หุ่นยนต์เคลื่อนที่ตามวัตถุ

OBJECT TRACKING ROBOT

โดย

นายไพศาล ฉัตรชนกิจ

นายเมธี ไทยอุทิศ

นายอวิรุทธ์ วิจิตรเมฆทอง

อาจารย์ที่ปรึกษา

รศ.ดร.สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

หุ่นยนต์เคลื่อนที่ตามวัตถุ

OBJECT TRACKING ROBOT

โดย

นายไพศาล ธีฉายนกิจ

นายเมธี ไทยอุทิศ

นายอวิรุทธ์ วิจิตรเมฆทอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ปริญญานิพนธ์ปีการศึกษา 2550

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์เคลื่อนที่ตามวัตถุ

ผู้จัดทำ

1.นายไพศาล ลีลาธนกิจ

2.นายเมธี ไทยอุทิศ

3.นายอวิรุทธ์ วิจิตรเมฆทอง

.....(อาจารย์ที่ปรึกษา)

(รศ.ดร.สุรพันธ์ เอื้อไพบูลย์)

วันที่ 14 / 03 / 2551

หุ่นยนต์เคลื่อนที่ตามวัตถุ

นายไพศาล ลีลาธนกิจ 47010544
นายเมธี ไทยอุทิศ 47010599
นายอวิรุทธ์ วิจิตรเมฆทอง 47010964
รศ.ดร.สุรพันธ์ เอื้อไพบูลย์ (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2550

บทคัดย่อ

โครงงานหุ่นยนต์เคลื่อนที่ตามวัตถุนี้ ถูกออกแบบให้ใช้กล้องวีดิโอในการรับสัญญาณภาพ และส่งสัญญาณภาพที่ได้ให้กับคอมพิวเตอร์ เพื่อให้คอมพิวเตอร์ทำการประมวลผลด้วยภาพ เมื่อได้ข้อมูลของภาพหลังทำการประมวลผลด้วยภาพแล้ว คอมพิวเตอร์จะส่งข้อมูลของสัญญาณภาพที่ได้ผ่านพอร์ตอนุกรมของคอมพิวเตอร์ ให้กับไมโครคอนโทรลเลอร์ อาร์ม7 ที่ได้เขียนโปรแกรมไว้เพื่อใช้ในการควบคุมการขับสเต็ปมอเตอร์และทิศทางการเคลื่อนที่ของหุ่นยนต์เคลื่อนที่ตามวัตถุต่อไป

สำหรับภาคการศึกษาที่ 1 หุ่นยนต์จะได้รับข้อมูลจากคอมพิวเตอร์ และส่งข้อมูลผ่านพอร์ตอนุกรมของคอมพิวเตอร์ ให้กับไมโครคอนโทรลเลอร์ อาร์ม7 ที่ได้เขียนโปรแกรมไว้ เพื่อใช้ในการควบคุมการขับสเต็ปมอเตอร์และทิศทางการเคลื่อนที่ของหุ่นยนต์

สำหรับภาคการศึกษาที่ 2 นี้ จะทำการติดต่อกับกล้องวีดีโอในการรับสัญญาณภาพ เพื่อให้คอมพิวเตอร์ทำการประมวลผลด้วยภาพ และนำผลที่ได้ไปใช้ในการควบคุมหุ่นยนต์ ที่ใช้ไมโครคอนโทรลเลอร์ อาร์ม7 ที่ได้เขียนโปรแกรมไว้ เพื่อใช้ในการควบคุมการขับสเต็ปมอเตอร์และทิศทางการเคลื่อนที่ของหุ่นยนต์ ที่ได้ออกแบบไว้ในภาคการศึกษาที่ 1

OBJECT TRACKING ROBOT

Mr. Paisarn Leelathanakig ID.47010544
Mr. Maytee Thaiutid ID.47010599
Mr. Awirut wijitmakthong ID.47010964
Assoc. Prof. Surapan Auepaiboon (advisor)
Educational Year 2007

Abstract

This object tracking robot is designed to receive input signals from the video camera and transfer the signals to the computer for process the signals with digital image processing. Then the processed image data is sent through the computer's serial port to the programmed microcontroller ARM7 in order to drive stepping motor and the direction of the robot movement.

In the 1st semester, this robot is received data by computer. Then the data is sent through the computer's serial port to the programmed microcontroller ARM7 in order to drive stepping motor and the direction of the robot movement.

In the 2nd semester, we interface with video camera which receives image signals to the computer for processing the signals with digital image processing. Then the processed images are used for controlling robot with the programmed microcontroller ARM7 in order to drive stepping motor and the direction of the robot movement that is designed in 1st semester.

กิตติกรรมประกาศ

โครงงานหุ่นยนต์เคลื่อนที่ตามวัตถุ ซึ่งประกอบด้วยชิ้นงานและเอกสารประกอบโครงงานนี้ ที่สำเร็จลุล่วงมาด้วยดีมิได้ หากขาดอาจารย์สุรพันธ์ เอื้อไพบูลย์ ผู้ให้คำแนะนำ ดูแลเรื่องอุปกรณ์และแนวทางการดำเนินงานอย่างใกล้ชิดมาโดยตลอด พร้อมทั้งการให้ความช่วยเหลือจากรุ่นพี่และเพื่อนๆภาคอิเล็กทรอนิกส์ สุดท้ายขอขอบพระคุณผู้ปกครอง ซึ่งคอยสนับสนุนเงินทุนและให้กำลังใจเสมอมา

นายไพศาล ลีลาธนกิจ

นายเมธี ไทยอุทิศ

นายอวิรุทธ์ วิจิตรเมฆทอง

ผู้จัดทำ

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่1

บทนำ

อุปกรณ์ทางอิเล็กทรอนิกส์นับได้ว่าเป็นส่วนหนึ่งที่สำคัญต่อการดำรงชีวิตของผู้คนบนโลกในปัจจุบันไปแล้ว การศึกษาวิทยาการต่างๆ และการพัฒนาเทคโนโลยี จึงเป็นปัจจัยอย่างหนึ่งในการสรรสร้างนวัตกรรมใหม่ๆ เพื่ออำนวยความสะดวกในการใช้ชีวิตประจำวัน และจัดสรรทรัพยากรที่มีอยู่ให้ใช้ได้อย่างคุ้มค่ามากที่สุด

ในงานด้านการประกอบวงจรทางอิเล็กทรอนิกส์ในปัจจุบัน เครื่องจักรกลระบบอัตโนมัติภายในโรงงานอุตสาหกรรม ต้องมีความสามารถ ความละเอียดและความแม่นยำที่สูงมาก เนื่องจากแนวโน้มของอุปกรณ์ที่ถูกใช้งานในด้านวงจรอิเล็กทรอนิกส์ในปัจจุบันและอนาคต จะเป็นอุปกรณ์ประเภท Surface Mount Device (SMD) ซึ่งถูกออกแบบและทำให้มีขนาดเล็กลงเรื่อยๆ จนยากที่จะประกอบเองด้วยแรงงานคน เครื่องจักรจึงเป็นทางเลือกหนึ่งที่ถูกใช้ในการประกอบวงจร ดังนั้นเครื่องจักรเหล่านั้นจึงต้องมีความละเอียด ความแม่นยำและความน่าเชื่อถือสูง ซึ่งในงานที่ต้องการความละเอียดสูงมักจะใช้การประมวลผลด้วยภาพถ่าย (Digital Image Processing) โดยใช้ภาพถ่ายจากกล้องถ่ายภาพหรือกล้องวีดีโอ มาทำการประมวลผลภาพ แทนการใช้การตรวจวัดจากแบบอื่นๆ เช่น เซนเซอร์ (Sensor) ที่นำมาใช้ในการตรวจวัดโดยการสัมผัส จะมีข้อเสียจากความยุ่งยากในการสร้าง การออกแบบ ไปจนถึงการติดตั้ง ดังนั้นจะเห็นได้ว่า การประมวลผลด้วยภาพถ่าย (Digital Image Processing) จะมีความยืดหยุ่นสูงกว่าในการปรับรูปแบบการใช้งาน และมีราคาถูกกว่าเมื่อเทียบกับวิธีอื่น

รายงานนี้เป็นโครงงานหุ่นยนต์ตรวจจับวัตถุ เป็นการรับภาพวัตถุที่มีสีและรูปร่างที่เรากำหนดจากกล้องวีดิโอเข้าสู่เครื่องคอมพิวเตอร์ จากนั้นจะใช้คอมพิวเตอร์ทำการประมวลผลด้วยภาพถ่าย แล้วส่งสัญญาณที่ได้ไปใช้ในการขับมอเตอร์และควบคุมทิศทางการเคลื่อนที่ผ่านไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) ที่ได้ทำการโปรแกรมไว้แล้ว

สำหรับภาคการศึกษาที่ 1/2550 จะเริ่มทำจากการโปรแกรมไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) โดยใช้ภาษาซี ในการควบคุมการขับมอเตอร์และทิศทางการเคลื่อนที่ผ่านการกดปุ่มส่งสัญญาณจากคีย์บอร์ดคอมพิวเตอร์ แทนการรับภาพเข้ามา และทำการออกแบบและทำส่วนของโครงสร้างรถ ซึ่งรายงานฉบับนี้ได้แสดงส่วนที่ได้จัดทำในภาคการศึกษานี้ไว้เป็นบทต่างๆ

