การควบคุมการเคลื่อนที่โดยใช้กล้องตรวจจับ

CONTROLLED ROBOT BY DETECTING CAMERA

โดย

นางสาวฉักษิกา วิบูลเทพาชาติ

นายเอกพล ชัยเจริญชนะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การควบคุมการเคลื่อนที่โดยใช้กล้องตรวจจับ

CONTROLLED ROBOT BY DETECTING CAMERA

อาจารย์ที่ปรึกษา

อาจารย์สุมิตร พนาอุดมทรัพย์

โดย

นางสาวลักษิกา วิบูลเทพาชาติ 43010362

นายเอกพล ชัยเจริญชนะ 43010554

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชา วิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมการเลื่อนที่โดยใช้กล้องตรวจจับ

ผู้จัดทำ

1. นางสาวลักษิกา วิบูลเทพาชาติ
2. นายเอกพล ชัยเจริญชนะ

_____ อาจารย์ที่ปรึกษา

(อาจารย์สุมิตร พนาอุดมทรัพย์)

การควบคุมการเคลื่อนที่โดยใช้กล้องตรวจจับ

CONTROLLED ROBOT BY DETECTING CAMERA

นางสาวลักษิกา วิบูลเทพาชาติ
นายเอกพล ชัยเจริญชนะ
อ. สุมิตร พนาอุดมทรัพย์ อาจารย์ที่ปรึกษา

ปีการศึกษา 2546

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการออกแบบและสร้างระบบควบคุมหุ่นยนต์ที่สามารถเคลื่อนที่โดยใช้กล้องตรวจจับและเคลื่อนที่ไปยังตำแหน่งที่ต้องการ พร้อมทั้งสามารถหลบหลีกสิ่งกีดขวางได้ตามต้องการ โดยจะใช้กล้องวีดีโอในการจับภาพหุ่นยนต์และสิ่งกีดขวาง และส่งให้คอมพิวเตอร์ส่วนบุคคลแสดงผล แล้วคอมพิวเตอร์จะทำการประมวลภาพและส่งสัญญาณควบคุมไร้สายให้กับไมโครคอนโทรลเลอร์ PIC16F84A ควบคุมการทำงานของหุ่นยนต์

ABSTRACT

This project presents the design and construction of robot control by detecting camera system in order to move the robot avoid the obstruction and sends the signal from camera to the personal computer for displaying the capture picture. Then personal computer will process the captured pictures and sends the wireless control signal to the microcontroller PIC16F84A to the movement of the robot.

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

1) ความเป็นมา

หุ่นยนต์หรือโรบอท (ROBOT) เป็นสิ่งประดิษฐ์หรือเครื่องจักรกลที่สามารถทำงานต่างๆ แทนมนุษย์ได้ ซึ่งลักษณะของหุ่นยนต์ต่างๆที่ใช้ในงานทางด้านอุตสาหกรรมนั้นสามารถแบ่งได้เป็น 2 ประเภท คือ

1) MOBILE ROBOT

2) FIXED ROBOT

โดยในอุตสาหกรรมหลายประเภทนั้นหากมีการนำเอาหุ่นยนต์เข้ามาใช้แทนแรงงานมนุษย์แล้ว พบว่าจะทำให้มีต้นทุนการผลิตลดลง เนื่องจากหุ่นยนต์มีความสามารถบางอย่างที่เหนือกว่ามนุษย์และมีความเหมาะสมกว่า เช่น งานที่ต้องใช้แรงมากๆหรืองานที่อันตรายที่มนุษย์ไม่สามารถทำได้ ในส่วนนั้นเองที่เหมาะสมที่จะใช้หุ่นยนต์เป็นอย่างยิ่ง อีกทั้งหุ่นยนต์สามารถทำงานได้มากกว่ามนุษย์และไม่มีปัจจัยด้านบุคคลต่างๆเข้ามาเกี่ยวข้อง

ในอดีตการควบคุมหุ่นยนต์จะใช้ระบบแมนนวล(MANUAL)ซึ่งควบคุมโดยมนุษย์ต่อมาได้มีเทคโนโลยี เซนเซอร์เข้ามาใช้ในการควบคุมหุ่นยนต์ทำให้หุ่นยนต์สามารถทำงานได้โดยตัวมันเอง หากแต่ยังมีข้อ จำกัดหลายประการ แต่ในปัจจุบันเทคโนโลยีการประมวลผลด้วยภาพ (image processing) ได้เข้ามามีบทบาทเป็นอย่างมากในการควบคุมหุ่นยนต์ ซึ่งช่วยลดข้อจำกัดในการใช้งานหุ่นยนต์แบบอัตโนมัติให้ลดลงและหุ่นยนต์สามารถทำงานได้หลากหลายและประสิทธิภาพสูงขึ้นได้

จากเหตุผลตามที่กล่าวมาข้างต้นเป็นเหตุผลที่ทำให้เกิดโครงงานนี้ขึ้น โดยในโครงงานนี้จะจำลองระบบการทำงานของโกดังหรือภายในโรงงานที่มีการเก็บของเป็นสัดส่วนและมีน้ำหนักมาก อีกทั้งยังมีการเคลื่อนย้ายขนส่งไม่เป็นเวลา เพื่อที่จะตัดปัญหาดังกล่าวในด้านแรงงานคน จึงมีความคิดที่จะใช้หุ่นยนต์ควบคุมด้วยระบบคอมพิวเตอร์ผ่านกล้องเข้ามาแทนในการตัดปัญหาดังกล่าว

2) จุดประสงค์

1) สั่งงานหุ่นยนต์ด้วยคอมพิวเตอร์ โดยประมวลผลจากภาพที่ได้จากกล้อง
2) ศึกษาการทำงานและใช้งานวงจรอินฟราเรดได้
3) ศึกษาการทำงานและการใช้งานไมโครคอนโทรลเลอร์ตระกูล PIC ได้
4) การเขียนโปรแกรม Visual C++ ในการประมวลผลจากภาพได้

3) ขอบเขต

ขอบเขตการทำงาน สามารถสร้างโปรแกรมควบคุมหุ่นยนต์ให้สามารถวิ่งไปตามเส้นทางที่ไม่ชนสิ่งกีดขวางไปยังตำแหน่งที่กำหนดไว้ สุดท้ายให้รถวิ่งกลับไปยังตำแหน่งเริ่มต้น ซึ่งขอบเขตของโครงงานจะแบ่งเป็น 2 ส่วน คือ

3.1) Software – ซอฟแวร์

- โปรแกรมประมวลผลและสั่งงานหุ่นยนต์จากกล้อง
- โปรแกรมควบคุมไมโครคอนโทรลเลอร์

3.2) Hardware – ฮาร์ดแวร์

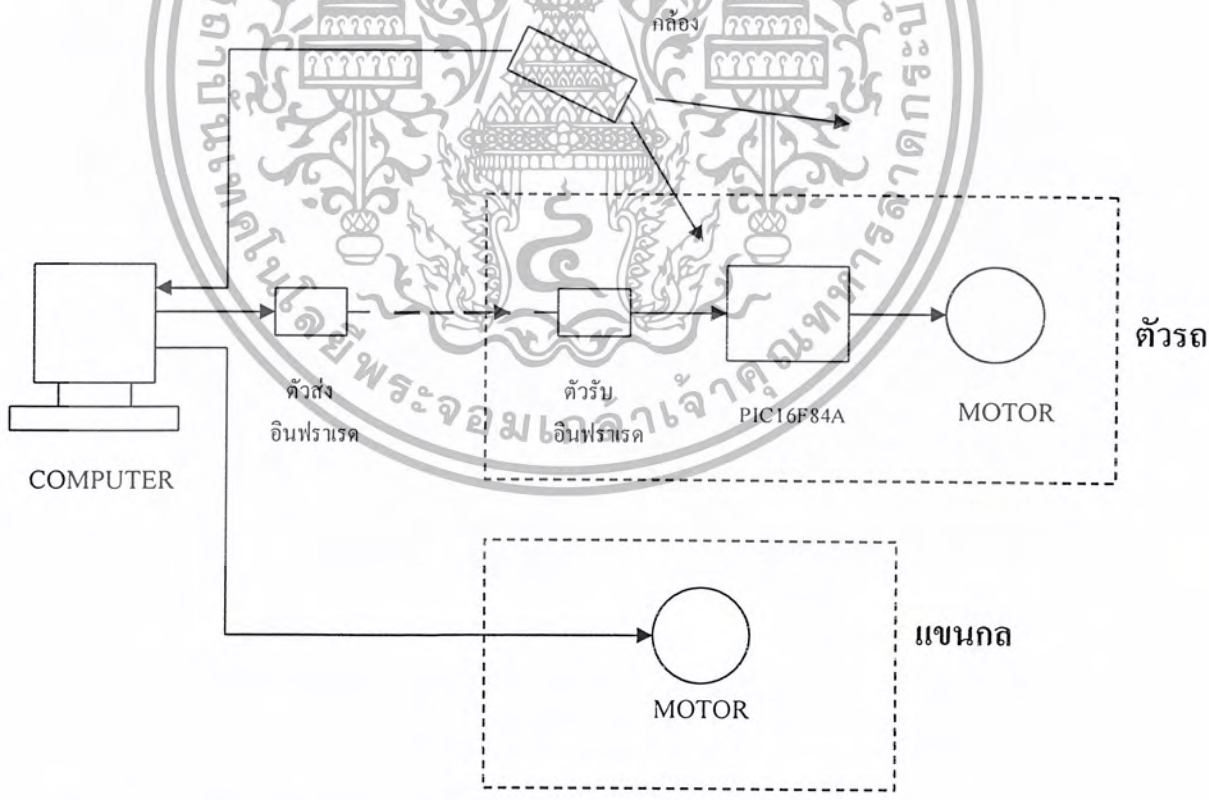
- ตัวรถ
- ตัวรับสัญญาณอินฟราเรด
- ตัวส่งสัญญาณอินฟราเรด

บทที่ 2
ทฤษฎีและหลักการ

1) ขั้นตอนการทำงาน

เริ่มต้นกล้องวิดีโอจะจับภาพมาแล้วส่งให้คอมพิวเตอร์ทำการจำลองภาพให้เป็นรูปภาพแบบง่ายๆ เมื่อทำการระบุตำแหน่งที่ต้องการให้หุ่นยนต์ไป เครื่องคอมพิวเตอร์จะทำการคำนวณหาเส้นทางที่หุ่นยนต์สามารถดำเนินไปได้ หลังจากนั้นคอมพิวเตอร์จะป้อนคำสั่งที่ใช้ในการควบคุมหุ่นยนต์โดยส่งข้อมูลออกทางพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ผ่านตัวรับส่งอินฟราเรด เมื่อไมโครคอนโทรลเลอร์ ในหุ่นยนต์ได้รับคำสั่งแล้วก็จะทำการเดินตามคำสั่งที่ได้รับมา จากนั้นกล้องต้องคอยจับภาพเพื่อคอยดูว่าหุ่นยนต์เดินไปยังตำแหน่งที่ต้องการหรือไม่ ถ้ายังไปไม่ถึงก็จะยังคงส่งคำสั่งไปให้หุ่นยนต์จนกว่าหุ่นยนต์จะอยู่ที่ตำแหน่งที่ต้องการ

2) รูปแบบการทำงานของระบบ



รูปที่ 2.1 แผนภาพบล็อกไดอะแกรมแสดงการทำงานของหุ่นยนต์ควบคุมด้วยคอมพิวเตอร์

ระบบสามารถถูกแบ่งออกเป็น 5 ส่วนดังนี้

ส่วนคอมพิวเตอร์ จะเป็นตัวรับคำสั่งจากผู้ใช้งานและรับภาพที่ส่งมาจากกล้อง เพื่อนำมาคำนวณ แล้วคอมพิวเตอร์จะทำการแปลงข้อมูลที่ใช้ในการควบคุมให้อยู่ในรูปแบบที่จะใช้ส่งผ่านพอร์ตอนุกรมเพื่อที่จะส่งให้ตัวส่งอินฟราเรด และ ส่งไปยังมอเตอร์ของแขนกล

ส่วนรับภาพ ใช้กล้องดิจิตอลเป็นตัวจับภาพจากทางด้านบน โดยครอบคลุมพื้นที่สำรวจแล้วส่งสัญญาณภาพมาให้คอมพิวเตอร์ผ่านทางพอร์ต USB ตัวกล้องจะถูกยึดติดกับเสาเพื่อให้กล้องสามารถมองจากมุมบนได้

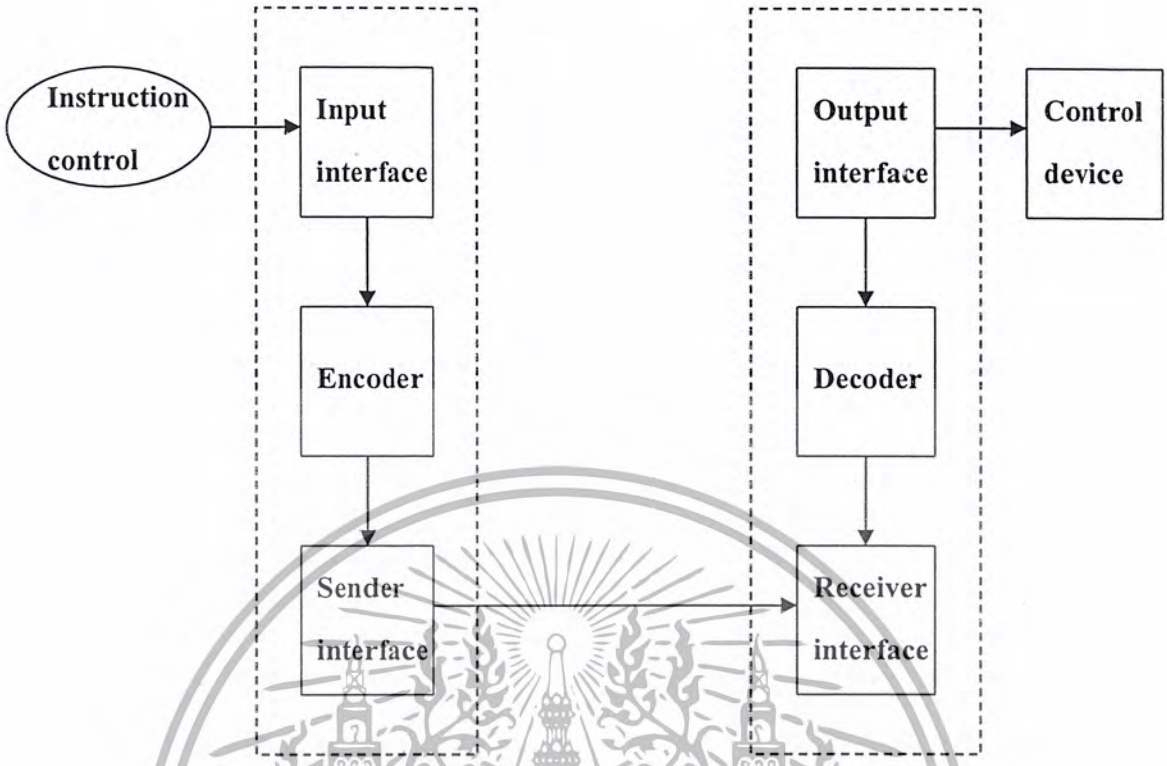
ส่วนการขับเคลื่อนดีซีมอเตอร์ของรถ เป็นส่วนที่ทำให้หุ่นยนต์สามารถเคลื่อนที่ได้ โดยจะใช้ดีซีมอเตอร์ทั้งหมด 2 ตัว ซึ่งสามารถควบคุมความเร็วและทิศทางของมอเตอร์ได้จากพัลส์ (pulse) ที่ส่งไปให้มอเตอร์

ส่วนไมโครคอนโทรลเลอร์ จะทำหน้าที่ในการควบคุมการทำงานของหุ่นยนต์ทั้งหมด โดยได้รับคำสั่งการควบคุมจากคอมพิวเตอร์ที่ส่งมาทางพอร์ตอนุกรม ผ่านตัวรับส่งอินฟราเรด และทำการควบคุมการหมุนของมอเตอร์ให้เป็นไปตามคำสั่งโดยการส่งพัลส์ไปให้

ส่วนการติดต่อสื่อสาร ทำหน้าที่เป็นตัวเชื่อมต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ส่วนนี้ประกอบด้วย การเชื่อมต่อผ่านพอร์ตอนุกรมและส่งข้อมูลต่อด้วยอินฟราเรด และ การเชื่อมต่อกับแขนกลโดยสายไฟธรรมดา

3) หลักการของรีโมทคอนโทรล

บล็อกไดอะแกรมในรูปที่ 2.2 แสดงโครงสร้างและหลักการทำงานของระบบควบคุมระยะไกล โดยทั่วไป ในลักษณะของการควบคุมแบบทางเดียว เริ่มจากตัวกำหนดคำสั่งที่ใช้สำหรับการควบคุมว่า มีคำสั่งอะไรบ้าง ชุดคำสั่งทั้งหมดมีกี่คำสั่ง เป็นต้น เมื่อมีการกำหนดรูปแบบของคำสั่งแล้ว รูปแบบของคำสั่งที่ถูกเลือกจะถูกส่งไปยังภาคส่งสัญญาณที่ทำหน้าที่แปลงสัญญาณ หรือรวมสัญญาณควบคุมให้มีรูปแบบที่เหมาะสมกับวงจร



