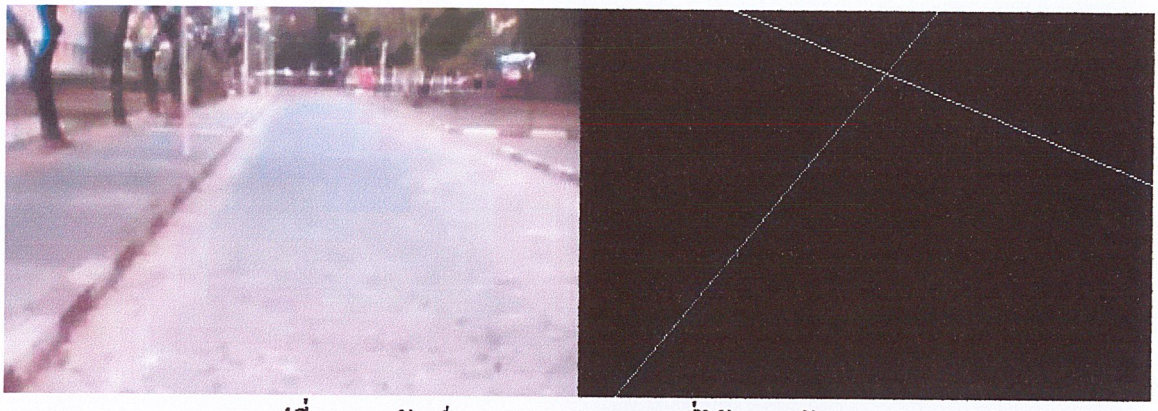
จะเห็นได้ว่ากล้องที่ใช้ในการจับภาพถนนนั้นจะวางอยู่ตรงส่วนกลางของยานพาหนะและกล้องที่ใช้สำหรับการจับภาพป้ายจราจรนั้นจะอยู่ด้านข้างฝั่งซ้ายของยานพาหนะ

5.3.2 การทดลองในส่วนของการหาขอบถนน

เป็นการทดลองโดยให้รถเดินทางเป็นเส้นตรงแล้วจับขอบถนนด้านใกล้แล้วเทียบรูปจริงที่รับเข้ามากับขอบทางที่ได้จากการรันโปรแกรมโดยการทดลองนั้นจะทำในสภาพถนนและช่วงเวลาที่ต่างกันซึ่งแต่ละถนนก็จะมีลักษณะสีและผิวของถนนและสิ่งแวดล้อมรอบๆที่ต่างกัน อีกทั้งในแต่ละช่วงเวลาก็จะมีลักษณะแสงแดดหรือร่มเงาที่ต่างกันด้วยซึ่งสิ่งต่างๆเหล่านี้มีผลอย่างมากกับผลการทดลอง

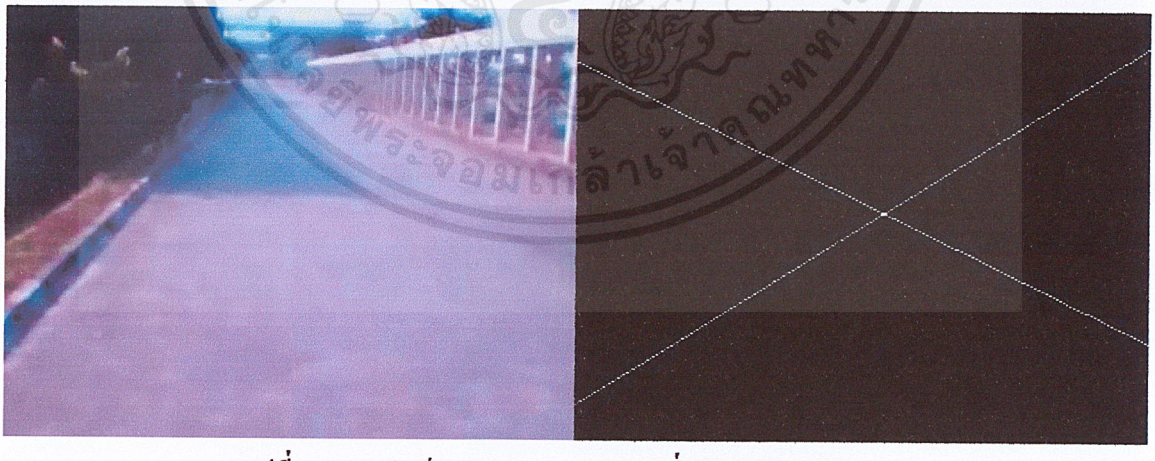
5.3.2.1 วิธีการทดลอง

นำวีดีโอตัวอย่างที่ถ่ายมาจากเวปแคมที่ติดตั้งกับตัวรถซึ่งทำการปรับแต่งกล้องโดยการปรับความต่างสี (contrast) หรือสมดุลแสงสีขาว (white balance) ที่ดีเพียงพอแล้ว มารันในโปรแกรมการหาขอบถนนที่ได้เขียนไว้ โดยค่าที่ได้ออกมานั้นจะเป็นค่าเวกเตอร์จากการทดลองจะยอมรับให้ค่าที่ได้ออกมานั้นผิดพลาดได้เล็กน้อย เพราะการทำการประมวลผลภาพนั้นไม่สามารถทำให้ค่าที่ได้ออกมาแม่นยำมากนักโดยขึ้นอยู่กับหลายๆอย่าง ซึ่งรูปด้านล่างเป็นตัวอย่างการหาขอบถนนที่ถูกต้องไม่ถูกต้องแต่ยังอยู่ในค่าเบี่ยงเบนที่ยอมรับได้ และไม่ถูกต้อง ตามลำดับ



