รถวิ่งตามเลนถนน

(Lane Detection for Vehicle)

โดย

นายประทีป เลาวกุล

นายนัฐพงค์ พันธุ์ขันคำ

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

รถวิ่งตามเลนถนน

(Lane Detection for Vehicle)

โดย

นายประทีป เลาวกุล เลขประจำตัว 43010243

นายนัฐพงศ์ พันธุ์ขันคำ เลขประจำตัว 43010210

อาจารย์ที่ปรึกษา

รศ.ดร. สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง รถวิ่งตามเลนถนน

จัดทำโดย

1. นาย ประทีป	เลาวกุล	43010243
2.นาย นัฐพงศ์	พันธุ์ขันคำ	43010210

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

..... อาจารย์ที่ปรึกษา

(รศ.ดร. สุรพันธ์ เอื้อไพบูลย์)

รถวิ่งตามเลนถนน

LANE DETECTION FOR VEHICLE

นายประทีป เลาวกุล 43010243

นายนัฐพงศ์ พันธุ์ขันคำ 43010210

โครงงานได้รับการตรวจสอบแล้วพร้อมที่จะทำการสอบได้

..... อาจารย์ที่ปรึกษา

(รศ.ดร. สุรพันธ์ เอื้อไฟบูลย์)

รถวิ่งตามเลนถนน

นาย ประทีป เลาวกุล 43010243
นาย นัฐพงค์ พันธุ์ขันคำ 43010210
รศ.ดร. สุรพันธ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ในการพัฒนาระบบการตรวจจับเลนถนนของระบบขับเคลื่อนใดๆ นับเป็นงานที่น่าสนใจแต่มีความยากยิ่งในการประมวลผลภาพ โครงงานนี้จึงเป็นเสมือนแบบจำลองการทำงานของระบบโดยจะรับภาพจากกล้องดิจิตอลที่ติดตั้งอยู่บนรถ และนำภาพอย่างหยาบๆ ที่ได้มาทำการประมวลผลเพื่อให้คอมพิวเตอร์ได้ทราบว่าขณะนั้นระบบขับเคลื่อนของเราอยู่ในสถานการณ์เช่นไร จากนั้นส่งผลการประมวลผลที่ได้ไปยังไมโครคอนโทรลเลอร์ ซึ่งการขับเคลื่อนของรถจำลองจะอาศัยการควบคุมความเร็วด้วยสเต็ปเปอร์มอเตอร์ ทำให้รถสามารถเคลื่อนที่ไปได้อัตโนมัติ พร้อมทั้งตรวจจับเลนของถนนไปในขณะเดียวกัน

Lane Detection for Vehicle

Mr. Prateep Laovakun 43010243

Mr. Nuttapong Phantkankum 43010210

Assoc.Prof.Dr. Surapan Airphaiboon (Advisor)

2nd Semester, 2003

Abstract

In the developing of a vision based on road detection system is very interesting to cope with but we are facing difficulty in a process to give the picture out. This project uses as a model to display system's function by taking the image from digital camera on vehicle. The image is then sent to the microcontroller to process. The vehicle movement is controlled by stepper motor. So the vehicle can move automatically and can detect the lane simultaneously.

สารบัญ

สารบัญรูปภาพ

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

เนื่องจากการศึกษาการตรวจจับเลนถนน เป็นเทคโนโลยีที่อาจจะเรียกได้ว่าใหม่สำหรับการขับเคลื่อนยานพาหนะระบบอัตโนมัติ ซึ่งปัจจุบันมีนักวิจัยหลายท่านกำลังศึกษาถึงแนวทางที่ดีที่สุด ซึ่งในทางคอมพิวเตอร์ก็คือการใช้เวลาในการประมวลผลให้น้อยที่สุด ซึ่งส่งผลให้สามารถตอบสนองต่อสภาวะต่างๆ ได้ทันท่วงที ในปัจจุบันได้มีการทดลองควบคุมการขับเคลื่อนยานพาหนะโดยใช้การประมวลผลของคอมพิวเตอร์จะทำได้ โดยต้องอาศัยกล้องที่มีประสิทธิภาพสูงและคอมพิวเตอร์ต้องมีความเร็วในการประมวลผลสูง ซึ่งในการประมวลผลภาพจะมีรายละเอียดขั้นตอนดังต่อไปนี้

1.) สร้างตัวกำหนดตำแหน่งของเลนถนน เพื่อใช้ในการกำหนดขอบเขตของเลนถนนโดยปราศจากสัญญาณรบกวนหรือวัตถุอื่นบนถนน

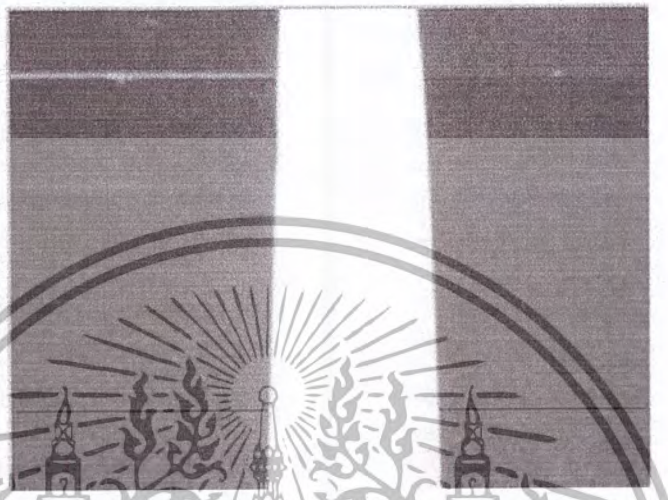
2.) การแบ่งส่วนย่อยต่างๆ ของภาพที่ได้จากถนน

3.) ขั้นตอนการประมาณรูปแบบของเลนถนน ประกอบด้วย พิกัดของจุดกึ่งกลาง ความโค้งของถนน จำนวนของเลนถนน และตำแหน่งของรถยนต์บนเลนถนน

รูปที่ 1.1 เลนถนนที่จะทำการตรวจจับ

1.2 การหาขอบเขตของถนนในเวลาจริง

เราจะวิเคราะห์ส่วนที่ใช้ในการหาขอบเขตของถนนในเวลาจริง การหาทิศทางของถนน และตำแหน่งของยานพาหนะ เพื่อนำไปใช้เป็นข้อมูลในการขับมอเตอร์ ในการหาข้อมูลในเวลาจริง เราจะสนใจเฉพาะสภาพแวดล้อมของถนน และจะประมวลผลโดยใช้ภาพขาวดำ



รูปที่ 1.2 เลนถนนที่ผ่านการดีเทคและจะนำไปประมวลผล

1.3 ตัวกำหนดตำแหน่งของเลนถนน

จากข้างต้นเราจะพบว่ามีสิ่งแวดล้อมต่างๆ เข้ามาเกี่ยวข้องในภาพของเลนถนน เช่น ปริมาณของแสง และสิ่งก่อสร้างต่างๆ เราจึงต้องพัฒนาระบบที่จะหาตำแหน่งของถนนในสภาวะที่ต่างกันได้ ซึ่งเมื่อเรานำภาพผ่านตัวกรองความถี่ต่ำผ่าน จะได้เซ็ทของจุดต่างๆ ที่ขึ้นกับขอบถนน ตัวกำหนดตำแหน่งของเลนถนนจะปรากฏในรูปจุดสองจุดที่มีเกรเดียนท์สูงมาก คือจุดทั้งสองแทนการเปลี่ยนจากบริเวณมืดเป็นบริเวณสว่าง

รูปที่ 1.3 ภาพที่นำเข้าไปเป็นอินพุทและเอาท์พุทที่ได้จากการประมวลผล

บทที่ 2

ระบบขับเคลื่อน (Driving System)

ระบบขับเคลื่อนในโครงงานนี้จะใช้สเต็ปเปอร์มอเตอร์ เพราะสเต็ปเปอร์มอเตอร์มีคุณสมบัติแตกต่างจากมอเตอร์ทั่วๆ ไป โดยเมื่อป้อนกำลังไฟฟ้าให้กับมัน มันจะหมุนเพียงเล็กน้อยตามเส้นรอบวงหยุดในขณะที่มอเตอร์ทั่วๆไปจะหมุนทันทีและหมุนทันทีและหมุนตลอดเวลาตราบเท่าที่ยังมีพลังงานจ่ายให้แก่ตัวมอเตอร์สเต็ปเปอร์มอเตอร์สามารถกำหนดตำแหน่งของการหมุนได้อย่างละเอียด โดยการใช้คอมพิวเตอร์เป็นตัวกำหนดและจัดเก็บตัวเลขข้อมูลของการหมุนเหล่านั้นไว้

สเต็ปเปอร์มอเตอร์ สามารถใช้งานในระบบเปิด (open system) นั้นคือมันสามารถทำงานได้โดยไม่ต้องการการป้อนกลับ (feedback) แต่ในการกำหนดตำแหน่งให้ถูกต้องจำเป็นที่จะต้องมีการป้อนกลับข้อมูลกลับไปยังระบบควบคุมให้รับรู้เพื่อตรวจสอบว่าตำแหน่งถูกต้องหรือ เกิดการผิดพลาด (error) หรือไม่วิธีหนึ่งที่ใช้กันโดยทั่วไปคือ การใช้ลิมิตสวิตช์ (limit switch) โดยติดตั้งไว้ที่ตำแหน่งที่ต้องการตรวจจับเมื่อสเต็ปเปอร์มอเตอร์เริ่มหมุนและหมุนจนกระทั่งถึง ตำแหน่งของสวิตช์ตรวจจับ สัญญาณก็จะถูกป้อนกลับเข้าสู่ระบบทำให้ทราบการทำงานของสเต็ปเปอร์มอเตอร์ได้ตลอดเวลา ซึ่งโดยปกติในวงจรควบคุมจะมีการกำหนดจุดอ้างอิง (reference point) ไว้ด้วยเพื่อให้การทำงานไม่ผิดพลาด ตัวอย่างง่ายๆ เช่น ถ้าเริ่มจ่ายกำลังไฟฟ้าให้กับฟลอปปี้ดิสก์ไดรฟ์จะได้ยินเสียงของมอเตอร์กำลังเคลื่อนที่เพื่อหาจุดอ้างอิงที่กำหนด หลังจากนั้นวงจรไดรฟ์คอนโทรลเลอร์จะเริ่มทำงานได้ โดยมันจะทราบถึงทุกๆ สเต็ปที่ทำการขับเคลื่อนหัวอ่าน/เขียนไปยังแต่ละแทร็กบนดิสก์

เช่นเดียวกับมอเตอร์ทั่วไป การที่จะทำให้เกิดการหมุนของโรเตอร์ (rotor) ได้ต้องมีการกระทำของสนามแม่เหล็กที่เกิดขึ้นระหว่างโรเตอร์และสเตเตอร์ (stator) ซึ่งขึ้นอยู่กับการจัดวางขั้วแม่เหล็ก(pole) การหมุนทำได้ทั้งแบบต่อเนื่องและกลับทิศทางไปมาโดยกระบวนการทางไฟสลับ, การจัดวางแปรงถ่าน, การจัดแยกคอมมิวเตเตอร์ และทำการสวิตชิ่งพลังงานไฟฟ้าเพื่อให้เกิดแรงดึงดูดของแม่เหล็ก (magnetic attraction) ที่ขั้วแม่เหล็กสร้างและหยุดสลับกัน ผลก็คือเกิดสนามแม่เหล็กหมุนขึ้นบนสเตเตอร์ โดยการจ่ายกำลังไฟฟ้าทีละคู่ของขั้วแม่เหล็กในทิศทางตรงกันข้ามไปตลอดเวลา และเมื่อต้องการให้หยุดหมุนทำได้โดยการเกิดขั้วแม่เหล็กที่จุดหนึ่งโดยหยุดหมุนทำได้โดยหยุดการสวิตชิ่งในลำดับต่อไปเสีย การหมุนกลับทิศทางก็ทำได้เช่น เดียวกับที่กล่าวมาแล้วเพียงแต่ทำการสวิตชิ่งกำลังไฟฟ้าให้เกิดสนามแม่เหล็กหมุนในทิศทางกลับกัน หรือกลับลำดับการสวิตช์ของมัน

โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ประกอบขึ้นจากแผ่นเหล็กวงแหวน ที่มีซี่ยื่นออกมาแต่ละซี่เหล่านั้นจะมีคอยล์พันสวมอยู่ ดังนั้น เมื่อป้อนกระแสไฟฟ้า ผ่านคอยล์ทำให้เกิด

สนามแม่เหล็กไฟฟ้าขึ้นด้านตรงข้ามของแต่ละขั้วแม่เหล็กจะได้รับ กระแสไฟฟ้าในเวลาเดียวกันแต่ว่าจะไหลวนในทิศทางตรงกันข้าม ทำให้เกิดสนามแม่เหล็กไฟฟ้าในทิศทางตรงข้ามขึ้น ดังนั้นถ้าเพิ่มจำนวนของขั้วแม่เหล็กมากขึ้น จะเป็นการเพิ่มจำนวนของสเต็ปเปอร์ต่อรอบมากขึ้นจะเป็นการเพิ่มจำนวนของสเต็ปต่อรอบมากขึ้นตามไปด้วย

อย่างไรก็ตามผู้ใช้งานสามารถเพิ่มจำนวนของสเต็ปได้อีกวิธีหนึ่ง โดยไม่ต้องปรับเปลี่ยนโครงสร้างภายในโดยทำการจ่ายกำลังไฟฟ้าไปยังขั้วแม่เหล็ก 2 ขั้วที่อยู่ใกล้กันในเวลาเดียวกัน ซึ่งจะทำให้โรเตอร์หยุดหมุนอยู่ระหว่างกลางของ 2 ขั้วแม่เหล็กนั้น หรือเคลื่อนที่ไปครึ่งสเต็ปเท่านั้น และวิธีการนี้ยังช่วยให้เกิดแรงบิต (torque) มากขึ้นด้วย

สเต็ปเปอร์มอเตอร์โดยทั่วไปมีจำนวนของขั้วแม่เหล็กหรือจำนวนสเต็ปต่อรอบเป็นจำนวนมาก ปกติอยู่ที่ประมาณ 100-400 สเต็ปต่อรอบ การมีจำนวนสเต็ปมากๆ นี้ไม่ได้เพิ่มที่จำนวนขั้วแม่เหล็กไฟฟ้าที่สเตเตอร์ แต่ทำได้โดยเพิ่มจำนวนซี่ขั้วแม่เหล็กที่โรเตอร์ จำนวนสเต็ปต่อรอบทั้งหมดจะได้จากการคูณจำนวนขั้วแม่เหล็กบนสเตเตอร์และจำนวนซี่ที่โรเตอร์ ดังเช่นถ้ามีขั้วแม่เหล็ก 3 ขั้วบนสเตเตอร์ และ 8 ซี่ขั้วแม่เหล็กบนโรเตอร์ สเต็ปเปอร์มอเตอร์ตัวนี้จะทำงานที่ 24 สเต็ปต่อรอบ หรือหมุนเป็นมุม 15 องศาต่อสเต็ป