สำหรับภาคการศึกษาที่ 2/2550 จะทำการติดต่อกับกล้องวิดีโอในการรับสัญญาณภาพเพื่อให้คอมพิวเตอร์ทำการประมวลผลด้วยภาพ และนำผลที่ได้ไปใช้ในการควบคุมหุ่นยนต์ ที่ใช้ไมโครคอนโทรลเลอร์ อาร์ม7 ที่ได้เขียนโปรแกรมไว้ เพื่อใช้ในการควบคุมการขับสเต็ปมอเตอร์และทิศทางการเคลื่อนที่ของหุ่นยนต์ ที่ได้ออกแบบไว้ในภาคการศึกษาที่ 1

บทที่ 1 บทนำ แสดงรายละเอียดอย่างคร่าวๆเกี่ยวกับการศึกษาโครงงาน แนวคิดในการทำโครงงาน วัตถุประสงค์ของโครงงาน ประโยชน์ที่ได้รับจากการทำโครงงาน และขอบเขตของการทำโครงงาน

บทที่ 2 ทฤษฎี กล่าวถึง การใช้งานไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) สถาปัตยกรรมภายใน คำสั่งบางคำสั่งที่ถูกใช้งาน รวมถึงการใช้ในรูปแบบ Interupt และการติดต่อกับพอร์ตอนุกรมกับคอมพิวเตอร์ ของไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) โดยใช้ภาษาซี และการทำงานของสเต็ปปิ้งมอเตอร์ (Stepping Motor) ในส่วนของดิจิตอลอิมเมจโปรเซสซิ่ง (Digital Image Processing) จะเป็นทฤษฎีต่างๆเกี่ยวกับการประมวลผลภาพ เช่น การแบ่งส่วนภาพ (Segmentation) การทำเทรสโฮลด์ (Thresholding technique) การแยกส่วนสี (Color Segmentation) และการติดตามวัตถุ (Object Tracking)

บทที่ 3 การออกแบบ กล่าวถึง แนวคิดและวิธีการทำงานของโปรแกรม การใช้งานและการควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ (Stepping Motor) การควบคุมทิศทางของสเต็ปปิ้งมอเตอร์ (Stepping Motor) ส่วนของการประมวลผลภาพ ก็จะเป็นการอธิบายวิธีคิดและหลักการทำงานของโปรแกรม แสดงการทำงานด้วย Flow Chart

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึง การทดสอบการทำงานของสเต็ปปิ้งมอเตอร์ (Stepping Motor) การวัดรูปแบบของสัญญาณที่ได้จากไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) ที่ทำการโปรแกรมไว้ ส่วนของการประมวลผลภาพ ก็จะเป็นการทดสอบโปรแกรมย่อยต่างๆของโปรแกรมหลัก ซึ่งเป็นไปตามกระบวนการประมวลผลภาพ

บทที่ 5 สรุปและวิจารณ์ กล่าวถึง ผลการทดลองที่ได้ ปัญหาที่พบในการทำงาน และแนวความคิดในการแก้ปัญหา

1.1 รายละเอียดพอสังเขปของโครงงาน

โครงงานหุ่นยนต์ตรวจจับวัตถุ เป็นการรับภาพวัตถุที่มีสีและรูปร่างที่เรากำหนดจากกล้องวีดิโอเข้าสู่เครื่องคอมพิวเตอร์ จากนั้นจะใช้คอมพิวเตอร์ทำการประมวลผลด้วยภาพถ่าย แล้วส่งสัญญาณที่ได้ไปใช้ในการขับมอเตอร์และควบคุมทิศทางการเคลื่อนที่ผ่านไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) ที่ได้ทำการโปรแกรมไว้แล้ว

สำหรับภาคการศึกษาที่ 1/2550 นี้ จะเริ่มทำจากการโปรแกรมไมโครคอนโทรลเลอร์ อาร์ม 7 (ARM7) โดยใช้ภาษาซี ในการควบคุมการขับสเต็ปปิ้งมอเตอร์และทิศทางการเคลื่อนที่ผ่านการกดปุ่มส่งสัญญาณจากคีย์บอร์ดคอมพิวเตอร์ แทนการรับภาพเข้ามา และทำการออกแบบและทำส่วนของโครงสร้างรถ

สำหรับภาคการศึกษาที่ 2/2550 จะทำการติดต่อกับกล้องวิดีโอในการรับสัญญาณภาพเพื่อให้คอมพิวเตอร์ทำการประมวลผลด้วยภาพ และนำผลที่ได้ไปใช้ในการควบคุมหุ่นยนต์ ที่ใช้ไมโครคอนโทรลเลอร์ อาร์ม7 ที่ได้เขียนโปรแกรมไว้ เพื่อใช้ในการควบคุมการขับสเต็ปมอเตอร์และทิศทางการเคลื่อนที่ของหุ่นยนต์ ที่ได้ออกแบบไว้ในภาคการศึกษาที่ 1

1.2 วัตถุประสงค์ของโครงงาน

โครงงานนี้จัดทำขึ้นเพื่อ

1. ศึกษาการประมวลผลภาพด้วยภาพจากคอมพิวเตอร์ ให้ทราบถึงแนวคิดในการออกแบบโปรแกรม

2. ศึกษาการทำงาน การใช้งาน และการควบคุมไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) ด้วยภาษาซีในการควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ (Stepping Motor)

3. ศึกษาการทำงานของสเต็ปปิ้งมอเตอร์ (Stepping Motor) เพื่อการใช้งานในอนาคต

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถที่จะประยุกต์และพัฒนาการใช้งานไมโครคอนโทรลเลอร์ อาร์ม7 (ARM7) ในงานควบคุมรูปแบบอื่นตามที่ต้องการได้

2. เข้าใจหลักการเบื้องต้นของ การประมวลผลด้วยภาพ (Digital Image Processing) เป็นพื้นฐานในการศึกษาและทำงานต่อไป

3. เข้าใจหลักการทำงานของ สเต็ปปิ้งมอเตอร์ (Stepping Motor) และสามารถนำมาใช้งานจริงได้

4. เพื่อเป็นพื้นฐานก่อนที่จะออกไปทำงานจริงต่อไป

บทที่ 2
ทฤษฎีและหลักการทำงานฮาร์ดแวร์

2.1 สถาปัตยกรรมซีพียู ARM7

สถาปัตยกรรมของ ARM7 เป็นซีพียูแบบRISCขนาด32บิตภายในมีบัสขนาด32บิตตัวเดียวที่ใช้สำหรับรับส่งข้อมูล และคำสั่ง ชุดคำสั่งจะมีขนาด 32 บิตคงที่ ในขณะที่ข้อมูลสามารถเลือกได้ว่าจะมีขนาด 8,16 หรือ 32 บิต โดยแสดงแกนกลาง (core) ของซีพียู ARM7 ได้ดังรูปที่ 1.1

โครงสร้างของ ARM7 จะเป็นแบบ load-and-store ในการประมาณผลข้อมูลใดๆต้องกระทำผ่านทางรีจิสเตอร์ เริ่มต้นด้วยการโหลดค่าจากหน่วยความจำเก็บในรีจิสเตอร์นำค่ามาประมวณผล เสร็จแล้วจะเขียนค่าเก็บในหน่วยความจำเดิม

รีจิสเตอร์ของARM7ที่ใช้งานได้สำหรับผู้ใช้มีทั้งหมด 16 ตัวคือ R0-R15 โดยทุกตัวมีขนาด 32 บิต โดย R0-R12 เป็นรีจิสเตอร์ทั่วไปที่ไม่ได้กำหนดหน้าที่การทำงานพิเศษ ส่วน R12 ทำหน้าที่เป็น stack pointer (SP) R14 ทำหน้าที่เป็น link register (LR) และ R15 ทำหน้าที่เป็น Program Counter (PC)

2.1.1 ARM7TDMI TDMI หมายถึงอะไร

ไมโครคอนโทรลเลอร์ตระกูล ARM7TDMI ภายในมีแกนกลางเป็นซีพียู ARM7 ที่เพิ่มความสามารถอีก 4 ประเภทที่นำตัวอักษรมาเขียนเป็นชื่อย่อที่ได้แก่

T : สนับสนุนคำสั่ง16บิตที่มีชื่อว่า Thumb instruction set

D : สนับสนุนการดีบัก (debug)

M : สนับสนุนการคูณแบบยาว (long multiplies)

I : มีโมดูล EmbeddedICE เพื่อสนับสนุนการดีบักภายในซีพียู

Thumb Mode (T)

ชุดคำสั่งของARM มีขนาด32บิตในซีพียู ARM7TDMI จะสนับสนุนชุดคำสั่งประเภทที่สองที่บีบอัดคำสั่งให้มีขนาด 16 บิต เรียกว่า Thumb instruction set เมื่อทำงานในโหมดนี้จะทำงานกับหน่วยความจำขนาด 16 บิตได้รวดเร็วขึ้น และบีบอัดโปรแกรมให้มีขนาดเล็กลงทำให้สามารถนำ ARM7TDMI ไปใช้งานสมองกลฝังตัวได้ดี