รูปที่ 5.6 ผลลัพท์ของการหาขอบถนนที่ได้ผลถูกต้อง

รูปที่ 5.7 ผลลัพท์ของการหาขอบถนนที่ได้ผลไม่ถูกต้องแต่ยังอยู่ในค่าเบี่ยงเบนที่ยอมรับได้



รูปที่ 5.8 ผลลัพท์ของการหาขอบถนนที่ได้ผลไม่ถูกต้อง

5.3.2.2 ผลการทดลอง

ตารางที่ 5.1 ผลการทดลองของการหาขอบถนน

สถานที่	จำนวนเฟรมทั้งหมด	จำนวนเฟรมที่ถูกต้อง	เปอร์เซ็นต์ความถูกต้อง
เวลา 8.30			
ถนนข้างหอใน	144	89	61.18
ถนนหลังหอใน	130	67	51.54
ถนนหลังสระว่ายน้ำ	77	42	54.55
รวม	351	198	56.41
เวลา 12.30			
ถนนข้างหอใน	144	73	50.69
ถนนหลังหอใน	144	105	72.92
ถนนหลังสระว่ายน้ำ	144	101	70.14
รวม	432	279	64.58
เวลา 16.00			
ถนนข้างสระว่ายน้ำ	144	77	53.47
ถนนหลังหอใน	96	50	52.08
ถนนข้างหอใน	128	72	56.25
รวม	368	199	54.08
เวลา 17.45			
ถนนข้าง ECC	227	206	90.75
ถนนหน้าร้านอันนา	38	36	94.74
ถนนข้างหอใน	86	53	61.63
รวม	351	295	84.05

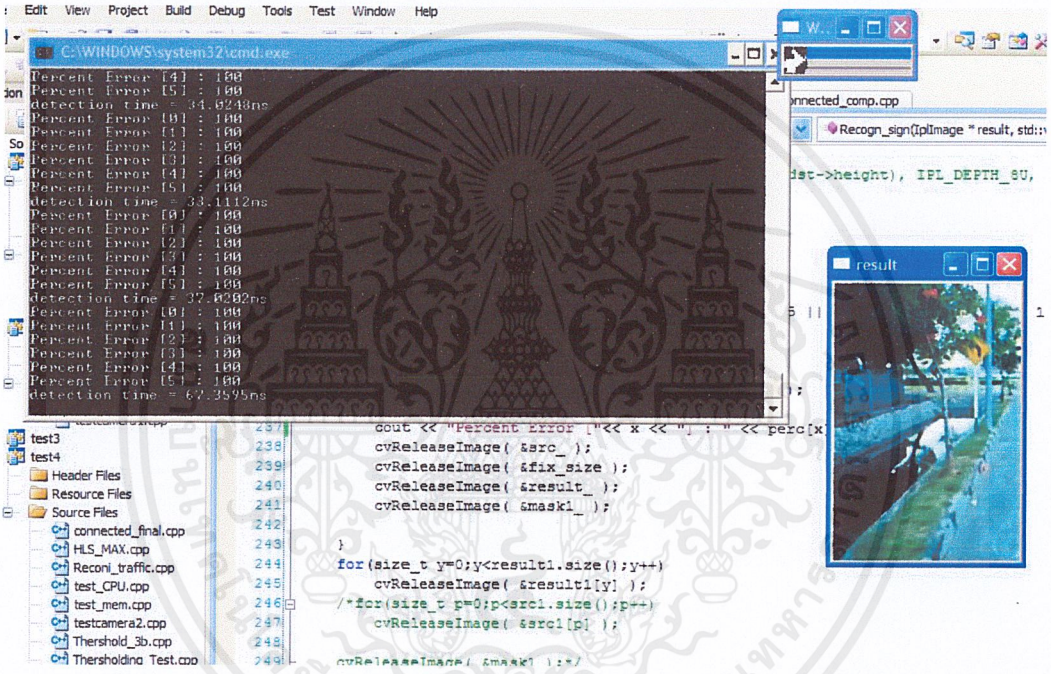
5.3.3 การทดลองในส่วนการหาและวิเคราะห์ป้ายจราจร

ในการทดลองในส่วนนี้ได้มีการทดลองตัวโปรแกรมกับป้ายภายในสถาบันโดยในการทดลองแต่ละครั้งจะต้องมีการทำการปรับแต่งค่าของกล้องก่อนที่จะทำการวิ่งจริงเนื่องจากในแต่ละเวลาของวันนั้นกล้องจะสามารถอ่านค่าสีและความสว่างของภาพออกมาได้ไม่คงที่จึงต้องมีการปรับแต่งก่อนทำการวิ่งจริงซึ่งโปรแกรมที่นำมาใช้จะต้องสามารถปรับค่าของความสว่าง

(brightness), ความต่างสี, ความเข้มสี (color intensity) และค่าความสมดุลของสีขาว ซึ่งเป็นค่าที่มีความสำคัญมากในการปรับแต่งให้กล้องมีความสามารถในการเก็บข้อมูลที่สามารถนำไปวิเคราะห์ต่อไปได้