2.1) การแยกประเภทของสเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์แบ่งตามพื้นฐานได้เป็น 3 ชนิดคือ

1.) วาริเอเบิลรีลักแตนซ์ (variable reluctance : VR)

2.) เพอร์มาเนนต์แม็กเนต (permanent magnet : PM)

3.) ไฮบริด (hybrid)

2.1.1 ชนิดวาริเอเบิลรีลักแตนซ์ มีโครงสร้างของโรเตอร์แบบมัลติทูธ (multitooth) ทำจากเหล็กอ่อน การทดสอบเพื่อให้ทราบว่าเป็นมอเตอร์ชนิดนี้ทำได้ง่ายมากคือ ใช้นิ้วหมุนเพลาของมอเตอร์มอเตอร์ชนิดนี้ที่โรเตอร์จะไม่เกิดปรากฏการณ์ทางแม่เหล็กที่โรเตอร์ เมื่อหมุนจะรู้สึกขัดๆ เหมือนเป็นฟันเฟือง สเต็ปเปอร์มอเตอร์ชนิดนี้มีจุดด้อยในเรื่องของความถูกต้องของตำแหน่งและทำงานได้ไม่ดีนักเมื่อมีสเต็ปในการหมุนสูง

2.1.2 ชนิดเพอร์มาเนนต์แม็กเนต มีโครงสร้างของโรเตอร์แบบเรียบไม่มีซี่ขั้วแม่เหล็ก บนโรเตอร์จะเป็นแบบแม่เหล็กถาวรการควบคุมทำได้ โดยป้อนกระแสกระตุ้นที่ขดลวดบนสเตเตอร์แบบ 4 เฟส จะมีขั้วแม่เหล็กอยู่ 4 ขั้วซึ่งมีคอยล์พันอยู่แยกจากกันขั้วแม่เหล็กถาวรบนโรเตอร์จะถูกแรงดึงดูดจากขั้วแม่เหล็กบนสเตเตอร์นั้นถึงแม้ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป ทำให้เกิดเป็นแรงยึดหน่วงขึ้น สเต็ปเปอร์มอเตอร์ชนิดนี้มีข้อดีในเรื่องของความถูกต้องของตำแหน่งเมื่อเปรียบเทียบกับชนิดอื่น

2.1.3 ชนิดไฮบริด เป็นชนิดที่นิยมใช้งานกันมากที่สุดโดยเฉพาะนำมาใช้ในอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ สเต็ปเปอร์มอเตอร์ชนิดนี้มีโครงสร้างภายในคือ สเตเตอร์เป็นชนิดวาริเอเบิลรีลักแตนซ์ ส่วนโรเตอร์เป็นชนิดเพอร์มาเนนต์แม็กเนตนำมาประกอบเข้าด้วยกัน จึงทำให้เป็นมอเตอร์ชนิดที่แรงยึดหน่วงสูง, มีแรงบิดดึงและผลักดี และยังคงทำงานได้ดีแม้ว่าจะมีจำนวนของสเต็ปต่อรอบในการหมุนสูงก็ตาม

ยังมีสเต็ปเปอร์มอเตอร์แบบใหม่อีกชนิดหนึ่ง เป็นชนิดที่ปรับปรุงมาจากชนิดเพอร์มาเนนต์แม็กเนตนั้นคือ ชนิดแรเอิร์ธเพอร์มาเนนต์แม็กเน็ต (rare earth permanent magnet) หรือที่เรียกกันว่าชนิดดิสก์แมกเน็ตสเต็ปเปอร์มอเตอร์ (disc magnet steppers)

2.2) การพันขดลวดหรือคอยล์บนสเต็ปเปอร์มอเตอร์มีอยู่ 2 วิธี

คือ แบบไบ โพลาร์ (Bipolar) และแบบยูนิโพลาร์ (Unipolar)

2.2.1 แบบไบโพลาร์(Bipolar)

สเต็ปเปอร์มอเตอร์แบบไบโพลาร์ มีการพันขดลวด 1 ขดบนแต่ละขั้วแม่เหล็ก ของสเตเตอร์ ขั้วแม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้า และสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามได้โดยการกลับทิศการไหลของกระแสไฟฟ้าทำได้ โดยการใช้วงจรสวิตชิ่งกลับขั้วไฟฟ้า ดังรูป 2.1

รูปที่ 2.1 โครงสร้างอย่างง่ายของสเต็ปเปอร์มอเตอร์แบบไบโพลาร์

2.2.2 สเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์

สเต็ปเปอร์มอเตอร์ได้รับการพัฒนามาอย่างต่อเนื่อง จนในปัจจุบันสเต็ปเปอร์มอเตอร์ที่นิยมใช้อย่างแพร่หลายมากที่สุด และหาได้ง่ายคือ สเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์ (Uni-polar stepper motor) มีสักษณะการพันขดลวดของมอเตอร์แสดงในรูปที่ 2.2

สเต็ปเปอร์มอเตอร์แบบนี้มีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเต็ปเปอร์ แต่ละขดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวจะมี 4 เฟสคือเฟส 1, 2, 3 และ 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเต็ปเปอร์มอเตอร์แบบนี้มีทั้งแบบ 5 สายและ 6 สาย ถ้าเป็นแบบ 5 สายจะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสองมาต่อรวมกันเป็นสายเดียว

รูปที่ 2.2 โครงสร้างอย่างง่ายของสเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์

2.3)การกระตุ้นและควบคุมการหมุนของสเต็ปเปอร์มอเตอร์

การกระตุ้นและการควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้อง ด้วยสามารถแบ่งได้เป็น 3 รูปแบบคือ แบบหนึ่งเฟส,แบบ 2 เฟส (two phase) และแบบครึ่งสเต็ป (half step)

แบบหนึ่งเฟสหรือแบบฟูลสเต็ป (Full step) เป็นการกระตุ้นที่มีรูปแบบง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งไล่เรียงถัดกันไป เช่น เริ่มต้นที่ขดที่ 1,2,3,4 แล้ววนกลับมาขดที่ 1 วนไปเรื่อยๆ หรือเริ่มที่ขดที่ 1 แล้วย้อนไปยังขดที่ 4,3,2 แล้วกลับมายังขดที่ 1 อีกครั้งซึ่งทำให้ทิศทางของการหมุนสวนกัน ในการกระตุ้นรูปแบบนี้จึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 2.1

แบบ 2 เฟสเป็นการกระตุ้นซึ่งคล้ายกับแบบหนึ่งเฟส แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไป เช่นเดียวกับแบบเวฟ ดังตัวอย่าง ขดลวดชุดแรกที่ถูกกระตุ้นจะเป็นขดที่ 1 และ 2 ตามด้วยการกระตุ้นขดที่ 2 และ 3 ต่อไปเป็นขดที่ 3 และ 4 ถัดไปเป็นขดที่ 4 และ 1 แล้วกลับมาที่ขดที่ 1 และ 2 วนไปตามลำดับเช่นนี้หรือเริ่มที่ขด 1 และ 4 ตามด้วยขดที่ 4 และ 3 ถัดไปเป็นขดที่ 3 และ 2 ต่อไปเป็นขดที่ 2 และ 1 แล้ววนกลับมาที่ขดที่ 1 และ 4 ทิศทางการหมุนจะสวนทางกัน การกระตุ้นสเต็ปเปอร์มอเตอร์แบบนี้สามารถเพิ่มแรงบิดได้มากกว่าแบบเวฟโรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน และต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือการกระตุ้นแบบนี้ต้องใช้กำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆ แสดงดังในตารางที่ 2.2

แบบครึ่งสเต็ปเป็นรูปแบบที่ผสมผสานระหว่างการกระตุ้นแบบเวฟและแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับ ดังนี้ เริ่มจากขดลวดที่ 1,1 และ 2,2,2 และ 3,3,3 และ 4,4,4 และ 1 แล้ววนกลับมายังขดลวดที่ 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลง แต่ละสเต็ปเกิดแรงเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังไว้อีกประการหนึ่งว่า เมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2 สเต็ป จึงจะได้เท่ากับระยะเท่ากับ 1 สเต็ปเต็มของการควบคุมใน 2 แบบแรก สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้ขนาดเท่ากับแบบ 2 เฟสเป็นอย่างน้อย จึงจะพอเพียง ขั้นตอนการทำงานต่างๆ แสดงดังในตารางที่ 2.3

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

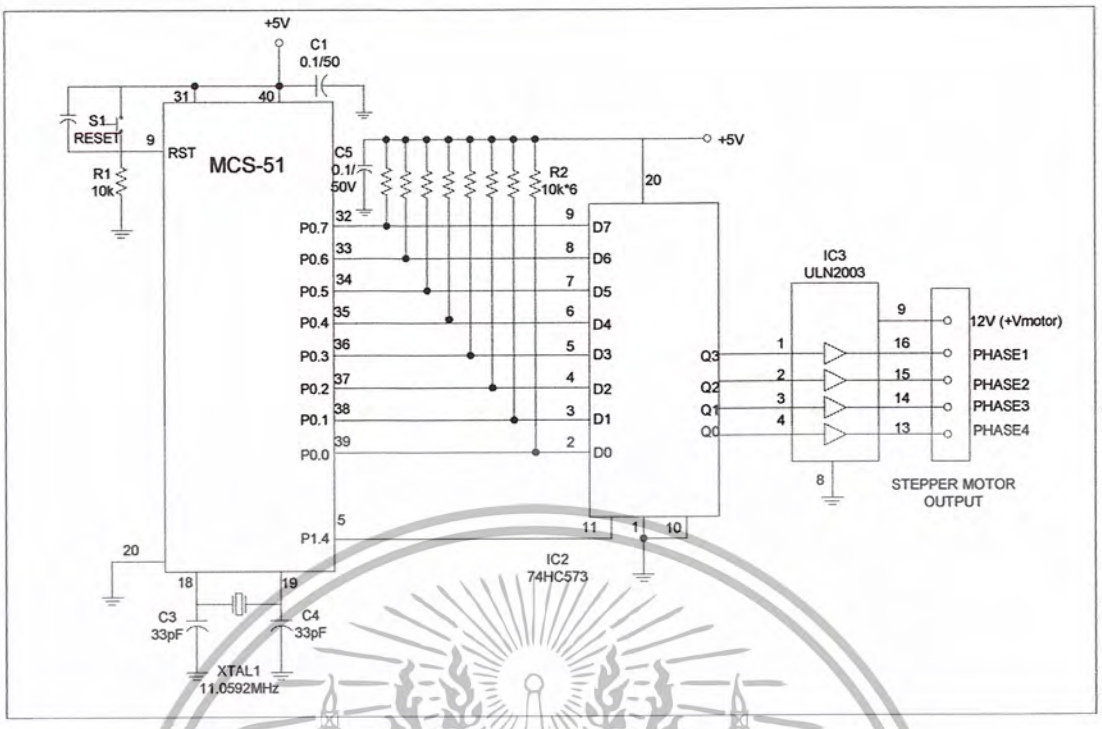
ตารางที่ 2.1 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบหนึ่งเฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

ตารางที่ 2.2 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบสองเฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

ตารางที่ 2.3 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบครึ่งสเต็ป



รูปที่ 2.3 ตัวอย่างวงจรการขับสเต็ปเปอร์มอเตอร์ของไมโครคอนโทรลเลอร์ MCS-51

ตัวอย่างวงจรขับสเต็ปเปอร์มอเตอร์ของไมโครคอนโทรเลอร์ MCS-51

แสดงในรูปที่ 2.3 จะใช้พอร์ต 0 ส่งข้อมูลผ่านไอซีแลตช์ 74HC573 แล้วส่งต่อไปยังไอซีไดรเวอร์กระแสสูงแบบคอลเล็กเตอร์เปิดเบอร์ ULN2003 ด้วยการใช้ไอซีแบบนี้ทำให้สามารถเลือกแรงดันสำหรับขับสเต็ปเปอร์มอเตอร์ได้กว้างขวางตั้งแต่ 5-30 V โดย ULN2003 มีความสามารถในการจ่ายกระแสได้สูงสุด 500 mA ต่อขา ทั้งนี้ต้องเตรียมแหล่งจ่ายไฟให้มีความสามารถในการจ่ายกำลังไฟฟ้าสูงเพียงพอด้วย

เมื่อต้องการให้มอเตอร์หมุนให้ส่งข้อมูล “1” ไล่เรียงไปตามลำดับจาก P0.0 ถึง P0.3 แล้ววนกลับมาไปที่ P0.0 ใหม่ หากต้องการให้มอเตอร์หมุนกลับทิศทางก็ให้ส่งข้อมูลย้อนกลับ โดยเริ่มจาก P0.3 ก่อนแล้วสิ้นสุดรอบที่ P0.0 แล้ววนกลับไปที่ P0.3 ใหม่ เมื่อ ULN2003 ได้รับข้อมูล “1” ก็จะทำการกลับลอจิก ทำให้เกิดกระแสไฟฟ้าไหลผ่านขดลวดของมอเตอร์ที่ต่ออยู่กับขาเอาต์พุตที่ทำงานส่งผลให้เกิดการเคลื่อนที่ของแกนมอเตอร์ขึ้น

2.4 ทฤษฎีและหลักการของพอร์ตอนุกรม

พอร์ตสื่อสารข้อมูลแบบอนุกรม MCS-51 สามารถรับส่งสัญญาณแบบอนุกรมได้โดยไม่ต้องพึ่งอุปกรณ์ภายนอกอื่นๆ แต่อย่างใด ในด้านอัตราเร็วของการรับส่งข้อมูลก็สามารถกำหนดได้ตามความต้องการของผู้ใช้ โดยสามารถเลือกอัตราเร็วในการรับ ส่งข้อมูล (baud rate) มาตรฐานตั้งแต่ 110 bps,1.2 kbps,4.8 kbps,9.6 kbps,19.2 kbps,375 kbps ตามมาตรฐานของ UART

2.4.1 การใช้งานพอร์ตอนุกรมของ MSC-51

โดยปกติแล้ว MCS-51 คือระบบคอมพิวเตอร์ ซึ่งมีความสามารถในการรับส่งข้อมูลจากภายนอกและนำมาประมวลผล พร้อมทั้งนำสัญญาณต่างๆ ออกไปควบคุมอุปกรณ์ภายนอกได้