รูปที่ 2.2 แผนภาพบล็อกไดอะแกรมแสดงการทำงานของรีโมทคอนโทรล

โดยอาจทำการเข้ารหัสสัญญาณให้แต่ละคำสั่งมีรหัสเฉพาะของตัวมันเอง ให้เป็นสัญญาณไฟฟ้าก่อนที่จะถูกส่งออกไปยังภาครับ โดยตัวอินเตอร์เฟสด้านตัวส่งเพื่อทำหน้าที่ส่งสัญญาณที่โดยอาจทำการเข้ารหัสสัญญาณให้แต่ละคำสั่งมีลักษณะเฉพาะของตัวมันเอง ให้เป็นสัญญาณทางไฟฟ้าก่อนที่จะถูกส่งออกไปยังภาครับ โดยตัวอินเตอร์เฟสด้านตัวส่ง เพื่อทำหน้าที่ส่งสัญญาณที่ภาครับต้องเข้าใจได้ นั่นก็คือต้องเป็นระบบเดียวกัน สัญญาณไฟฟ้า สัญญาณแสง หรือสัญญาณความถี่สูง สัญญาณนี้สามารถเดินทางผ่านตัวกลางที่เป็นสายนำสัญญาณ หรือผ่านตัวกลางอากาศ ขึ้นอยู่กับระบบที่ถูกออกแบบ

หากใช้สายสัญญาณเป็นตัวนำสัญญาณจะเรียกว่าระบบใช้สาย ซึ่งถ้าใช้สัญญาณไฟฟ้าเป็นสัญญาณควบคุม (ที่มีการจัดรูปแบบหรือเข้ารหัสแล้ว) ก็จะใช้สายไฟฟ้าเป็นตัวนำสัญญาณแต่หากถ้าใช้สัญญาณแสงเป็นตัวควบคุม ตัวนำสัญญาณจะเป็นเส้นใยแก้วนำแสงหรือไฟเบอร์ออฟติก ในกรณีที่สัญญาณควบคุมถูกส่งไปในอากาศ เพื่อเดินทางไปยังเครื่องรับ เช่น การใช้สัญญาณไฟฟ้าในรูปของคลื่นวิทยุ หรือการใช้สัญญาณแสงอินฟราเรดโดยตรง ระบบจะมีชื่อว่า ระบบไร้สายระบบนี้เองที่กำลังเป็นที่นิยมใช้กันอยู่มากในปัจจุบัน

สัญญาณที่เข้ามายังเครื่องรับหรือภาครับ จะถูกตัวอินเตอร์เฟสทำหน้าที่แปลงสัญญาณ ให้อยู่ในรูปสัญญาณไฟฟ้าเข้ากับระบบของตัวรับ ก่อนถูกถอดรหัสเพื่อทราบวัตถุประสงค์ของคำสั่ง จากนั้นส่วนของวงจรอินเตอร์เฟสด้านเอาท์พุทจะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ที่ต้องการตามลักษณะคำสั่งที่
ได้รับ

ระบบที่กล่าวถึงมานั้นเป็นระบบรีโมทคอนโทรลหรือการควบคุมระยะไกลแบบทางเดียวที่มีการสั่งงานจากจุดหนึ่งแล้วเกิดการทำงานขึ้นอีกจุดหนึ่ง หากจุดที่ถูกสั่งให้ทำงาน มีความสามารถในการกลับมายังจุดเริ่มต้น ให้ทำงานได้ด้วยแล้วแสดงว่าการทำงานของจุดหรือตำแหน่งทั้งสองมีความเสมอภาคกัน อย่างนี้ถือเป็นระบบควบคุมระยะไกลแบบสองทาง ซึ่งมักปรากฏให้เห็นอย่างชัดเจนในระบบการสื่อสารทั่วไป

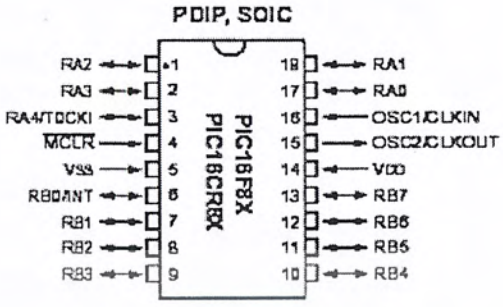
4) ไมโครคอนโทรลเลอร์ตระกูล PIC

เนื่องจากโครงสร้างนี้ มีการใช้ไมโครคอนโทรลเลอร์เพียงเป็นตัวเข้าและถอดรหัสเท่านั้น ดังนั้นจึงไม่ขอกล่าวถึงอย่างละเอียดนัก จึงจะขอกล่าวเพียงแค่คุณสมบัติโดยทั่วไป โครงสร้างภายนอก และ การต่อขาในการใช้งานเท่านั้น

4.1) คุณสมบัติทั่วไปโดยหลักๆ

- เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต
- RAM 68 Bytes
- EEPROM 64 Bytes
- ขาอินพุทเอาท์พุทจำนวน 35 ชุดต่อคำสั่ง
- มีไทเมอร์/เคาน์เตอร์
- แหล่งจ่ายไฟขนาด 2 – 6 Volt
- มีฟังก์ชันการทำงานพิเศษที่ช่วยลดอุปกรณ์ภายนอก
- สามารถ FLASH ข้อมูลเข้าสู่ IC ได้โดยตรง

4.2) โครงสร้างภายนอก

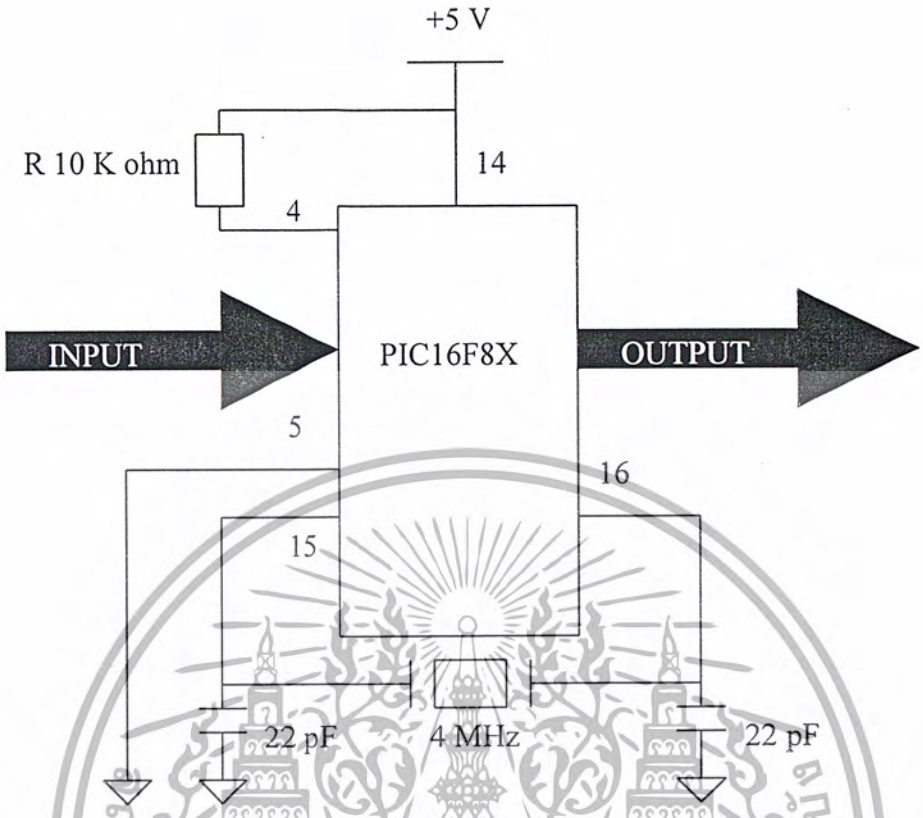


รูปที่ 2.3 โครงสร้างภายนอกของไมโครคอนโทรลเลอร์ PIC 18 ขา

การจัดขาของไมโครคอนโทรลเลอร์ PIC 18 ขา

- OSC1/CLKIN : เป็นขาต่อคริสตัลด้านอินพุท แหล่งกำเนิดสัญญาณนาฬิกาจากภายนอก
- OSC2/CLKOUT : เป็นขาต่อคริสตัลด้านเอาท์พุท
- MCLR : ขามาสเตอร์เคลียร์ (รีเซต) แอคทิฟโลว์
- RA0 – RA3 : พอร์ตอินพุท/เอาท์พุท
- RA 4/T0CK1 : เป็นทั้งพอร์ทอินพุท/เอาท์พุตและใช้เป็นขาอินพุทสำหรับไทเมอร์ TMR0
- RB0/INT : เป็นทั้งพอร์ทอินพุท/เอาท์พุทและเป็นขาสำหรับอินเทอร์รัพจากภายนอก
- RB1 – RB2 : พอร์ทอินพุท/เอาท์พุท
- RB6 : เป็นทั้งพอร์ทอินพุท/เอาท์พุท และใช้เป็นขาสัญญาณนาฬิกาสำหรับการ FLASH
- RB7 : เป็นทั้งพอร์ทอินพุท/เอาท์พุท และใช้เป็นขารับข้อมูลสำหรับการ FLASH
- V_{SS} : กราวด์ของ IC
- V_{DD} : ไฟเลี้ยง IC

การต่อขาสำหรับใช้งาน



รูปที่ 2.4 การต่อใช้งานไมโครคอนโทรลเลอร์

5) ทฤษฎีของการประมวลผลภาพ

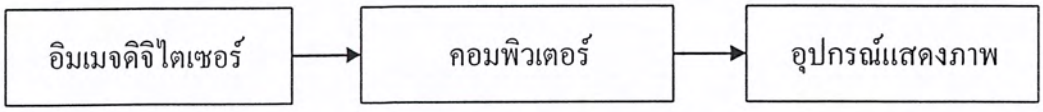
ในการประมวลผลภาพดิจิตอล (digital image processing) นั้นต้องใช้อุปกรณ์พื้นฐาน 3 ชนิดคือ

(1) หน่วยประมวลผล (process unit) ได้แก่ คอมพิวเตอร์

(2) อุปกรณ์อินพุท (input device) ได้แก่ อิมเมจดิจิไตเซอร์ (image digitizer)

(3) อุปกรณ์เอาท์พุต (output device) ได้แก่ อุปกรณ์แสดงภาพ

ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 ระบบประมวลผลภาพดิจิตอล

โดยทั่วไปนั้น ข้อมูลภาพยังไม่สามารถนำไปวิเคราะห์ด้วยคอมพิวเตอร์ได้ทันที เนื่องจากคอมพิวเตอร์ทำงานเกี่ยวกับระบบตัวเลข จึงต้องแปลงข้อมูลภาพให้อยู่ในรูปแบบของตัวเลขก่อน เราเรียกการแปลงนี้ว่าดิจิไตเซชั่น (digitization) ดังรูปที่ 2.6 โดยภาพจะถูกแบ่งออกเป็นส่วนเล็กๆ เรียกว่า พิกเซล (pixel) ซึ่งมีลักษณะเป็นตะแกรงสี่เหลี่ยมจัตุรัส โดยแต่ละพิกเซลจะมีระดับความสว่างและความมืดแตกต่างกัน โดยในแต่ละพิกเซลจะถูกระบุตำแหน่งโดย (x,y)

โดยทั่วไปแล้ว ข้อมูลภาพจะมีความเข้มตั้งแต่ 2 ระดับขึ้นไป แต่ที่ใช้กันมากจะใช้กันที่ค่าระดับความเข้มของจุดภาพเท่ากับ 256 ระดับ ซึ่งจะทำให้ค่าของจุดภาพอยู่ในช่วง 0 - 255 โดยใช้เนื้อที่เก็บข้อมูลขนาด 1 ไบท์ (8 บิต) สำหรับข้อมูล 1 จุดภาพ ($2^8 = 256$) ในกรณีที่ต้องการภาพที่มีความละเอียดของระดับความเข้มสูงๆ ก็จะต้องการบิตสำหรับเก็บข้อมูลมากกว่า 8 บิต อาจจะเป็น 16 หรือ 24 บิต 2^{16} และ 2^{24} ระบบการประมวลผลภาพนั้นจะมีขนาดขึ้นกับความละเอียดของภาพ คือถ้าภาพมีความละเอียดมากก็จะใช้เนื้อที่ความจำของระบบมากในการเก็บข้อมูล

5.1) การประมวลผลภาพเชิงเลข

การประมวลผลภาพเชิงตัวเลข หมายถึง การนำภาพที่พบทั่วไปมาประมวลผลด้วยเครื่องคอมพิวเตอร์ โดยภาพที่นำมาประมวลด้วยเครื่องคอมพิวเตอร์นี้จะถูกแทนที่ด้วยตัวเลขให้อยู่ในรูปของเมตริกซ์ แต่ภาพที่ได้โดย ส่วนมากแล้วจะเป็นภาพที่ได้จากตัวรับสัญญาณ ซึ่งอยู่ในรูปของฟังก์ชั่น f(x,y) ที่ต่อเนื่องในระบบ 2 มิติ โดยจะเป็นสัดส่วนกับความสว่างหรือความเข้มของภาพที่ตำแหน่ง (x,y) ซึ่งเรียกว่าระดับสีเทา (gray level)

5.1.1) การแทนภาพด้วยข้อมูลแบบดิจิตอล ภาพข้อมูลแบบดิจิตอล เป็นภาพที่ถูกดัดแปลงมาจากอนาลอกอยู่ในรูปของตัวเลข โดยภาพอนาลอกถูกแบ่งเป็นพื้นที่สี่เหลี่ยมเล็กๆที่เรียกว่าพิกเซล ในแต่ละพิกเซลจะถูกระบุตำแหน่งโดย (x,y) และระดับค่าสีเทาของพิกเซล โดยเราสามารถแปลงภาพเป็นข้อมูลแบบดิจิตอลได้โดยมีขั้นตอนและวิธีการดังนี้

เมื่อเรานำสัญญาณอนาลอกที่ต้องการประมวลผลมาผ่านส่วนที่เรียกว่าดิจิไตเซอร์ (digitizer) ซึ่งมีหน้าที่ในการเปลี่ยนสัญญาณดิจิตอล อุปกรณ์ส่วนนี้ ได้แก่ กล้องโทรทัศน์ดิจิไตเซอร์ จากนั้นทำการควอนไตซ์ (quantizing) เพื่อที่จะประมวลผลสัญญาณด้วยระบบคอมพิวเตอร์ฟังชันก์ของภาพ f(x,y) จะถูกทำให้เป็นสัญญาณที่ไม่ต่อเนื่องทั้งระนาบของภาพ ซึ่งเราเรียกว่า การสุ่มภาพ (image sampling) ของฟังก์ชันที่ได้เรียกว่า การควอนไตเซชันระดับสีเทา (gray level quantization) ก็จะได้ข้อมูลที่เป็นดิจิตอล

สมมุติว่าสัญญาณภาพต่อเนื่อง f(x,y) ถูกดิจิไตซ์ระนาบ x และ y เป็นช่วงเท่าๆกัน เราสามารถจัด f(x,y) ให้อยู่ในรูปเมตริกซ์ NxN ได้ดังสมการที่ (1)

$$f(x,y) = \begin{bmatrix} f(0,0) & f(1,0) & f(0,2)\ldots f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2)\ldots f(1,N-1) \\ . & . & . \quad . \\ . & . & . \quad . \\ f(N-1,0) & f(N-1,1) & f(N-1,2)\ldots f(N-1,N-1) \end{bmatrix} \quad (1)$$

ซึ่งทางขวาของสมการจะเรียกได้ว่า ภาพดิจิตอล และทุกๆสมาชิกของเมตริกซ์จะเรียกว่า พิกเซล จากขบวนการสร้างดิจิตอลข้างต้นจะเห็นได้ว่าเราสามารถทราบขนาดของความละเอียดของภาพ NxN พิกเซล และจำนวนระดับของเกรย์สเกล(gray scale)ในทางปฏิบัติการทำควอนไตเซชั่นในระบบภาพดิจิตอล จะมีค่าดังสมการที่ (2)

เมื่อ $B = NxNxM$ บิต (2)

B = ขนาดของข้อมูลภาพที่เป็นดิจิตอล

G = จำนวนของเกรย์สเกลที่ต้องใช้ในการเก็บภาพ

M = จำนวนบิตที่ใช้ในการแทนข้อมูลภาพ 1 พิกเซล

โดย M สามารถหาได้จาก

$$G = 2^M$$

5.1.2) ลักษณะการจัดเก็บข้อมูลภาพดิจิตอล โดยทั่วไปแล้ว ข้อมูลภาพจะมีความเข้มตั้งแต่ 2 ระดับขึ้นไป แต่ที่ใช้กันมากจะใช้กันที่ค่าระดับความเข้มของจุดภาพเท่ากับ 256 ระดับ ซึ่งทำให้ค่าของจุดภาพอยู่ในช่วง (0-255) โดยใช้เนื้อที่เก็บข้อมูลภาพขนาด 1 ไบต์ หรือ 8 บิต สำหรับข้อมูล 1 จุดภาพ ($2^8 = 256$) ในกรณีที่ต้องการภาพที่มีความละเอียดของระดับความเข้มสูงๆอาจจะต้องการจำนวนบิตสำหรับเก็บข้อมูลมากกว่า 8 บิต อาจเป็น 16 หรือ 24 บิต โดยค่าความเข้มของจุดภาพเท่ากับ 2^6 และ 2^{24} โดยจะแยกให้ชัดเจน ดังนี้

1. ภาพ 2 ระดับ คือ มีเพียงแต่จุดขาวกับจุดดำเท่านั้น โดยแต่ละจุดภาพเป็นข้อมูลขนาด 1 บิต
2. ภาพ 16 ระดับ คือ ในแต่ละจุดภาพจะมีขนาดของข้อมูล 4 บิต ซึ่งทำให้สามารถแสดงได้ 16 ระดับสี หรือ 16 เกรย์สเกล ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือขาวดำ
3. ภาพ 256 ระดับ คือในแต่ละจุดภาพจะมีขนาดของข้อมูล 8 บิต ซึ่งทำให้สามารถแสดงได้ 256 ระดับสี หรือ 256 เกรย์สเกล ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือขาวดำ
4. ภาพทรูคัลเลอร์(true color) คือในแต่ละจุดภาพมีขนาดของข้อมูล 24 บิต ทำให้สามารถแสดงภาพได้เหมือนจริงที่สุด เพราะสามารถแสดงสีได้ถึง 16,777,216 ระดับสี โดยจะแสดงได้แต่ภาพสีเท่านั้น ไม่สามารถแสดงภาพขาวดำได้