5.3.3.1 การทดลองของการหาป้ายจราจรจากภาพที่รับเข้ามา

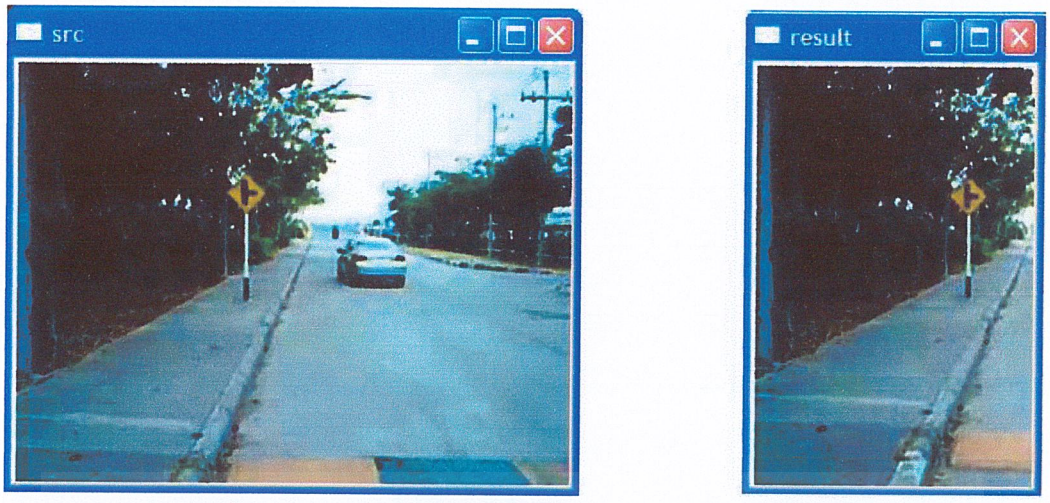
เริ่มต้นนี้เราจะรับภาพเข้ามาในโปรแกรมผ่านทางยูเอสบีพอร์ต (USB Port) ซึ่งในตัวโปรแกรมของเราจะทำการเรียกภาพเข้ามาที่ละเฟรม โดยจะรับเฟรมต่อไปเข้ามาก็ต่อเมื่อทำการประมวลผลเฟรมก่อนหน้าเสร็จแล้ว



รูปที่ 5.9 การนำภาพเข้ามาที่ละเฟรมเมื่อประมวลผลเฟรมก่อนหน้าเสร็จ

จากรูปที่ 5.9 จะเห็นได้ว่าเราได้ทำการนำวิดีโอ มาทำการทดลองในการรันโปรแกรม ซึ่งจะพบว่าเมื่อภาพจากวิดีโอภาพหนึ่งถูกประมวลผลเรียบร้อยแล้วนั้นภาพที่เข้ามา หรือเฟรมที่เข้ามาถัดไปไม่ใช่เฟรมที่ต่อจากเฟรมที่แล้วแต่เป็นเฟรมที่แสดงผลอยู่ขณะนั้นดังนั้นการประมวลผลที่เกิดขึ้นจึงสามารถทำได้ทันในเวลาที่ต้องการ

หลังจากจัดการเรื่องการนำภาพเข้ามาจากกล้องเรียบร้อยแล้วเราก็จะนำภาพที่รับมาทำการลดขนาดลงเพื่อที่จะลดเวลาในการประมวลผลลดลงโดยเราจะตัดภาพที่จะนำมาวิเคราะห์เฉพาะส่วนซ้ายของภาพดังรูปที่ 5.10



รูปที่ 5.10 ภาพเริ่มต้นที่รับเข้ามาและภาพที่ผ่านการลดขนาด

เมื่อทำการลดขนาดภาพเรียบร้อยก็จะนำภาพที่ลดขนาดแล้วไปเข้าโปรแกรมในส่วนของการหาภาพป้ายจราจรซึ่งในขั้นตอนนี้นั้นเราจะทำในขั้นตอนของแบ่งช่วงของค่าสี เพื่อหาป้ายออกมาจากภาพโดยผ่านขั้นตอนนี้แล้วจะทำให้ได้ภาพที่เป็นภาพขาว-ดำ ออกมาดังรูปที่ 5.11

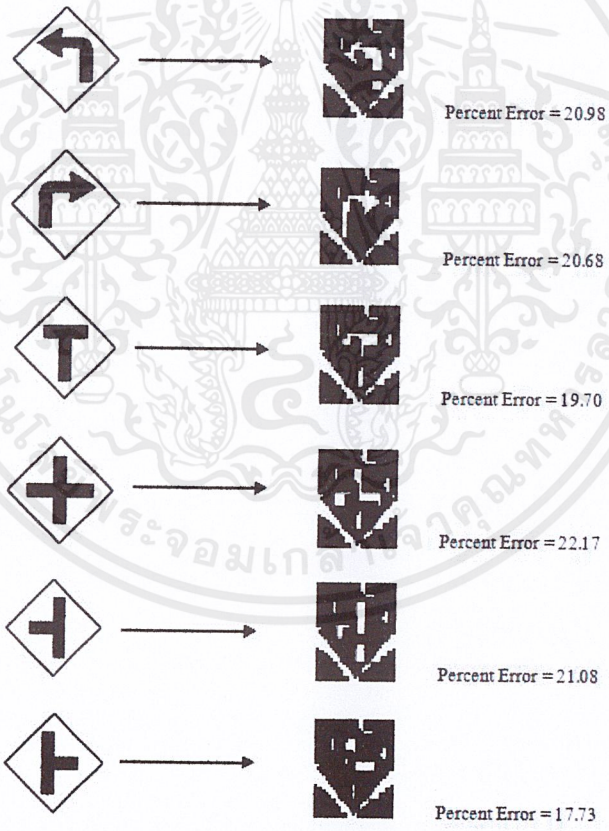
รูปที่ 5.11 ภาพ ขาว-ดำ (Binary Image) ที่ได้จากการทำการแบ่งช่วงสี