อย่างไรก็ตาม Thumb mode มีข้อจำกัดคือ เมื่อใช้กับงานประเภทเดียวกันมันจะใช้จำนวนคำสั่งมากกว่าโหมด 32 บิต ทำให้ทำงานช้ากว่า ดังนั้นสำหรับงานความเร็วสูงยังคงต้องใช้โหมด 32 บิตปกติ

ประการที่สองใน Thumb instruction set ไม่มีคำสั่งที่จะเป็นสำหรับจัดการกับเอ็กเซปชั่น (exception handling)

Long multiplies (M)

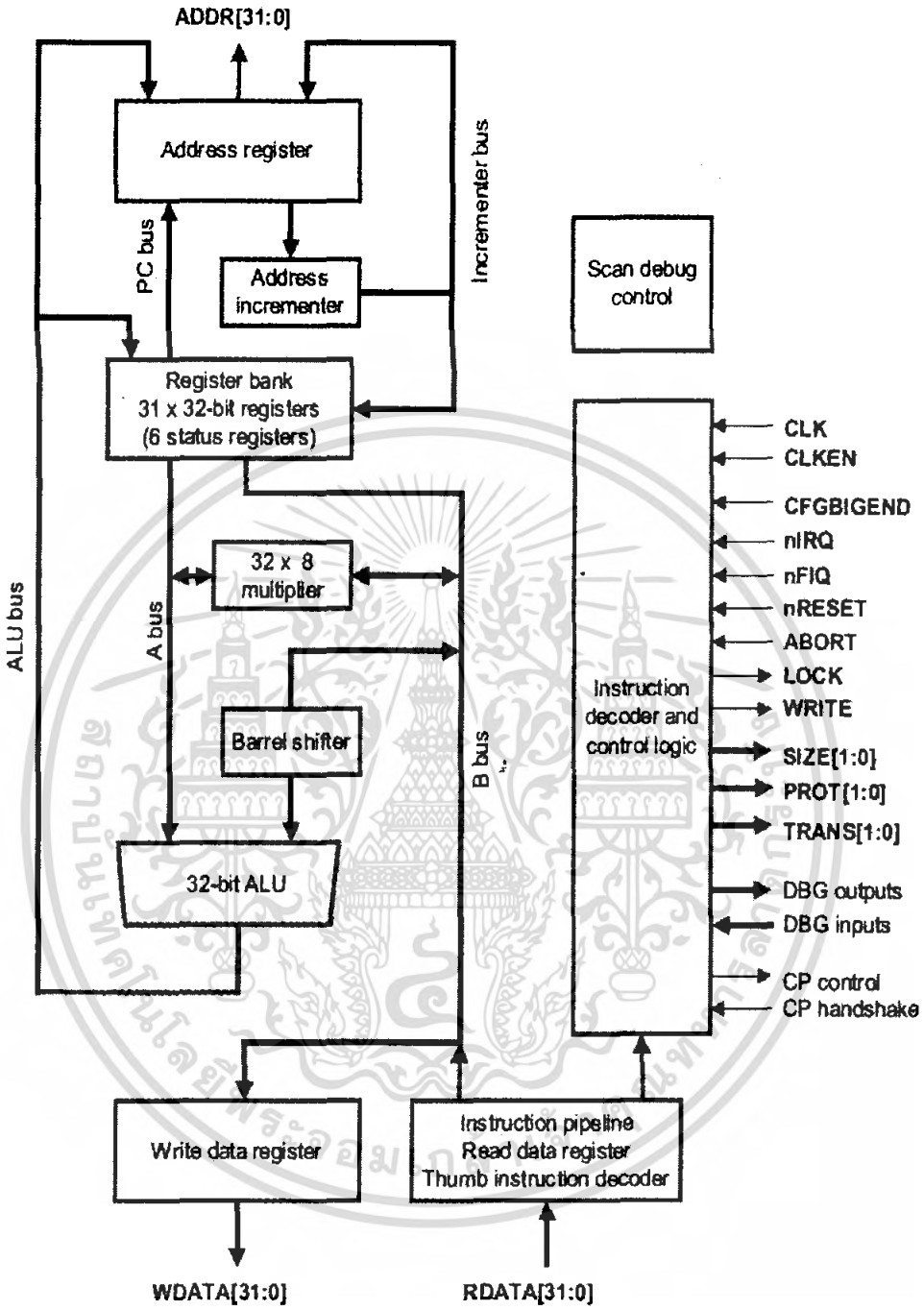
ในชุดคำสั่งของ ARM7TDMI มีการเพิ่มคำสั่งพิเศษอีก 4 คำสั่งที่สามารถคูณเลขขนาด 32 x 32 บิตได้ผลลัพธ์เป็น 64 บิต และการคูณสะสมค่า (multiplication accumulation:MAC) โดยสามารถคูณข้อมูลขนาด 32 x 32 บิตจำนวนหลายชุดได้ผลลัพธ์เป็น 64 บิตทำให้สามารถทำการคำนวณคณิตศาสตร์ที่ซับซ้อนได้โดยไม่จำเป็นต้องใช้ชิปประมวณผลสัญญาณดิจิตอล (Digital Signal Processor:DSP) ช่วย

Debugging (D)

ภายในมีส่วนขยายของฮาร์ดแวร์เพื่อรองรับการดีบักโปรแกรมได้ในขณะที่ทำงานซึ่งทำงานกับพอร์ต JTAG และ TAP contronller

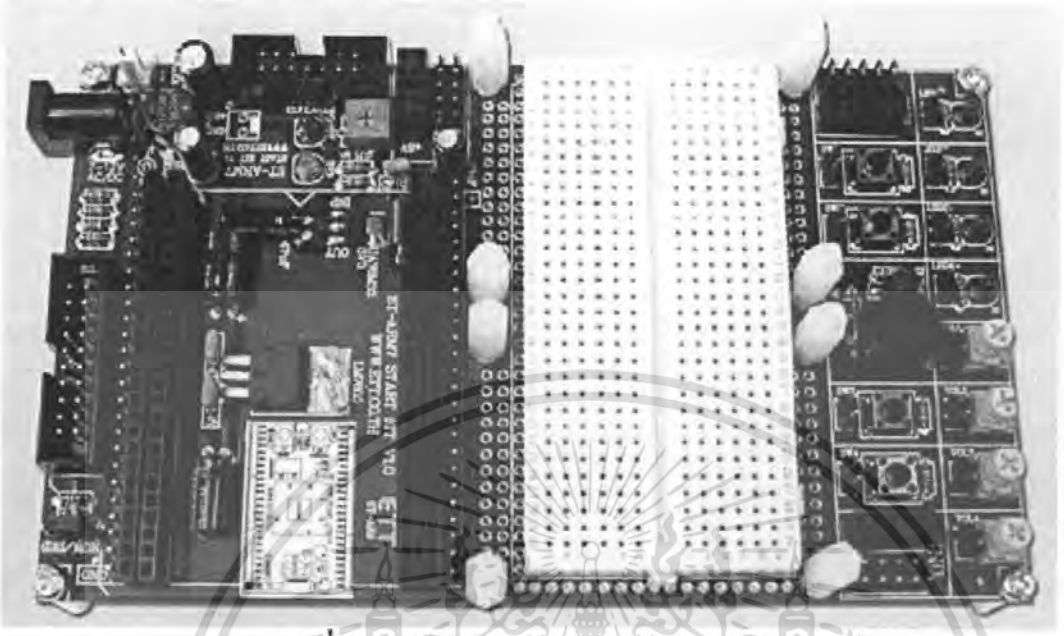
EmbeddedICE (I)

ส่วนของ EmbeddedICE ช่วยเพิ่มฟังก์ชันการทำงานการดีบักโปรแกรม และการภายในโมดูลนี้มีเบรกพอยต์ และวอทช์พอยต์รีจิสเตอร์ ทำให้สามารถพักการทำงานของโปรแกรมเพื่อดีบักการทำงาน เราสามารถควบคุมรีจิสเตอร์เหล่านี้ผ่านทาง JTAG test port และซอฟต์แวร์ดีบักกิ่งทูลที่ทำงานบนเครื่องคอมพิวเตอร์ เมื่อพบเบรกพอยต์หรือวอทช์พอยต์ ตัวซีพียูจะหยุดการทำงาน และเข้าสู่สถานะดีบัก เมื่ออยู่ในสถานะดีบักจะสามารถดูค่าของรีจิสเตอร์หรือค่าของหน่วยความจำทั้งแบบ Flash/EEPROM, SRAM และค่าของรีจิสเตอร์ที่จัดเทียบตำแหน่งกับหน่วยความจำ (Memory Mapped Registers)