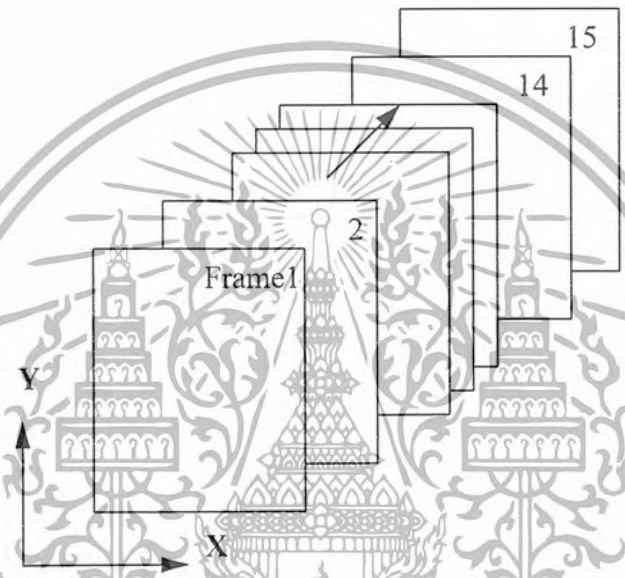
การแสดงภาพนี้ใช้วิธีตั้งค่าของแม่สีในตารางสี โดยอาจเลือกสีเป็นแบบ 16 สี จาก 64 สี หรือ 16 สี จาก 262,144 สี หรือ 256 สี จาก 262,144 สี ขึ้นกับโหมดการแสดงผล สำหรับโหมดทรูคัลเลอร์นั้น จะไม่มีการเลือกสี แต่จะแสดงผลโดยการส่งค่าสี RGB ผ่าน D/A สีละ 8 บิต ออกไป ความแตกต่างของการแสดงภาพสีและขาวดำ คือ ภาพขาวดำจะต้องตั้งสีให้แม่สีทั้งสามมีค่าเท่ากัน เนื่องจาก VGA กำหนดให้แม่สีแต่ละสีใช้ได้เพียง 64 ระดับสีเท่านั้น หากต้องการให้เห็นทั้ง 256 ระดับ จะต้องแสดงในโหมดทรูคัลเลอร์ แล้วให้ RGB ที่ค่าเท่ากัน ซึ่งในโหมดนี้จะสามารถใช้รีจิสเตอร์ได้ 8 บิต สำหรับแต่ละแม่สี

โดยทั่วไปวิธีการประเมินภาพเชิงเลข ที่ทำให้คอมพิวเตอร์สามารถรู้จักวัตถุภายในภาพนั้นแบ่งได้เป็น 2 ระดับด้วยกัน คือ การประมวลภาพในระดับต่ำ (low level image processing) และการประมวลภาพในระดับสูง (high level image processing) การประมวลผลภาพระดับต่ำจะเป็นการประมวลผลเชิงตัวเลขเกือบหมด เพื่อหาตัวแปรต่างๆมาอธิบายข้อมูลภาพ โดยมีจุดประสงค์เพื่อนำตัวแปรเหล่านั้นไปใช้ในการประมวลผลระดับสูงต่อไป

การประมวลผลภาพในระดับสูง เป็นการนำผลลัพธ์ที่ได้จากการประมวลผลภาพในระดับต่ำมาตีความหรือเพื่อส่งให้คอมพิวเตอร์สามารถรู้จักและเข้าใจภาพได้ สำหรับความแตกต่างของการประมวลผลภาพระดับต่ำและระดับสูง คือ ข้อมูลที่นำมาใช้ในการประมวลผลโดยที่การประมวลผลภาพในระดับต่ำจะใช้ความสว่างของจุดโดยตรง ส่วนการประมวลผลภาพระดับสูงนั้นข้อมูล ภาพที่นำมาประมวลผลจะถูกแสดงในรูปของสัญลักษณ์ ซึ่งจะแสดงถึงสิ่งต่างๆ ที่อยู่ในภาพ เช่น ขนาด หรือ รูปร่างของวัตถุในภาพ

5.2) สัญญาณข้อมูลภาพจากดิจิตอลวีดีโอ

การส่งสัญญาณข้อมูลภาพจาก วีดีโอจะมีลักษณะการส่งที่เป็นลำดับภาพเดี่ยว หรือ เฟรม (frame) ที่ฉายต่อเนื่องกันดังรูปที่ 2.6 เช่น ระบบวีดีโอ NTSC จะส่งด้วยอัตราเร็ว 30 เฟรมต่อวินาที โดยดิจิตอลวิดีโอแต่ละเฟรมจะเป็นข้อมูลภาพดิจิตอลในลักษณะของเมตริกซ์ ซึ่งแต่ละจุดจะเรียกว่า พิกเซล มีค่าของระดับความเข้มสี โดยทั่วไปจะใช้เกรย์สเกลที่มีค่าตั้งแต่ 0 - 255 โดย 0 แทนความมืดมากที่สุด และ 255 แทนความสว่างมากที่สุด



รูปที่ 2.6 ภาพการส่งสัญญาณวีดีโอ ที่อัตรา 15 เฟรมต่อวินาที

5.3) ข้อมูลภาพชนิดบิตแมป

5.3.1) รูปแบบของไฟล์ข้อมูลภาพชนิดบิตแมป รูปแบบของไฟล์ภาพข้อมูลชนิดบิตแมป (bitmap) เป็นฟอร์แมทของวินโดว์บิตแมป ซึ่งเป็นมาตรฐานสำหรับไฟล์กราฟฟิกบนวินโดว์ ซึ่งจะใช้ในการตัดต่อ หรือสำเนาภาพต่างๆลงบน คลิปบอร์ด (clipboard) เมื่อเวลาจัดเก็บไฟล์ที่มีนามสกุล .BMP ซึ่งเป็นฟอร์แมตที่ยังสามารถใช้เป็นวอลเปเปอร์ (wallpaper) ได้อีกด้วย

5.3.2) โครงสร้างของไฟล์ข้อมูลภาพชนิดบิตแมปโครงสร้าง ของไฟล์ข้อมูลภาพชนิดบิต แมป จะประกอบด้วย 3 ส่วน คือ

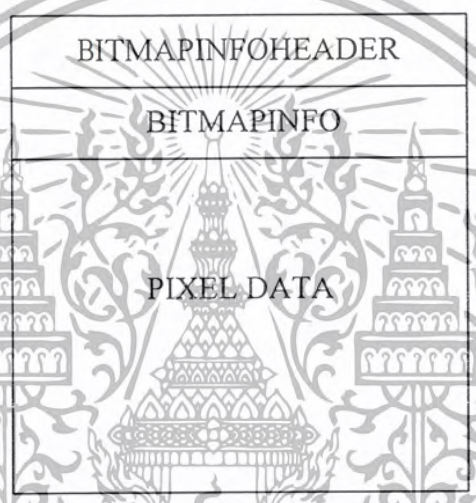
(1) ข้อมูลเฮดเดอร์ (header)

(2) ข้อมูลจานสี (palette)

(3) ข้อมูลภาพ (data)

(1) ข้อมูลเฮดเดอร์ คือ ข้อมูลที่อยู่บริเวณส่วนหัวของไฟล์ ซึ่งจะประกอบด้วยข้อมูลที่บอกรายละเอียดต่างๆของภาพ เช่น ความกว้าง ความยาวของภาพ จำนวนสี จำนวนบิต ความละเอียดของภาพ

(2) ข้อมูลจานสี เป็นค่าแม่สี RGB แล้วนำไปผสมบนหน้าจอแทน ไฟล์ข้อมูลชนิดบิตแมป มีโครงสร้างดังรูปที่ 2.7 แบ่งออกเป็น 3 ส่วน ได้แก่ บิตแมปไฟล์เฮดเดอร์ (Bitmapfileheader) เป็นส่วนที่บอกข้อมูลของไฟล์ บิตแมปอินโฟ (bitmapinfo) เป็นส่วนที่แสดงขนาดและข้อมูลสีของภาพ ส่วนสุดท้าย คือ พิกเซลดาต้า (pixel data) เป็นส่วนเก็บข้อมูลแต่ละพิกเซล



รูปที่ 2.7 โครงสร้างของบิตแมปไฟล์

บิตแมปอินโฟเฮดเดอร์

Byte	Data	Detail
1-2	File Type	Must be ASC II test “BM”
3-6	Size of file	In double word (32 – bit integer)
7-10	Reverse for future	Must be zero
11-14	Byte offset to bitmap data	Offset from bitmapinfoheader

ตารางที่ 2.1 แสดงข้อมูลในบิตแมปไฟล์เฮเดอร์

BITMAPINFO บิตแมปอินโฟ

โครงสร้างของ บิตแมปอินโฟ เขียนได้เป็นดังนี้

```
typedef struct  tagBITMAPINFO {// bmi
        BITMAPINFOHEADER        bmiHeader;
        RGBQUAD                 bmiColor[1];
}BITMAPINFO;
```

บิตแมปอินโฟ ประกอบด้วย 2 ส่วน คือ บิตแมปอินโฟเฮดเดอร์เป็นส่วนที่บอกขนาดและข้อมูลสีของภาพบิตแมป และ อาร์จีบีควอด (RGBQUAD) ซึ่งจะเก็บค่าตารางสีสำหรับเทียบจากค่าสีของแต่ละพิกเซล

บิตแมปอินโฟเฮดเดอร์

โครงสร้างสามารถเขียนได้ดังนี้

```
type struct  tagBITMAPINFOHEADER {// bmih
        DWORD           biSize;
        LONG            biWidth;
        LONG            biHeight;
        WORD            biPlanes;
        WORD            biBitCount;
        DWORD           biCompression;
        DWORD           biSizeImage;
        LONG            biXiPelsPerMeter;
        LONG            biYiPelsPerMeter;
        DWORD           biClrUsed;
        DWORD           biClrIMportant;

}BITMAPINFOHEADER;
```

โดยแต่ละฟิลด์ (field) จะมีความหมายดังนี้

biSize	จำนวนไบต์ของเฮดเดอร์ (Header file)
biWidth,biHeight	บอกขนาดความกว้าง และความยาวของรูปภาพในพิกเซล
biPlanes	เป็น 1 เสมอ
biBitCount	จำนวนบิตต่อ 1 พิกเซล
biCompression	แสดงการบีบอัดข้อมูล
biSizeImage	บอกขนาดของไฟล์
biXiPelsPerMeter	ความยาวแนวนอนในหน่วยพิกเซลต่อเมตร
biYiPelsPerMeter	ความยาวแนวตั้งในหน่วยพิกเซลต่อเมตร
biClrUsed	จำนวนสีในตารางสีที่จะถูกชี้ด้วยค่าพิกเซลในบิตแมป
biClrIMportant	เป็นเลขที่แสดงว่าข้อมูลสีมีความสำคัญในการแสดงผลของบิตแมป ถ้าเป็นเลข 0 แสดงว่าทุกสีมีความสำคัญ

อาร์จีบีควอด

มีโครงสร้างดังนี้

```
type struct tagRGBQUAD {// rgbq
    BYTE    rgbBlue
    BYTE    rgbGreen
    BYTE    rgbRed
    BYTE    rgbReserved;
}RGBQUAD;
```

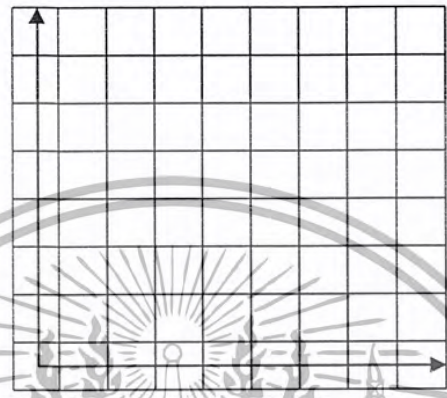
อาร์จีบีควอด จะเป็นโครงสร้างที่แสดงความเข้มของสีแดง เขียว และน้ำเงิน โดยมีความหมายของแต่ละฟิลด์ดังนี้

rgbBlue	แสดงความเข้มของสีน้ำเงิน
rgbGreen	แสดงความเข้มของสีเขียว
rgbRed	แสดงความเข้มของสีแดง
rgbReserved	ต้องมีค่าเป็น 0

ในส่วนของคำสั่ง bmiColors ของโครงสร้าง บิตแมปอินโฟ จะประกอบด้วยอาเรย์ (Array)

ของอาร์จีบีควอด เพื่อเป็นตารางเปรียบเทียบสีของข้อมูลในแต่ละพิกเซล

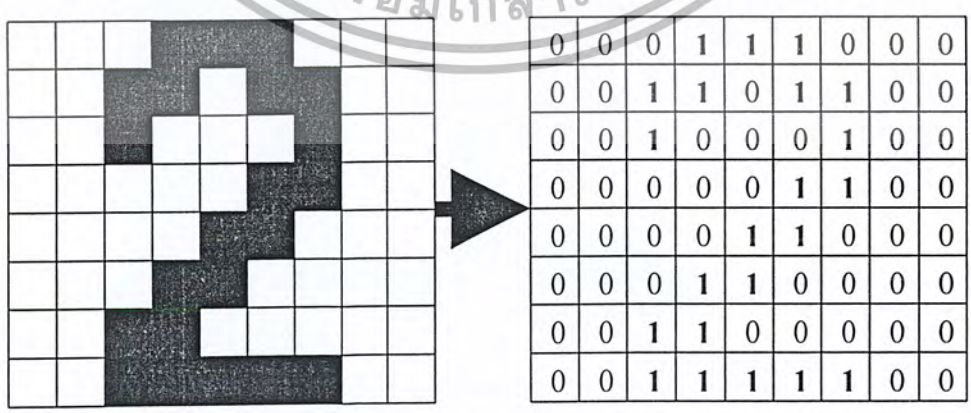
(3) ข้อมูลภาพ เป็นส่วนเก็บข้อมูลสีของแต่ละพิกเซลของภาพ โดยข้อมูลแรกจะเป็นค่าสีของ พิกเซลที่อยู่แถวล่างสุดที่ตำแหน่งซ้ายสุด ข้อมูลลำดับต่อไปจะเรียงทางขวาจากแถวล่างจนถึงแถวบนสุด



รูปที่ 2.8 แสดงการเก็บข้อมูลของแต่ละพิกเซล

5.4) การสร้างภาพไบนารี

การสร้างภาพไบนารี หมายถึง การแปลงข้อมูลภาพที่มีความเข้มหลายระดับ ให้เป็นภาพที่มีความเข้มเพียง 2 ระดับ คือ 1 จุดภาพ มีค่าได้แค่ 2 ค่าเท่านั้น โดยเป็น 0 กับ 1 ซึ่งจะหมายถึง จุดภาพที่มีสีดำ และ 0 จะหมายถึง จุดภาพที่มีสีขาว การแปลงข้อมูลภาพหลายระดับไปเป็นภาพไบนารีจึงมีความจำเป็นและมีประโยชน์มากในการแสดงผลได้ 2 ระดับ ประโยชน์อีกประการ คือ การลดเนื้อที่การเก็บข้อมูลภาพให้เหลือเพียง 8 บิต



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

รูปที่ 2.9 ภาพแบบไบนารีและข้อมูลของแต่ละพิกเซล

ในการสร้างภาพไบนารี สามารถทำโดยใช้เทคนิคการทำเทรชโฮล (Thresholding technics) โดยพิจารณาว่าจุดใดควรเป็นจุดขาวหรือจุดดำ จะกระทำโดยการเปรียบเทียบระหว่างจุดภาพเริ่มต้นกับค่าคงที่ค่าหนึ่ง เรียกว่า ค่าเทรชโฮล เทคนิคนี้ใช้กันมากในกรณีที่ข้อมูลภาพมีลักษณะแตกต่างกันระหว่างวัตถุ และพื้นหลัง โดยค่าของจุดภาพใดๆ ที่มีค่าน้อยกว่าค่าเทรชโฮลจะถูกกำหนดค่าเป็น ค่า 1 (จุดสีดำ) และถ้าค่าของจุดภาพมีค่ามากกว่าค่าเทรชโฮล จะถูกเปลี่ยนให้เป็นค่า 0 (จุดสีขาว) ซึ่งการทำงานสามารถแสดงได้ดังสมการที่ (3)

$$b(x,y) = \begin{cases} 1 \; ; \; g(x,y) < Thr \\ 0 \; ; \; g(x,y) > Thr \text{ or } 0 \; ; \; g(x,y) = Thr \end{cases} \qquad (3)$$

b(x,y) คือ ข้อมูลภาพผลลัพธ์ภาพเป็นไบนารี

g(x,y) คือ ข้อมูลภาพอินพุทที่มีระดับความเข้ม 0 ถึง L ระดับ

Thr คือ ค่าเทรชโฮลเป็นค่าคงที่ระหว่าง 0 ถึง L ระดับ

1 คือ จุดดำ

0 คือ จุดขาว

โดยที่ L คือ ระดับความเข้มของจุดภาพสูงสุด

ในการสร้างภาพไบนารีโดยใช้เทคนิคเทรชโฮลเพื่อให้ได้ผลลัพธ์ที่ได้คมชัดและเหมาะสม สิ่งที่สำคัญที่สุด คือ ค่าเทรชโฮล เนื่องจากถ้าเลือกค่าเทรชโฮลที่ไม่เหมาะสม ภาพที่ได้จะไม่คมชัด และรายละเอียดบางส่วนอาจขาดหายไป ดังนั้นปัญหาการสร้างภาพด้วยวิธีเทรชโฮล คือ ทำอย่างไรจึงสามารถคำนวณหาค่าเทรชโฮลที่เหมาะสมได้

5.4.1) การหาค่าเทรชโฮลโดยการกำหนดไว้ล่วงหน้า

การหาค่าเทรชโฮลโดยวิธีการกำหนดล่วงหน้า (preassigned threshold value) นี้ เป็นวิธีที่ง่ายที่สุด เป็นการกำหนดค่าเทรชโฮลเองจากผู้ใช้ ซึ่งจะขึ้นอยู่กับประสบการณ์ของผู้ใช้ โดยการเลือกค่าคงที่ค่าหนึ่งจะเป็นค่าที่อยู่ระหว่างค่าต่ำสุดและค่าสูงสุด ของระดับความเข้มของข้อมูลอินพุท