หลังจากได้ภาพที่พบป้ายจราจรมาแล้วนั้นก็จะทำการตัดเอาเฉพาะส่วนที่เป็นป้ายออกมาเท่านั้นดังรูปที่ 5.12

รูปที่ 5.12 ภาพของป้ายจราจรที่หาได้

5.3.3.2 การวิเคราะห์ภาพป้ายจราจรที่สามารถหาได้

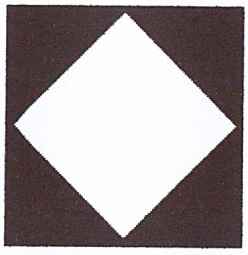
ในการทดลองนี้เราจะนำภาพป้ายจราจรที่หาได้มาทำการลบด้วยภาพป้ายจราจรที่ถูกต้องและมีความเป็นไปได้ทั้งหมดแล้วดูผลค่าของความผิดพลาดที่เกิดขึ้น โดยภาพที่จะนำมาลบนั้นต้องทำการปรับขนาดของภาพให้มีขนาดเท่ากับขนาดของป้ายที่หาได้ก่อนที่จะนำมาลบ ซึ่งผลของภาพที่ผ่านการลบแล้วนั้นจะเป็นดังรูปที่ 5.13



รูปที่ 5.13 ผลลัพธ์ของการนำภาพมาลบกับภาพต้นแบบ

จากผลลัพธ์ที่ได้ แสดงให้เห็นว่าเปอร์เซ็นต์น้อยที่สุดที่หาได้นั้นเป็นผลลัพธ์ที่ถูกต้อง และเปอร์เซ็นต์ผิดพลาดสูงสุดคือ 22.17 โดยเปอร์เซ็นต์ที่กำหนดเป็นมาตรฐานที่เรากำหนดคือ 25

เพราะภาพป้ายจราจรที่ไม่มีข้อมูลใดอยู่เลยมีเปอร์เซ็นต์ผิดพลาดเท่ากับ 44.34 เปอร์เซ็นต์ดังรูปที่ 5.14



รูปที่ 5.14 เปอร์เซ็นต์ผิดพลาดของป้ายที่ว่างเปล่า

และป้ายจราจรที่มีค่าเปอร์เซ็นต์ความผิดพลาดมากที่สุดคือ 35.03 เปอร์เซ็นต์ดังรูปที่ 5.15

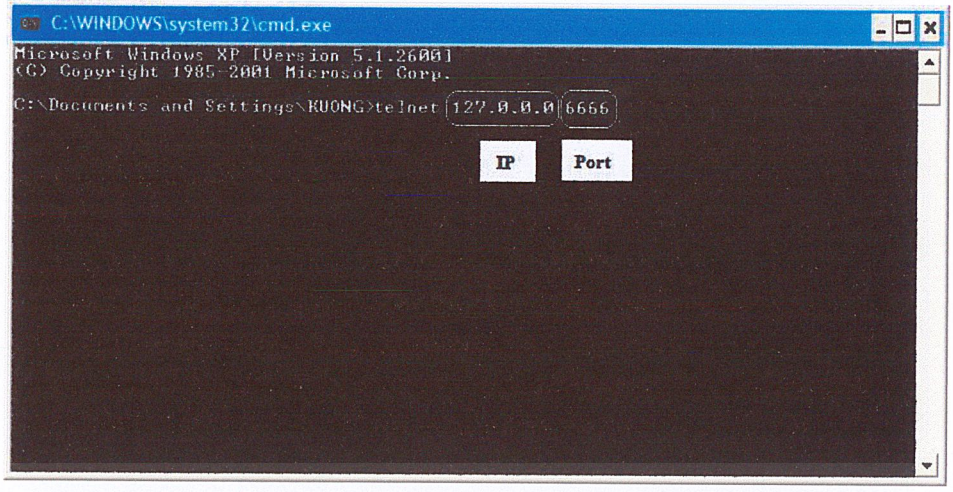
รูปที่ 5.15 ป้ายจราจรที่มีเปอร์เซ็นต์ความผิดพลาดสูงสุด

5.3.3.3 การทดลองการคำนวณหาระยะทางระหว่างป้ายจราจรกับพาหนะและส่งข้อมูลไปให้ส่วนตัดสินใจผ่านทีซีพีซ็อคเก็ต

ในการทดลองนี้เราได้ทำการหาค่าระยะทางระหว่างพาหนะกับป้ายจราจรเมื่อเราสามารถตรวจพบและวิเคราะห์ป้ายจราจรได้โดยในขั้นตอนของการส่งค่าเราจะทำการกรองข้อมูลก่อนส่งซึ่งก่อนจะส่งนั้นเราจะทำการส่งค่าหมายเลขของป้ายจราจรที่วิเคราะห์ได้และระยะห่างของพาหนะและป้ายจราจร โดยเราจะสามารถหาค่าระยะทางได้จากสมการที่ 5.1