รีจิสเตอร์ที่เกี่ยวข้องในการใช้งานพอร์ตอนุกรม

1.รีจิสเตอร์ควบคุมไทม์เมอร์ สิ่งที่ต้องคำนึงถึงในการใช้งานพอร์ตอนุกรมคือ อัตราการรับส่งข้อมูลหรืออัตราบอด (Baud rate) คือจังหวะการเลื่อนข้อมูลเข้าหรือออกจาก MSC-51 โดยสามารถสร้างจากไทม์เมอร์แชลแนล 1 โดยทำงานในโหมด 2 รีจิสเตอร์ที่ต้องการทำโปรแกรม มีดังนี้

-TMOD ตำแหน่ง 89 H ทำหน้าที่เลือกโหมดของไทม์เมอร์

-TCON ตำแหน่ง 88 H ทำหน้าที่เริ่มต้นสร้างบอด

-TH1 ตำแหน่ง 88 H ทำหน้าที่ใส่ข้อมูลการนับของไทม์เมอร์ 1 เพื่อสร้างบอด

2. รีจิสเตอร์ควบคุมการลดกำลัง เนื่องจาก การสร้างบอดนั้นต้องนำบิตในรีจิสเตอร์ PCON มาใช้ในการคำนวณ ข้อมูลของ TH1 ดังนั้นรีจิสเตอร์ที่ใช้คือ

- PCON ตำแหน่ง 87H ทำหน้าที่ในการคำนวณข้อมูลที่จะใส่ในรีจิสเตอร์ TH1 ดังตารางที่ 2.4

7	6	5	4	3	2	1	0
SMOD	-	-	-	GF1	GF0	PD	IDL

ตารางที่ 2.4 แสดงตำแหน่งบิตในรีจิสเตอร์ PCON

บิต	สัญลักษณ์	รายละเอียด
7	SMOD	เป็นบิตที่ใช้ในการแก้ไขอัตราการบอด
6-4	-	ไม่ได้ใช้งาน
3	GF1	แฟล็กใช้งานทั่วไป
2	GF2	แฟล็กใช้งานทั่วไป
1	PD	บิตที่แสดงการลดลงของกำลังไฟ
0	IDL	บิตที่แสดงในโหมดไอเดิล

3. รีจิสเตอร์ควบคุมการอินเตอร์รัพท์ มีรีจิสเตอร์ที่เกี่ยวข้องดังนี้

- IE ตำแหน่ง A8H ทำหน้าที่ยอมให้เกิดการอินเตอร์รัพท์
- IP ตำแหน่ง B8H ทำหน้าที่จัดลำดับความสำคัญของกาอินเตอร์รัพท์

4. รีจิสเตอร์ควบคุมพอร์ตอนุกรม ขึ้นอยู่กับรีจิสเตอร์โดยตรงคือ

- SBUF ตำแหน่ง 99H ทำหน้าที่ที่เป็นบัฟเฟอร์การรับหรือส่งข้อมูล
- SCON ตำแหน่ง 98H ทำหน้าที่ควบคุมและกำหนดโหมดการใช้งาพอร์ตอนุกรมทั้งหมดดังตารางที่ 2.5

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ตารางที่ 2.5 แสดงตำแหน่งบิตภายในรีจิสเตอร์ SCON

บิต	สัญลักษณ์	รายละเอียด
7	SM0	โหมดของพอร์ตอนุกรมบิต 0 ทำการเซ็ต โดยใช้โปรแกรมสั่งงาน
6	SM1	โหมดของพอร์ตอนุกรมบิต 1 ทำการเซ็ต โดยใช้โปรแกรมสั่งงานเช่นกัน

SM0	SM1	โหมด	รายละเอียด
0	0	0	รีจิสเตอร์แบบเลื่อนบิต,อัตราการส่ง = f/12
0	1	1	UART ชนิด 8 บิต,อัตราการส่งเปลี่ยนแปลงได้
1	0	2	UART ชนิด 9 บิต,อัตราการส่ง = f/32 หรือ f/64
1	1	3	UART ชนิด 9 บิต,อัตราการส่งเปลี่ยนแปลงได้

บิต	สัญลักษณ์	รายละเอียด
5	SM2	ใช้เป็นบิตแสดงการติดต่อระหว่างไมโครโปรเซสเซอร์ ในกรณีนี้ใช้เฉพาะโหมด 2 และ 3 เมื่อบิตถูกเซ็ตเป็น 1 การอินเตอร์รัพท์จะเกิดขึ้น
4	REN	บิตอีนาเบิ้ลการรับบิตจะถูกเซ็ตเป็น 1 เมื่อต้องการสัญญาณอนุกรม
3	TB8	ใช่ว่าจะเลือกส่ง 8 บิตหรือไม่ใช่ สำหรับในโหมด 2 หรือ 3 บิตหยุดในโหมด 1 ส่วนในโหมด 0 จะไม่ใช้บิตนี้
2	RB8	ใช่ว่าจะเลือกรับ 8 บิตหรือไม่ใช้ สำหรับในโหมด 2 หรือ3 บิตหยุดในโหมด 1 ส่วนในโหมด 1 ส่วนในโหมด 0 จะไม่ใช้บิตนี้
1	TI	แฟลกอินเตอร์รัพท์เมื่อส่งข้อมูลในโหมด 0 จะถูกเซ็ตเป็น 1 หลังจากส่งบิต 7 ไปแล้ว
0	RI	แฟลกอินเตอร์รัพท์เมื่อรับข้อมูลในโหมด 0 จะถูกเซ็ตเป็น 1 หลังจากส่งบิต 7 เข้าแล้ว

การกำหนดโหมดการใช้งาน

การกำหนดโหมดการใช้งานพอร์ตสื่อสารอนุกรมในโหมดต่างๆ ดังตารางที่ 2.6

โหมด	SCON	SM2 VARIATION
0	10H	Single processor Environment (SM2 = 0)
1	50H	
2	90H	
3	D0H	
0	NA	Multiprocessor Environment (SM2 = 1)
1	70H	
2	B0H	
3	F0H	

ตารางที่ 2.6 รูปตารางแสดงการใช้งานพอร์ตสื่อสารอนุกรมโหมดต่าง ๆ

วิธีการคำนวณและกำหนดค่า baudrate ในการใช้พอร์ตสื่อสารอนุกรมโหมดต่าง ๆ

โหมด 0 ค่า baud rate ถูกกำหนดไว้คงที่ ที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้ โดยไม่จำเป็นต้องมีการกำหนดหรือใช้รีจิสเตอร์ที่ใช้เป็น ไทม์เมอร์หรือเคาน์เตอร์แต่อย่างใด ดังนั้น baud rate ของพอร์ตสื่อสารอนุกรมในโหมดนี้สามารถเขียนเป็นสมการได้ง่ายๆดังนี้

$$\text{Baud rate ในโหมด 0} = \frac{f_{osc}}{12}$$

โหมด 1 ค่า baud rate สามารถแปรค่าได้โดยการกำหนดไทม์เมอร์ 1 หรือ ไทม์เมอร์ 2 (มีใน 8052)ดังนี้

การใช้ไทม์เมอร์ 1 เป็นตัวกำหนด baud rate เมื่อใช้ไทม์เมอร์ 1 เป็นตัวกำหนด baud rate จะใช้ไทม์เมอร์ 1 ในโหมด 2 (Auto-Reload) โดยมีสูตรในการคำนวณค่า baud rate ดังนี้

$$\text{Baud rate ในโหมด 1} = \frac{2^{\text{mod}} \times f_{osc}}{32 \times 12 \times [256 - (TH1)]}$$

ในการใช้งานทั่วไป เรามักจะทราบว่าต้องการใช้ baud rate ค่าเท่าใด ดังนั้นค่าที่เราต้องการหาก็คือค่าที่ต้องโหลดไว้ในรีจิสเตอร์ใช้งานเฉพาะ TH1 สมการในการคำนวณหาค่าที่ต้องโหลดไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ TH1 เมื่อทราบค่า baud rate คือ

$$TH1 = 256 - \frac{s\,\text{mod} \times f_{osc}}{384 \times baudrate}$$

ค่าในรีจิสเตอร์ใช้งานเฉพาะ TH1 จำเป็นต้องเป็นเลขจำนวนเต็ม การปัดเศษที่ได้จากการคำนวณทิ้งหรือปัดขึ้นทำให้ไม่ได้ค่า baud rate ได้ตามต้องการวิธีแก้ปัญหานี้คือ การเปลี่ยนค่าความถี่ของคริสตอลที่ใช้

ในการเซตบิต SMOD ให้ใช้คำสั่งต่อไปนี้

```
ORL   PCON,#10000000B
```

เนื่องจากรีจิสเตอร์ใช้งานเฉพาะ PCON ไม่สามารถเข้าถึงได้ในระดับบิต

การใช้ไทม์เมอร์ 2 เป็นตัวกำหนด baud rate ในไทม์เมอร์ 2 มีการทำงานที่ใช้สำหรับเป็นตัวกำหนด baud rate

เมื่อใช้สัญญาณนาฬิกาจากภายนอกเป็นอินพุทเป็นอินพุทให้แก่ไทม์เมอร์ 2 ที่ขา T2(P1.0) ค่า baud rate สามารถคำนวณได้ดังนี้

Baud rate = อัตราการเกิด overflow ของไทม์เมอร์2 / 16

เมื่อใช้สัญญาณนาฬิกาจากภายในชิป(กำหนดจากความถี่ออสซิลเลเตอร์) ค่า baud rate สามารถคำนวณได้ดังนี้

$$\text{Baud rate} = \frac{f_{OSC}}{32 \times [65535 - (RCAP2H, RCAP2L)]}$$

หากต้องการหาค่าที่จะโหลดไปไว้ในรีจิสเตอร์ ใช้งานเฉพาะ RCAP2H,RCAP2L เมื่อทราบค่า baud rateที่ต้องการจะหาได้จากสมการต่อไปนี้

$$\text{RCAP2H,RCAP2L} = 65335 - \frac{f_{OSC}}{32 \times baudrate}$$

โหมด 2 ค่า baud rate มีให้เลือกได้เพียง 2 ค่า ขึ้นอยู่กับการกำหนดค่าบิต SMOD ในรีจิสเตอร์ใช้งานเฉพาะ PCON ดังนี้

บิต SMOD = 0 : baud rate จะเป็น 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้

บิต SMOD = 1 : baud rate จะเป็น 1/32 ของความถี่ออสซิลเลเตอร์ที่ใช้

ในโหมดนี้ไม่จำเป็นต้องใช้รีจิสเตอร์สำหรับใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์เพื่อกำหนดค่า baud rate แต่อย่างใด

ในการเซตบิต SMOD ให้ใช้คำสั่งต่อไปนี้

```
ORL   PCON,#10000000B
```

เนื่องจากรีจิสเตอร์ใช้งานเฉพาะ PCON ไม่สามารถเข้าถึงได้ในระดับบิต

สูตรการคำนวณค่า baud rate ในโหมด 2 มีดังนี้

$$\text{Baud rate ในโหมด 2} = \frac{2^{smod} \times f_{OSC}}{64}$$

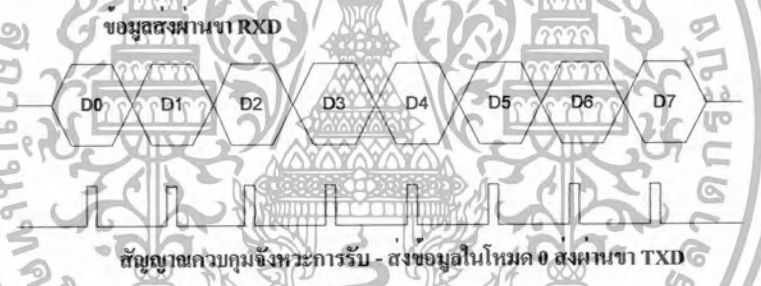
หากใช้คริสตอลความถี่ 12 เมกกะเฮริตซ์ baud rate สูงสุดในการทำงานในโหมดนี้คือ 375 Kbps

โหมด 3 ค่า baud rate ในโหมดนี้คำนวณและกำหนดได้จากวิธีเดียวกันกับในโหมด 1

พอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51

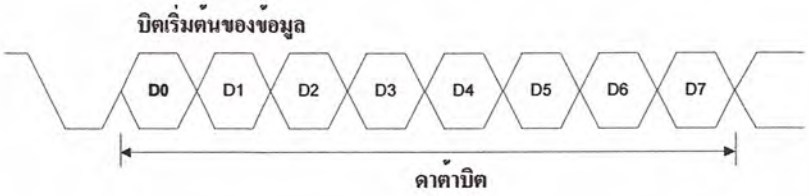
การใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51 มีความสะดวกและคล่องตัวสูง ทั้งนี้เนื่องจากผู้ใช้สามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 ประเภท โดยสามารถกำหนดได้จากค่าของบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON ดังแสดงในรูปที่ 2.1 การใช้งานที่แตกต่างกัน 4 ประเภท นี้มีจุดประสงค์เพื่อความคล่องตัวในการรับส่งข้อมูลแบบอนุกรมแต่ล่ะประเภทดังนี้

โหมด 0 การทำงานของพอร์ตสื่อสารอนุกรมในโหมด 0 ขา RXD จะใช้สำหรับและส่งข้อมูล ส่วนขา TXD มีไว้เพื่อใช้สร้างสัญญาณ shift clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูล(ข้อมูลจะถูกรับหรือส่งตามจังหวะของสัญญาณ clock) ในโหมดนี้การรับส่งข้อมูลจะเป็นแบบ 8 บิต โดยเริ่มรับและส่งบิตไบต์ต่ำก่อน (LSB first) อัตราการรับส่งข้อมูลในการทำงานโหมด 0 ถูกกำหนดที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้ การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 จะไม่มีบิตเริ่มต้นของข้อมูล และบิตสิ้นสุดของข้อมูล เพราะจังหวะการรับส่งข้อมูลถูกกำหนดจากสัญญาณ shift clock แล้ว