รูปที่ 2.1 แกนกลางของซีพียู ARM7

2.2 บอร์ด ET-ARM7 START KIT V1 EXP



รูปที่ 2.2 บอร์ด ET-ARM7 START KIT V1 EXP

เป็นอีกบอร์ดหนึ่งของ อีทีที ในตระกูล ARM7 ซึ่งเป็น CPU แบบ 16/32 BIT ของบริษัท PHILIPS เบอร์ LPC2138 สามารถ DOWNLOAD โปรแกรมเข้าหน่วยความจำภายในตัวแบบ FLASH ผ่านทาง PORT RS232 ได้โดยตรง บนบอร์ด ออกแบบเป็นบอร์ด CONTROL ขนาดเล็ก ใช้งานอิสระ หรือต่อบน PROJECT BOARD ใช้ต่อทดลองวงจรต่างๆ

> ใช้ ARM เบอร์ LPC2138 16/32 BIT MCU 64 PIN LQFP TYPE
> หน่วยความจำโปรแกรมภายในตัว MCU แบบ FLASH 512KBYTE, RAM ภายใน 32KBYTE
> ใช้ X' TAL 19.6608 MHz โดยตัว MCU สามารถประมวลผลด้วยความเร็วสูงสุดถึง 58.9824 MHz
> รองรับการโปรแกรมแบบ IN-SYSTEM PROGRAMMING (ISP) ผ่านทาง ON-CHIP-BOOL-LOADER SOFTWARE UART 0 โดยต่อเข้ากับ PORT RS232 ของเครื่อง พีซี ได้โดยตรง
> 47 I/O PIN สามารถต่อกับระบบ I/O ที่เป็นระดับสัญญาณ 5V ได้
> ใช้กับ POWER SUPPLY 3.3VDC
> UART แบบ FULL-DUPLEX จำนวน 2 ช่อง คือ UART 0 มาตรฐาน 4 PIN ETT เป็นสัญญาณระดับ RS232 และ UART1 เป็นสัญญาณระดับ TTL
> SPI จำนวน 2 ช่อง, I2C จำนวน 2 ช่อง

> A TO D ขนาด 10 BIT จำนวน 8 ช่อง, D TO A ขนาด 10 BIT จำนวน 1 ช่อง

> TIMER 32 BIT 2 ช่อง, PWM 6 ช่อง, WATCHDOG TIMER, REAL TIME CLOCK ในตัว CPU พร้อม X' TAL 32.768 KHz และขั้วต่อ BATTERY

> บอร์ด ET-ARM STAMP LPC2138 วางตัวบนขั้ว PIN HEADER ด้านละ 25 PIN รวม 50 PIN ระยะห่าง 2.54 mm. สามารถนำไปใส่ลงบนบอร์ดทดลอง ET-ARM7 START KIT V1, V1 EXP หรือ ต่อกับบอร์ดทดลอง PROJECT BOARD ก็สามารถทำได้

> PCB SIZE 40X65 mm

ชุด ET-ARM STAMP LPC 2138 ประกอบด้วย

> บอร์ด ARM STAMP LPC 2138

> สาย DOWNLOAD ET-RS232 9 PIN

> แผ่น CD-ROM คู่มือและตัวโปรแกรมทำงานบน WINDOWS 98/ME/XP/2000

2.3 การกำหนดหน้าที่การทำงานของพอร์ต

หลังจากเกิดการรีเซ็ตไมโครคอนโทรลเลอร์ ARM7 วงจรภายในสั่งรีเซ็ตค่ารีจิสเตอร์ PINSEL ทุกตัวกำหนดค่าให้ Port0 และ Port1 ทุกขาเป็น GPIO และให้ทุกขาเป็นอินพุต

ขาแต่ละขาของ Port0 และ Port1 มีหน้าที่การทำงานได้หลายหน้าที่ โดยกำหนดการทำงานของ Port0 ที่รีจิสเตอร์ PINSEL0,PINSEL1 และกำหนดหน้าที่การทำงานของ Port1 ที่รีจิสเตอร์ PINSEL2 โดยรีจิสเตอร์แต่ละตัวจะมีแอดเดรสแสดงในตารางที่ 2.1

ตารางที่ 2.1 รีจิสเตอร์ PINSEL0, PINSEL1 และ PINSEL2

ชื่อ	ความหมาย	การติดต่อ	แอดเดรส
PINSEL0	Pin function select register 0 กำหนดการทำงานของ P0.0-P0.15	อ่าน/เขียน	0xE002 C000
PINSEL1	Pin function select register I กำหนดการทำงานของ P0.16-P0.31	อ่าน/เขียน	0xE002 C004
PINSEL2	Pin function select register 2 กำหนดการทำงานของ P1.0-P1.31	อ่าน/เขียน	0xE002 C014

ค่าแต่ละบิตของรีจิสเตอร์ PINSEL0 แสดงได้ในตารางที่ 2.2 จากตารางจะพบว่าแต่ละขาจะมีหน้าที่การทำงานได้ 3 หรือ 4 หน้าที่ ตัวอย่างเช่น ขา P0.1 จะมีหน้าที่การทำงนได้ถึง 4 หน้าที่ คือ GPIO0.1, RxD0 (UART0), PWM3 และ EINT0 ถ้าต้องการกำหนดค่าให้ขา P0.0-P0.15 มีการทำงานเป็นGPIO ทำได้โดยการเขียนค่า 0 ทุกบิตให้กับ PINSEL0 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

PINSEL0 = 0x00000000; // Set P0.0 – P0.15 to GPIO Function

ตารางที่2.2 ค่าประจำบิตของรีจิสเตอร์ PINSEL0

PINSEL0	ชื่อขา	การทำงานเมื่อมีค่า00	การทำงานเมื่อมีค่า01	การทำงานเมื่อมีค่า10	การทำงานเมื่อมีค่า11	ค่าหลังรีเซ็ต
1:0	P0.0	GPIO Port 0.0	TxDo (UARTO)	PWM1	Reserved	00
3:2	P0.1	GPIO Port 0.1	RxDo (UARTO)	PWM3	EINT0	00
5:4	P0.2	GPIO Port 0.2	SCL0 (I^2C)	Capture 0.0(TIMER0)	Reserved	00
7:6	P0.3	GPIO Port 0.3	SDA0 (I^2C)	Match 0.0 (TIMER0)	EINT1	00
9:8	P0.4	GPIO Port 0.4	SCK0 (SPI0)	Capture 0.1 (TIMER0)	AD0.6	00
11:10	P0.5	GPIO Port 0.5	MISO0 (SPI0)	Match 0.1 (TIMER0)	AD0.7	00
13:12	P0.6	GPIO Port 0.6	MOSI0 (SPI0)	Capture 0.2 (TIMER0)	AD1.0	00
15:14	P0.7	GPIO Port 0.7	SSEL0 (SPI0)	PMW2	EINT2	00
17:16	P0.8	GPIO Port 0.8	TxD1 (UART1)	PMW4	AD1.1	00
19:18	P0.9	GPIO Port 0.9	RxD1 (UART1)	PMW6	EINT3	00
21:20	P0.10	GPIO Port 0.10	RTS1 (UART1)	Capture 1.0 (TIMER1)	AD1.2	00
23:22	P0.11	GPIO Port 0.11	CTS1 (UART1)	Capture 1.1 (TIMER1)	SCL0 (I^2C1)	00
25:24	P0.12	GPIO Port 0.12	DSR1 (UART1)	Match 1.0(TIMER1)	AD1.3	00
27:26	P0.13	GPIO Port 0.13	DTR1 (UART1)	Match 1.1(TIMER1)	AD1.4	00
29:28	P0.14	GPIO Port 0.14	DCD1 (UART1)	EINT1	SDA1 (I^2C1)	00
31:30	P0.15	GPIO Port 0.15	RI1 (UART1)	EINT2	AD1.5	00