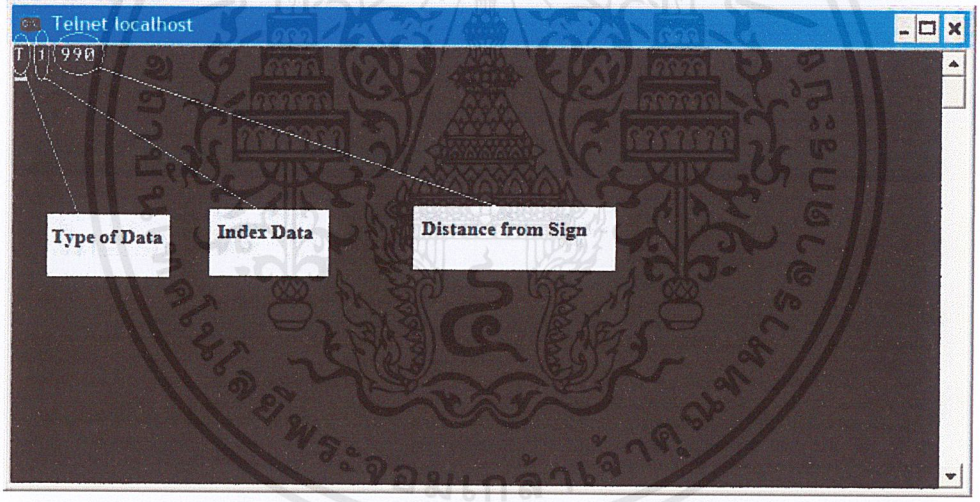
$$\text{ระยะทาง} = 1250 - [10 \times (\text{ความกว้างของภาพที่หาได้} - 25)] \quad (5.1)$$

โดยค่าของระยะห่างที่หาได้มีหน่วยเป็นเซนติเมตรซึ่งเราจะทำการติดต่อเพื่อขอรับข้อมูลผ่านทางทีซีพี ซ็อคเก็ต ดังรูปที่ 5.16



รูปที่ 5.16 ขั้นตอนการติดต่อขอรับข้อมูลผ่านทางทีซีพี ซ็อคเก็ต

โดยเมื่อมีการวิเคราะห์ข้อมูลเสร็จและผ่านการกรองเรียบร้อยแล้วข้อมูลจะถูกส่งมาให้ ดังรูปที่ 5.17



รูปที่ 5.17 ข้อมูลที่วิเคราะห์ได้และส่งมาให้ผู้ติดต่อ

จากภาพจะเห็นได้ว่าข้อมูลที่ส่งมามีทั้งหมด 3 ข้อมูลโดยข้อมูลตัวแรกคือ ชนิดของข้อมูลซึ่งในที่นี้ตัว T หมายถึงการส่งข้อมูลของการหาป้ายจราจร ข้อมูลตัวที่ 2 คือหมายเลยของป้ายจราจรที่วิเคราะห์ได้และข้อมูลตัวสุดท้ายคือระยะห่างของพาหนะกับป้ายจราจรโดยมีหน่วยเป็นเซนติเมตร

5.3.3.4 ผลการทดลองในส่วนของการหาและวิเคราะห์ป้ายจราจร

ในส่วนของผลการทดลองนี้เราจะนำวิดีโอที่เก็บมามาทำการทดสอบและดูว่าเราสามารถหาป้ายเจอได้ตั้งแต่ระยะเท่าใดของแต่ละป้ายและสามารถวิเคราะห์ได้ถูกต้องเป็นจำนวนกี่เปอร์เซ็นต์ของค่าที่วิเคราะห์ออกมาทั้งหมด ซึ่งจะแสดงผลดังตารางที่ 5.2

ตารางที่ 5.2 ผลการหาและวิเคราะห์ป้ายจราจร

ลำดับ / ป้ายที่มีในสถาบัน	ป้ายลำดับที่ 0	ป้ายลำดับที่ 1	ป้ายลำดับที่ 2	ป้ายลำดับที่ 5
เจอ 0 ป้ายเลี้ยวซ้าย	14	2	0	4
เปอร์เซ็นที่หาได้	77.78%	10%	0%	18.18%
เจอ 1 ป้ายเลี้ยวขวา	4	15	0	5
เปอร์เซ็นที่หาได้	22.33%	75%	0%	22.72%
เจอ 2 ป้ายสามแยกตัวที	0	1	0	3
เปอร์เซ็นที่หาได้	0%	5%	0%	13.63%
เจอ 3 ป้ายบอกสี่แยก	0	0	0	0
เปอร์เซ็นที่หาได้	0%	0%	0%	0%
เจอ 4 ป้ายบอกสามแยกทางซ้าย	0	0	0	2
เปอร์เซ็นที่หาได้	0%	0%	0%	9.09%
เจอ 5 ป้ายบอกสามแยกทางขวา	0	2	0	8
เปอร์เซ็นที่หาได้	0%	10%	0%	36.36%

ดังที่เห็นในตารางแสดงให้เห็นว่าการหาป้ายจราจรนั้นป้ายที่สามารถหาและวิเคราะห์อย่างถูกต้องและมีผลส่งกลับไปยังผู้ขอข้อมูลคือป้ายที่ 0 และ 1 ซึ่งทางป้ายที่ 5 นั้นก็สามารถหาและวิเคราะห์ได้แต่เนื่องจากไม่ผ่านการกรองเพราะข้อมูลที่วิเคราะห์มีค่าใกล้เคียงกันมากเพราะในภาพนั้นแสงแดดทำให้วัตถุที่อยู่ภายในภาพบางส่วนมีสีเดียวกันกับป้ายและวัตถุนั้นมีขนาดใหญ่กว่าป้ายและในอัลกอริทึมการหาป้ายของเราก็จะดึงเอาส่วนที่มีขนาดใหญ่สุดเมื่อทำการแบ่งช่วงสี ออกมาจึงนำภาพออกมาผิดทำให้การวิเคราะห์ผิดพลาดไป ส่วนป้ายหมายเลข 2 นั้นไม่สามารถหาป้ายในภาพออกมาได้เนื่องจากขณะวิ่งนั้นพาหนะของเราได้วิ่งย้อนแสงทำให้ภาพที่รับเข้ามานั้นมีคุณภาพ