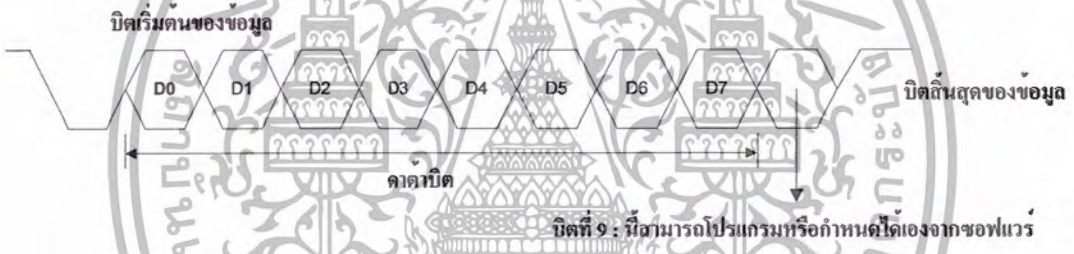
รูปที่ 2.4 แสดงข้อมูลที่รับและส่งในการทำงานของพอร์ต

โหมด 1 การทำงานแบบที่สอง หรือการทำงานในโหมดที่ 1 มีการรับส่งข้อมูลครั้งละ 10 บิต ข้อมูลจะถูกส่งออกไปภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาทาง RXD ข้อมูลทั้ง 10 บิต ประกอบไปด้วยบิตเริ่มต้นข้อมูล 1 บิต(มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) และบิตสิ้นสุดข้อมูลอีก 1 บิต (มีค่าเป็น 1 เสมอ) ในขณะทำการรับข้อมูล ค่าในบิตสิ้นสุด ของข้อมูลที่รับได้จะไปอยู่ในบิต RB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON อัตราเร็วในการรับหรือส่งข้อมูลของพอร์ตสื่อสารแบบอนุกรมในโหมดนี้สามารถเปลี่ยนแปลงได้



รูปที่ 2.5 แสดงการรับและส่งข้อมูลอนุกรมในโหมด 1

โหมด 2 การทำงานแบบที่ สามหรือการทำงานในโหมด 2 จะมีการรับและส่งข้อมูลครั้งละ 11 บิตข้อมูลจะถูกส่งออกภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาผ่านทางขา RXD ข้อมูลที่รับส่งเข้ามาทั้ง 11 บิต ประกอบด้วยบิตเริ่มต้นข้อมูล 1 บิต บิตข้อมูล 8 บิต ตามด้วยบิตที่ 9 (ต่อจากบิตข้อมูลบิตสุดท้าย) ซึ่งเป็นบิตที่สามารถกำหนดให้มีค่าเป็น 1 หรือ 0 ก็ได้ (programmable 9th data bit) และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล ดังนั้นจำนวนบิตที่รับและส่งทั้งหมด 11 บิต ประกอบด้วยบิตต่างๆ ดังนี้



รูปที่ 2.6 แสดงข้อมูลรับและส่งข้อมูลอนุกรมโหมด 2 และ 3

โหมด 3 การทำงานของพอร์ตสื่อสารอนุกรมแบบสุดท้าย คือ การทำงานในโหมดที่ 3 ในการทำงานโหมดนี้ข้อมูลจำนวน 11 บิต ถูกส่งผ่านขา TXD และถูกรับเข้ามาทางขา RXD ข้อมูลทั้ง 11 บิต ประกอบด้วยบิตเริ่มต้นข้อมูล 1 บิต ตามด้วยบิตที่ 9 ซึ่งสามารถกำหนดค่าได้เหมือนโหมด 2 และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล อัตราการส่งและรับข้อมูลสามารถเปลี่ยนแปลงได้ ดังนั้นจะเห็นได้ว่ารูปแบบการรับและส่งข้อมูลในโหมด 3 จะเหมือนกับในโหมด 2 ทุกอย่าง แต่ในโหมดนี้สามารถกำหนดอัตราเร็วในการรับส่งข้อมูลได้ตามความต้องการของผู้ใช้

การเขียนหรือส่งข้อมูลออกจากพอร์ตอนุกรม

ข้อมูลที่ต้องการส่งออกทุกค่าต้องนำไปเก็บไว้รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมซึ่งก็คือรีจิสเตอร์ SBUF ดังตัวอย่าง

```
MOV   SBUF,#'A'
```

จากคำสั่งข้างต้นเป็นการส่งข้อมูลของตัวอักษร A ออกไปยังพอร์ตอนุกรมของไมโครคอนโทรลเลอร์อย่างไรก็ตามก่อนทำการส่งข้อมูลทุกครั้ง ต้องแน่ใจว่าบิต TI เคลียร์หรือมีค่าเป็น "0"และเมื่อทำการส่งข้อมูลเรียบร้อยแล้ว ก็จะเกิดการเซตบิต TI เพื่อแจ้งให้ทราบ ดังตัวอย่างโปรแกรมต่อไปนี้

```
CLR   RI          เคลียร์บิต TI เพื่อเตรียมการส่งข้อมูลออก
MOV   SBUF,#'A'   ส่งข้อมูลของตัวอักษร A ไปยังพอร์ตอนุกรม
JNB   TI,$        รอการเซตของบิต TI เพื่อแจ้งการส่งข้อมูลที่เสร็จ
                  สมบูรณ์
```

การอ่านหรือรับข้อมูลจากพอร์ตอนุกรม

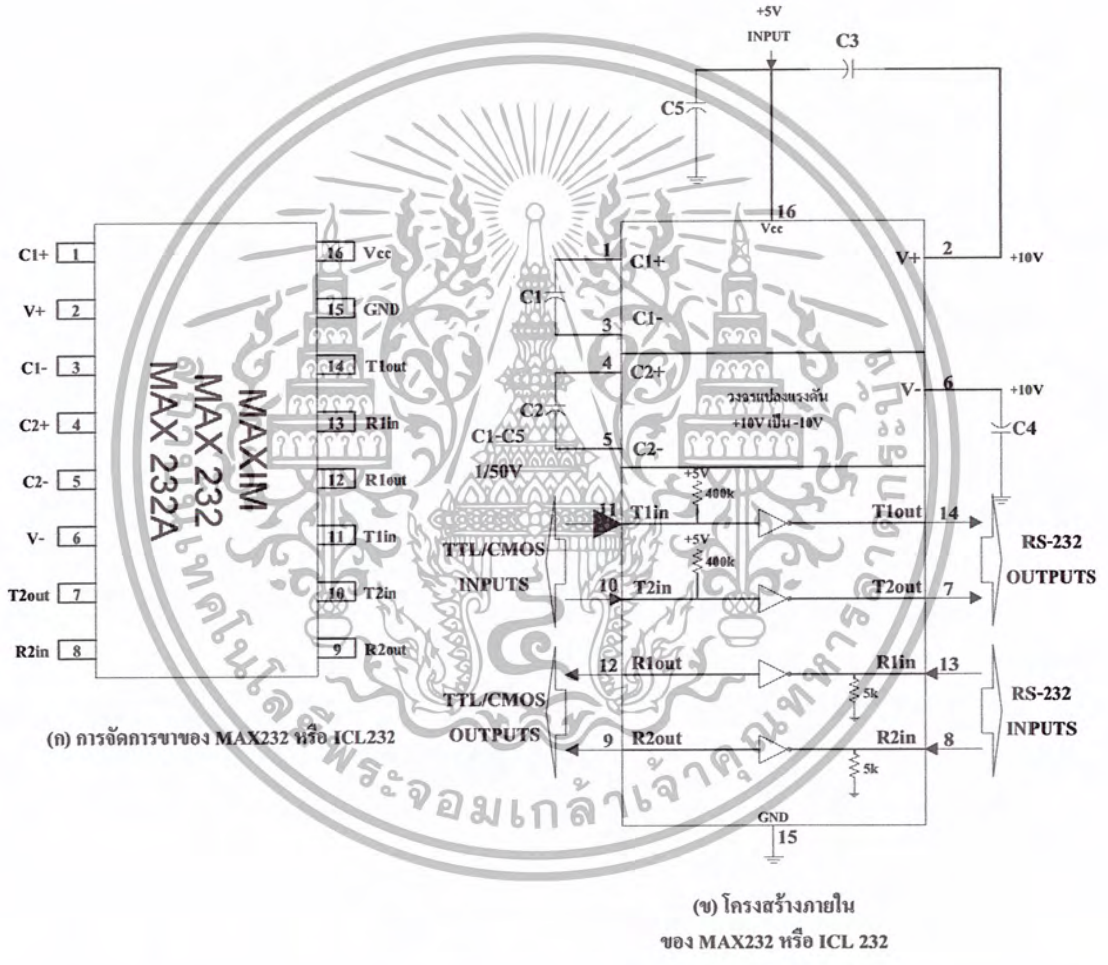
การรับข้อมูลจากพอร์ตอนุกรมสามารถกระทำได้ง่ายมาก เพียงทำการตรวจสอบว่าบิต RI เกิดการเซตขึ้นหรือไม่ ถ้าพบว่ามีการเซตเกิดขึ้นแล้ว ให้ทำการอ่านค่าจากรีจิสเตอร์ SBUF โดยต้องทำการโอนย้ายข้อมูลผ่านทางแอกคิวมูเลเตอร์หรือรีจิสเตอร์ A ดังตัวอย่าง

```
CLR   RI          เคลียร์บิต TI เพื่อเตรียมการส่งข้อมูลออก
JNB   RI,$        รอคอยการเซตของบิต RI อันเป็นการแจ้งให้ทราบว่า
                  การรับข้อมูลเสร็จสมบูรณ์และมีข้อมูลเกิดขึ้นที่
                  รีจิสเตอร์ SBUF
MOV   A,SBUF      อ่านค่าจากรีจิสเตอร์ โดยการโอนข้อมูลผ่านทาง
                  รีจิสเตอร์ A
CLR   RI          หลังจากทำการอ่านข้อมูลเรียบร้อยแล้ว ต้องทำการ
                  เคลียร์บิต RI เสมอ
```

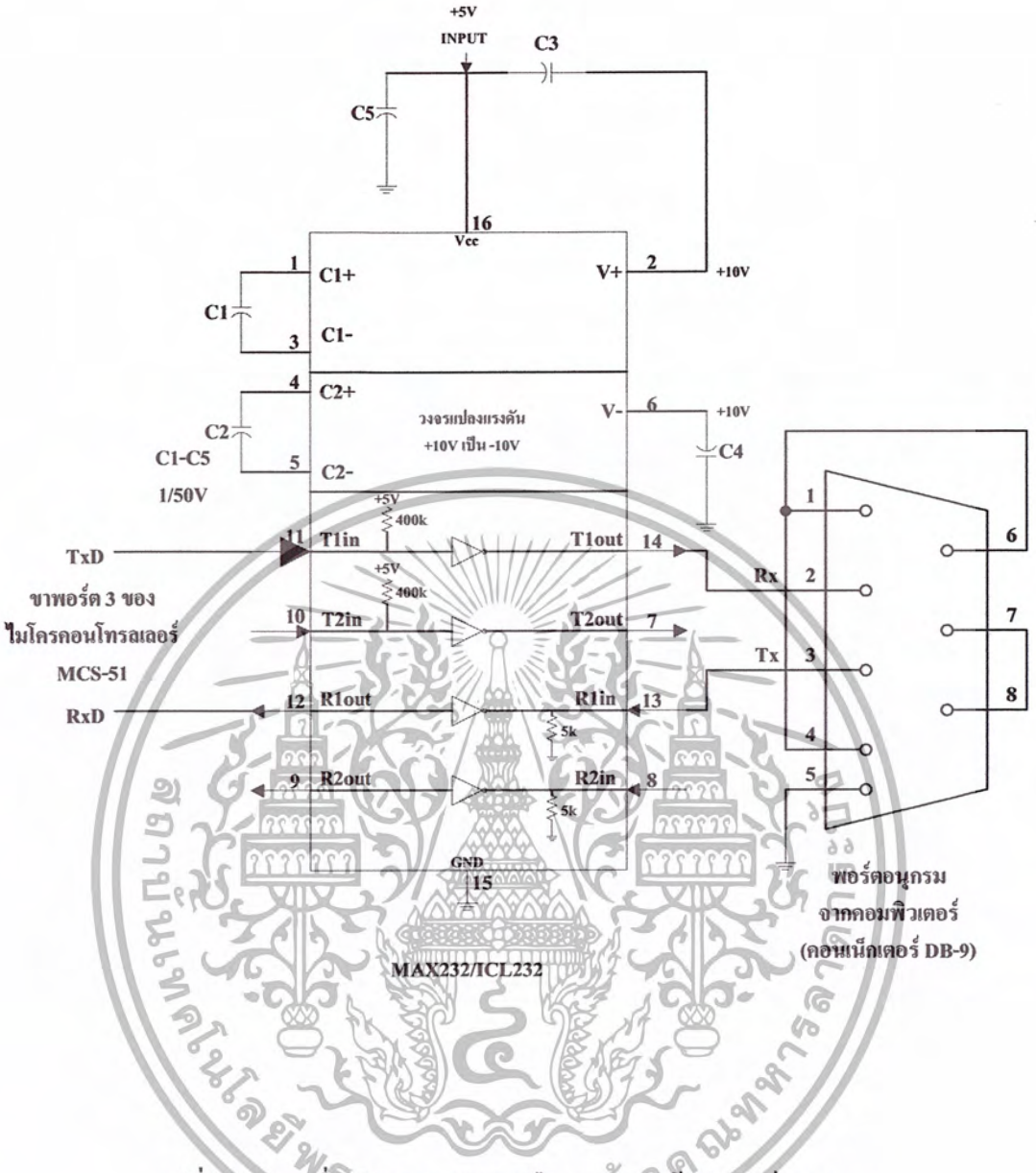
การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ ±3 ถึง ±12 V. ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

ไอซี ที่ทำหน้าที่ ในการแปลงระดับสัญญาณนี้ ต้องทำการแปลงข้อมูล ส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จาก MAXIM หรือ ICL232 จาก HARRIS เป็นต้น ในรูปที่ 2.7 แสดงการจัดขาของไอซี ICL232 ซึ่งใช้ในการแปลงสัญญาณ RS-232 ส่วนวงจรของการต่อกับไมโครคอนโทรลเลอร์ MCS-51 แสดงในรูปที่ 2.8



รูปที่ 2.7 รายละเอียดเบื้องต้นของไอซีแปลงสัญญาณที่เชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์



รูปที่ 2.8 วงจรเชื่อมต่อ MAX232 หรือ ICL232 เข้ากับพอร์ตอนุกรม
ของคอมพิวเตอร์และไมโครคอนโทรลเลอร์ MCS-51