ดยส่วนมากแล้วจะเลือกค่า ดังนี้ Gray scale ดังนี้

GrayValue = (BYTE)(0.299 * redValue + 0.587 *greenValue + 0.114 * blueValue)

5.4.2) การหาค่าเทรชโฮลจากค่ากลาง (midrange threshold value)

การหาวิธีนี้จะพิจารณาจากค่ากลางโดยอาศัยการคำนวณพื้นฐานทางสถิติ ในการหาค่ากลางหรือค่าเฉลี่ย (mean) มาประยุกต์ ค่าที่ได้จะเป็นค่ากึ่งกลางระหว่างค่าระดับความเข้มสูงสุดกับค่าต่ำสุดของข้อมูลภาพอินพุท สำหรับการหาค่ากึ่งกลางจะแสดงได้ดังสมการที่ (4)

$$Thr = \frac{Max(g(x,y)) + Min(g(x,y))}{2} \quad (4)$$

โดยที่ Thr ค่าเทรชโฮล

g(x,y) ข้อมูลภาพอินพุทที่มีระดับความเข้ม 0 ถึง L ระดับ

Max(g(x,y)) ค่าสูงสุดเกรย์สเกลของข้อมูลอินพุท

Min(g(x,y)) ค่าต่ำสุดเกรย์สเกลของข้อมูลอินพุท

การหาค่าเทรชโฮลจากค่าเฉลี่ยเลขคณิต หาได้จากสมการที่ 5

$$Thr = \frac{g_i(x,y)}{NxN} \quad (5)$$

5.5) เทคนิคการติดตามรอยขอบภาพ

เทคนิคการติดตามรอยขอบภาพ ถูกนำมาใช้ในการแยกและคัดลอกส่วนของรูปภาพใดๆที่อยู่บนรูปใหญ่ ข้อมูลภาพที่จะนำมาประมวลผลด้วยเทคนิคนี้ต้องอยู่ในรูปของข้อมูลไบนารี นั่นคือจุดภาพจะแสดงด้วยรูป 0 กับ 1 เท่านั้น

การทำงานของเทคนิคการติดตามรอยขอบของภาพ เป็นการเดินไต่ไปตามขอบระหว่างส่วนที่เป็นรูปภาพ กับส่วนที่เป็นพื้นหลัง โดยจะตรวจกวาดไปทุกๆพิกเซล โดยจะเริ่มจากจุดมุมซ้ายของภาพ ตรวจกวาดไปในทิศทางจากซ้ายไปขวา และเลื่อนลงจากบนลงล่าง เมื่อตรวจกวาดมาพบจุดใดๆที่มีค่าของจุดภาพเป็น 1 ก็จะเปลี่ยนลักษณะการเคลื่อนไหว ไปยังจุดถัดไปใหม่ โดยมีเงื่อนไขการเคลื่อนที่ ดังนี้

ลักษณะการทำงานของเทคนิคการติดตามรอยขอบภาพ ซึ่งจะแสดงการเคลื่อนที่ไปตามจุดต่างๆที่เป็นขอบภาพเริ่มจากจุดที่ถูกแรเงาไว้ซึ่งเป็นจุดของภาพแรกที่ตรวจกวาดตามเงื่อนไขที่กำหนด

เมื่อการเคลื่อนที่วนกลับมาถึงจุดเริ่มต้น ก็จะทราบจุที่เป็นขอบของภาพทั้งหมด ขณะที่เดินไต่ขอบนั้นก็จดจำค่าพิกัดไปด้วย เมื่อเดินถึงจุดเดิม ก็นำค่าพิกัดที่น้อยที่สุดและมากที่สุดมากำหนดขอบเขตของขนาดวัตถุเพื่อใช้เปรียบเทียบกันต่อไป

6) พอร์ตขนาน (parallel port)

เป็นพอร์ตที่ใช้ส่งและรับข้อมูลที่นิยมใช้มากที่สุด พอร์ตขนานจะส่งข้อมูลได้อย่างรวดเร็วและสามารถส่งได้หลายสาย และมีความผิดพลาดของสัญญาณน้อยมาก (ในระยะทาง น้อยกว่า 0.66เมตร) ซึ่งเราสามารถใช้ช่องการส่งข้อมูลที่มีมากนั้นรับและส่งค่าได้ในพอร์ตเดียว

7) ยูนิเวอร์แซลซีเรียลบัส (USB Port)

เป็นพอร์ตที่มีไว้ใช้รับส่งข้อมูลพอร์ตหนึ่ง ที่มีการส่งข้อมูลที่เร็วมาก (ถ้าเทียบกับพอร์ตอนุกรมและพอร์ตขนานและพอร์ต PS/2) ใช้กับอุปกรณ์ที่ต้องการความเร็วของสัญญาณ มีความผิดพลาดของสัญญาณน้อยมาก (ในระยะทางน้อยกว่า 100 เมตร) ซึ่งเราใช้ในการส่งสัญญาณจากกล้องดิจิตอลมายังคอมพิวเตอร์

บทที่ 3
การออกแบบ และ การประมวลผล

1) หุ่นยนต์

โครงหุ่นยนต์ทำจากกล่องไม่อัดซึ่งเจาะรูให้มอเตอร์ยื่นออกมาได้ หุ่นยนต์จะประกอบได้ด้วยล้อขับเคลื่อน 2 ล้อและล้ออิสระอีก 1 ล้อ

ใช้ถ่านไฟฉายขนาด 9V. 2 ก้อนเป็นแหล่งจ่ายไฟมอเตอร์ และ แหล่งจ่ายไฟรวมทั้งหมดโดย จะวางไว้ที่หน้าตัวหุ่น เหตุผลที่เราติดลังถ่านไว้แบบนี้ เพื่อให้หุ่นยนต์อยู่ในสภาพสมดุล ลดภาระของล้อไม่ให้ล้อใดล้อหนึ่งรับน้ำหนักมากเกินไป

รูปข้างล่างแสดงโครงสร้างของหุ่นยนต์ที่ออกแบบไว้ โดยจะแสดงเฉพาะมุมมองข้างบน

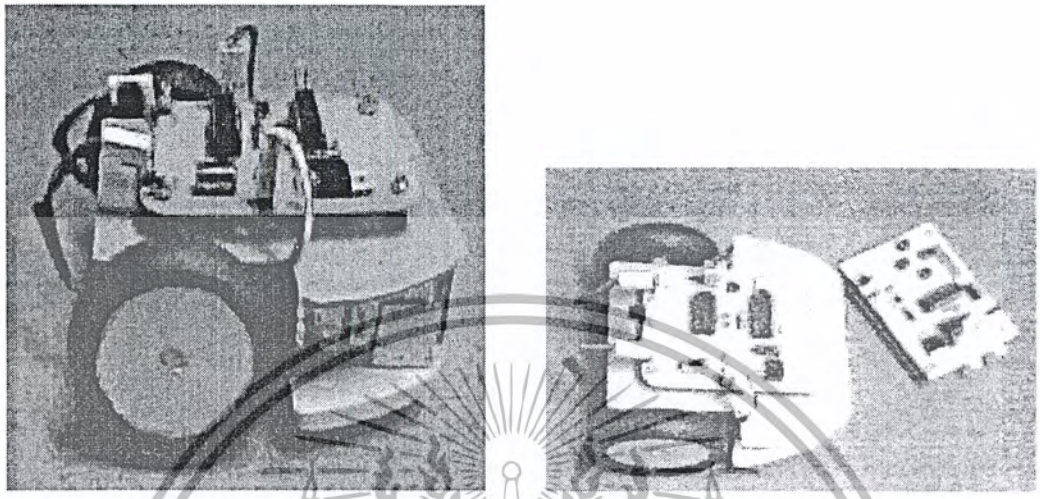
รูปที่ 3.1 แสดงรูปโครงสร้างของหุ่นยนต์เมื่อมองจากข้างบน

จากรูปที่ผ่านมาเราสามารถแบ่งการเคลื่อนที่ของหุ่นยนต์ได้เป็น 6 แบบคือ

1. **เดินหน้า** สามารถทำได้โดยการสั่งให้มอเตอร์ทั้งคู่หมุนไปข้างหน้า
2. **ถอยหลัง** ทำได้โดยการสั่งให้มอเตอร์ทั้งคู่หมุนไปด้านหลัง
3. **เลี้ยวขวา** ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนไปหน้า ส่วนมอเตอร์ทางขวาอยู่กับที่
4. **เลี้ยวซ้าย** ทำได้โดยการสั่งให้มอเตอร์ทางขวาหมุนไปหน้า ส่วนมอเตอร์ทางซ้ายอยู่กับที่

5. หันขวา ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนไปหน้า ส่วนมอเตอร์ทางขวาหมุนกลับหลัง

6. หันซ้าย ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนกลับหลัง ส่วนมอเตอร์ทางขวาหมุนไปหน้า

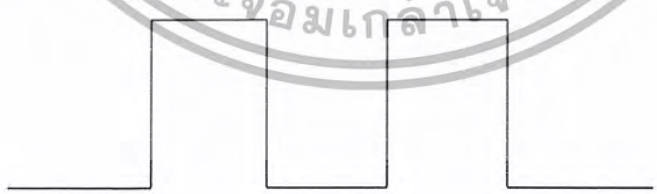


รูปที่ 3.2 แสดงรูปตัวรถ

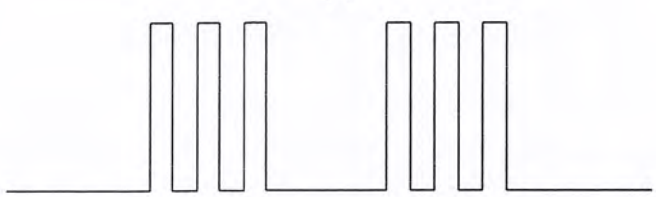
2) วงจรรับส่งอินฟราเรด

2.1) วงจรส่งอินฟราเรด

การออกแบบวงจรโดยใช้การส่งสัญญาณแบบโทนเบิรสต์ (Tone Burst) ซึ่งประกอบด้วยพัลส์ความถี่สูงแบบต่อเนื่องตลอดช่วงความกว้างของบิตข้อมูลที่เป็น “1” ในขณะที่บิตข้อมูลอยู่ในสภาวะต่ำ สัญญาณจะคงเดิมไม่มีการเปลี่ยนแปลงแต่อย่างใด ซึ่งส่งออกด้วยบอร์ดเรท 2400 bit/sec



(ก)



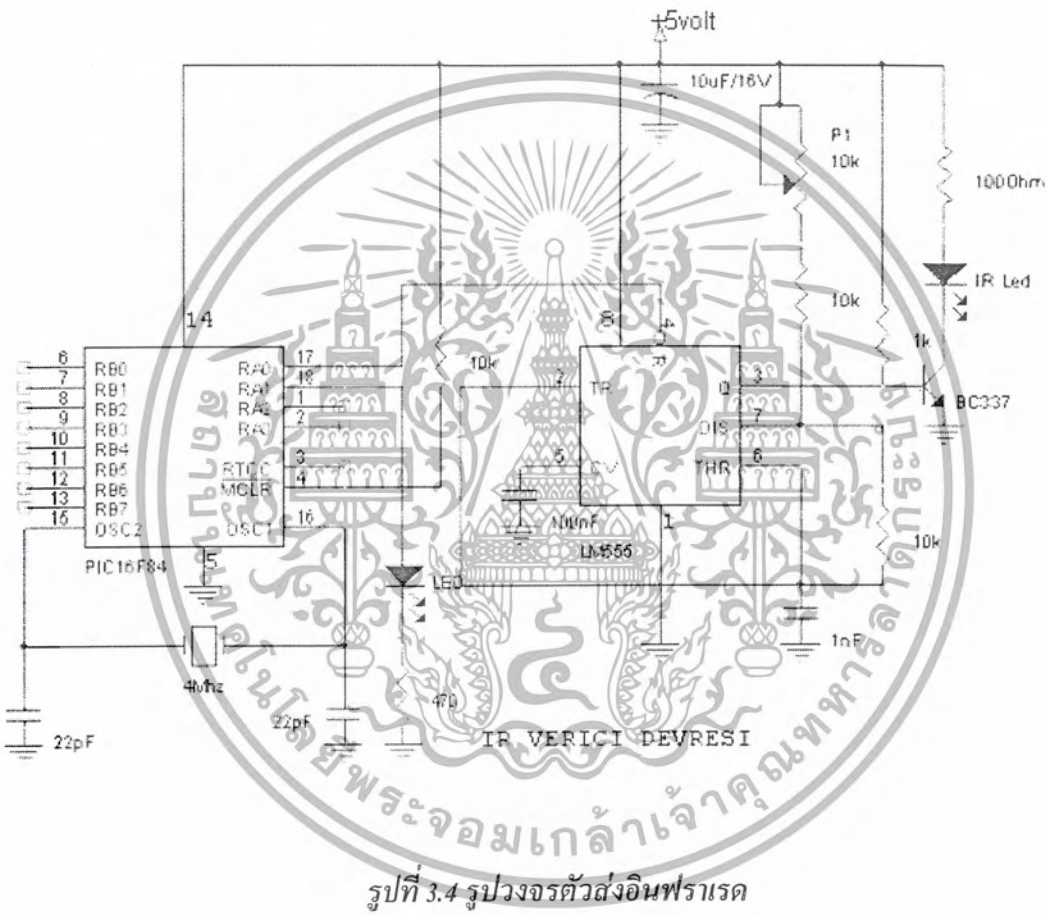
(ข)

รูปที่ 3.3 ลักษณะสัญญาณโทนเบิรสต์

(ก) สัญญาณพัลส์ข้อมูล

(ข) สัญญาณโทนเบิรสต์

วงจรตัวส่งอินฟราเรดที่ใช้มีรูปดังนี้

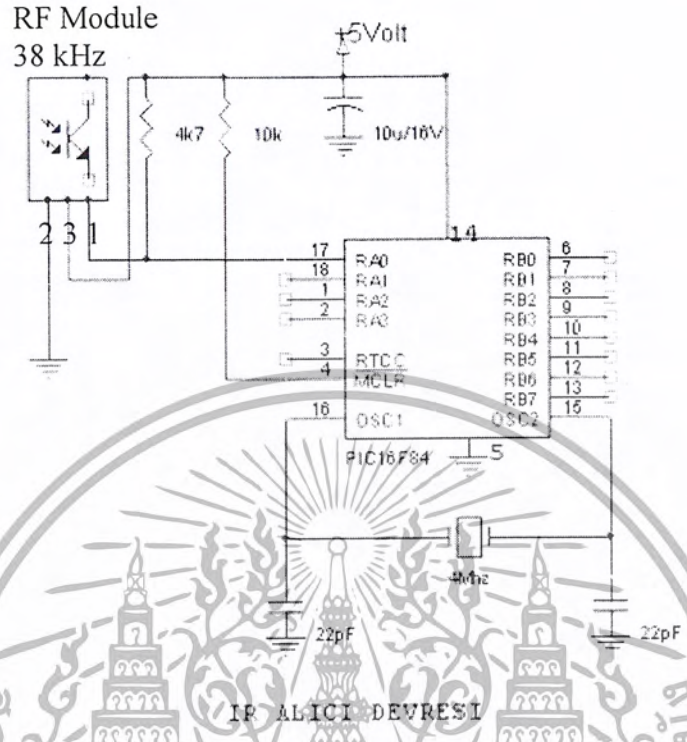


รูปที่ 3.4 รูปวงจรตัวส่งอินฟราเรด

2.2) วงจรรับอินฟราเรด

วงจรรับอินฟราเรดถูกออกแบบ โดยใช้โมดูลอินฟราเรดเบอร์ V4838 โมดูลตัวนี้จะตอบสนองเฉพาะสัญญาณความถี่ที่ 38 kHz เท่านั้น โดยจะทำการกรองความถี่พาหะ 38 kHz ออกแล้วเอาเฉพาะสัญญาณข้อมูลส่งให้ภาคขยายสัญญาณเพื่อส่งไปยังไมโครคอนโทรลเลอร์เพื่อทำการเช็คสถานะของอินพุท จากนั้นจึงส่งสัญญาณควบคุมไปควบคุมมอเตอร์ต่อไป