ของสีภายในภาพมืดและรายละเอียดไม่พอจึงไม่สามารถหาส่วนที่เป็นป้ายจราจรออกมาได้โดยจะได้ค่าระยะทางห่างจากป้ายเป็นดังตารางที่ 5.3

ตารางที่ 5.3 ข้อมูลระยะห่างจากป้ายที่ส่งมา

ป้ายที่หาเจอ	ค่าระยะห่างจากป้ายที่ส่งข้อมูลให้
0 ป้ายบอกเลี้ยวซ้าย	1130
1 ป้ายบอกเลี้ยวขวา	1130
2 ป้ายบอกทางสามแยกรูปตัว T	ไม่มีการส่งข้อมูล
5 ป้ายบอกทางสามแยกด้านขวา	ไม่มีการส่งข้อมูล

จากผลการทดลองของตารางที่ 5.3 แสดงให้เห็นว่าระยะทางที่เราสามารถหาและวิเคราะห์ได้ของป้ายจราจรที่เราหาเจอเป็นระยะ 1130 เซนติเมตร ส่วนอีกทั้งสองป้ายนั้นเนื่องจากเราไม่สามารถระบุได้ชัดเจนว่าเป็นป้ายนั้นได้จริงตามข้อกำหนดจึงไม่ได้มีการส่งข้อมูลออกไป

5.4 ผลการทดลองระบบโดยรวม

จากการทดลองพบว่ารถสามารถหาเส้นทางที่เหมาะสมไปยังปลายทางได้ สามารถเคลื่อนที่ไปยังปลายทางได้ แต่ยังต้องอาศัยการปรับแก้ตำแหน่งจากผู้ใช้อยู่บ้าง ค่าตำแหน่งที่คลาดเคลื่อนไปนั้นอยู่ที่ระยะประมาณ 2-3 เมตรต่อการวิ่งระยะ 100 เมตร รถมีการเลือกทิศทางการกลับรถที่เหมาะสม โดยในบางกรณีต้องอาศัยข้อมูลที่ถูกต้องของส่วนประมวลผลภาพด้วย ระหว่างการเดินรถถ้าหากว่าสีของพื้นถนนไม่สม่ำเสมอ เช่นในกรณีที่มีเงาหรือมีแดดบนพื้นถนนเพียงบางส่วน จะทำให้ระบบประมวลผลภาพ ประมวลผลได้ข้อมูลที่ไม่ถูกต้อง ทำให้รถมีการวิ่งส่ายซ้ายขวาอยู่บ้าง แต่อย่างไรก็ตามรถก็ยังคงรักษาเส้นทางต่อไปได้โดยใช้ข้อมูลจากส่วนอื่นๆ

รถสามารถตรวจสอบพบสิ่งกีดขวางได้เฉพาะในบางกรณีเนื่องจากข้อจำกัดของมอดูลโซนาส์ ซึ่งส่งผลให้การหลบหลีกสิ่งกีดขวางทำได้ไม่เป็นที่น่าพอใจ

บทที่ 6
ข้อสรุปและข้อเสนอแนะ

6.1 ข้อสรุป

ระบบสามารถทำงานได้ดีในระดับหนึ่ง โดยสามารถขับเคลื่อนไปยังเป้าหมายด้วยเส้นทางที่เหมาะสมที่สุดได้ แต่ยังต้องอาศัยการปรับแก้ตำแหน่งรถที่ถูกต้องจริงๆจากผู้ใช้งานในบางกรณี รถสามารถหลบหลีกสิ่งกีดขวางหรือหยุดเมื่อตรวจพบสิ่งกีดขวางได้ดีในระดับหนึ่ง

6.2 ปัญหาและอุปสรรค

6.2.1 ปัญหาเกี่ยวกับฮาร์ดแวร์

- เซอร์โวสำหรับควบคุมล้อหน้า มีความไม่แม่นยำ สามารถแก้ไขได้โดยการเทียบวัดความผิดพลาดและปรับแก้โดยส่วนซอฟต์แวร์
- เซ็นเซอร์วัดระยะทางมีความละเอียดน้อยไป ทำให้การตรวจวัดระยะทางและหาตำแหน่งทำได้ไม่แม่นยำ สามารถแก้ไขได้โดยการออกแบบเซ็นเซอร์วัดระยะทางให้มีความละเอียดมากขึ้นกว่านี้
- ขนาดของล้อหน้าเปลี่ยนแปลงไปตามปริมาณลมในล้อและน้ำหนักของรถ ทำให้การวัดระยะทางด้วยเซ็นเซอร์วัดระยะทางที่ติดอยู่บนล้อหน้ามีความผิดพลาด สามารถแก้ไขได้โดยการวัดขนาดของล้อหน้าใหม่ทุกครั้งก่อนการใช้งาน
- ความเร็วของรถขึ้นอยู่กับน้ำหนักของรถและแรงดันของแบตเตอรี่ จึงทำให้ควบคุมความเร็วรถให้มีค่าตามที่ต้องการได้ยาก สามารถแก้ไขโดยการเขียนซอฟต์แวร์ให้ปรับกำลังขับเคลื่อนรถตามความเร็วที่อ่านได้
- เข็มทิศมีความผิดพลาดเมื่อถูกรบกวนจากสิ่งรอบข้าง หรือวางไว้ใกล้โลหะ หรือวางไม่ขนานกับพื้นโลก สามารถแก้ไขได้โดยการเทียบวัดความผิดพลาดและปรับแก้โดยส่วนซอฟต์แวร์
- จีพีเอสจะส่งค่าตำแหน่งซ้ำๆออกมาเรื่อยๆ เมื่อจีพีเอสเข้าสู่สภาวะคงที่ทั้งๆที่มีการเปลี่ยนตำแหน่งรถแล้ว