บทที่ 3

หลักการทำงาน

รูปที่ 3.1 แสดง Flow Chart การทำงานโดยรวม

การทำงานโดยรวมของ Lane Detection for Vehicle

การทำงานเริ่มต้นจากการที่ กล้อง รับภาพจากเลนถนน จากนั้นจะส่งภาพที่ได้นั้นไปให้คอมพิวเตอร์ทำการประมวลผล โดยการประมวลผลนั้นจะใช้โปรแกรม Visual Basic ในการประมวลผล ซึ่งจะทำการเขียน Browser ในการประมวลผล เมื่อทำการประมวลผลเสร็จแล้วนั้นจะทำการส่งข้อมูลที่ประมวลผลออกไปให้กับ Mcs-51 ต่อไปเพื่อทำการ Drive Motor โดยการส่งจะทำการส่งผ่านทางพอร์ตอนุกรม (Serial Prot) โดยการทำงานนั้นสามารถอธิบายได้คร่าว ๆ ดังนี้คือ เมื่อการที่เลนถนนเกิดมีการเลี้ยวซ้าย เลี้ยวขวา หรือ ตรง นั้นคอมพิวเตอร์จะทำการนำภาพที่ได้มาประมวลผล โดยถ้าเลนถนนเลี้ยวซ้าย จะทำการส่งข้อมูลที่ได้ไปให้กับ Mcs-51 ทำการสั่งให้มอเตอร์ด้านขวาหมุน แล้วด้านซ้ายหยุด จะทำให้รถ เลี้ยวซ้ายนั้นเอง ในทำนองเดียวกันถ้าเลนถนนเลี้ยวขวา ก็จะทำงานตรงข้ามกันนั่นเอง และถ้าเลนส์ถนนตรงนั้นก็จะทำการสั่งให้มอเตอร์หมุนไปพร้อม ๆ กัน นั่นเอง ซึ่งสามารถสรุปการทำงานโดยรวมได้ดัง Flow Chart ข้างต้นนั่นเอง

หลักการทำงาน

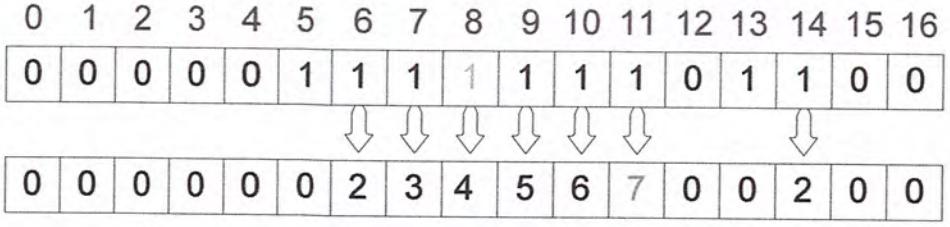
3.1 การทำงานของส่วน Browser ประมวลผล

โดยการนำภาพที่เก็บมาจากกล้องนั้น โดยการใช้ Component ในการรับภาพชื่อ ezVidCap แล้วทำการเก็บภาพไว้ชื่อว่า capture.bmp แล้วทำการโหลดภาพที่ได้มาคำนวณ โดยจะทำการแบ่งแยกระดับสีของภาพ (Threshold) โดยจะทำ Threshold บนภาพที่ capture มาเพียงสองเส้นเพื่อกันการDetect Error เนื่องจากการเจอเส้นประ ซึ่งจะแบ่งสีออกเป็น 2 ระดับ คือ ขาว กับ ดำ โดยดูที่ความเข้มของแสง ซึ่งในโปรแกรมนั้น ซึ่งระดับค่าสีในโปรแกรมที่จับภาพได้นั้นจะมีตั้งแต่ 0-255 (0 คือสีดำ และ 255 คือสีขาว)โดยในโปรแกรมได้ทำการให้ค่าสีที่มีความเข้มแสงน้อยกว่า 100 เป็นสีดำ และมากกว่า 100 เป็นสีขาว และทำการเก็บค่าไว้ในตัวแปร แบบ array โดยค่าสีดำให้เป็น 0 และให้ค่าสีขาวเป็น 1 ดังรูปที่ 3.2

0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0

รูปที่ 3.2 แสดงการเก็บค่าใน Array โดยกำหนดให้สีดำเก็บเป็นค่า 0 สีขาวเก็บเป็นค่า 1

ต่อจากนั้นเราก็หาตำแหน่งจุดกึ่งกลางของเลนถนน โดยเราจะยึดเอาเส้นกลางเลนถนนที่มีขนาดใหญ่ที่สุดเป็นตำแหน่งที่เรายึดโดยจากรูปที่ 3.2 นั้นเราจะเห็นว่ามีเส้นสีขาวอยู่สองเส้น (ดูจากกลุ่มเลข 1 ที่ติดกัน) เพื่อจะให้การ Detect มีผลกับเส้นขนาดใหญ่สุด แล้วไม่มีผลกับเส้นที่เล็กกว่าหรือ Noise นั้นเราต้องเปลี่ยนจาก Array แรกแปลงเป็น Array ดังรูปที่ 3.3



รูปที่ 3.3 การแปลง Array เพื่อจะ Detect หาเส้นสีขาวที่มีขนาดใหญ่ที่สุด

โดยเราจะเขียนโปรแกรมโดยถ้าข้อมูลใน Array ที่ข้อมูลข้างมีค่าเป็น 1 ให้ทำการบวกไปเรื่อยๆ ดังรูปที่ 3.3 แล้วเราจะหาจุดกึ่งกลางเลนที่มีขนาดใหญ่สุดโดย อย่างแรกเราจะเขียนโปรแกรมเพื่อหาข้อมูลที่มากที่สุดใน Array ที่เราแปลงแล้ว โดยจากรูปที่ 3.3 นั้นเราจะเห็นเป็นเลข 7 ซึ่งจะเป็นตำแหน่ง Array ที่ 11 ซึ่งเราสามารถหาตำแหน่งกึ่งกลางเลนโดย

ตำแหน่งกึ่งกลางเลนของภาพ = ตำแหน่ง Array ของข้อมูลที่มากที่สุด- [(ข้อมูลที่มากที่สุด-1)/2]

= 11-[(7-1)/2]

= 8

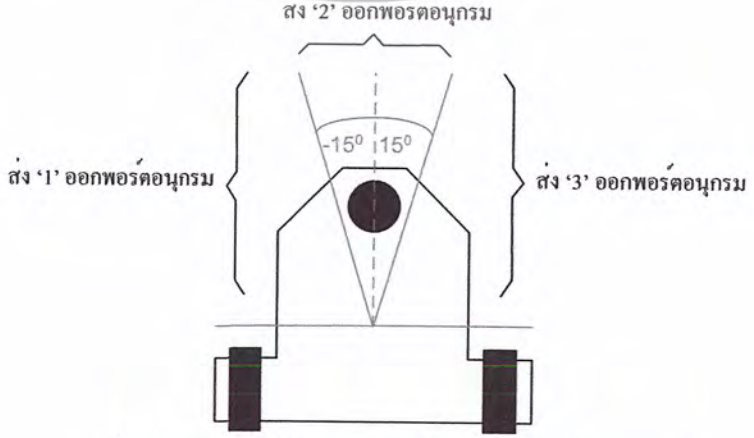
ซึ่งเมื่อได้จุดกึ่งกลางของเลนถนนแล้ว ก็จะสามารถนำค่าที่ได้นั้นไปเปรียบเทียบกับจุดกึ่งกลางของภาพที่ Capture มา ซึ่งจะทำให้ทราบว่ารถเบี่ยงเบนไปจากเลนถนนเป็นมุมเท่าไหร่นั่นเอง แล้วต่อไปก็จะนำค่าที่ได้ส่งออกพอร์ตอนุกรม(Serial Port)โดยการส่งจะมีข้อแม้ดังนี้ คือ

เมื่อค่ามุม(-∞ ,-15)จะส่ง ASCII เลข '1' ออกพอร์ตอนุกรม

เมื่อค่ามุม(-15,15)จะส่ง ASCII เลข '2' ออกพอร์ตอนุกรม

เมื่อค่ามุม(15, ∞)จะส่ง ASCII เลข '3' ออกพอร์ตอนุกรม

ดังรูปที่ 3.4



รูปที่ 3.4 แสดงข้อแม้ในการส่งข้อมูลออกพอร์ตอนุกรม

ไปสู่คอนโทรลเลอร์ เพื่อทำการDrive Motor ต่อไป

3.2 หลักการทำงานของส่วน Controller

เมื่อ Controller ได้รับข้อมูลที่ส่งมาทางพอร์ตอนุกรม ซึ่ง Browser ได้ส่งมาเป็น 3 ตัวเลขคือ 1,2,3 ซึ่ง ในส่วน Controller นั้นได้เห็นเลขซึ่งเป็นรหัส ASCII ที่ Browser ส่งเป็นเป็นเลขฐาน 16 ดังนี้

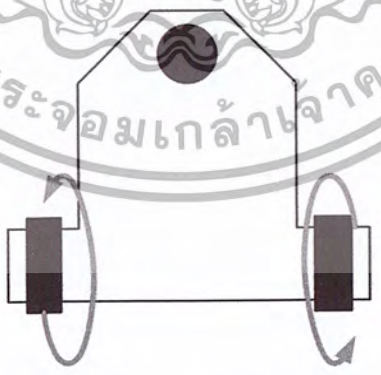
ตัวอักษรที่ส่งทางพอร์ตอนุกรม	ASCII
1	31
2	32
3	33

ตารางที่ 3.1 แสดงค่ารหัสที่ส่งออกจากพอร์ตอนุกรมและค่าที่คอนโทรเลอร์ได้รับ

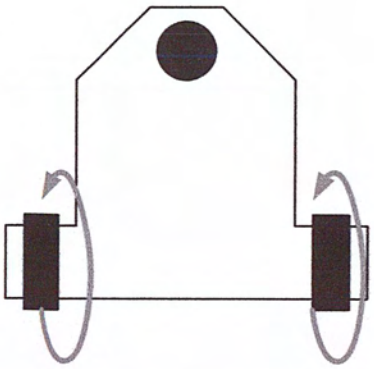
ดังนั้นเราจึงเขียนโปรแกรมให้ MSC-51 ให้เมื่อดังรับเลข 3 ตัวนี้แล้วเป็นข้อแม้ในการขับเคลื่อน STEPPERMOTOR แบบ หนึ่งเฟส ดังนี้

ตัวเลขที่ได้รับ	มอเตอร์ตัวที่ 1(ซ้าย)	มอเตอร์ตัวที่ 2(ขวา)	ทิศทาง
31	หมุนด้านหลัง	หมุนด้านหน้า	ซ้าย
32	หมุนด้านหน้า	หมุนด้านหน้า	ตรง
33	หมุนด้านหน้า	หมุนด้านหลัง	ขวา

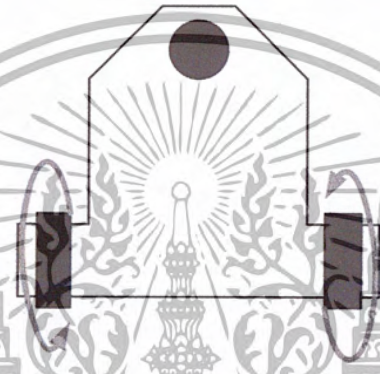
ตารางที่ 3.1 แสดงทิศทางการเคลื่อนที่ของรถเมื่อได้รับรหัสจากคอนโทรลเลอร์



รูปที่ 3.5 เมื่อรถได้รับข้อมูล '1' ที่ Browser ส่งมาทางพอร์ตอนุกรม



รูปที่ 3.6 เมื่อรถได้รับข้อมูล '2' ที่ Browser ส่งมาทางพอร์ตอนุกรม



รูปที่ 3.7 เมื่อรถได้รับข้อมูล '3' ที่ Browser ส่งมาทางพอร์ตอนุกรม

รูปที่ 3.8 แสดงวงจรขับมอเตอร์สเต็ปเปอร์ 2 ตัว ควบคุมโดยไมโครคอนโทรลเลอร์ MCS-51

บทที่4

ผลการทดลอง

4.1 การทดลองที่ 1 การทดลองการป้อนขับสเต็ปมอเตอร์

เราจะพบว่า

4.1.1 เมื่อเราป้อนแรงดันทริกให้ล้อหมุนแบบที่ 1 โดย

- เริ่มแรกปรับล้อรถให้ล้อรถอยู่ในตำแหน่งล้อตรง

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 0011 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 0011 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหลัง และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 0110 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 0110 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหลัง และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 1100 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 1100 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหลัง และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 1001 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 1001 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหลัง และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

สรุปการป้อนแรงดันทริกตามแบบที่ 1 นั้นจะทำให้สเต็ปเปอร์มอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) หมุนไปทางด้านหลัง และจะทำให้สเต็ปเปอร์มอเตอร์ตัวที่ 2 (ด้านขวาของรถ) หมุนไปทางด้านหน้า ดังนั้นจึงทำให้ทิศทางรถไปทางด้านซ้าย

4.1.2 เมื่อเราป้อนแรงดันทริกให้ล้อหมุนแบบที่ 2 โดย

- เริ่มแรกปรับล้อรถให้ล้อรถอยู่ในตำแหน่งล้อตรง

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 1100 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 0011 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 0110 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 0110 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 0011 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 1100 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 1001 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 1001 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหน้า จำนวน 1 สเต็ป

สรุปการป้อนแรงดันทริกตามแบบที่ 2 นั้นจะทำให้สเต็ปเปอร์มอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) หมุนไปทางด้านหน้า และจะทำให้สเต็ปเปอร์มอเตอร์ตัวที่ 2 (ด้านขวาของรถ) หมุนไปทางด้านหน้า ดังนั้นจึงทำให้ทิศทางรถไปทิศทางตรง

4.1.3 เมื่อเราป้อนแรงดันทริกให้ล้อหมุนแบบที่ 3 โดย

- เริ่มแรกปรับล้อรถให้ล้อรถอยู่ในตำแหน่งล้อตรง

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 1100 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 1100 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหลัง จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 0110 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 0110 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหลัง จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 1100 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 1100 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหลัง จำนวน 1 สเต็ป

- ทำการป้อนแรงดันทริกมอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) เป็น 1001 และป้อนแรงดันทริกกับมอเตอร์ตัวที่ 2 (ด้านขวาของรถ) เป็น 1001 จะทำให้สเต็ปเปอร์มอเตอร์ตัวที่1 หมุนไปด้านหน้า และสเต็ปมอเตอร์ตัวที่ 2 หมุนไปด้านหลัง จำนวน 1 สเต็ป