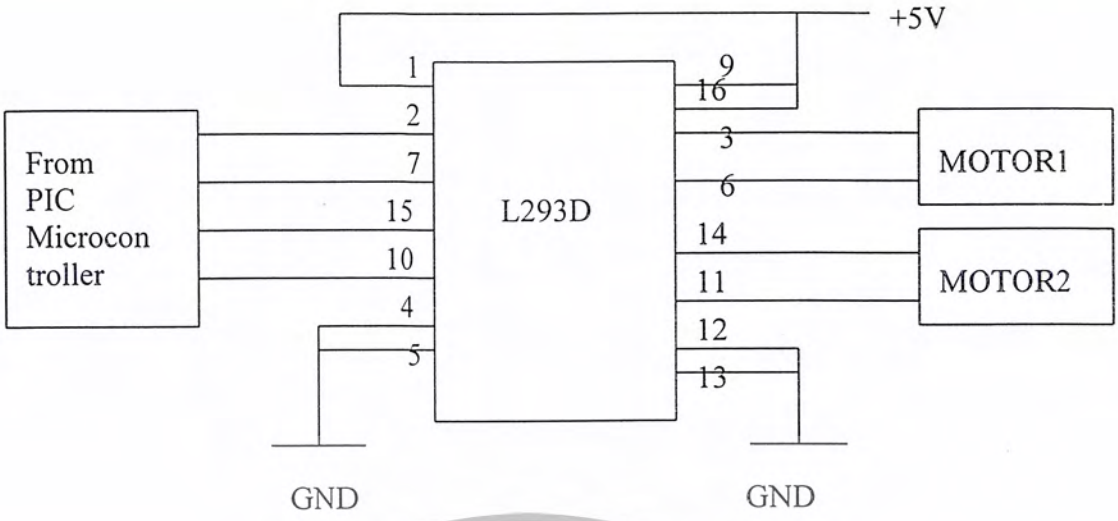
วงจรตัวรับอินฟราเรดที่ใช้มีรูปดังนี้



รูปที่ 3.5 รูปวงจรตัวรับอินฟราเรด

2.3) วงจรขับมอเตอร์

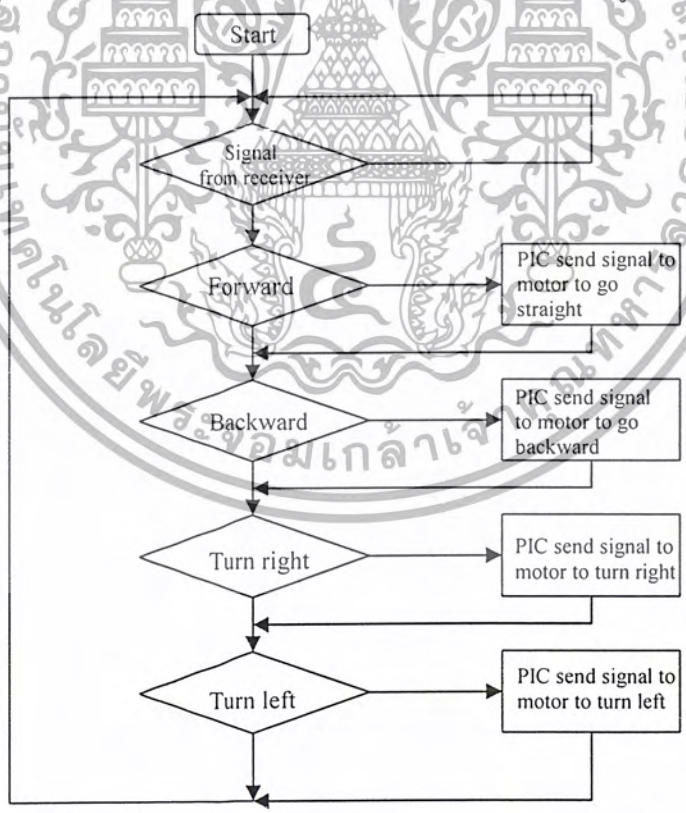
ในโครงงานนี้จะใช้ IC ในการขับมอเตอร์ เพื่อที่จะลดขนาดของตัวรถให้มีขนาดเล็กที่สุด เพื่อที่จะได้มีความคล่องตัวในการทำงาน โดย IC ที่ใช้เบอร์ L293D สามารถขับมอเตอร์ที่มีขนาดตั้งแต่ 3 – 36 VOLT ได้ถึง 2 ตัว ซึ่งสามารถดูรายละเอียดอื่นๆ โดยรายละเอียดได้จากดาต้าชีทท้ายรายงาน



รูปที่ 3.6 รูปวงจรตัวขับมอเตอร์

2.4) การควบคุมตัวรถ

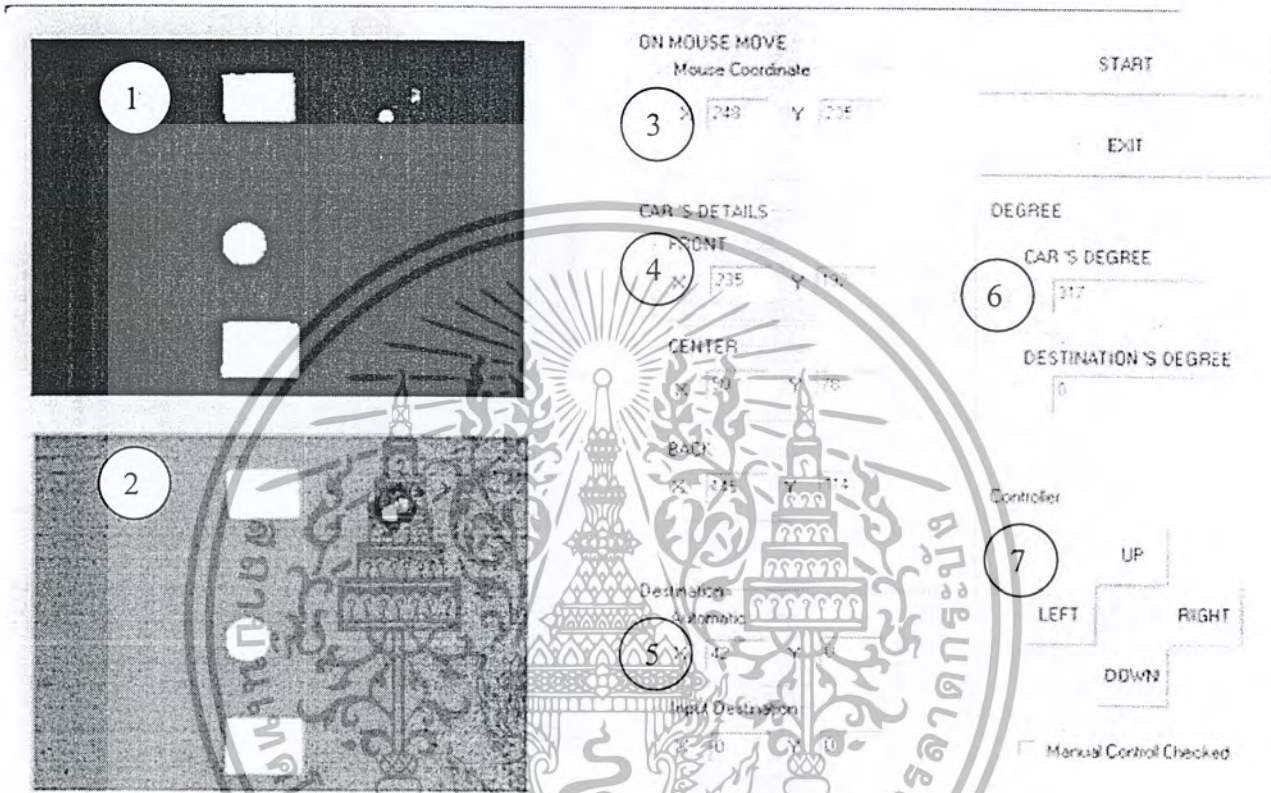
เนื่องจากในเทอมนี้ขอบเขตการทำงานยังไม่ได้ติดต่อกับคอมพิวเตอร์ ดังนั้น การควบคุมตัวรถ จึงกระทำผ่านวงจรอินฟราเรดแบบ MANUAL ซึ่งเป็นการทดสอบวงจรรับส่งอินฟราเรดและตัวหุ่นพร้อมๆกัน FLOW CHART ข้างล่างจึงเป็นของโปรแกรมที่อยู่ภายใน PIC



รูปที่ 3.7 รูปโฟลวชาร์ตการทำงานของตัวหุ่น

2.5) ส่วนติดต่อกับผู้ใช้งาน

ใช้โปรแกรม Visual C++ ในการ สร้างโปรแกรมทั้งในส่วนติดต่อกับผู้ใช้ และ ในส่วนของการประมวลผล ดังรูปข้างล่าง



รูปที่ 3.8 ส่วนติดต่อกับผู้ใช้งาน

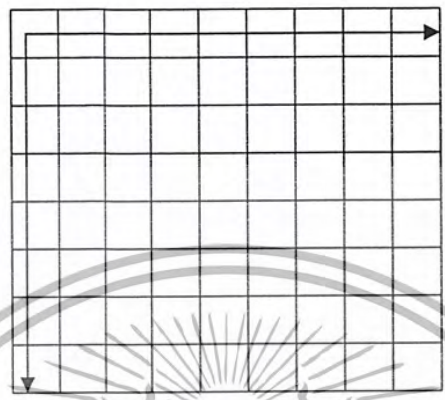
โดยจะแบ่งการทำงานเป็นส่วนๆ ดังต่อไปนี้

1. เฟรมในส่วนการประมวลผลซึ่งในโปรแกรมนี้จะทำการประมวลผลในโหมด เกรย์สเกล ดังนั้น ส่วนนี้จะทำการ แสดงในส่วนที่ทำการแปลงโหมดภาพแล้ว
2. เฟรมในส่วนการแสดงภาพที่รับมาจากกล้อง โดยยังไม่ได้มีการประมวลผลใดๆ
3. เป็นส่วนที่ทำการแสดงตำแหน่งของเมาท์ในขณะนั้น
4. เป็นส่วนที่ทำการแสดงตำแหน่งของพิกัดของตัวรถ ในตำแหน่งต่างๆ
5. เป็นส่วนที่ทำการแสดงตำแหน่งของเป้าหมาย
6. เป็นส่วนที่ทำการแสดงตำแหน่งของมุมของรถ และ เป้าหมายในขณะนั้น
7. ส่วนคอนโทรลในแบบแมนนวล

2.6) ส่วนของการประมวลผล

การทำงานของโปรแกรมจะเริ่มเมื่อเราคลิกที่ปุ่ม start จากนั้นโปรแกรมจะทำการประมวลผล

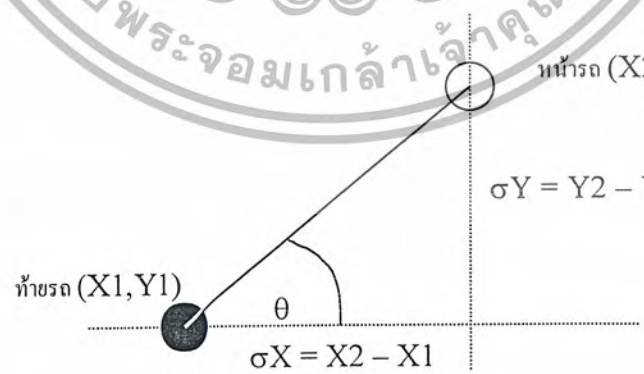
2.6.1) โปรแกรมจะทำการประมวลภาพ โดยการสแกนหน้าจอดังรูป



รูปที่ 3.9 แสดงการแสกนจอภาพ

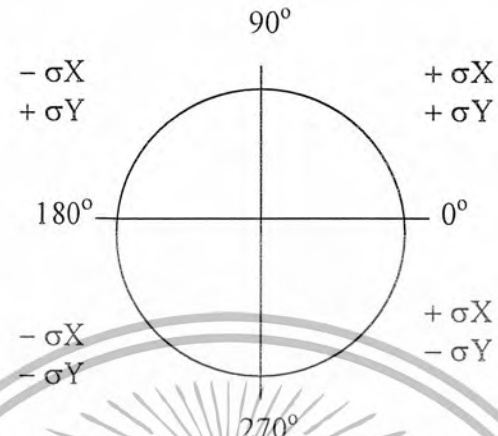
จากรูปพบว่าจะทำการสแกนจากขวาไปซ้าย และบนลงล่างตามลำดับ เพื่อที่จะหาจุดต่างๆ ตามที่ได้เขียนโปรแกรมไว้ ซึ่งจะทำให้ได้ตำแหน่งของรถ ตำแหน่งของสิ่งกีดขวาง และตำแหน่งของเป้าหมาย

2.6.2) จากนั้นจะทำการคำนวณองศาของรถ และ ของเป้าหมายว่า เท่ากันหรือไม่ ถ้าไม่จะทำการสั่งตัวรถให้ขยับปรับมุมให้มีขนาดเท่ากัน โดยการหามุมจะหาได้ตามรูป



$$\theta = \text{Tan}^{-1}(\sigma Y / \sigma X)$$

โดย การหาองศาของเป้าหมายก็หาแบบเดียวกันนี้ จากนั้นจึงนำมาประมวลผลการพิจารณามุม เมื่อพิจารณาร่วมกับวงกลม 1 หน่วย จะสามารถแบ่งเงื่อนไขในการคำนวณได้เป็น



- ถ้า σX มีเครื่องหมายเป็น - ทิศทางของหุ่นยนต์สามารถหาได้จาก

  DegreeCar = 180 + 180/ θ *arctan(σY / σX) องศา

- ถ้า σX = 0 แสดงว่าหุ่นยนต์วางตัวอยู่ตามแนวแกน Y กรณีนี้ arctan(σY / σX) จะหาค่าไม่ได้ จึงต้องใช้วิธีพิจารณาเครื่องหมายของ σY แทนซึ่งจะได้ว่า

  ถ้า σY มีเครื่องหมายเป็น + จะได้ทิศทางของหุ่นยนต์เท่ากับ 90 องศา

  ถ้า σY มีเครื่องหมายเป็น - จะได้ทิศทางของหุ่นยนต์เท่ากับ 270 องศา

- ถ้า σX มีเครื่องหมายเป็น + การหาทิศทางของหุ่นยนต์แบ่งได้เป็น 2 กรณีคือ

  ถ้า σY มีเครื่องหมายเป็น - จะสามารถคำนวณหาทิศทางได้คือ

  DereeCar = 360 + 180/ θ *arctan(σY / σX) องศา

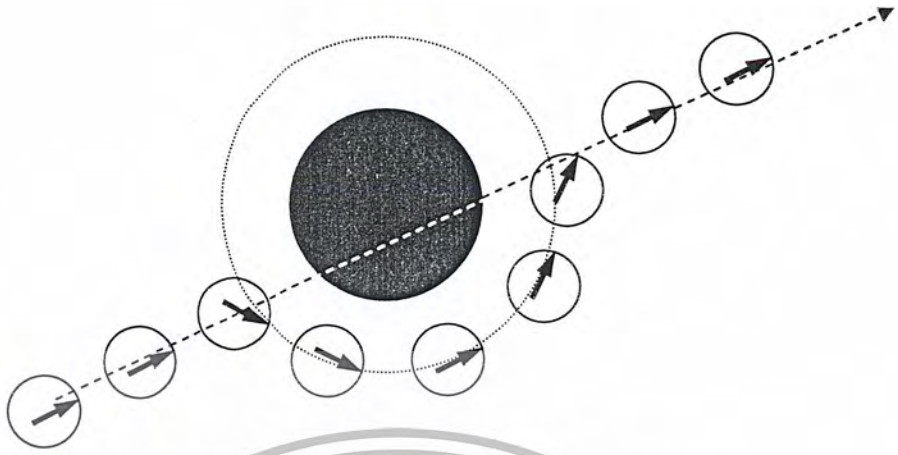
  ถ้า σY มีเครื่องหมายเป็น + หรือมีค่าเป็น 0 จะสามารถคำนวณหาทิศทางได้

  DereeCar = 180/ θ *arctan(σY / σX) องศา

2.6.3) หลังจากที่ได้ปรับองศาแล้ว ก็จะสั่งให้หุ่นเดินไปยังตำแหน่งเป้าหมาย

กรณีที่มีสิ่งกีดขวาง

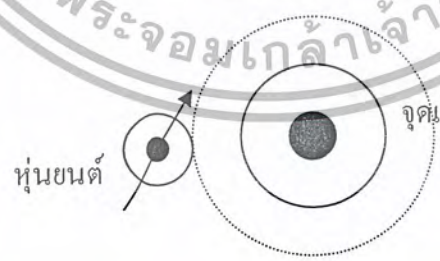
ในกรณีที่มีสิ่งกีดขวางนั้น โปรแกรมจะต้องทำการประมวลผลเพื่อที่จะหาเส้นทางในการหลบหลีกสิ่งกีดขวางเพื่อที่จะไปยังเป้าหมายให้ได้ตามที่ต้องการ โดยการหลบสิ่งกีดขวางสามารถทำได้โดย



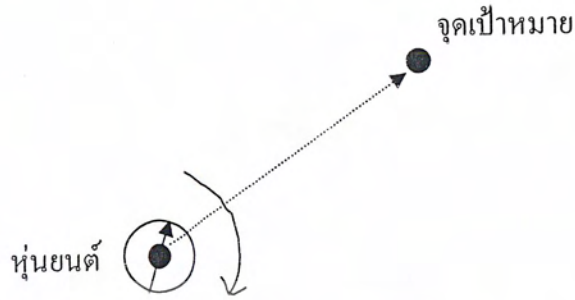
จากรูปพบว่า เส้นทางเป้าหมายจะมีสิ่งกีดขวางๆ อยู่ทำให้ไม่สามารถที่จะทำการเคลื่อนที่ไปตรงๆ ได้ จึงต้องทำการหลบหลีกสิ่งกีดขวาง โดยเมื่อโปรแกรมพบสิ่งกีดขวางแล้วก็ทำการสร้างรัศมีสมมุติขึ้นในโปรแกรมขึ้นเพื่อเป็นการบอก ให้ตัวหุ่นรับรู้ว่ามีสิ่งกีดขวางห้ามที่จะเคลื่อนที่เข้าไปยังอาณาเขตดังกล่าว ซึ่ง หุ่นจะพยายามที่จะหลบหลีกสิ่งกีดขวางไปตามรัศมีของสิ่งกีดขวางนั้นเมื่อสามารถหลบได้แล้วก็จะเคลื่อนที่ไปยังตำแหน่งที่ต้องการ

จากกระบวนการข้างต้นจะสามารถสรุปเป็นการทำงาน โดยเลือกคำสั่งที่จะสั่งให้กับหุ่นยนต์โดยพิจารณาจาก

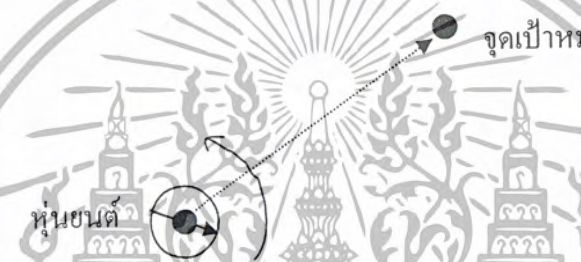
ถ้าจุดศูนย์กลางของรถอยู่ใกล้จุดเป้าหมายในรัศมีที่กำหนด ก็จะสั่งให้หุ่นยนต์หยุดเดินแล้วพร้อมที่จะเดินไปยังจุดเป้าหมายอันต่อไป