- โซนาร์ไม่สามารถตรวจสอบวัตถุที่มีมุมตกกระทบที่ทำให้เกิดการสะท้อน ของคลื่นโซนาร์ได้ สามารถแก้ไขปัญหาให้เกิดขึ้นน้อยลงโดยการตรวจสอบหลายๆครั้ง
- ในการทำงานของกล้อง 2 ตัวถ้าใช้กล้องจากบริษัทเดียวกันที่เป็นรุ่นที่ใช้ไดร์เวอร์เดียวกันแล้วนำ มาใช้กับตัวบอร์ดอัลตราไคแอน (UltraClient) ซึ่งยูเอสบี (USB) บนตัวบอร์ดนั้นมีลักษณะต่อกันแบบฮับ (USB Hub) ซึ่งจะทำให้ไม่สามารถเปิดกล้อง 2 ตัวพร้อมกันได้เนื่องจากมีการแชร์ไดร์เวอร์กันในยูเอสบีพอร์ตเดียวกัน

6.2.2 ปัญหาเกี่ยวกับซอฟต์แวร์

- ทำการทดลองระบบตัดสินใจได้ไม่สะดวก เนื่องจากการทดลองแต่ละครั้งจะต้องเสียเวลาในการเคลื่อนย้ายรถ สามารถแก้ปัญหาได้โดยการใช้การจำลองรถเพื่อทดสอบระบบตัดสินใจ แต่การทดสอบโดยการจำลองก็มีความไม่เหมือนจริงอยู่บางส่วน เช่น ไม่สามารถจำลองข้อมูลที่ได้จากระบบประมวลผลภาพได้สมจริง ไม่สามารถจำลองข้อมูลที่ได้จากจีพีเอสได้สมจริง
- กล้องเวปแคม (Web Cam) นั้นถ้าเราใช้กล้องที่เป็นไดร์เวอร์ (Driver) รุ่นใหม่ๆจะทำให้วินโดวส์มองเห็นกล้องนั้นเป็นอุปกรณ์ชิ้นหนึ่งของบริษัทนั้น จึงไม่สามารถใช้ไดร์เวอร์วีเอฟดับบิว (VFW driver) ได้ซึ่งจะทำให้โปรแกรมโอเพนซีวีไม่สามารถเปิดใช้งาน 2 กล้องได้

6.2.3 ปัญหาเกี่ยวกับสิ่งแวดล้อม

- พื้นถนนในบางจุดไม่เรียบ ทำให้ค่าที่อ่านได้จากเซ็นเซอร์วัดระยะทางมีความผิดพลาด และเกิดการโยกของรถทำให้เข็มทิศอิเล็กทรอนิกส์ไม่ขนานกับพื้นโลก ซึ่งเป็นสาเหตุให้เข็มทิศอิเล็กทรอนิกส์บอกค่าองศาผิดพลาด
- พื้นที่ที่ใช้ทดสอบมีรถวิ่งผ่านในบางจังหวะ ทำให้การทดสอบไม่ต่อเนื่อง

เนื่องจากในการทดลองให้รถเดินทางตามถนนของจริงและสภาวะแวดล้อมจริง สิ่งแวดล้อมจึงมีผลอย่างมากกับผลการทดลองโดยเฉพาะในส่วนของการประมวลผลภาพ ซึ่งปัจจัยที่ส่งผลต่อความถูกต้องของข้อมูลจากการประมวลผลภาพ มีดังนี้

- สภาพแวดล้อมบริเวณรอบๆถนน เช่นขอบทางเดิน หรือลูกระนาดเป็นต้น เนื่องจากมีการใช้ color threshold เป็นหลัก หากสีของสิ่งที่ไม่ใช่ถนนมีส่วนคล้ายกับถนนจะทำให้ผลการทดลองมีโอกาสผิดพลาดได้ง่าย

- แสงแดดก็เป็นปัจจัยสำคัญเนื่องจากบางบริเวณเมื่อโดนแสงจะสะท้อน ทำให้สีผิดเพี้ยนหรือสว่างขาวมากเกินความจริง
- เงาจากสิ่งต่างๆเช่น ตึกหรือต้นไม้ทำให้ถนนมีหลายสีทั้งส่วนที่มีเงาและไม่มีเงาซึ่งทำให้เกิดความผิดพลาดได้มาก
- สีของถนน บางพื้นที่ที่ทำการทดลองถนนบางที่มีสีที่ไม่แน่นอน ไม่เรียบกลืนเป็นสีเดียวกันทำให้หาถนนไม่เจอ
- ป้ายจราจรบางป้ายอาจจะเอียง โดนบังหรือโดนทำให้บิดเบี้ยวไปจากปกติซึ่งทำให้การหาป้ายจราจรหาไม่เจอในบางกรณี
- ในกรณีการวิ่งย้อนแสงจะทำให้ภาพที่ได้มีค่าของสีผิดเพี้ยนไปทำให้การหาส่วนป้ายออกจากภาพที่รับมานั้นทำได้ยากและมีข้อผิดพลาดสูง
- ในภาพที่รับเข้ามีส่วนที่มีสีเหมือนกับป้ายและมีขนาดใหญ่กว่าขนาดของป้ายทำให้ การดึงภาพส่วนที่หาออกมาผิดพลาดได้จึงไม่สามารถวิเคราะห์ป้ายที่ถูกต้องได้