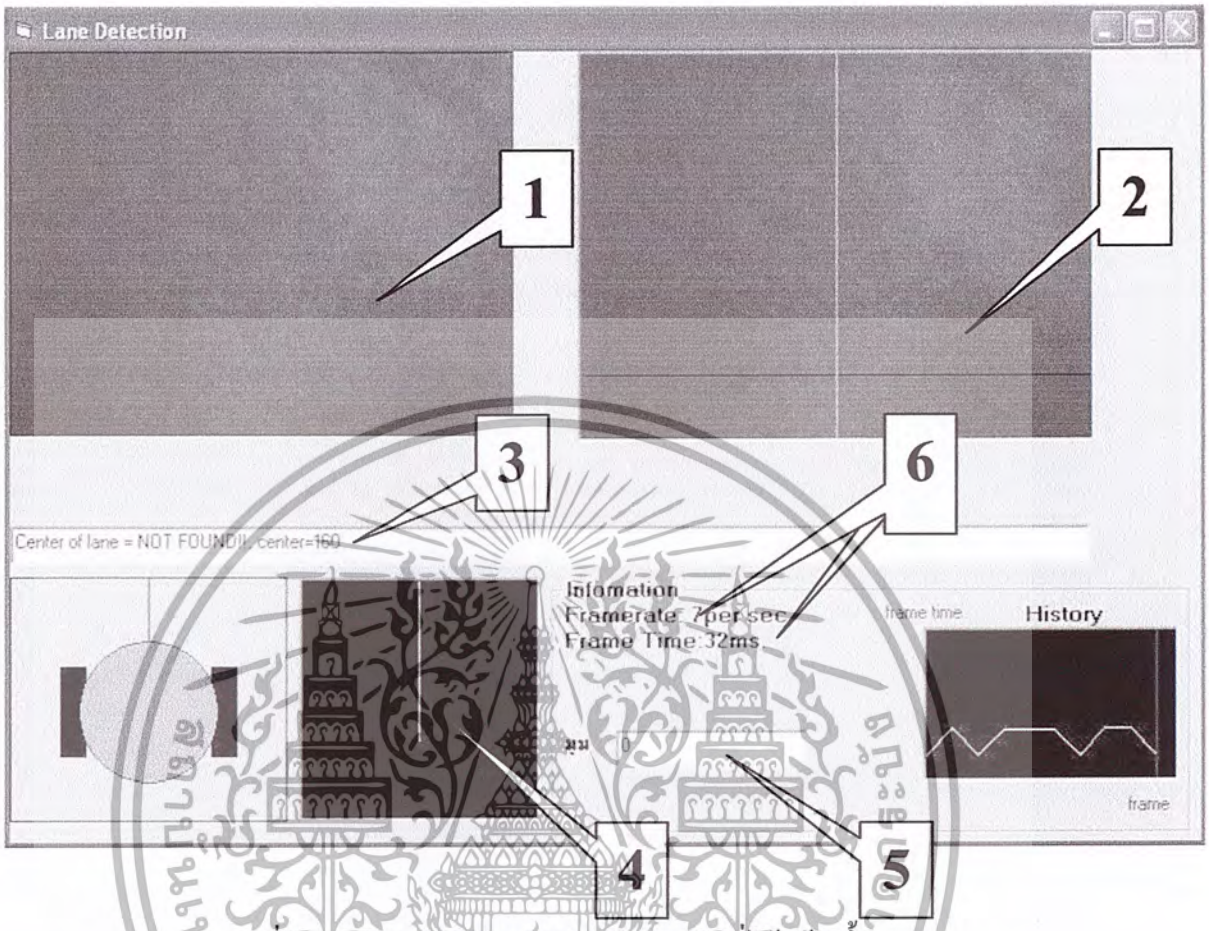
สรุปการป้อนแรงดันทริกตามแบบที่ 1 นั้นจะทำให้สเต็ปเปอร์มอเตอร์ตัวที่ 1 (ด้านซ้ายของรถ) หมุนไปทางด้านหน้า และจะทำให้สเต็ปเปอร์มอเตอร์ตัวที่ 2 (ด้านขวาของรถ) หมุนไปทางด้านหลัง ดังนั้นจึงทำให้ทิศทางรถไปทางด้านขวา

โดยทั้งหมดเราสรุปได้ดังตารางนี้

การป้อนแรงดันทริก	มอเตอร์ตัวที่1 (ด้านซ้าย)	มอเตอร์ตัวที่2 (ด้านขวา)	ทิศทางของรถ
แบบที่ 1	หมุนหลัง	หมุนหน้า	เลี้ยวซ้าย
แบบที่ 2	หมุนหน้า	หมุนหน้า	ตรง
แบบที่ 3	หมุนหน้า	หมุนหลัง	เลี้ยวขวา

ตารางที่ 4.1 แสดงผลการตรวจสอบการควบคุมสเต็ปเปอร์มอเตอร์

4.2 การทดลองที่ 2 เรื่องการรับภาพ การประมวลผลภาพและการส่งค่าควบคุม



รูปที่ 4.1 แสดงรายละเอียดออปเจคโปรแกรมที่ได้สร้างขึ้น

การทดลองนี้ได้ทำการเขียนโปรแกรมรับภาพจากกล้องดิจิตอล เมื่อกดปุ่ม Run ภาพที่ได้จะแสดงที่หน้าจอหลัก โดยจะขอทำการอธิบายลักษณะของโปรแกรมซึ่งจะบอกรายละเอียดได้เป็นส่วนๆ ทั้งหมด 6 ส่วน ดังต่อไปนี้

ส่วนที่ 1หน้าจอหลัก เป็นส่วนของภาพที่ รับได้จากกล้องดิจิตอลโดยยังไม่ผ่านกระบวนการประมวลผลภาพ

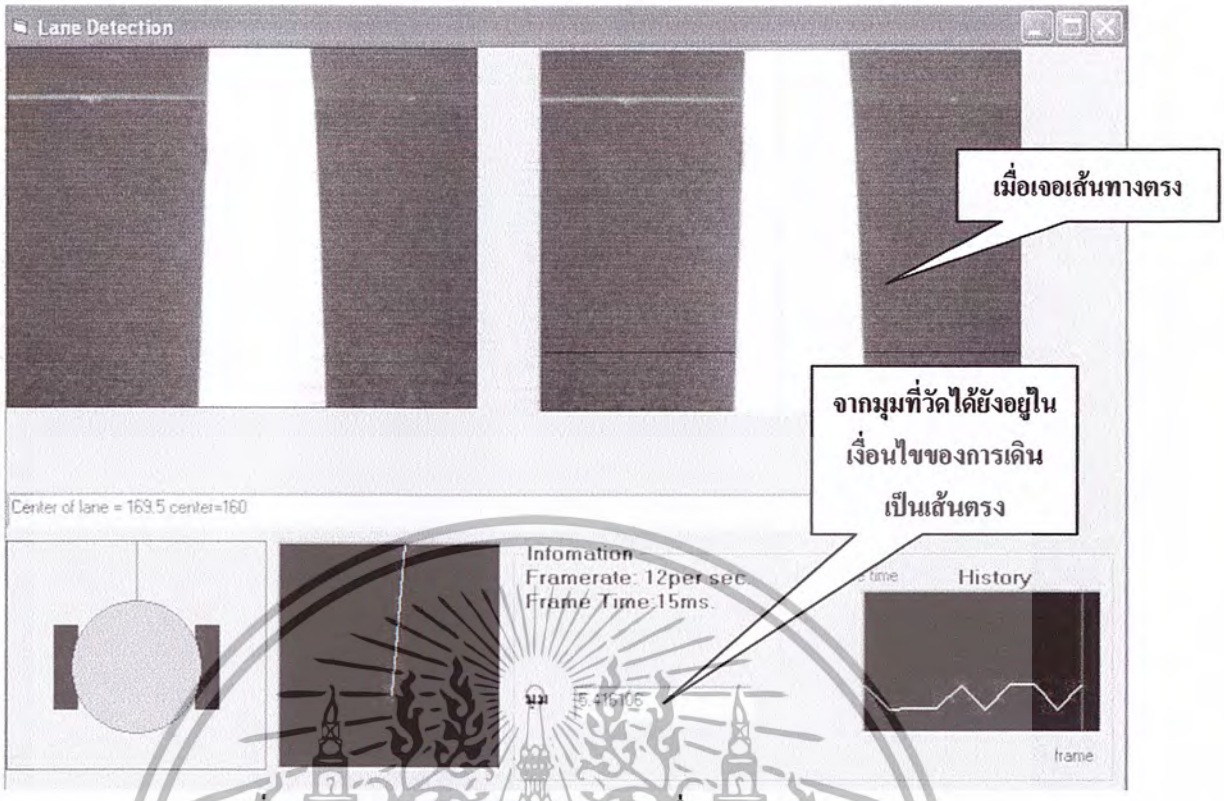
ส่วนที่ 2หน้าจอแสดงภาพที่ได้รับการประมวลผลภาพเรียบร้อยแล้ว

ส่วนที่ 3แสดงตำแหน่งจุดกึ่งกลางของภาพ และจุดกึ่งกลางของเลน ที่ทำการประมวลผลแล้ว

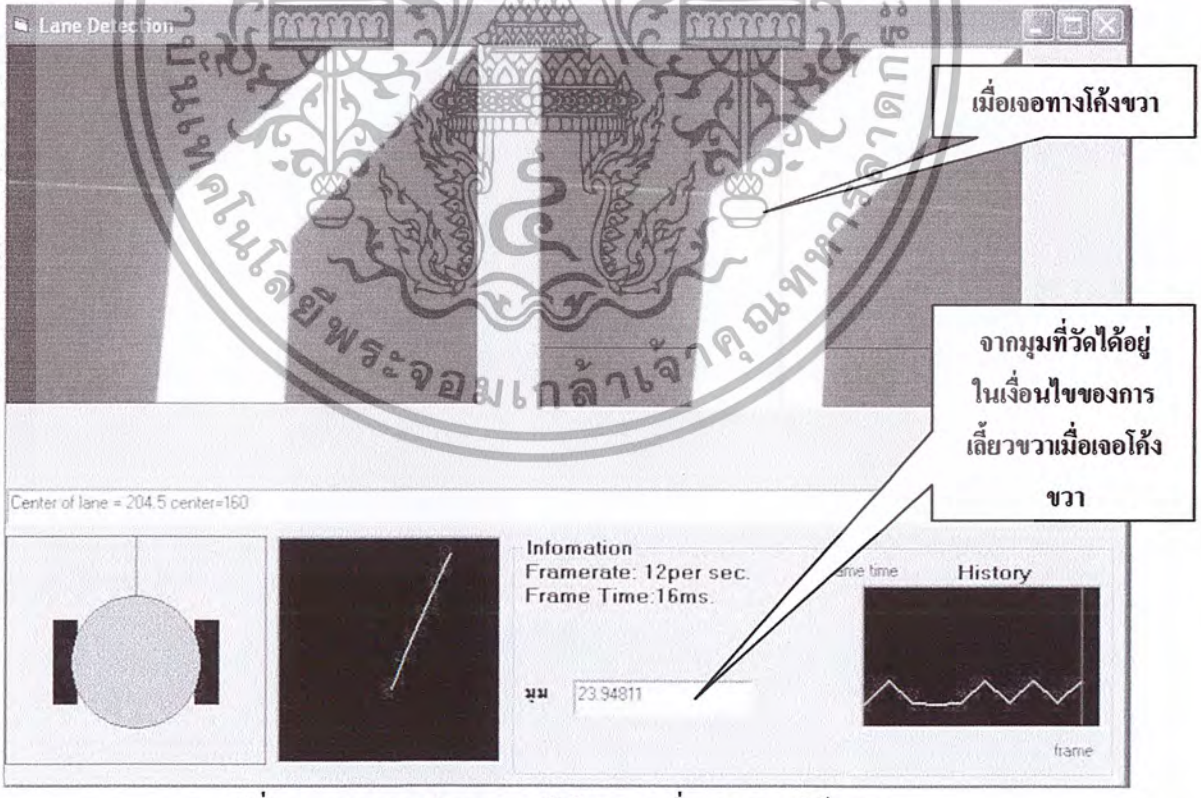
ส่วนที่ 4แสดงมาตรวัดทิศทางของตัวรถว่ากำลังเคลื่อนที่ไปในทิศใด

ส่วนที่ 5แสดงมุมเป็นองศาของตัวรถโดยทำการเทียบ กับจุดกึ่งกลางจอเป็นหลัก

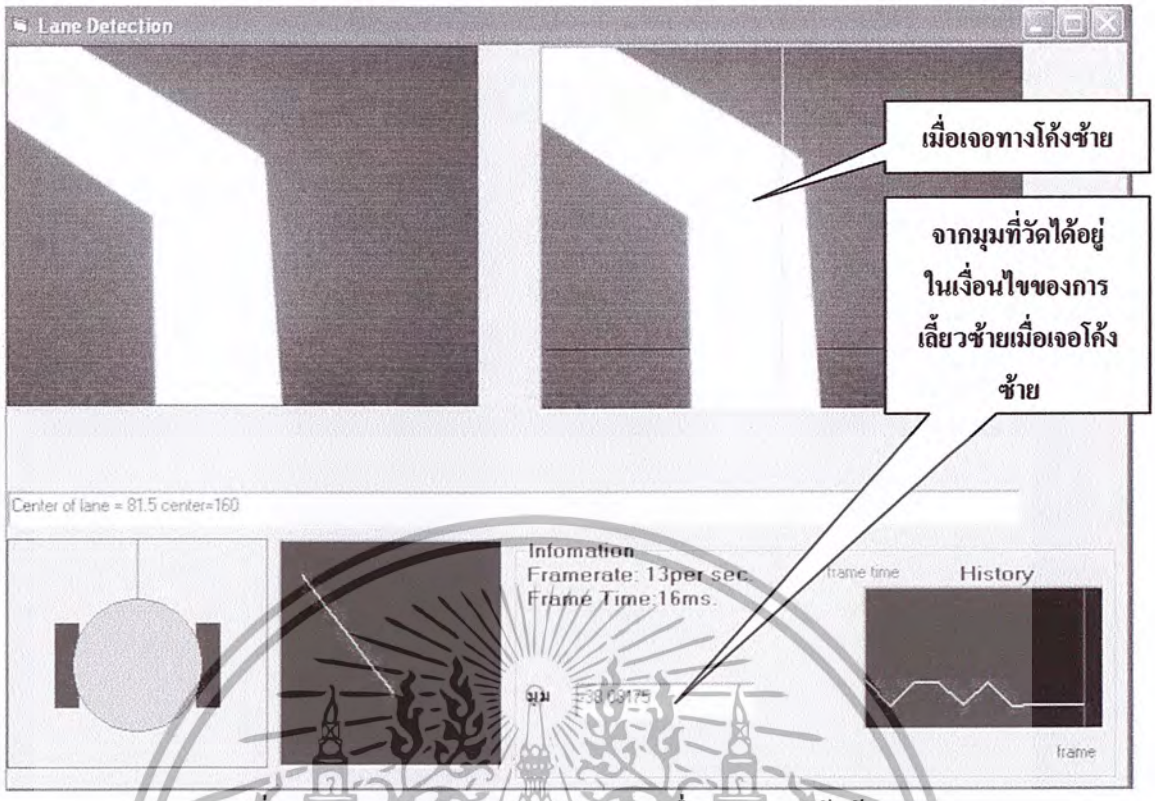
ส่วนที่ 6 แสดงค่า Frame Rate และ Frame Time ของโปรแกรม