ค่าแต่ละบิตของรีจิสเตอร์ PINSEL1 แสดงได้ในตารางที่ 2.3 ตัวอย่างถ้าต้องการกำหนดค่าให้ขา P0.16 – P0.31 มีการทำงานเป็น GPIO ทำได้โดยการเขียนค่า 0 ทุกบิตให้กับ PINSEL1 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
PINSEL1 = 0x00000000;          // Set P0.16-P0.31 to GPIO Function
```

ตารางที่ 2.3 ค่าประจำบิตของรีจิสเตอร์ PINSEL1

PINSEL1	พอร์ต	การทำงานเมื่อมีค่า 00	การทำงานเมื่อมีค่า 01	การทำงานเมื่อมีค่า 10	การทำงานเมื่อมีค่า 11	ค่าหลังรีเซ็ต
1:0	P0.16	GPIO Port 0.16	EINT0	Match 0.2 (TIMER0)	Capture 0.2 (TIMER0)	00
3:2	P0.17	GPIO Port 0.17	Capture 1.2 (TIMER1)	SCK1 (SPI1)	Match 1.2 (TIMER1)	00
5:4	P0.18	GPIO Port 0.18	Capture 1.3 (TIMER1)	MISO1 (SPI1)	Match 1.3 (TIMER1)	00
7:6	P0.19	GPIO Port 0.19	Match 1.2 (TIMER1)	MOSI1 (SPI1)	Capture 1.2 (TIMER1)	00
9:8	P0.20	GPIO Port 0.20	Match 1.3 (TIMER1)	SSEL1 (SPI1)	EINT3	00
11:10	P0.21	GPIO Port 0.21	PWM5	AD1.6	Capture 1.3 (TIMER1)	00
13:12	P0.22	GPIO Port 0.22	AD1.7	Capture 0.0 (TIMER0)	Match 0.0 (TIMER0)	00
15:14	P0.23	GPIO Port 0.23	V_{BUS}	สงวนไว้	สงวนไว้	00
17:16	P0.24	สงวนไว้				00
19:18	P0.25	GPIO Port 0.25	AD0.4	AOUT	สงวนไว้	00
21:20	P0.26	สงวนไว้				00
23:22	P0.27	สงวนไว้				00
25:24	P0.28	GPIO Port 0.28	AD0.1	Capture 0.2 (TIMER0)	Match 0.2 (TIMER0)	00
27:26	P0.29	GPIO Port 0.29	AD0.2	Capture 0.3 (TIMER0)	Match 0.3 (TIMER0)	00
29:28	P0.30	GPIO Port 0.30	AD0.3	EINT3	Capture 0.0(TIMER0)	00
31:30	P0.31	GPO Port 0.31	UP_LED	CONNECT		00

ค่าแต่ละบิตของรีจิสเตอร์ PINSEL2 ที่ใช้กำหนดหน้าที่การทำงานของPort P1 แสดงได้ในตารางที่4.4 ตัวอย่างถ้าต้องการกำหนดค่าให้ขา P1.16 – P1.31 มีการทำงานเป็น GPIO ทำได้โดยการเขียนค่า 0 ทุกบิตให้กับ PINSEL1 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
PINSEL2 = 0x00000000:          // Set P1.16 – P1.31 to GPIO Function
```

ตารางที่ 2.4 ค่าประจำบิตของรีจิสเตอร์ PINSEL2

PINSEL1	ความหมาย	ค่าหลังรีเซ็ต
1:0	สงวนไว้	00
2	เป็น 0 ขาP1.31:26 จะใช้เป็น GPIO ถ้าเป็น 1 ขา P1.31:26 จะใช้เป็น Debug port.	P1.26/ $\overline{RTCK}$
3	เป็น 0 ขาP1.25:26 จะใช้เป็น GPIO ถ้าเป็น 1 ขา P1.25:26 จะใช้เป็น Trace port.	P1.20/ $\overline{TRACESYNC}$

2.4 การกำหนดค่าควบคุมการทำงานของพอร์ตอเนกประสงค์ (GPIO)

หลังจากที่กำหนดค่าให้พอร์ตแต่ละตัวมีการทำงานเป็น GPIO แล้ว การควบคุมการทำงานของ GPIO จะสั่งงานผ่านทางรีจิสเตอร์ 4 ตัวได้แก่ IOPIN, IOSET, IOCLR, IODIR โดยรีจิสเตอร์แต่ละตัวจะมีแอดเดรสแสดงในตารางที่ 2.5

ในไมโครคอนโทรลเลอร์ LPC2131/2/4/6/8 ได้เพิ่มความสามารถของพอร์ตอเนกประสงค์ให้ทำงานได้เร็วขึ้นซึ่งจะเรียกว่าเป็น Fast GPIO โดยได้กำหนดรีจิสเตอร์เพิ่มเติมเท่านั้น ถ้าเขียนเป็นภาษาซีจะสังเกตผลไม่ออก ดังนั้นในหนังสือเล่มนี้จะเน้นใช้งาน GPIO เป็นโหมดปกติ ซึ่งจะมีข้อดีตรงที่สามารถไปใช้งานกับไมโครคอนโทรลเลอร์ ตระกูล LPC2000 รุ่นอื่นได้

ตารางที่ 2.5 ชื่อ และแอดเดรสของรีจิสเตอร์ที่ใช้ควบคุม GPIO

ชื่อทั่วไป	ความหมาย	การติดต่อ	ชื่อ และ แอดเดรสของ Port0	ชื่อ และ แอดเดรสของ Port1
IOPIN	GPIO Port value register ใช้อ่านสถานะปัจจุบันของพอร์ต	อ่าน	0xE002 8000 IOPIN0	0xE002 8010 IOPIN1
IOSET	GPIO Port Output set register ใช้กำหนดค่าให้ขาของพอร์ตมีค่าเป็น1ถ้าบิตใดเป็น1ขาของพอร์ตที่ตรงกันจะมีค่าเป็น 1	อ่าน/เขียน	0xE002 8004 IOSET0	0xE002 8014 IOSET1

IODIR	GPIO Port Direction control register ใช้กำหนดว่าขาใดเป็นเอาต์พุต	อ่าน/เขียน	0xE002 8008 IODIR0	0xE002 8018 IODIR1
IOCLR	GPIO Port Output clear register ใช้กำหนดให้ขาของพอร์ตมีค่าเป็น0	เขียน	0xE002 800C IOCLR0	0xE002 801C IOCLR1

การสั่งงานของGPIOเริ่มต้นด้วยการกำหนดทิศทางของพอร์ตก่อนว่าจะให้เป็นอินพุตหรือเอาต์พุต โดยกำหนดค่าที่รีจิสเตอร์ IODIR โดยค่าบิตใดที่เป็น 0 คือให้ขาของบิตนั้นเป็นอินพุท ถ้าให้ค่าเป็น1ขาของบิตนั้นจะเป็นเอาต์พุต

ตัวอย่างเช่น ต้องการให้ขา P0.22, P0.20, P0.19 และ P0.16 เป็นเอาต์พุต โดยขาของ P0 ที่เหลือเป็นอินพุต จะต้องกำหนดค่าให้กับรีจิสเตอร์ IODIR0 แต่ละบิตดังนี้

บิตที่	31	30	29	28	27	26	25	24	23	**22**	21	**20**	**19**	18	17	**16**
ค่า	0	0	0	0	0	0	0	0	0	**1**	0	**1**	**1**	0	0	**1**

0 0 5 9

บิตที่	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0 0 0 0

ซึ่งสามารถเขียนเป็นส่วนของโปรแกรมภาษาซีได้ดังนี้

```
IODIR0 = 0x00590000;
```

หลังจากกำหนดให้พอร์ตเป็นเอาต์พุตแล้ว ถ้าต้องการสั่งให้พอร์ตที่เป็นเอาต์พุตนี้มีค่าเป็น 1 ต้องสั่งที่รีจิสเตอร์ IOSET ตัวอย่างเช่นต้องการสั่งให้ขา P0.16 และ P0.19 เป็น 1 จะต้องกำหนดค่าให้กับรีจิสเตอร์ IOSET0 ดังนี้

บิตที่	31	30	29	28	27	26	25	24	23	22	21	20	**19**	18	17	**16**
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**

- 0 0 0 9 -

บิตที่	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0 0 0 0

ซึ่งสามารถเขียนเป็นเป็นส่วนของโปรแกรมภาษาซีได้ดังนี้

IOSET = 0x00090000;

การเขียนค่าเป็น 0 ให้รีจิสเตอร์ IOSET จะไม่มีผลต่อขาของพอร์ตที่ตรงกับบิตนั้นขาของพอร์ตจะมีค่าคงเดิม

ถ้าต้องการสั่งให้พอร์ตที่เป็นเอาต์พุตนี้มีค่าเป็น0 ต้องสั่งที่รีจิสเตอร์ IOCLR ตัวอย่าง เช่นต้องการสั่งให้ขา P0.22 และ P0.20 เป็น 0 จะต้องกำหนดค่าให้กับรีจิสเตอร์ IOCLR0 ดังนี้

บิตที่	31	30	29	28	27	26	25	24	23	**22**	21	**20**	19	18	17	16
ค่า	0	0	0	0	0	0	0	0	0	**1**	0	**1**	0	0	0	0
	0					0				5				0		

บิตที่	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0					0				0				0		

ซึ่งสามารถเขียนเป็นส่วนของโปรแกรมภาษาซีได้ดังนี้

IOCLR0 = 0x00500000;

ข้อควรระวัง ในการสั่งให้ขาพอร์ตมีค่าเป็น 0 นี้ไม่ได้สั่งให้เขียนค่า 0 ให้กับรีจิสเตอร์ IOCLR ต้องเขียนค่าเป็น 1 เท่านั้น

ในการเขียนค่าให้กับพอร์ต จะต้องแยกข้อมูล ถ้าบิตใดเป็น 0 จะต้องเขียนสั่งที่รีจิสเตอร์ IOCLR ถ้าบิตใดเป็น 1 จะต้องเขียนสั่งที่รีจิสเตอร์ IOSET

2.5 กลไกการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ ARM7

ในไมโครคอนโทรลเลอร์ ARM7 มีโมดูล Vectored Interrupt Controller (VIC) เป็นตัวควบคุมกลไกของการอินเตอร์รัปต์ โดยสามารถรับอินเตอร์รัปต์จากอุปกรณ์ทั้งภายใน และภายนอกไมโครคอนโทรลเลอร์ ARM7 ได้ทั้งหมด 32 อินพุตตามที่แสดงในตารางที่ 5.1