ถ้ามุมของหุ่นยนต์มากกว่ามุมของจุดเป้าหมายก็จะสั่งให้หุ่นยนต์หันขวา



ถ้ามุมของหุ่นยนต์น้อยกว่ามุมของจุดเป้าหมายก็จะสั่งให้หุ่นยนต์หันซ้าย



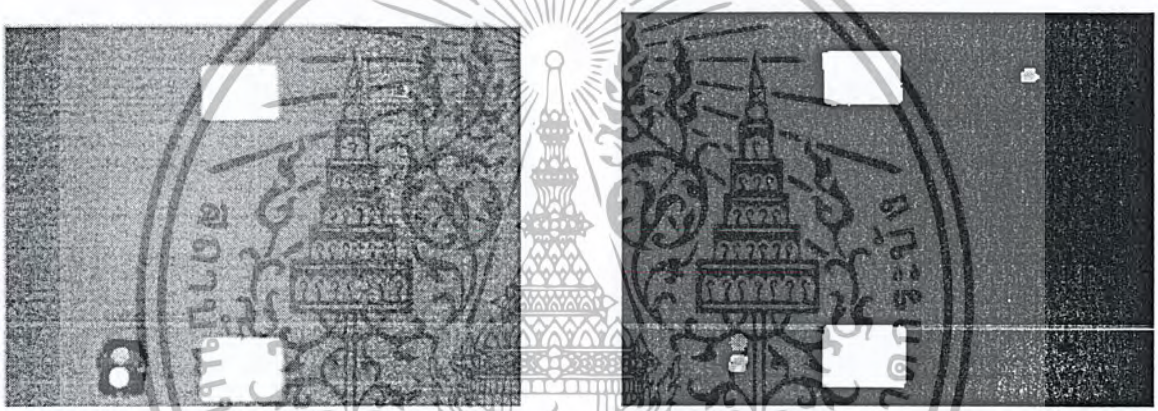
การแสดงการทำงานของส่วนประมวลผลจะเป็นดังโฟล์วชาร์ทที่ภาคผนวก

บทที่ 4

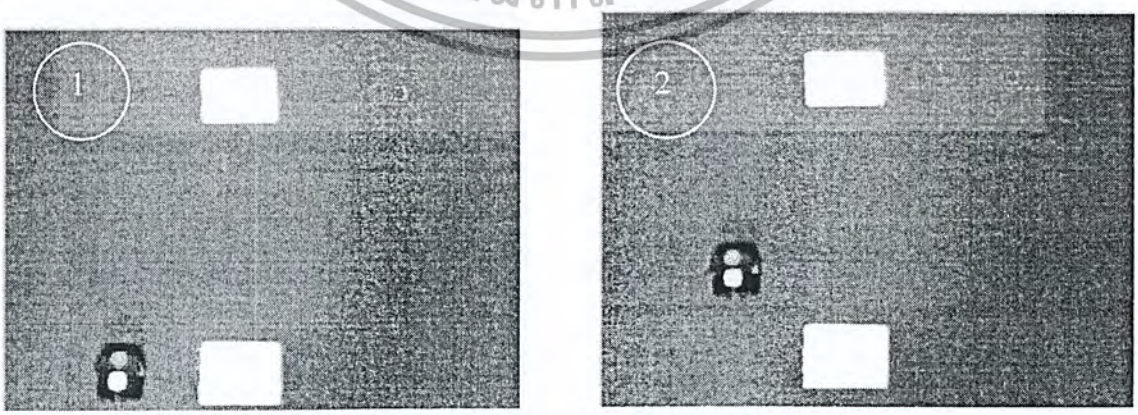
ผลการทดลอง

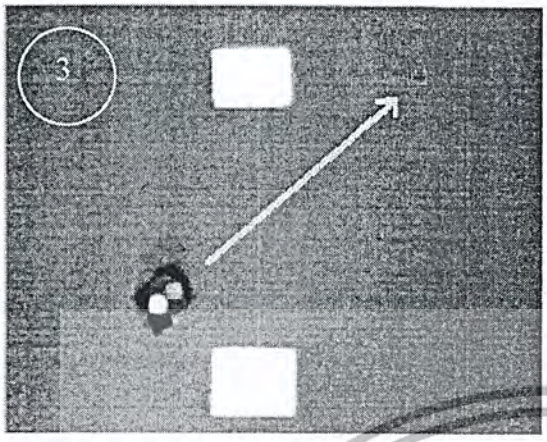
จากการทดลองนี้ได้จำลองการทำงานในแบบโรงงาน หรือ โกดังสินค้าที่มีการเก็บของอย่างเป็นระเบียบ ดังนั้นการทำงานจึงแบ่งเป็นแบบที่มาตรฐานที่มีการกำหนดตำแหน่งแน่นอน และ การทำงานที่เป็น การใช้อิมเมจโปรเซสซิ่ง

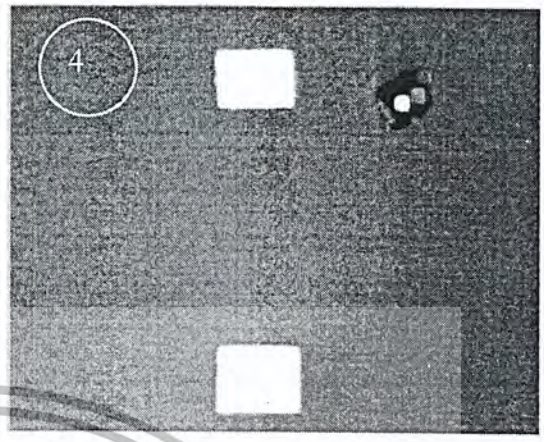
จากรูปเป็นการจับภาพและประมวลผลโดยใช้ GrayScale



1) ผลการทดลองโดยที่ไม่มีสิ่งกีดขวาง

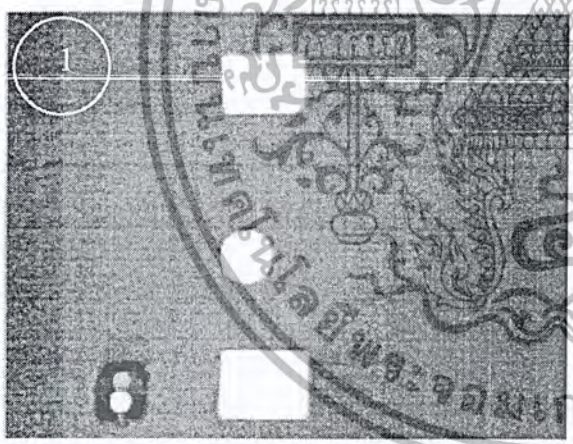


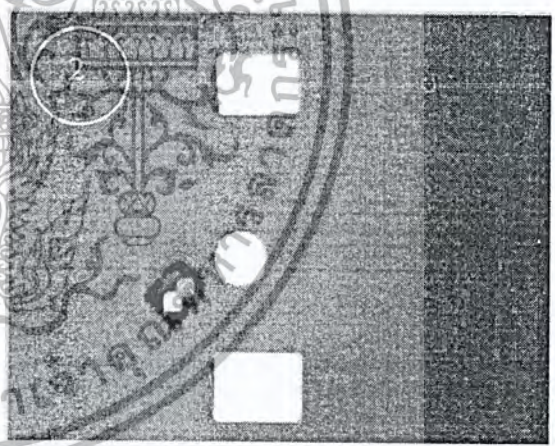


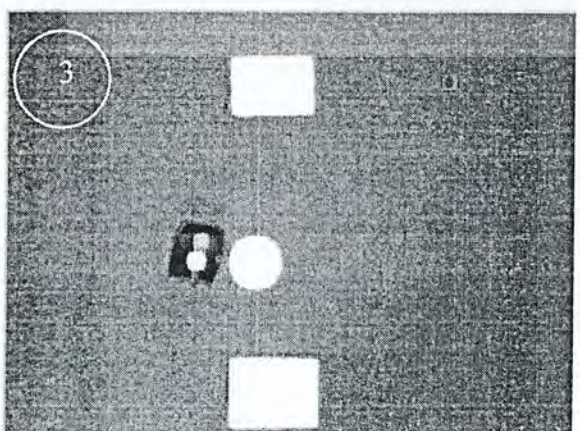


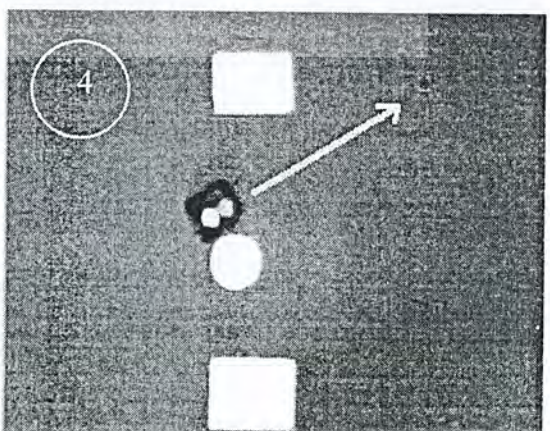
2) ผลการทดลองโดยที่มีสิ่งกีดขวาง

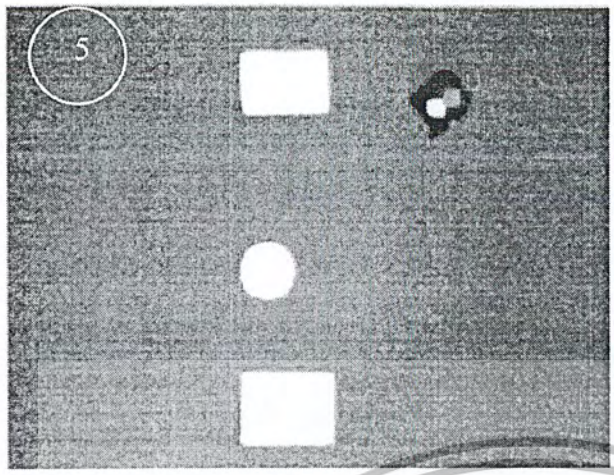
ซึ่งสิ่งกีดขวางในที่นี้จะสมมุติให้เป็นเหตุการณ์ที่อาจจะเกิดขึ้นจากเหตุบังเอิญ เช่น ของตกลงมาขวางคนขี่นขวางเส้นทางวิ่งของตัวหุ่นอยู่ หุ่นก็จำเป็นจะต้องเคลื่อนไหวหลบได้











จากผลการทดลองพบว่าหุ่นสามารถทำงานได้ โดยที่หลบสิ่งกีดขวางและเดินทางไปยังตำแหน่งที่ต้องการได้อย่างที่ต้องการ

บทที่ 5
สรุปผลการทดลอง

1) สรุปผลการดำเนินงาน

โครงงานนี้เกี่ยวกับการสร้างหุ่นยนต์ที่ควบคุมด้วยคอมพิวเตอร์โดยใช้รีโมตคอนโทรลแบบอินฟราเรดเป็นตัวส่ง ข้อมูลระหว่างหุ่นยนต์กับคอมพิวเตอร์ การควบคุมหุ่นยนต์จะควบคุมผ่านคอมพิวเตอร์โดยจะมีกล้องวีดีโอจับความเคลื่อนไหว ตำแหน่งของหุ่นยนต์และสิ่งกีดขวางเพื่อนำไปใช้ในการออกคำสั่งกับหุ่นยนต์

จากการดำเนินงาน หุ่นยนต์สามารถเดินทางตามเส้นทางที่กำหนดโดยมีความสามารถพื้นฐาน คือ เลี้ยวซ้าย เลี้ยวขวา หันหน้า หันหลัง พร้อมทั้งมีความสามารถสำคัญ คือ หลบสิ่งกีดขวางได้ตามที่ต้องการ และสามารถหาเส้นทางในการที่ไปยังจุดหมายได้ด้วย

2) ปัญหาที่เกิดและแนวทางแก้ไข

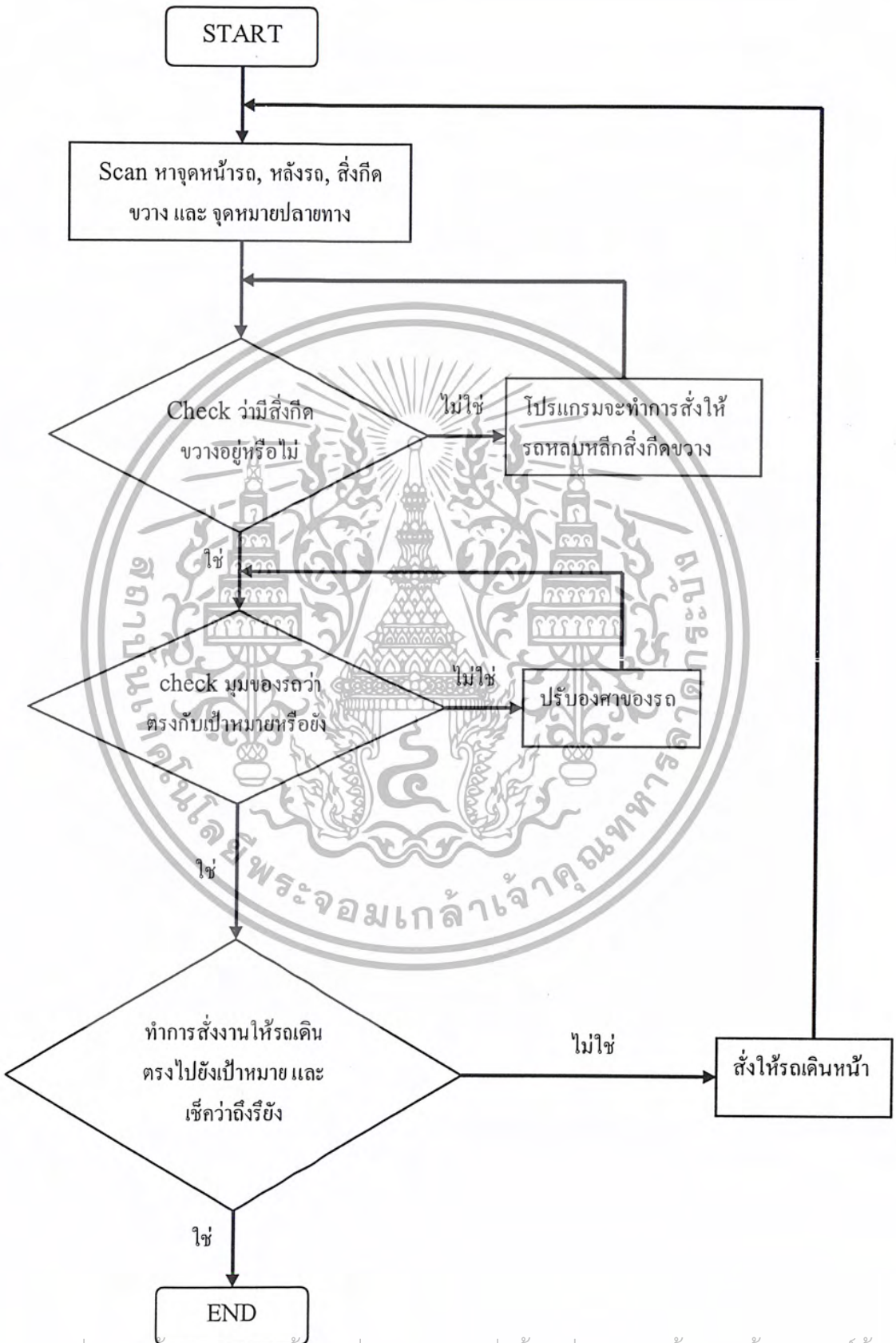
1. ปัญหาเรื่องของกล้องที่ใช้ในการตรวจจับภาพ เพราะไม่สามารถควบคุมความสามารถในการปรับเรื่องแสง เนื่องจากตัวกล้องมีการปรับค่าแสงโดยอัตโนมัติ แต่ สามารถปรับแก้ทางฮาร์ดแวร์ หรือ ทางซอฟแวร์ได้
2. เรื่องการเขียนโปรแกรม เนื่องจากยังไม่มีความชำนาญในการเขียน โปรแกรมเพียงพอที่จะเขียนให้โปรแกรมมีความกะทัดรัด และสามารถทำงานได้อย่างมีประสิทธิภาพ