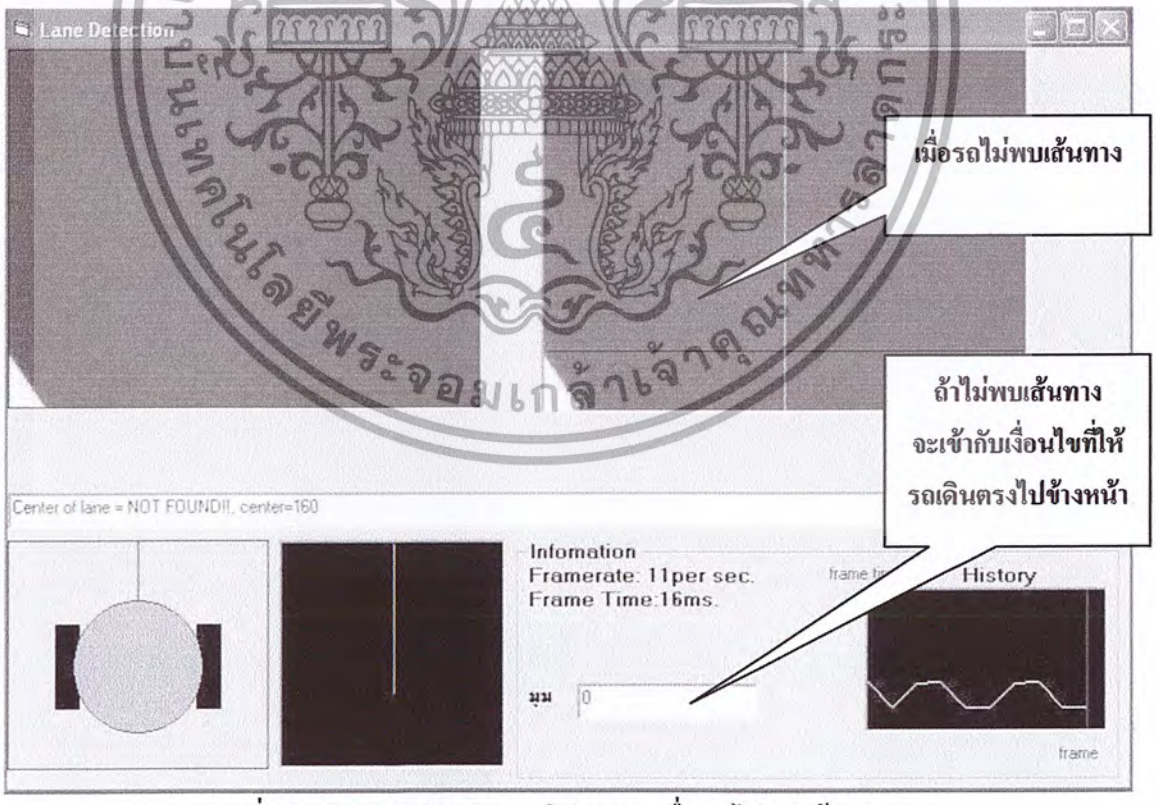
รูปที่ 4.2 แสดงการทำงานของโปรแกรมเมื่อพบถนนทางตรง



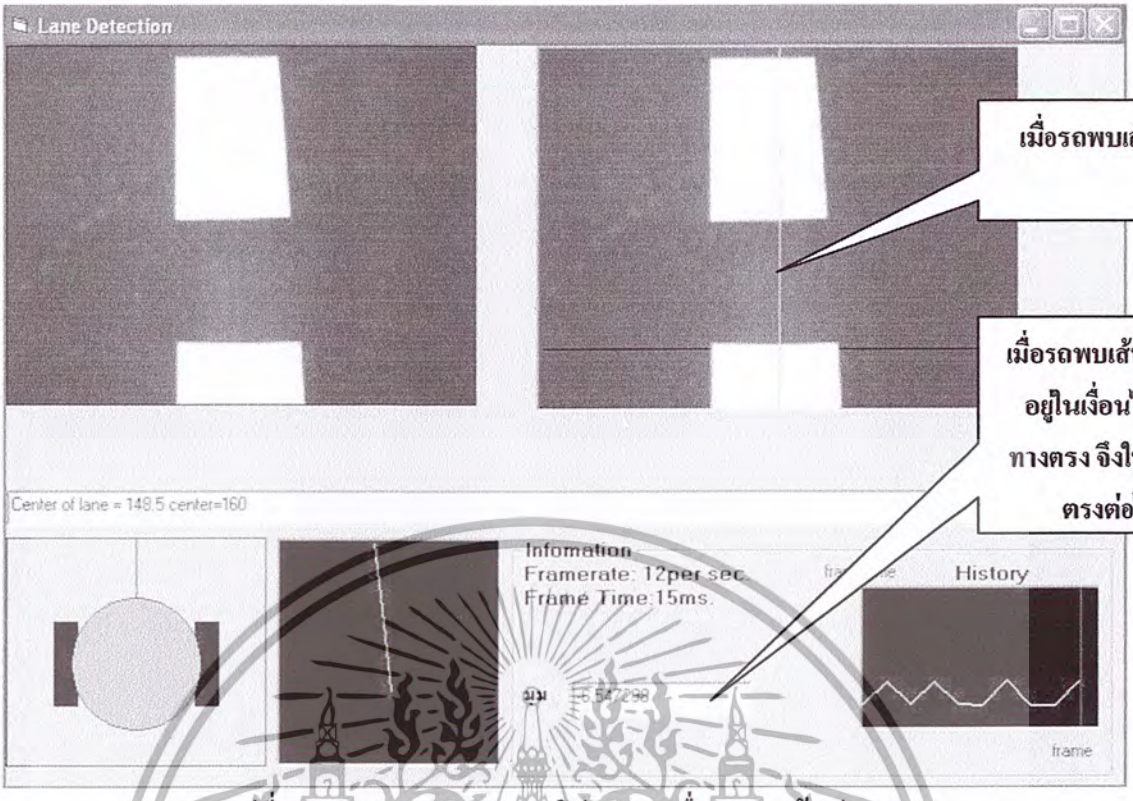
รูปที่ 4.3 แสดงการทำงานของโปรแกรมเมื่อพบถนนโค้งขวา



รูปที่ 4.4 แสดงการทำงานของโปรแกรมเมื่อพบถนนโค้งซ้าย



รูปที่ 4.5 แสดงการทำงานของโปรแกรมเมื่อรถไม่พบเส้นทาง



รูปที่ 4.6 แสดงการทำงานของโปรแกรมเมื่อรถพบเส้นประ

จากการทดลองการควบคุมรถอัตโนมัติได้ผลการทดลองดังรูปข้างต้นและสามารถเขียนเป็นตารางได้ดังนี้

ลักษณะถนน	ตรวจสอบพบ	จำนวนครั้งตรวจสอบถูกต้อง	จำนวนครั้งตรวจสอบผิดพลาด	% ความผิดพลาด
ถนนโค้งซ้าย	เลี้ยวซ้าย	9	1	10
ถนนโค้งขวา	เลี้ยวขวา	9	1	10
ถนนตรง	ตรง	10	10	0

ตารางที่ 4.2 แสดงการตรวจสอบลักษณะถนน

จากการทดลองวัดมุมเลี้ยวของเลนถนนในองศาต่าง ๆ เป็นตารางได้ดังนี้

มุมจริง	มุมที่ทำการคำนวณได้
30	24
45	44
60	54
90	0 เนื่องจากคำนวณจากค่า Tan

ตารางที่ 4.3 แสดงค่ามุมที่วัดได้และค่าที่ได้จากการคำนวณ

4.3 การทดลองที่ 3 เรื่องทดสอบมุมของการเลี้ยวรถ

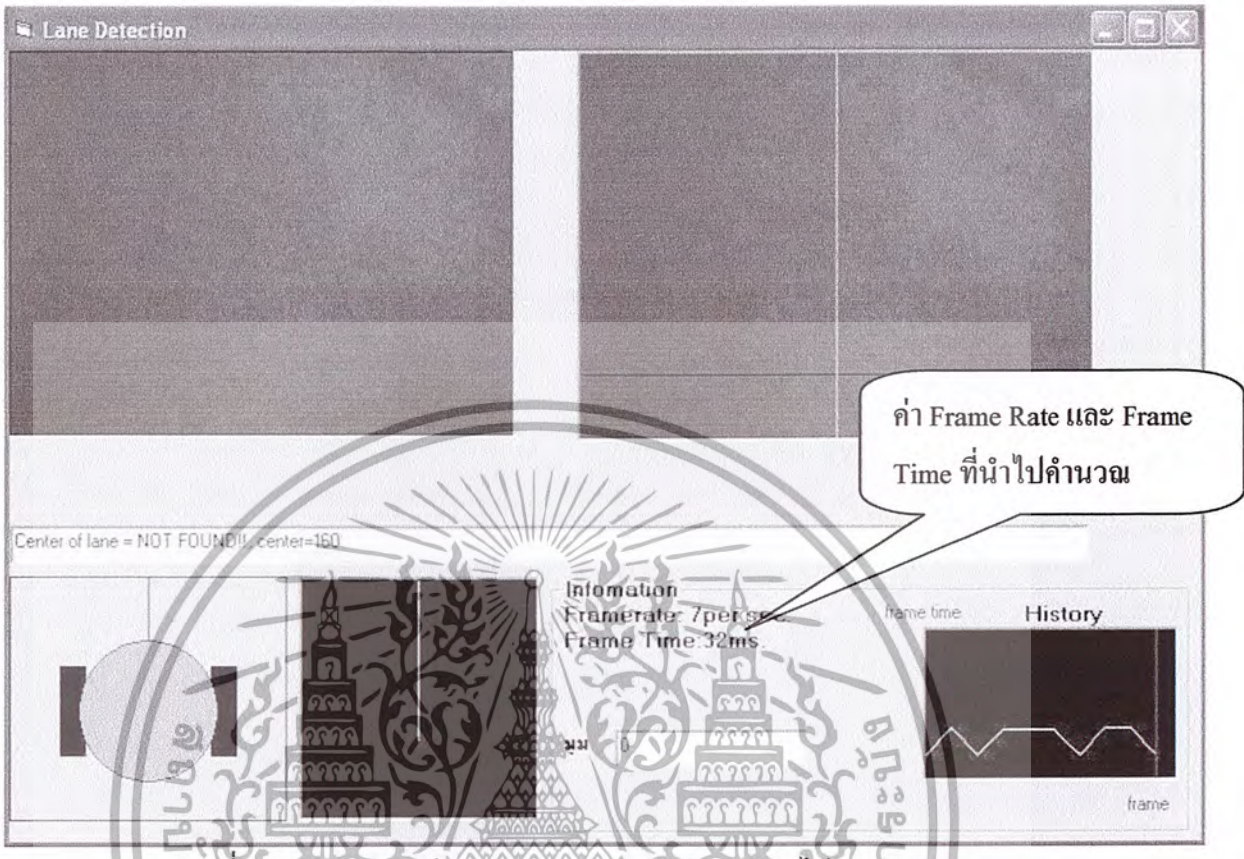
มุมของถนน (ทำกับทางตรง)	5°	10°	15°	20°	25°	30°	35°	40°
การเคลื่อนที่ตามเส้น	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ

มุมของถนน (ทำกับทางตรง)	45°	50°	55°	60°	65°	70°	75°	80°
การเคลื่อนที่ตามเส้น	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ	ปกติ

มุมของถนน (ทำกับทางตรง)	85°	90°	95°	100°	105°	110°	115°	120°
การเคลื่อนที่ตามเส้น	ปกติ	ไม่ปกติ	ไม่ปกติ	ไม่ปกติ	ไม่ปกติ	ไม่ปกติ	ไม่ปกติ	ไม่ปกติ

ตารางที่ 4.4 แสดงการทำงานของรถเทียบกับมุมต่างๆ

4.4 การคำนวณหาค่าเวลาการทำงานต่างๆ ของโปรแกรม



รูปที่ 4.7 แสดงการนำค่า Frame Rate และ Frame Time ไปคำนวณ

จากการเขียนโปรแกรม จับเวลาจะได้ Frame Time ของโปรแกรมอยู่ประมาณ 15-32 ms และค่า Frame Rate ของกล้องที่ทำการทดลองมีค่าสูงสุด 15 เฟรมต่อวินาที (f/s)

เพราะฉะนั้น กล้องใช้เวลาในการทำงานต่อ Frame มีค่า $\frac{1}{15}$ หรือ 66.7 ms ซึ่งทำให้โปรแกรมสามารถทำงานได้ ภาพต่อภาพเมื่อได้รับภาพจากกล้องมา

เมื่อคิดคำนวณเวลารวมทั้งระบบคือเวลาการทำงานรวมกับการทำงานของคอนโทรลเลอร์ ซึ่งจากการคำนวณเวลาการทำงานของคอนโทรเลอร์จะใช้เวลาประมาณ 25 ms ซึ่งเมื่อรวมเวลากับการทำงานของโปรแกรมแล้ว จะมีค่าประมาณ 40-57 ms

เพราะฉะนั้นระบบจะทำการประมวลผลโดยรับภาพจาก Frame ต่อ Frame เหมือนเดิม ซึ่งคอนโทรลเลอร์จะได้รับคำสั่งจากคอมพิวเตอร์ทุก $\frac{1}{15}$ หรือ 66.7 ms เนื่องจาก Frame Rate ของกล้องมีค่าจำกัด

บทที่ 5
บทสรุป

จากโครงงานนี้จะเห็นว่าประกอบด้วยส่วนหลัก ๆ สองส่วน คือ ส่วนของฮาร์ดแวร์และส่วนของซอฟแวร์ ซึ่งส่วนของฮาร์ดแวร์เป็นส่วนของตัวรถ ส่วนของซอร์ฟแวร์จะเป็นส่วนของการประมวลผลภาพเลนถนน โดยทั้งสองส่วนจะทำงานร่วมกันเพื่อให้รถเคลื่อนที่ไปตามเลนถนน

ในการขับเคลื่อนรถนั้นจะอาศัยการประสานงานกันระหว่างกล้องดิจิตอล วงจรไดร์มอเตอร์ และไมโครคอนโทรลเลอร์ โดยจะแสดงผลทางจอคอมพิวเตอร์ พร้อมทั้งควบคุมการเคลื่อนที่ของรถ