VIC จะนำอินพุตจากการอินเตอร์รัปต์ทั้ง 32 แหล่งมาจัดแบ่งหมวดหมู่ได้เป็น 3 ประเภท คือ FIQ, vectored IRQ และ non-vectored IRQ ทำให้สามารถปรับลำดับความสำคัญของอินเตอร์รัปต์จากอุปกรณ์ประกอบต่างๆได้ตามที่ต้องการ

Fast Interrupt request (FIQ) จะลำดับความสำคัญสูงสุด โดยรีบตอบสนองเร็วสุด

Vectored IRQs จะมีลำดับความสำคัญอยู่กึ่งกลาง โดยสามารถนำอินเตอร์รัปต์แค่ 16 หรือ 32 แหล่งมาจัดให้เป็น Vectored IRQs ได้ 16 ตัว โดย Slot 0 จะมีความสำคัญสูงสุด Slot 15 มีความสำคัญต่ำสุด

Non-vectored IRQ มีความสำคัญต่ำสุด

ตารางที่ 2.6 แหล่งกำเนิดอินเตอร์รัปต์ทั้ง 32 ตัว

อุปกรณ์	แหล่งกำเนิดอินเตอร์รัปต์	VIC Channel #
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgCommTx	3
TIME0	Match 0-3 (MR0, MR1,MR2, MR3) Capture 0-3 (CR0, CR1, CR2, CR3)	4
TIME1	Match 0-3 (MR0, MR1,MR2, MR3) Capture 0-3 (CR0, CR1, CR2, CR3)	5
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)	7
PWM0	Match 0-6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8
I^2C0	SI (state change)	9
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	10
SPI1 (SSP)	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	11
PLL	PLL Lock (PLOCK)	12
RTC	Counter Indrement (RTCCIF) Alarm (RTCALF)	13
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15

System Control	External Interrupt 2 (EINT2)	16
System Control	External Interrupt 3 (EINT3)	17
ADC0	A/D Converter 0 end of conversion	18
I^2C1	SI (state change)	19
BOD	Brown our detect	20
ADC1	A/D Converter 1 end of conversion	21
USB	USB Interrupt, DMA Interrupt	22
	Reserved	23-31

รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์มีทั้งหมด 43 ตัว ดังแสดงในตารางที่ 2.7 โดยในหัวข้อนี้เป็นการเลือกเฉพาะรีจิสเตอร์ที่เกี่ยวกับการอินเตอร์รัปต์ในแบบ Vector IRQ ซึ่งก็คือ รีจิสเตอร์ VICVectAddr0-15, VICVectCntl0-15 และ VICIntEnable

ตารางที่ 2.7 รีจิสเตอร์ที่เกี่ยวกับการอินเตอร์รัปต์

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
VICIRQStatus	IRQ Status Request อ่านค่าสถานะว่าอนุญาตให้มี interrupt request จากตัวใด และถูกจัดเป็น IRQ	อ่าน	0	0xFFFF F000
VICFIQStatus	FIQ Status Request อ่านค่าสถานะว่าอนุญาตให้มี interrupt request จากตัวใด และถูกจัดเป็น FIQ	อ่าน	0	0xFFFF F004
VICRawIntr	Raw Interrupt Status Register ใช้อ่านสถานะของอินเตอร์รัปต์จากทั้ง 32 แหล่งหรือจากซอฟต์แวร์โดยไม่สนว่าถูกเปิดใช้หรือถูกจัดประเภทหรือไม่	อ่าน	0	0xFFFF F008
VICIntSelect	Interrupt Select Register ใช้ในการระบุว่าอินเตอร์รัปต์ทั้ง 32 แหล่งแต่ละตัวเป็น FIQ หรือ IRQ	อ่าน/เขียน	0	0xFFFF F00C
VICIntEnable	Interrupt Enable Register ใช้ในการระบุว่าอินเตอร์รัปต์ทั้ง 32 แหล่งแต่ละตัวเป็น FIQ หรือ IRQ	อ่าน/เขียน	0	0xFFFF F010

VICIntEnClr	Interrupt Enable Clear Register ใช้ล้างค่าบิตหนึ่งบิตหรือมากกว่าหนึ่งบิตของรีจิสเตอร์ Interrupt Enable Register	เขียน	0	0xFFFF F014
VICSoftInt	Software Interrupt Register ค่าของรีจิสเตอร์นี้จะถูกนำไป OR กับอินเตอร์รัปต์จากอุปกรณ์ต่างๆ ทั้ง32 แหล่ง	อ่าน/เขียน	0	0xFFFF F018
VICSoftIntClear	Software Interrupt Clear Register ใช้ล้างค่าบิตหนึ่งบิตหรือมากกว่าหนึ่งบิตของรีจิสเตอร์ Software Interrupt Register	เขียน	0	0xFFFF F01C
VICProtection	Protection Enable Register ใช้จำกัดการเข้าถึง VIC register	อ่าน/เขียน	0	0xFFFF F020
VICVectAddr	Vector Address Register เมื่อเกิดการอินเตอร์รัปต์แบบ IRQ ตัว IRQ service routine จะอ่านค่าจากรีจิสเตอร์เพื่อกระโดดไปยังแอดเดรสที่ระบุไว้	อ่าน/เขียน	0	0xFFFF F030
VICDefVectAddr	Default Vector Address Register ใช้เก็บค่าแอดเดรสของ interrupt service routine (ISR) สำหรับ non-vectored IRQ	อ่าน/เขียน	0	0xFFFF F034
VICVectAddr0	Vector address 0 register รีจิสเตอร์ VICVectAddr0-15 จะเก็บค่าแอดเดรสของ Interrupt Service Routine (ISRs) สำหรับ IRQ Slot ทั้ง16 ตัว	อ่าน/เขียน	0	0xFFFF F100
VICVectAddr1	Vector address 1 register	อ่าน/เขียน	0	0xFFFF F104
VICVectAddr2	Vector address 2 register	อ่าน/เขียน	0	0xFFFF F108
VICVectAddr3	Vector address 3 register	อ่าน/เขียน	0	0xFFFF F10C

VICVectAddr4	Vector address 4 register	อ่าน/เขียน	0	0xFFFF F110
VICVectAddr5	Vector address 5 register	อ่าน/เขียน	0	0xFFFF F114
VICVectAddr6	Vector address 6 register	อ่าน/เขียน	0	0xFFFF F118
VICVectAddr7	Vector address 7 register	อ่าน/เขียน	0	0xFFFF F11C
VICVectAddr8	Vector address 8 register	อ่าน/เขียน	0	0xFFFF F120
VICVectAddr9	Vector address 9 register	อ่าน/เขียน	0	0xFFFF F124
VICVectAddr10	Vector address 10 register	อ่าน/เขียน	0	0xFFFF F128
VICVectAddr11	Vector address 11 register	อ่าน/เขียน	0	0xFFFF F12C
VICVectAddr12	Vector address 12 register	อ่าน/เขียน	0	0xFFFF F130
VICVectAddr13	Vector address 13 register	อ่าน/เขียน	0	0xFFFF F134
VICVectAddr14	Vector address 14 register	อ่าน/เขียน	0	0xFFFF F138
VICVectAddr15	Vector address 15 register	อ่าน/เขียน	0	0xFFFF F13C
VICVectCntl0	Vector control 0 register รีจิสเตอร์ Vector Control Register0-15 แต่ละตัว จะควบคุม IRQ slot แต่ละตัว โดย Slot 0 มีความสำคัญสูงสุด และ Slot 15 มีความสำคัญต่ำสุด	อ่าน/เขียน	0	0xFFFF F200
VICVectCntl1	Vector control 1 register	อ่าน/เขียน	0	0xFFFF F204
VICVectCntl2	Vector control 2 register	อ่าน/เขียน	0	0xFFFF F208
VICVectCntl3	Vector control 3 register	อ่าน/เขียน	0	0xFFFF F20C
VICVectCntl4	Vector control 4 register	อ่าน/เขียน	0	0xFFFF F210
VICVectCntl5	Vector control 5 register	อ่าน/เขียน	0	0xFFFF F214
VICVectCntl6	Vector control 6 register	อ่าน/เขียน	0	0xFFFF F218
VICVectCntl7	Vector control 7 register	อ่าน/เขียน	0	0xFFFF F21C
VICVectCntl8	Vector control 8 register	อ่าน/เขียน	0	0xFFFF F220
VICVectCntl9	Vector control 9 register	อ่าน/เขียน	0	0xFFFF F224
VICVectCntl10	Vector control 10 register	อ่าน/เขียน	0	0xFFFF F228
VICVectCntl11	Vector control 11 register	อ่าน/เขียน	0	0xFFFF F22C
VICVectCntl12	Vector control 12 register	อ่าน/เขียน	0	0xFFFF F230
VICVectCntl13	Vector control 13 register	อ่าน/เขียน	0	0xFFFF F234

VICVectCntl14	Vector control 14 register	อ่าน เขียน	0	0xFFFF F238
VICVectCntl15	Vector control 15 register	อ่าน เขียน	0	0xFFFF F23C