6.3 ข้อเสนอแนะ

- ในการติดตั้งอุปกรณ์ฮาร์ดแวร์ควรวางแผนและคำนวณวัดระยะให้แน่นอนก่อนลงมือติดตั้งและออกแบบรวมทั้งระบบ เนื่องจากอุปกรณ์บางอย่างติดตั้งแล้วไม่สามารถแก้ไขได้
- ในการเขียนซอฟต์แวร์ที่มีไอซีเกี่ยวข้องด้วย ถ้าหากว่าพบปัญหาควรจะตรวจสอบหาสาเหตุของปัญหาจากส่วนซอฟต์แวร์ก่อน ไม่ควรไปตรวจสอบหาสาเหตุของปัญหาจากไอซีเพราะโอกาสที่ไอซีจะเสียมีน้อยมาก
- ในการทำงานที่ต้องอาศัยเฟรมเวิร์คควรจะศึกษาเฟรมเวิร์คนั้นๆให้ดีเสียก่อนใช้ เพราะปัญหาบางอย่างเกิดเพราะผู้ใช้งานไม่รู้จักกระบวนการทำงานหรือ สถาปัตยกรรมของเฟรมเวิร์คนั้นดีพอ
- ก่อนเริ่มต้นทำงานตามที่ออกแบบ ควรจะสละเวลามาตรวจสอบก่อนว่าสิ่งที่ออกแบบมานั้นมีจุดวิกฤตที่ใดบ้าง และจุดวิกฤตนั้นสามารถทำได้ในทางปฏิบัติจริงๆ
- การทดสอบการทำงานควรทดสอบส่วนย่อยๆก่อน ก่อนที่จะนำมารวมกันเป็นระบบใหญ่เพื่อให้สามารถหาสาเหตุของปัญหาได้ง่าย

- โปรโตคอลในการติดต่อสื่อสารระหว่างระบบต่างๆควรออกแบบและทำเป็นเอกสารให้เรียบร้อยก่อนการเริ่มทำงาน
- ในการเขียนโค้ดส่วนของการทำการประมวลผลภาพควรคำนึงถึงความเร็วในการทำงานเพื่อให้โปรแกรมทำงานให้ทันเวลา
- ควรปรับค่ากล้องให้ดีก่อนที่จะทำการให้รถวิ่งทุกครั้ง
- ในการวางแผนปฏิบัติงานควรเผื่อเวลาในการทดลองให้มากพอ เพราะในการทดลองจริงมักมีปัญหาเฉพาะหน้าให้แก้ไข
- ควรใช้กล้องที่ไม่มีการใช้งานไดร์เวอร์(Driver) ร่วมกันบนเครื่องอัลตราไคแอน (UltraClient) เพราะจะทำให้ไม่สามารถเปิดกล้องทั้ง 2 ตัวพร้อมกันได้
- ควรใช้งานกล้องรุ่นเก่าที่สามารถทำงานบนไดร์เวอร์วีเอฟดับบิว (VFW:Video For Windows Driver) ได้เพราะจะทำให้สามารถใช้ไลบรารี่โอเพนซีวี (OpenCV library) ในการเปิดกล้อง 2 ตัวได้

บรรณานุกรม

ณันธนา ตระการรัตนกุล และ เสฏฐวุฒิ ขันต้นธง. 2549. "รถยนต์ขับเคลื่อนอัตโนมัติ." วิทยานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรม คอมพิวเตอร์ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

Rafael, C. G. and Richard, E. W. 2002. **Digital Image Processing.** 2nd ed. New Jersey: Prentice-Hall.

Newman, D.jay. 2000. **Linux Robotics**.

Player wiki. [online] Available : http://playerstage.sourceforge.net/wiki/Main_Page

Roland Siegwart and Illah R. NourBakhsh, 2004. **Introduction to Autonomous Mobile Robots.** London : The MIT Press.

Wikipedia. 2001. Mathematical morphology. [Online] Available : http://en.wikipedia.org/wiki/Mathematical_morphology

OpenCVWiki. 2006. OpenCV. [Online] Available : http://opencv.willowgarage.com/wiki/

Boost. 2008. Boost 1.37.0 Library Documentation. [Online] Available : http://www.boost.org/doc/libs/1_37_0/

Microchip.2007.MCP2515.[Online] Available : http://ww1.microchip.com/downloads/en/DeviceDoc/21801e.pdf

Rick Stoneking.2002.A Simple CAN Node using the MCP2510 and PIC12C67X.[Online] Available : http://ww1.microchip.com/downloads/en/AppNotes/00215b.pdf

Microchip.2007.MCP2551.[Online] Available : http://ww1.microchip.com/downloads/en/DeviceDoc/21667E.pdf

Pat Richards.2005.A CAN Physical Layer Discussion.[Online] Available : http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en012057

Microchip.2008.PIC16F/882/883/884/886/887.[Online]

Available : http://www.microchip.com/stellent/idcplg?

IdcService=SS_GET_PAGE&nodeId=1824&appnote=en012057

National Semiconductor.2004.LM2576.[Online]

Available : http://www.national.com/ds/LM/LM2576.pdf

ON Semiconductor.2008.MC7800, MC7800A, MC7800AE, NCV7800.[Online]

Available : http://www.onsemi.com/pub_link/Collateral/MC7800-D.PDF

Robot-electronics.2008.CMPS-Robot Compass Module.[Online]

Available : http://www.robot-electronics.co.uk/htm/cmps3doc.shtml

Maxbotix.2007.LV-MaxSonar-EZ3 Datasheet.[Online]

Available : http://www.maxbotix.com/uploads/LV-MaxSonar-EZ3-Datasheet.pdf

Klaus Betke.2000.The NMEA 0183 Protocol.[Online]

Available : http://www.tronico.fi/OH6NT/docs/NMEA0183.pdf