โดยเมื่อสั่งให้รถเคลื่อนที่ กล้องดิจิตอลจะรับภาพจากเลนส์ถนนแล้ว ประมวลผลโดยผ่านกระบวนการได้แก่ การเทรชโฮลด์ และการหาจุดกึ่งกลางของเลนส์ โดยการเทรชโฮลด์จะแปลงจากภาพปกติ (Normal View) เป็นภาพที่ประกอบด้วยพิกเซลขาวดำ แล้วจะทำการหาจุดกึ่งกลางของถนนต่อไปจากตาราง array จากจุดกึ่งกลางเลนส์ที่ได้จึงนำมาประมวลผลเพื่อควบคุมทิศทางการเคลื่อนที่ของรถต่อไป

ในส่วนวงจรควบคุมการเคลื่อนที่ของรถนั้น จะใช้คอนโทรเลอร์ (MCS-51) ซึ่งรับข้อมูลจาก Browser โดยผ่านมาทางพอร์ตอนุกรมเพื่อให้คอนโทรเลอร์ควบคุมการหมุนของ Stepper Motor ทำให้ตัวรถเคลื่อนที่ไปตามทิศทางที่ถูกต้อง

ดังนั้นแนวทางในการปรับปรุงก็คือ ควรใช้กล้องดิจิตอลและอุปกรณ์ในการประมวลผลที่มีความเร็วและประสิทธิภาพสูง ควรที่จะมีการจัดสภาพแวดล้อมให้เหมาะกับการประมวลผล ส่วนสภาพของแสงที่ตกกระทบสามารถปรับเปลี่ยนได้ที่โปรแกรม Setup ให้เข้ากับสภาพของแสงได้ รวมทั้งการคอนโทรลตัวรถควรจะทำให้นุ่มนวลเพื่อให้กล้องจับภาพได้อย่างมีประสิทธิภาพมากขึ้น

ภาคผนวก

- Source code โปรแกรมภาษาแอสเซมบลี่ของไมโครคอนโทรลเลอร์ควบคุมการหมุนของสเต็ปเปอร์มอเตอร์
- Source code ของโปรแกรมรับ ประมวลผลภาพและควบคุมการเคลื่อนที่ของรถ

Source code โปรแกรมภาษาแอสเซมบลี่ของไมโครคอนโทรลเลอร์ควบคุมการหมุนของสเต็ปเปอร์มอเตอร์

```
DRIVER_LE   BIT     P1.4
STEP    EQU         030H    ; For keep Stepper State
ORG     0000H
; กำหนดตั้งค่าให้กับ Timer1 เพื่อใช้สร้าง Baud Rate ในการรับส่งข้อมูล
        MOV         IE,#00000000B
        MOV         TMOD,#00100000B
        MOV         TL1,#0FDH
        MOV         TH1,#0FDH
; กำหนดค่าให้กับ SCON Register โดยเลือกโหมด 1 และให้รับข้อมูล REN=1
        MOV         SCON,#01010000B
        SETB        TR1
        MOV         P0,#00000000B
INDEX:  ACALL       SUB_RXD
        MOV         R0,#7
        SETB        DRIVER_LE
LED1:   CJNE        A,#31H,LED2
STEPLEFT:   MOV     STEP,#00110011B     ;Drive Motor ให้เลี้ยวซ้าย
        MOV         P0,STEP
        ACALL       DELAY_100ms
        MOV         STEP,#01100110B
        MOV         P0,STEP
        ACALL       DELAY_100ms
        MOV         STEP,#11001100B
        MOV         P0,STEP
        ACALL       DELAY_100ms
        MOV         STEP,#10011001B
        MOV         P0,STEP
        ACALL       DELAY_100ms
        DJNZ        R0,STEPLEFT
        SJMP        INDEX
```

```
LED2:       CJNE    A,#32H,LED3     ;Drive Motor ให้ตรง
STEPGO:     MOV     STEP,#11000011B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            MOV     STEP,#01100110B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            MOV     STEP,#00111100B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            MOV     STEP,#10011001B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            DJNZ    R0,STEPGO
            SJMP    INDEX
LED3:       CJNE    A,#33H,INDEX    ;Drive Motor ไปทางขวา
STEPRIGHT:  MOV     STEP,#11001100B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            MOV     STEP,#01100110B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            MOV     STEP,#00110011B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            MOV     STEP,#10011001B
            MOV     P0,STEP
            ACALL   DELAY_100ms
            DJNZ    R0,STEPRIGHT

            SJMP    INDEX
```

```
; รูทีนรับข้อมูลทาง Serial Port

SUB_RXD:        JNB     RI,$
                CLR     RI
                MOV     A,SBUF
                RET

; รูทีนส่งข้อมูลทาง Serial Port เพื่อบอกให้โปรแกรมที่ PC รู้ว่าส่งคำสั่งทำงานเรียบร้อยแล้ว

SUB_TXD:        MOV     SBUF,#43
                JNB     TI,$
                CLR     TI
                RET

;/---------------------------------------------------------

DELAY_10ms:     MOV     R7,#010
DELAY_10ms_1:   MOV     R6,#0E6H
DELAY_10ms_2:   NOP
                NOP
                DJNZ    R6,DELAY_10ms_2
                DJNZ    R7,DELAY_10ms_1
                RET

DELAY_1s:       MOV     R5,#100
DELAY_1s_1:     ACALL   DELAY_10ms
                DJNZ    R5,DELAY_1s_1
                RET

;////////////////////////////////////////////////////////////////

DELAY_100ms:    MOV     R7,#10
DELAY_100ms_1:  MOV     R6,#080H
DELAY_100ms_2:  NOP
                NOP
```

```
DJNZ        R6,DELAY_100ms_2
DJNZ        R7,DELAY_100ms_1
RET

END
```

Source code ของโปรแกรมรับ ประมวลผลภาพและควบคุมการเคลื่อนที่ของรถ

```
Option Explicit

Private Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long,
ByVal y As Long) As Long
Private Declare Function SetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long,
ByVal y As Long, ByVal crColor As Long) As Long
Private Declare Function GetTickCount Lib "kernel32" () As Long


Dim bEndProgram As Boolean
Dim frameRate As Long
Dim nextTimeChkFPS As Long
Dim frameCnt As Long
Dim startFrameTime As Long
Dim endFrameTime As Long
Dim frameTime As Long

Dim frameTimeHistory(10) As Long

Private Function doCheckLane(ByVal y As Long) As Single
    ezVidCap1.SaveDIB App.Path & "\capture.bmp"
    picCapture.Picture = LoadPicture(App.Path & "\capture.bmp")
    picCapture.Refresh
    'picCapture.Line (0, 230)-(picCapture.ScaleWidth, 230), QBColor(15)

    Dim x As Long
    Dim tempColor As Long
    Dim red As Long
    Dim green As Long
    Dim blue As Long
    Dim picData() As Long
    Dim picdatafi() As Long
```

```
ReDim picData(picCapture.ScaleWidth) As Long
ReDim picdatafi(picCapture.ScaleWidth) As Long
Dim x1 As Long
Dim x2 As Long
Dim i As Long
Dim a As Long
Dim b As Long
Dim Fi_Arr As Long
Dim Fi_Sum As Long
Dim Nav As Long
Dim Sum1 As Long
For x = 2 To picCapture.ScaleWidth - 2
   tempColor = GetPixel(picCapture.hdc, x, y)
   red = tempColor And &HFF
   green = (tempColor \ 256) And &HFF
   blue = (tempColor \ 65536) And &HFF
   If (red > 100) And (green > 100) And (blue > 100) Then
      SetPixel picCapture.hdc, x, y, &HFFFFFF
      picData(x) = 1
   Else
      SetPixel picCapture.hdc, x, y, &H0
      picData(x) = 0
   End If
Next

For i = 0 To UBound(picData) - 1
   If (picData(i) > 0) And (picData(i + 1) = 1) Then
      picdatafi(i) = picData(i) + picData(i + 1)
      picData(i + 1) = picdatafi(i)

   Else:  picdatafi(i) = 0
End If
```

```
    Next

    a = 1
    b = 2
    Do Until a = UBound(picdatafi)
    If picdatafi(a) > picdatafi(b) Then
    b = a
    End If
    a = a + 1
    Loop

    Fi_Arr = b
    Fi_Sum = picdatafi(b)

    picCapture.Line (picCapture.ScaleWidth / 2, 0)-(picCapture.ScaleWidth / 2,
            picCapture.ScaleHeight), QBColor(14)

    doCheckLane = Fi_Arr - ((Fi_Sum + 1) / 2)

End Function

Private Sub Form_Load()
    bEndProgram = False
    Me.Show
    Dim centerlane As Single
    Dim beforecenterlanemain As Single
    Dim beforecenterlanesub As Single

    '//------------------
    '// k vary
    Dim k As Single
    Dim carAngle As Single
```

```
Dim carAngleR As Single
k = 100.25
Dim i1 As Long
Dim ii As Long
'//------------------
MSComm1.Settings = "9600,N,8,1"
MSComm1.CommPort = 1
MSComm1.InputLen = 1
MSComm1.PortOpen = True
MSComm1.RThreshold = 1

Do
  startFrameTime = GetTickCount()
  beforecenterlanemain = doCheckLane(60)
  beforecenterlanesub = doCheckLane(200)

  If (beforecenterlanemain = 1.5) And (beforecenterlanesub = 1.5) Then
    Text1.Text = "Center of lane = NOT FOUND!!, center=" & picCapture.ScaleWidth / 2
    carAngleR = 0
    carAngle = 0
    txtAngle.Text = carAngle
    Picture1.Cls
    Picture1.Line (75, 100)-(75, 0), QBColor(15)
  ElseIf beforecenterlanemain = 1.5 Then
    centerlane = beforecenterlanesub
    Text1.Text = "Center of lane = " & centerlane & " center=" & picCapture.ScaleWidth / 2
    carAngleR = Atn((centerlane - (picCapture.ScaleWidth / 2)) / k)
    carAngle = carAngleR * 180 / 3.14
    txtAngle.Text = carAngle
    Picture1.Cls
    Picture1.Line (75, 100)-(Cos(carAngleR - 1.57) * 100 + 75, Sin(carAngleR - 1.57) * 100
     + 100), QBColor(15)
```

```
    Else
       centerlane = beforecenterlanemain
       Text1.Text = "Center of lane = " & centerlane & " center=" & picCapture.ScaleWidth / 2
       carAngleR = Atn((centerlane - (picCapture.ScaleWidth / 2)) / k)
       carAngle = carAngleR * 180 / 3.14
       txtAngle.Text = carAngle
       Picture1.Cls
       Picture1.Line (75, 100)-(Cos(carAngleR - 1.57) * 100 + 75, Sin(carAngleR - 1.57) * 100
+ 100), QBColor(15)
    End If

    frameCnt = frameCnt + 1
    If GetTickCount() >= nextTimeChkFPS Then
       frameRate = frameCnt
       frameCnt = 0
       nextTimeChkFPS = GetTickCount() + 1000
    End If

    endFrameTime = GetTickCount()
    frameTime = endFrameTime - startFrameTime

    lblFPS.Caption = "Framerate:" & frameRate & "per sec." & vbCrLf & "Frame Time:" &
       frameTime & "ms."
    For i1 = 0 To 9
       frameTimeHistory(i1) = frameTimeHistory(i1 + 1)
    Next
    frameTimeHistory(10) = frameTime
    picHistory.Line (0, 0)-(picHistory.ScaleWidth, picHistory.ScaleHeight), QBColor(0), BF
    picHistory.Line (0, picHistory.ScaleHeight - frameTimeHistory(0))-(0,
       picHistory.ScaleHeight - frameTimeHistory(0)), QBColor(14)
    ii = 0
    For i1 = 0 To picHistory.ScaleWidth Step picHistory.ScaleWidth / 10
```

```
        picHistory.Line -(i1, picHistory.ScaleHeight - frameTimeHistory(ii)), QBColor(14)
        If ii = 9 Then
           picHistory.Line (i1, 0)-(i1, picHistory.ScaleHeight), QBColor(4)
        End If
        ii = ii + 1
      Next

      If carAngle < -15 Then
      MSComm1.Output = "1"
      ElseIf carAngle > 15 Then
      MSComm1.Output = "3"
      Else
      MSComm1.Output = "2"

      End If

      DoEvents
   Loop Until bEndProgram = True
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
   bEndProgram = True
End Sub

Private Sub ¢¹Ò´àÊé¹_Click()


End Sub
```

กิตติกรรมประกาศ

โครงงานฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี ทั้งนี้ได้รับคงามช่วยเหลือ คำแนะนำและให้ทำปรึกษาจาก รศ.ดร. สุรพันธ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษาเป็นอย่างดี อีกทั้งให้ใช้ห้องทดลองและให้ยืมอุปกรณ์ทดลองต่างๆ ที่สำคัญๆ รวมทั้งเพื่อนๆ ที่ทำโปรเจคในห้องโปรเจคเดียวกันและห้องข้างเคียงที่ให้ความช่วยเหลือ ให้หยิบยืมอุปกรณ์ในส่วนที่ยังขาดให้สมบูรณ์ยิ่งขึ้น และคำแนะนำวิธีแก้ปัญหาจากเพื่อนๆ ที่มีประสบการณ์ รวมทั้งกำลังใจและน้ำใจจากเพื่อนๆ ทุกคน ขอขอบคุณทุกๆ ท่านที่ให้การสนับสนุนเป็นอย่างดีและจะขอเก็บความรู้สึกดีๆ นี้ไว้ตลอดไป

บรรณานุกรม

1. นายเกศ บัวลำใย, น.ส.ปรียาภรณ์ ภักดี; "ปริญญานิพนธ์เรื่องการตรวจจับเลนถนนโดยคอมพิวเตอร์"
2. นายมนูญ สันถวะคุปต์, นายมานะ รวมกิจธรรม; "ปริญญานิพนธ์เรื่องรถขับเคลื่อนอัตโนมัติ"
3. เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51;
   กฤษฎา ใจเย็น, ชัยวัฒน์ ลิ้มพรจิตรวิไล; บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด
4. เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51แบบแฟลช;
   วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล; บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

Paper Reference

- การตรวจจับและติดตามเลนถนนโดยใช้การประมวลผลภาพ;
  รศ.ดร.สุรพันธ์ เอื้อไพบูลย์ และ รศ.ดร.มนัส สังวรณ์ศิลป์

เว็บไซต์ที่ได้ค้นหาข้อมูลทางอินเตอร์เน็ต

http://www.thaiio.com