รีจิสเตอร์ VICVectAddr0-15 เป็นรีจิสเตอร์ที่เก็บค่าแอดเดรสของโปรแกรมที่ตอบสนองต่ออินเตอร์รัปต์ โดยVICVectAddr0 จะเป็นของอินเตอรรัปต์ Slot0 ที่มีความสำคัญสูงสุดตามลำดับไปจนถึง VICVectAddr15 จะเป็น Slot15 ที่มีความสำคัญต่ำสุด

VICVectCntl0-15 ใช้ในการควบคุมว่าที่ Slot หมายเลขที่ตรงกับ VICVectCntl นี้จะรับการอินเตอร์รัปต์จากแหล่งใด จากทั้งหมด 32 ที่ตามตารางที่ 2.6 โดยกำหนดค่าในบิตที่4:0โดยบิตที่5 ถ้าเป็น1 หมายถึงอนุญาตให้อินเตอร์รัปต์จากอุปกรณ์ที่กำหนดในSlot นี้

VICIntEnable ใช้เปิดอินเตอร์รัปต์จากแหล่งต่างๆ ทั้ง 32 แหล่งตามตารางที่ 2.6 โดยบิตใดมีค่าเป็น 1 หมายถึงอนุญาตให้อินเตอร์รัปต์ได้ถ้าบิตใดเป็น 0 ไม่อนุญาตให้อุปกรณ์นั้นๆ อินเตอร์รัปต์

2.6 การทดลองต่อกับพอร์ตอนุกรม UART0

ในไมโครคอนโทรลเลอร์ตระกูล LPC2000 มีวงจรพอร์ตสื่อสารอนุกรมอเนกประสงค์ (Universal Asynchronous Receiver Transmitter: UART) 2 วงจรที่เหมือนกันคือ UART0 และ UART1 โดย UART1 มีขาเพิ่มเติมสำหรับการติดต่อกับโมเด็ม ภายใน UART ทั้งสองตัวทีวงจรกำเนิดบอดเรตอัตโนมัติ (Baud rate generator) สำหรับสร้างสัญญาณนาฬิกาควบคุมจังหวะในการรับส่งข้อมูล และมีบัฟเฟอร์แบบ FIFO (First In First Out) ขนาด 14 ไบต์สำหรับรับหรือส่งข้อมูล

2.6.1 การกำหนดค่าเริ่มต้นสำหรับ UART0

ใน UART0 มีขาสัญญาณแค่สองขาคือขาสัญญาณ TxD0 ไว้สำหรับส่งข้อมูลอยู่ที่ขา P0.0 และขา RxD0 ไว้สำหรับรับข้อมูลอยู่ที่ขา P0.1

การใช้งาน UART0 เริ่มต้นด้วยการเขียนค่ายังรีจิสเตอร์ PINSEL0 เพื่อกำหนดให้ขา P0.0 และ P0.1 มีการทำงานเป็นUART0 โดยต้องกำหนดบิตที่3-0 ของ PINSEL0 ให้มีค่าเป็น0101 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
PINSEL0 |= 0x00000005;
```

รีจิสเตอร์ที่เกี่ยวข้องกับ UART0 มีทั้งหมด 10 ตัวโดยแต่ละตัวมีขนาด 8 บิต ดังแสดงในตารางที่ 2.13

หลังจากที่กำหนดให้ขา P0.0 และ P0.1 ทำงานเป็น UART0 แล้วถัดมาเป็นการกำหนดรูปแบบการติดต่อเช่น ติดต่อแบบ 8 บิต ใช้การตรวจสอบบิตผิดพลาดแบบใด เช่น even parity จำนวนของ Stop bit โดยการกำหนดค่าผ่านทางรีจิสเตอร์ UART Line Control Register : LCR ซึ่งมีรายละเอียดของรีจิสเตอร์ดังแสดงในตารางที่ 2.13

ตัวอย่างเช่นกำหนดรูปแบบการติดต่อเป็นแบบ 8 บิต ไม่ใช้พาริตี้บิต Stop bit 1 บิต จะต้องเขียนค่า 0x83 ให้กับรีจิสเตอร์ U0LCR ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

U0LCR = 0x83;

ตารางที่ 2.8 รีจิสเตอร์ที่เกี่ยวข้องกับ UART0

ชื่อ	ความหมาย	บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
U0RBR	Receiver Buffer Register	MSB	อ่านข้อมูล						LSB	อ่าน	ไม่กำหนด	0xE000C000 DLAB = 0
U0THR	Transmit Holding Register	MSB	เขียนข้อมูล						LSB	เขียน	ไม่กำหนด	0xE000C000 DLAB = 0
U0IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	อ่าน/เขียน	0	0xE000C004 DLAB = 0
U0IIR	Interrupt ID Register	FIFOs Enable		0	0	IIR3	IIR2	IIR1	IIR0	อ่าน	0x01	0xE000C008
U0FCR	FIFO Control Register	Rx Trigger		สงวนไว้		-	TxFIFO Reset	RxFIFO Reset	FIFO Enable	เขียน	0	0xE000C008
U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bit	Word Length Select		อ่าน/เขียน	0	0xE000C00C
U0LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	อ่า	0x60	0xE000C014
U0SCR	Scratch Pad Register	MSB							LSB	อ่า เขี	0	0xE000C01C
U0DLL	Divisor Latch LSB	MSB							LSB	อ่า เขี	0x01	0xE000C000 DLAB = 1
U0DLM	Divisor Latch MSB	MSB							LSB	อ่า เขี	0x01	0xE000C004 DLAB = 1

ตารางที่ 2.9 ค่าประจำบิตของรีจิสเตอร์ UART0 Line Control Register (U0LCR)

ค่าบิตของ U0LCR	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
1 : 0	World Length Select	00 ตัวอักษรขนาด 5 บิต 01 ตัวอักษรขนาด 6 บิต 10 ตัวอักษรขนาด 7 บิต 11 ตัวอักษรขนาด 8 บิต	0
2	Stop Bit Select	0 Stop bit 1 บิต 1 Stop bit 2 บิต (1.5 บิต ถ้า U0LCR[1:0]=00)	0
3	Parity Enable	0 ยกเลิกการเพิ่มพาริตี้บิตและการตรวจพาริตี้ 1 ใช้งานการเพิ่มพาริตี้บิตและการตรวจพาริตี้	0
5 : 4	Parity Select	00 Odd parity 01 Even parity 10 ให้พาริตี้บิตมีค่าเป็น 1 ตลอดเวลา 11 ให้พาริตี้บิตมีค่าเป็น 0 ตลอดเวลา	0
6	Break Control	0 ยกเลิกการหยุดการส่ง 1 อนุญาตให้หยุดการส่งได้ ขาเอาต์พุตขิง UART0 TxD จะมีค่าลอจิก 0	0
7	Divisor Latch Access Bit	0 ไม่อนุญาตให้แก้ไขค่าตัวหาร Divisor Latch 1 อนุญาตให้แก้ไขค่าตัวหาร Divisor Latch ได้	0

ในรีจิสเตอร์ LCR มีบิตที่เรียกว่า DLAB (Divisor Latch Access bit) ถ้าเราต้องการปรับค่าของวงจรกำเนิดบอดเรตจะต้องเซ็ตบิตนี้ให้เป็น 1

ค่าของBaud rate generator เป็นค่าของตัวหารขนาด 16 บิต เพื่อนำไปใช้หารค่าของ PCLK เพื่อให้ได้ความถี่ที่สูงกว่าค่าความเร็วบอดเรต 16 เท่า ทำให้ได้สมการค่าของตัวหารดังนี้

Divisor = PCLK/ (16*Baud)

ในกรณีของแผงวงจร JX-2148 และ CP-JR ARM7 USB-LPC2148 EXP ใช้คริสตัลความถี่ 12.000MHz สั่ง PLL คูณ 5 จะได้ CCLK = 60.0MHz และสั่ง VPBDIV = 2 จะได้ PCLK = 30.0 MHz ด้วย ถ้าต้องการอัตราบอดที่ 9600 bps ตัวหารจะมีค่าเท่ากับ

Divisor = 30,000,000 / (16*Divisor) = 30,000,000/(16*195)

ปัดเศษลงได้ = 195 หรือ 0xC3

ทดลองนำค่าที่ได้ไปคำนวณหาอัตราบอดจะได้

Baud = PCLK / (16*Divisor) = 30,000,000/(16*195)

= 9615 bps

ซึ่งผิดพลาดไป 0.156 % สามารถใช้งานได้ เนื่องมาตรฐานของการสื่อสารแบบอนุกรมสามารถรับกับอัตราบอดที่ผิดพลาดได้ถึง 5 %

เราต้องนำค่าตัวหารนี้ไปเก็บลงในรีจิสเตอร์ขอนาด 8 บิตสองตัวคือ Divisor Latch MSB (DLM) และ Divisor Latch LSB (DLL) ในขณะที่เขียนค่าเก็บในรีจิสเตอร์ DLM และ DLL นี้ค่าบิต DLAB ต้องมีค่าเป็น 1 เมื่อเขียนเสร็จแล้วต้องรีเซ็ตค่าบิตนี้ให้กลับเป็น 0 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
U0DLM = 0x00;
U0DLL  = 0Xc3;
U0LCR &= 0x7F;
```