3) แนวทางพัฒนาในอนาคต

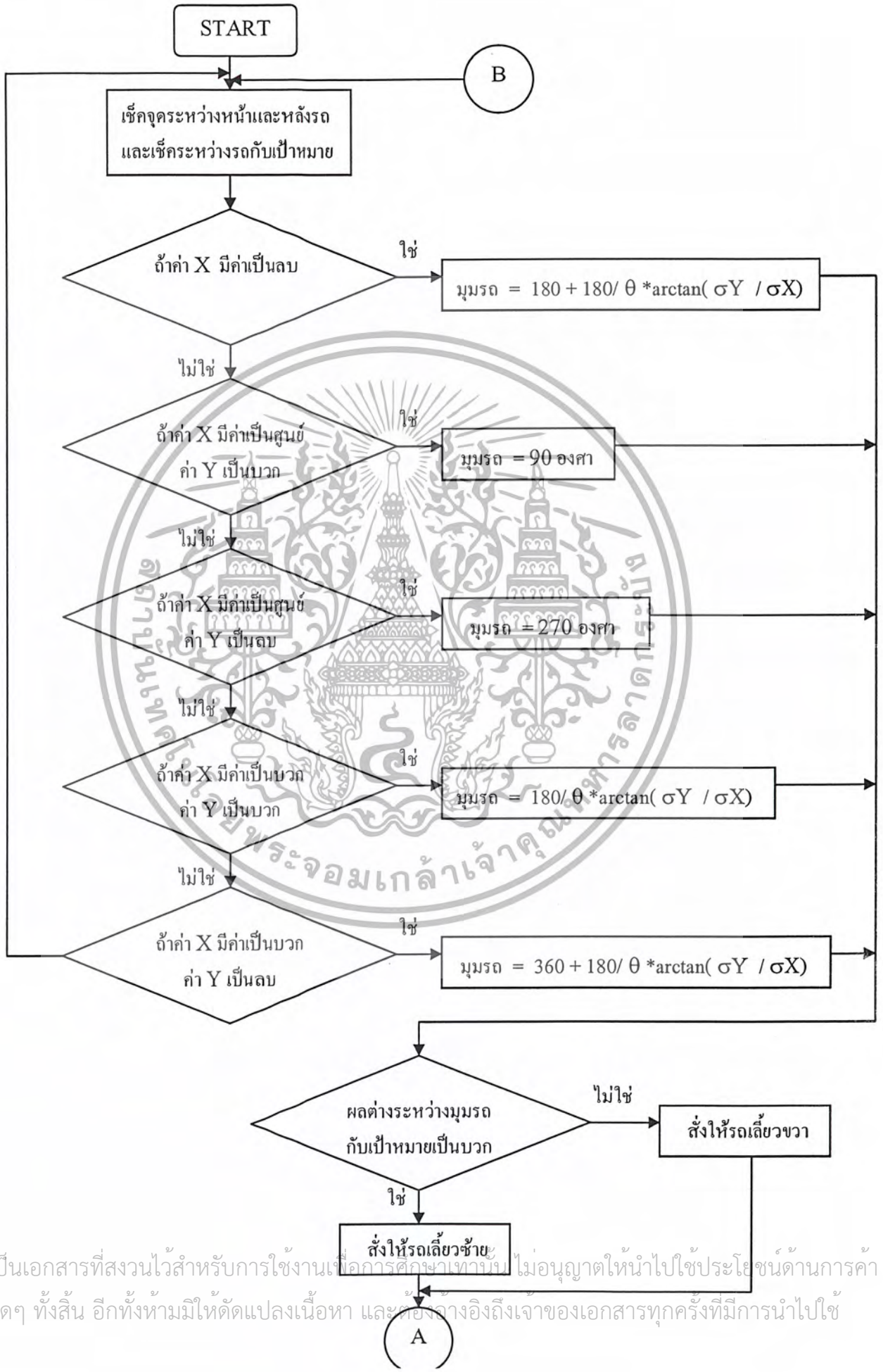
1. ทำระบบการสื่อสารด้วยคลื่นวิทยุ เพื่อลดข้อจำกัดการส่งข้อมูลให้เป็นเส้นตรง และเพื่อให้สามารถส่งผ่านสิ่งกีดขวางได้
2. เลือกใช้กล้องที่มีประสิทธิภาพมากขึ้นเพื่อลดปัญหาเรื่องภาพที่ออกมา

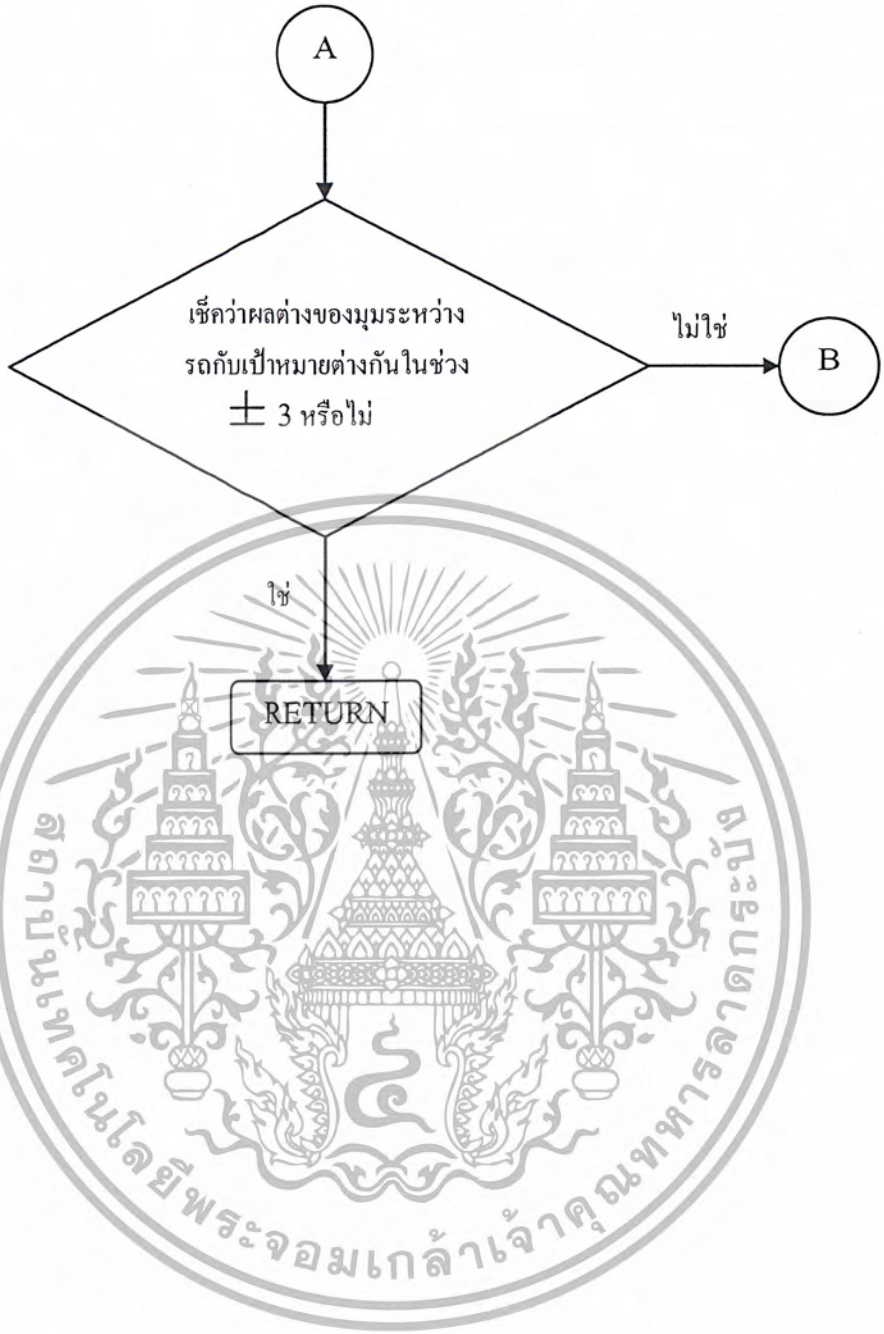
ภาคผนวก

โฟลวชาร์ทการทำงานทั้งหมดของโปรแกรม

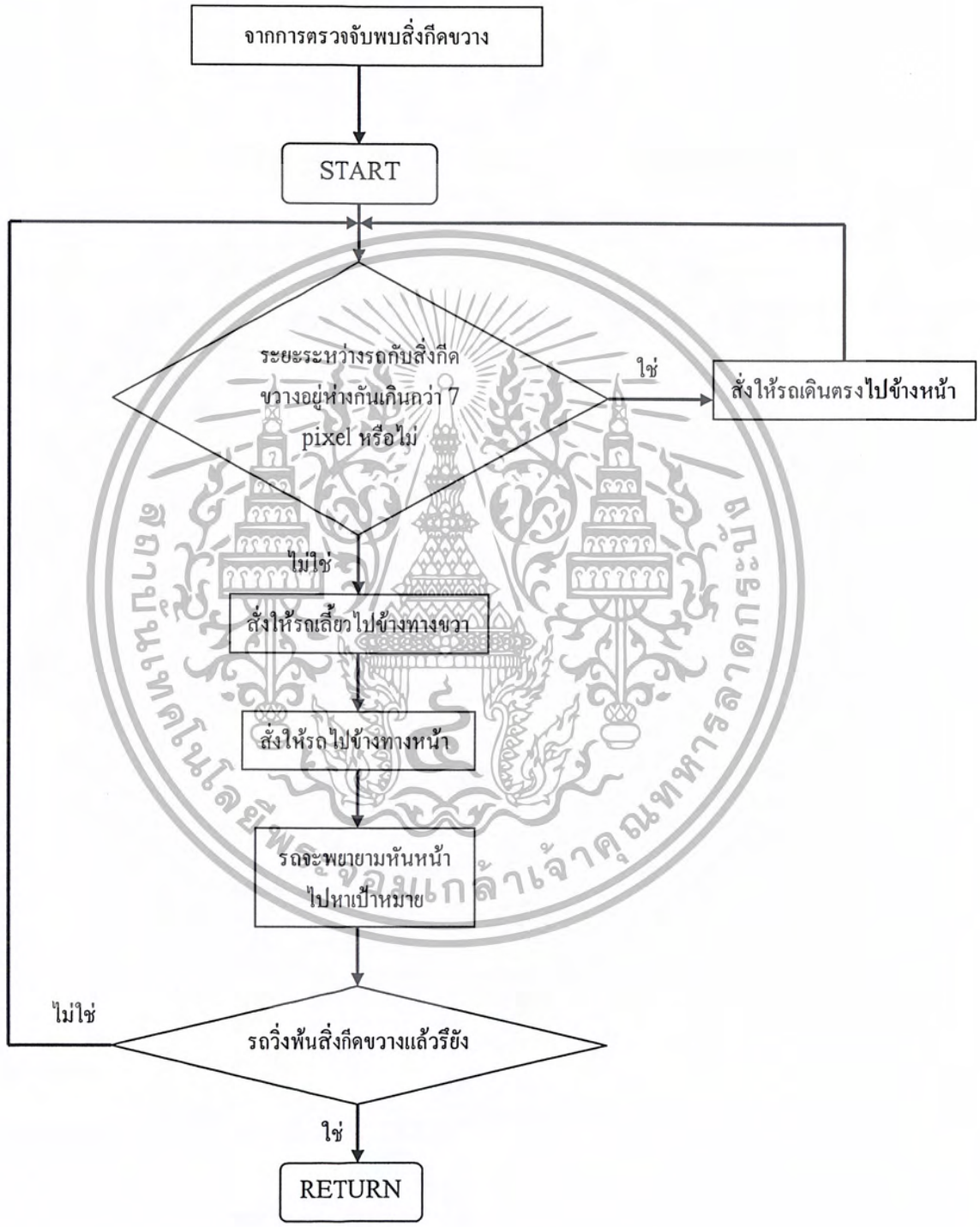


โฟลวชาร์ทการปรับองศาของรถ

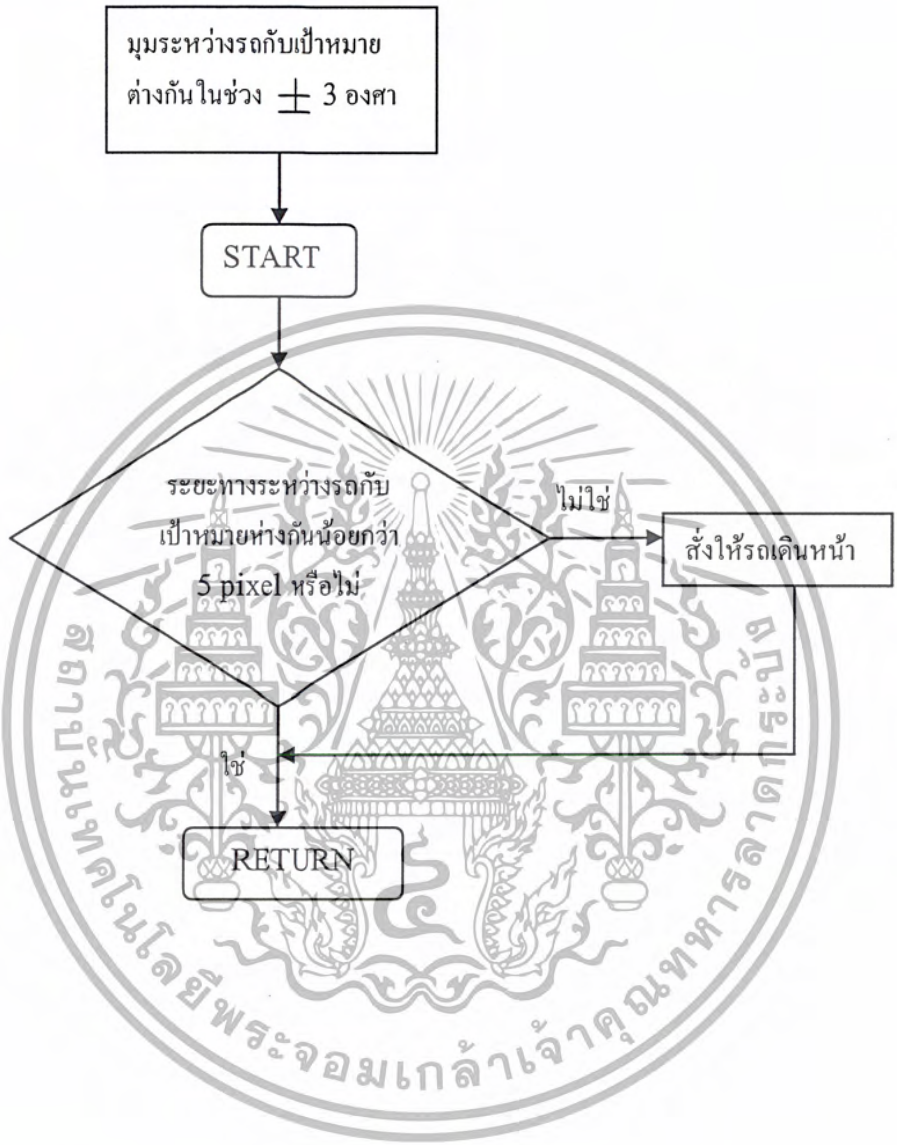




โฟลวชาร์ทในส่วนการสั่งให้รถหลบหลีกสิ่งกีดขวาง



โฟลวชาร์ททำการสั่งงานให้รถเดินตรงไปยังเป้าหมาย



```
// This is the path that use to process this project
// ImageFinalDlg.cpp : implementation file
//

#include "stdafx.h"
#include "ImageFinal.h"
#include "ImageFinalDlg.h"
#include "math.h"
#include "conio.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV supp
ort
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
```

```
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
//////
// CImageFinalDlg dialog

CImageFinalDlg::CImageFinalDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CImageFinalDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CImageFinalDlg)
    m_xCross = 0;
    m_yCross = 0;
    m_backX = 0;
    m_backY = 0;
    m_carDegree = 0;
    m_centerX = 0;
    m_centerY = 0;
    m_desDegree = 0;
    m_frontX = 0;
    m_frontY = 0;
    m_autoDestinationX = 0;
    m_autoDestinationY = 0;
    m_turn = _T("");
    m_checkDistance = 0;
    m_distanceObject = 0;
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in
Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);

//==================My Code=============================================
==
////////////////////// Set destination BMP to NULL first
    m_destinationBitmapInfoHeader = NULL;

}

////////////////////// Additional generic functions

static unsigned PixelBytes(int w, int bpp)
{
    return (w * bpp + 7) / 8;
}

static unsigned DibRowSize(int w, int bpp)
{
    return (w * bpp + 31) / 32 * 4;
}
```

```
static unsigned DibRowSize(LPBITMAPINFOHEADER pbi)
{
    return DibRowSize(pbi->biWidth, pbi->biBitCount);
}

static unsigned DibRowPadding(int w, int bpp)
{
    return DibRowSize(w, bpp) - PixelBytes(w, bpp);
}

static unsigned DibRowPadding(LPBITMAPINFOHEADER pbi)
{
    return DibRowPadding(pbi->biWidth, pbi->biBitCount);
}

static unsigned DibImageSize(int w, int h, int bpp)
{
    return h * DibRowSize(w, bpp);
}

static size_t DibSize(int w, int h, int bpp)
{
    return sizeof (BITMAPINFOHEADER) + DibImageSize(w, h, bpp);
}

//////////////////////// end of generic functions

void CImageFinalDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CImageFinalDlg)
    DDX_Control(pDX, IDC_VIDEOPROCESS, m_videoProcess);
    DDX_Text(pDX, IDC_X, m_xCross);
    DDX_Text(pDX, IDC_Y, m_yCross);
    DDX_Text(pDX, IDC_BACK_X, m_backX);
    DDX_Text(pDX, IDC_BACK_Y, m_backY);
    DDX_Text(pDX, IDC_CAR_DEGREE, m_carDegree);
    DDX_Text(pDX, IDC_CENTER_X, m_centerX);
    DDX_Text(pDX, IDC_CENTER_Y, m_centerY);
    DDX_Text(pDX, IDC_DES_DEGREE, m_desDegree);
    DDX_Text(pDX, IDC_FRONT_X, m_frontX);
    DDX_Text(pDX, IDC_FRONT_Y, m_frontY);
    DDX_Text(pDX, IDC_AUTO_X, m_autoDestinationX);
    DDX_Text(pDX, IDC_AUTO_Y, m_autoDestinationY);
    DDX_Text(pDX, IDC_EDIT1, m_turn);
    DDX_Text(pDX, IDC_DISTANCE, m_checkDistance);
    DDX_Text(pDX, IDC_DIS_OBJ, m_distanceObject);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CImageFinalDlg, CDialog)
    //{{AFX_MSG_MAP(CImageFinalDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
```

```
    ON_WM_QUERYDRAGICON()
    ON_WM_MOUSEMOVE()
    ON_BN_CLICKED(IDC_LEFT, OnLeft)
    ON_BN_CLICKED(IDC_RIGHT, OnRight)
    ON_BN_CLICKED(IDC_UP, OnUp)
    ON_BN_CLICKED(IDC_DOWN, OnDown)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
//////
// CImageFinalDlg message handlers

BOOL CImageFinalDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu)
;
        }
    }

    // Set the icon for this dialog.  The framework does this automatic
ally
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);            // Set big icon
    SetIcon(m_hIcon, FALSE);           // Set small icon

    // TODO: Add extra initialization here
    _outp(0x378, 0x00);                // initial port

    char sProcessReg[] = "HKEY_LOCAL_MACHINE\\Software\\Eakle\\ImageFin
al\\Camera";
    m_videoProcess.PrepareControl("VideoProcess", sProcessReg, 0 );
    m_videoProcess.ConnectCamera2();
    m_videoProcess.SetEnablePreview(TRUE);

    // Find the screen coodinates of the binary videoportal
    m_videoProcess.GetWindowRect(m_rectForProcessedView);
    ScreenToClient(m_rectForProcessedView);
    allocateDib(CSize(320, 240));
```

```
    // Start grabbing frame data for Procssed videoportal (bottom one)
    m_videoProcess.StartVideoHook(0);

    return TRUE;  // return TRUE  unless you set the focus to a control
}

void CImageFinalDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

// If you add a minimize button to your dialog, you will need the code
below
//  to draw the icon.  For MFC applications using the document/view mod
el,
//  this is automatically done for you by the framework.

void CImageFinalDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this to obtain the cursor to display while the user
 drags
//  the minimized window.
HCURSOR CImageFinalDlg::OnQueryDragIcon()
```

```
{
    return (HCURSOR) m_hIcon;
}

BEGIN_EVENTSINK_MAP(CImageFinalDlg, CDialog)
    //{{AFX_EVENTSINK_MAP(CImageFinalDlg)
    ON_EVENT(CImageFinalDlg, IDC_VIDEOPROCESS, 1 /* PortalNotification
*/, OnPortalNotificationVideoprocess, VTS_I4 VTS_I4 VTS_I4 VTS_I4)
    //}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

void CImageFinalDlg::OnPortalNotificationVideoprocess(long lMsg, long l
Param1, long lParam2, long lParam3)
{
    // TODO: Add your control notification handler code here
    // This function is called at the camera's frame rate

#define NOTIFICATIONMSG_VIDEOHOOK   10


    // Declare some useful variables
    // QCSDKMFC.pdf (Quickcam MFC documentation) p. 103 explains the va
riables lParam1, lParam2, lParam3 too

    LPBITMAPINFOHEADER lpBitmapInfoHeader; // Frame's info header conta
ins info like width and height
    LPBYTE lpBitmapPixelData; // This pointer-to-long will point to the
 start of the frame's pixel data
    unsigned long lTimeStamp; // Time when frame was grabbed

    if ((IsDlgButtonChecked(IDC_GOGOGO)) && (!IsDlgButtonChecked(IDC_CH
ECK_MANUAL))){
    switch(lMsg) {
        case NOTIFICATIONMSG_VIDEOHOOK:
            {
                lpBitmapInfoHeader = (LPBITMAPINFOHEADER) lParam1;
                lpBitmapPixelData = (LPBYTE) lParam2;
                lTimeStamp = (unsigned long) lParam3;

                MyImage(lpBitmapInfoHeader, lpBitmapPixelData);
                MyImageProcessing(lpBitmapInfoHeader);
                displayMyResults(lpBitmapInfoHeader);

            }
            break;

        default:
            break;
    }
    }
}

void CImageFinalDlg::allocateDib(CSize sz)
{
    // Purpose: allocate information for a device independent bitmap (D
```

```
IB)
    // Called from OnInitVideo

    if(m_destinationBitmapInfoHeader) {
        free(m_destinationBitmapInfoHeader);
        m_destinationBitmapInfoHeader = NULL;
    }

    if(sz.cx | sz.cy) {
        m_destinationBitmapInfoHeader = (LPBITMAPINFOHEADER)malloc(DibS
ize(sz.cx, sz.cy, 24));
        ASSERT(m_destinationBitmapInfoHeader);
        m_destinationBitmapInfoHeader->biSize = sizeof(BITMAPINFOHEADER
);
        m_destinationBitmapInfoHeader->biWidth = sz.cx;
        m_destinationBitmapInfoHeader->biHeight = sz.cy;
        m_destinationBitmapInfoHeader->biPlanes = 1;
        m_destinationBitmapInfoHeader->biBitCount = 24;
        m_destinationBitmapInfoHeader->biCompression = 0;
        m_destinationBitmapInfoHeader->biSizeImage = DibImageSize(sz.cx
, sz.cy, 24);
        m_destinationBitmapInfoHeader->biXPelsPerMeter = 0;
        m_destinationBitmapInfoHeader->biYPelsPerMeter = 0;
        m_destinationBitmapInfoHeader->biClrImportant = 0;
        m_destinationBitmapInfoHeader->biClrUsed = 0;
    }
}

void CImageFinalDlg::displayMyResults(LPBITMAPINFOHEADER lpThisBitmapIn
foHeader)
{
    // displayMyResults: Displays results of doMyImageProcessing() in t
he videoport
    // Notes: StretchDIBits stretches a device-independent bitmap to th
e appropriate size

    CDC         *pDC;   // Device context to display bitmap data

    pDC = GetDC();
    int nOldMode = SetStretchBltMode(pDC->GetSafeHdc(),COLORONCOLOR);

    StretchDIBits(
        pDC->GetSafeHdc(),
        17,                                         // videoportal left
-most coordinate
        17,                                         // videoportal top-
most coordinate
        320,                                        // videoportal widt
h
        240,                                        // videoportal heig
ht
        0,                                          // Row position to
display bitmap in videoportal
        0,                                          // Col position to
display bitmap in videoportal
```

```
        lpThisBitmapInfoHeader->biWidth,            // m_destinationBmp
's number of columns
        lpThisBitmapInfoHeader->biHeight,           // m_destinationBmp
's number of rows
        m_destinationBmp,                           // The bitmap to di
splay; use the one resulting from doMyImageProcessing
        (BITMAPINFO*)m_destinationBitmapInfoHeader, // The bitmap's hea
der info e.g. width, height, number of bits etc
        DIB_RGB_COLORS,                             // Use default 24-b
it color table
        SRCCOPY                                     // Just display
    );

    SetStretchBltMode(pDC->GetSafeHdc(),nOldMode);

    ReleaseDC(pDC);

    // Note: 04/24/02 - Added the following:
    // Christopher Wagner cwagner@fas.harvard.edu noticed that memory w
asn't being freed

    // Recall OnPortalNotificationProcessedview, which gets called ever
ytime
    // a frame of data arrives, performs 3 steps:
    // (1) grayScaleTheFrameData - which malloos m_destinationBmp
    // (2) doMyImageProcesing
    // (3) displayMyResults - which we're in now
    // Since we're finished with the memory we malloc'ed for m_destinat
ionBmp
    // we should free it:

    free(m_destinationBmp);

    // End of adds
}

void CImageFinalDlg::MyImage(LPBITMAPINFOHEADER lpThisBitmapInfoHeader,
 LPBYTE lpThisBitmapPixelData)
{

    unsigned int    W, H;           // Width and Height of current fr
ame [pixels]
    BYTE            *sourceBmp;     // Pointer to current frame of da
ta
    unsigned int    row, col;
    unsigned long   i;
    BYTE            grayValue;

    BYTE            redValue;
    BYTE            greenValue;
    BYTE            blueValue;

    W = lpThisBitmapInfoHeader->biWidth;  // biWidth: number of columns
    H = lpThisBitmapInfoHeader->biHeight; // biHeight: number of rows
```

```
    // Store pixel data in row-column vector format
    // Recall that each pixel requires 3 bytes (red, blue and green byt
es)
    // m_destinationBmp is a protected member and declared in binarizeD
lg.h

    m_destinationBmp = (BYTE*)malloc(H*3*W*sizeof(BYTE));

    // Point to the current frame's pixel data
    sourceBmp = lpThisBitmapPixelData;

    for (row = 0; row < H; row++) {
        for (col = 0; col < W; col++) {

            // Recall each pixel is composed of 3 bytes
            i = (unsigned long)(row*3*W + 3*col);

            // The source pixel has a blue, green andred value:
            blueValue  = *(sourceBmp + i);
            greenValue = *(sourceBmp + i + 1);
            redValue   = *(sourceBmp + i + 2);

            // A standard equation for computing a grayscale value base
d on RGB values
            grayValue = (BYTE)(0.299*redValue + 0.587*greenValue + 0.11
4*blueValue);

        if ((blueValue < 80) && (greenValue < 80) && (redValue > 180))
  // red Dot
        {
            *(m_destinationBmp + i)     =
            *(m_destinationBmp + i + 1) =
            *(m_destinationBmp + i + 2) = 150;
        }

        else if ((blueValue > 200) && (greenValue < 120) && (redValue
> 200))  // violet Dot
        {
            *(m_destinationBmp + i)     =
            *(m_destinationBmp + i + 1) =
            *(m_destinationBmp + i + 2) = 50;
        }

        else if ((blueValue > 200) && (greenValue > 200) && (redValue
> 200))  // white Dot
        {
            *(m_destinationBmp + i)     =
            *(m_destinationBmp + i + 1) =
            *(m_destinationBmp + i + 2) = 255;
        }

       else if ((blueValue > 120) && (greenValue < 70) && (redValue <
90))  // blue Dot
        {
            *(m_destinationBmp + i)     =
```

```
            *(m_destinationBmp + i + 1) =
            *(m_destinationBmp + i + 2) = 100;
        }

        else
        {
            *(m_destinationBmp + i)     =
            *(m_destinationBmp + i + 1) =
            *(m_destinationBmp + i + 2) = 0;
        }
       }
    }

}

void CImageFinalDlg::MyImageProcessing(LPBITMAPINFOHEADER lpThisBitmapI
nfoHeader)
{
    // doMyImageProcessing:  This is where you'd write your own image p
rocessing code
    // Task: Read a pixel's grayscale value and process accordingly

    unsigned int    m_carFrontX1,m_carFrontX2,m_carFrontX;
    unsigned int    m_carFrontY1,m_carFrontY2,m_carFrontY;

    unsigned int    m_carBackX1,m_carBackX2,m_carBackX;
    unsigned int    m_carBackY1,m_carBackY2,m_carBackY;

    unsigned int    m_objectX1,m_objectX2,m_objectX;
    unsigned int    m_objectY1,m_objectY2,m_objectY;

    unsigned int    m_distance,m_distanceObj;

    unsigned int    m_destinationX1,m_destinationX2,m_destinationX;
    unsigned int    m_destinationY1,m_destinationY2,m_destinationY;

    unsigned int    W, H;           // Width and Height of current fr
ame [pixels]
    unsigned int    row, col;       // Pixel's row and col positions
    unsigned long   i;              // Dummy variable for row-column
vector

    W = lpThisBitmapInfoHeader->biWidth;  // biWidth: number of columns
    H = lpThisBitmapInfoHeader->biHeight; // biHeight: number of rows

    m_carFrontX1 = 321;
    m_carFrontY1 = 241;
    m_carBackX1 = 321;
    m_carBackY1 = 241;
    m_destinationX1 = 321;
    m_destinationY1 = 321;
    m_objectX1 = 321;
    m_objectY1 = 321;

    for (row = 0; row < H; row++) {
```

```
        for (col = 0; col < W; col++) {

            // Recall each pixel is composed of 3 bytes
            i = (unsigned long)(row*3*W + 3*col);

            if (( *(m_destinationBmp + i) == 50))
            {
                if (col < m_carFrontX1)
                    m_carFrontX1=col,m_carFrontY1=row;
                else if (col > m_carFrontX1)
                    m_carFrontX2=col,m_carFrontY2=row;
            }

            if (( *(m_destinationBmp + i) == 255))
            {
                if (col < m_carBackX1)
                    m_carBackX1=col,m_carBackY1=row;
                else if (col > m_carBackX1)
                    m_carBackX2=col,m_carBackY2=row;
            }

            if (( *(m_destinationBmp + i) == 150))
            {
                if (col < m_destinationX1)
                    m_destinationX1=col,m_destinationY1=row;
                else if ( col > m_destinationX1)
                    m_destinationX2=col,m_destinationY2=row;
            }

            if (( *(m_destinationBmp + i) == 100))
            {
                if (col < m_objectX1)
                    m_objectX1=col,m_objectY1=row;
                else if ( col > m_objectX1)
                    m_objectX2=col,m_objectY2=row;
            }
    }

    // calculate the center of Dot
    m_carFrontX = ((m_carFrontX2 + m_carFrontX1)/2);
    m_carFrontY = ((m_carFrontY2 + m_carFrontY1)/2);

    m_carBackX = ((m_carBackX2 + m_carBackX1)/2);
    m_carBackY = ((m_carBackY2 + m_carBackY1)/2);

    m_objectX = ((m_objectX2 + m_objectX1)/2);
    m_objectY = ((m_objectY2 + m_objectY1)/2);

    m_destinationX = ((m_destinationX2 + m_destinationX1)/2);
    m_destinationY = ((m_destinationY2 + m_destinationY1)/2);

    // distance to Goal
    m_distance = sqrt(((m_destinationX - m_frontX)^2) + ((m_destination
Y - m_frontY)^2));
```

```
    // distance to Object
    m_distanceObj = sqrt(((m_objectX - m_frontX)^2) + ((m_objectY - m_f
rontY)^2));

    // display it on block
    m_frontX = m_carFrontX;
    m_frontY = m_carFrontY;


    m_backX = m_carBackX;
    m_backY = m_carBackY;

    m_centerX = m_objectX;
    m_centerY = m_objectY;

    m_checkDistance = m_distance;
    m_distanceObject = m_distanceObj;

    // calculate the angle and display it

    double x, y, x1, y1, cd, cd1;

    y = m_carFrontY - m_carBackY;   // car
    x = m_carFrontX - m_carBackX;

    y1 = m_destinationY - m_backY;  // destination
    x1 = m_destinationX - m_backX;

    cd  = (atan(y/x)*180)/(22/7);   // degree car
    cd1 = (atan(y1/x1)*180)/(22/7); // degree destination

    m_autoDestinationX = m_destinationX;
    m_autoDestinationY = m_destinationY;

    m_desDegree = cd1;
    m_carDegree = cd;



    // CONTROL ROBOT BY COMPUTER

    CheckAndControl();
    CheckObject();
    CheckGo();


    UpdateData(FALSE);

    }
}

void CImageFinalDlg::OnMouseMove(UINT nFlags, CPoint point)
{
    m_xCross = point.x;     // Display coordinate X,Y
    m_yCross = point.y;
```

```
    if (m_xCross > 337)
    {
        m_xCross = m_yCross = 0;
    }
    else if (m_xCross < 17)
    {
        m_xCross = m_yCross = 0;
    }
    else if (m_yCross > 257)
    {
        m_xCross = m_yCross = 0;
    }
    else if (m_yCross < 17)
    {
        m_xCross = m_yCross = 0;
    }
    else
    {
        m_xCross = (point.x-17);
        m_yCross = ((point.y-17)-240)*(-1);
    }

    UpdateData(FALSE);
    //CDialog::OnMouseMove(nFlags, point);
}

void CImageFinalDlg::OnLeft()
{
    if (IsDlgButtonChecked(IDC_CHECK_MANUAL))
    {
        _outp(0x378, 0xFB);
        _sleep(100);
        _outp(0x378, 0x00);
    }
}

void CImageFinalDlg::OnRight()
{
    if (IsDlgButtonChecked(IDC_CHECK_MANUAL))
    {
        _outp(0x378, 0xF7);
        _sleep(100);
        _outp(0x378, 0x00);

    }
}

void CImageFinalDlg::OnUp()
{
    if (IsDlgButtonChecked(IDC_CHECK_MANUAL))
    {
        _outp(0x378, 0xFE);
        _sleep(100);
        _outp(0x378, 0x00);
```

```
    }
}

void CImageFinalDlg::OnDown()
{
    if (IsDlgButtonChecked(IDC_CHECK_MANUAL))
    {
        _outp(0x378, 0xFD);
        _sleep(100);
        _outp(0x378, 0x00);
    }
}

void CImageFinalDlg::CheckAndControl()
{
    int check;
    check = m_desDegree - m_carDegree;

    if (m_frontY < 80)
        _outp(0x378, 0xFE);
    else
    {
    if (check > 7)
        _outp(0x378, 0xFB);                    // left
    else if (check < -7)
        _outp(0x378, 0xF7);                    // right
    else if ((check > -7) && (check < 7))
        _outp(0x378, 0xFF);
    }

    return;
}

void CImageFinalDlg::CheckGo()
{
    int result;
    result = m_carDegree - m_desDegree;

    if (m_checkDistance > 5)
    {
    if ((abs(result) >= 0) && (abs(result) <= 5))
        _outp(0x378, 0xFE);
    }
    else
        _outp(0x378, 0xFF);

    return;
}

void CImageFinalDlg::CheckObject()
{
    if (m_distanceObject < 50)
    {
        if (m_distanceObject < 7)
            _outp(0x378, 0xFB);
```

```
        else
            _outp(0x378, 0xFF);
    }
    else
        _outp(0x378, 0xFF);

    return;
}
```

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายๆฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อ.สุมิตร พนาอุดมทรัพย์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่แนะนำ และช่วยเหลือเสมอมา ซึ่งขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลัง เอาใจใส่เสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสูงสุดประมาณ และ ขอกราบขอบพระคุณมา ณ ที่นี้

ลักษิกา วิบูลเทพาชาติ
เอกพล ชัยเจริญชนะ

บรรณานุกรม

1. "หนังสือรีโมท เครื่องควบคุมไร้สาย", บริษัท ซีเอ็ดยูเคชัน จำกัด (มหาชน), พิมพ์ครั้งแรก พ.ศ. 2538
2. นส.พรีสา อารีกุล, นายเอกรัตน์ พนมฤทธิ์, "หุ่นยนต์ควบคุมด้วยคอมพิวเตอร์", สาขาวิศวกรรมระบบควบคุม, สจล. ,2543
3. นิรุธ อำนวยศิลป์, "เขียนเกมอย่างมืออาชีพ visual c++ และ DirectX", พิมพ์ครั้งแรก พ.ศ. 2545
4. พีรภัทร์ สว่างเพียร, "เทคนิคการเขียนโปรแกรมและเกมด้วย Visual C++", พิมพ์ครั้งแรก พ.ศ. 2546
5. www.microship.com
6. www.fairchildsemi.com
7. www.st.com