หรือจะนำไปเขียนเป็นฟังก์ชันสำหรับกำหนดการทำงานของ UART0 ได้ดังนี้

```
Void uart0_init (unsigned int baudrate)
{
    unsigned short u0dl;
    u0d1 = 30000000/(16*baudrate);
    PINSEL | = 0x00000005;
    U0LCR = 0x83;
    U0DLL = u0dl & 0xFF;
    U0DLM = (u0dl>>8);
    U0LCR &= 0x7F;
}
```

ในการเรียกใช้ฟังก์ชัน uart0_init() จะต้องส่งค่าอัตราบอดที่ต้องการให้ฟังก์ชันตัวอย่างเช่นต้องการอัตราบอดที่ 9600 bps จะต้องเรียกใช้ฟังก์ชันดังนี้ uart_init(9600);

เมื่อกำหนดการทำงานให้กับ UART แล้ว จะสามารถรับส่งค่าผ่านพอร์ตอนุกรมได้ในการส่งข้อมูลต้องเขียนข้อมูลไปยังรีจิสเตอร์ Transmit Holding Register (THR) ถ้าต้องการอ่านข้อมูลที่รับพอร์ตอนุกรมต้องอ่านค่าจากรีจิสเตอร์ Receiver Buffer Register (RBR) จากตารางที่ 7.1 จะพบว่ามีค่าอยู่ที่ตำแหน่งเดียวกัน แสดงว่าการเขียนค่าให้กับ THR เป็นการเขียนค่าลงในบัฟเฟอร์แบบ FIFO ของ UART0 การอ่านค่าจากรีจิสเตอร์ RBR เป็นการอ่านค่าลงในบัฟเฟอร์แบบ FIFO

ก่อนที่จะอ่านหรือเขียนค่าลงในรีจิสเตอร์ THR หรือ RBR จะต้องอ่านค่าสถานะของ UART ก่อนว่ามีการผิดพลาดหรือไม่ โดยอ่านค่าสถานะที่รีจิสเตอร์ Line Statuos Register (LSR) ก่อน โดยค่าประจำบิตของรีจิสเตอร์แสดงได้ในตารางที่ 2.15

ตารางที่ 2.10 แสดงค่าประจำบิตของ UART0 Line Statuos Register (U0LSR)

ค่าบิตของ U0LSR	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
0	Receiver Data Ready (RDR)	0 : U0RBR ว่าง 1 : U0RBR มีข้อมูลที่ถูกต้อง U0LSR0 เป็น 1 เมื่อ U0RBR มีค่าตัวอักษรที่ยังไม่อ่าน และล้างค่าเมื่อ UART0 RBR FIFO ว่าง	0
1	Overrun Error (OE)	0 : ไม่มี Overrun error 1 : มีOverrun error เกิดขึ้น Overrun error เกิดขึ้นเมื่อ UART0 RSR ได้รับจะสูญหายไป เมื่ออ่านค่าของU0LSR จะล้างค่าของบิตนี้	0
2	Parity Error (PE)	0 : ไม่มี Parity error 1 : มี Parity error เกิดขึ้น ใช้ตรวจสอบว่าข้อมูลที่รับมาใน UART0 RBR FIFO มีการผิดพลาดที่พาริตี้บิตหรือไม่ เมื่ออ่านค่าของU0LSR จะล้างค่าของบิตนี้	0
3	Framing Error (FE)	0 : ไม่มี Framing error 1 : มี Framing error เกิดขึ้น เมื่อ Stop bit ของข้อมูลที่รับมามีค่าเป็น 0 แสดงว่าเกิดการผิดพลาดที่เฟรมข้อมูล ในขณะที่เกิดการผิดพลาดที่เฟรมข้อมูลนี้ UART0จะพยายามรับข้อมูล โดยนำ Stop bit ที่ผิดพลาดนี้มาใช้เป็น Start bit ข้อมูลตัวถัดไป ซึ่งอาจจะอ่านข้อมูลได้ถูกต้องหรือไม่ถูกต้อง	0
4	Break Interrupt (BI)	0 : ไม่ใช้ Break Interrupt 1 : ใช้ Break Interrupt เมื่อ RxD0 ได้รับข้อมูลที่เป็น 0 ทุกบิต (start,data,parity,stop) จะเกิด Break Interrupt ภาครับจะหยุดทำงาน จนกว่าจะได้รับ	0

		ข้อมูลที่เป็น 1 ทุกบิตอีกครั้ง	
5	Transmitter Holding Register Empty (THRE)	0 : U0THR มีข้อมูลที่ถูกต้อง 1 : U0THR ว่าง บิต THRE จะถูกเซ็ตค่าเป็น 1 ทันทีที่พบว่า UART0 THR ว่าง และถูกล้างค่าทันทีที่มีการเขียนค่าไปยังU0THR	1
6	Transmitter Empty (TEMT)	0 : U0THR และ/หรือ U0TSR มีข้อมูลที่ถูกต้อง 1 : U0THR และ U0TSR ว่าง บิตนี้จะถูกล้างค่าเมื่อ U0THR หรือ U0TSR มีข้อมูลที่ถูกต้อง	1
7	Error in Rx FIFO (RXFE)	0 : ข้อมูลที่เก็บใน UART0 RBR FIFO ไม่ผิดพลาด 1 : ข้อมูลที่เก็บใน UART0 RBR FIFO อย่างน้อยหนึ่งตัวมีการผิดพลาด	0

ก่อนที่จะเขียนข้อมูลให้ UART ต้องตรวจสอบดูที่บิต Transmitter Empty (TEMT) ของรีจิสเตอร์ LSR ก่อนว่าบัฟเฟอร์สำหรับส่งค่าว่างหรือไม่ ถ้าว่างจะได้ค่า 1 จึงส่งข้อมูลได้ก่อนที่จะอ่านข้อมูลจาก UART ต้องตรวจสอบบิต UART ต้องตรวจสอบบิต Receiver Data Ready (RDR) ก่อน ถ้ามีข้อมูลพร้อมแล้วบิตนี้จะมีค่าเป็น 1 จึงอ่านค่าจาก UART ได้

เราสามารถนำมาเขียนเป็นฟังก์ชัน putchar(); สำหรับเขียนข้อมูลจำนวน 1 ไบต์ให้กับ UART และเขียนเป็นฟังก์ชัน getchar() สำหรับอ่านค่าจาก UART ได้ดังต่อไปนี้

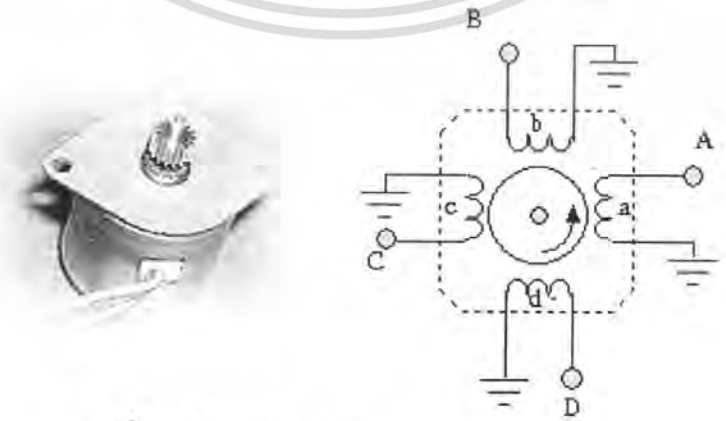
```
Int putchar (int ch) /* Write character to Serial Port */
{
        If      (ch == '\n')
        {
                While   (!(U0LSR & 0x20));
                U0THR = CR;   /* output CR */
        }
        while    (! (U0LSR & 0x20));
        return   (U0THR = ch);
}
Int getchar (void) /* Read character from Serial Port */
```

```
{
        While (! (U0LSR & 0x01));
        Return (U0RBR);
}
```

สำหรับการเขียนข้อมูลออกพอร์ตอนุกรม ในโปรแกรม Keil uVision3 ได้จัดเตรียมฟังก์ชัน printf(); ไว้ให้แล้ว และถ้าต้องการรับข้อมูลจากพอร์ตอนุกรม สามารถใช้ฟังก์ชัน scanf(); โดยต้องสั่ง #inclode <stdio.h> ที่ส่วนพรีโปรเซสเซอร์ของโปรแกรมภาษาซี เมื่อศึกษาตัวฟังก์ชัน printf() และ scanf() จะพบว่าตัวฟังก์ชันจะเรียกใช้ฟังก์ชัน putchar() และ getchar() ตามที่เขียนไว้ในโปรแกรมที่7.1 ซึ่งผู้ใช้สามารถนำไปดัดแปลงได้เช่น ให้ฟังก์ชัน printf() เป็นการเขียนค่าออกจอแสดงผล LCD ให้แก้ไขที่ตัวฟังก์ชัน putchar() เท่านั้น

2.7 สเตปปิ้งมอเตอร์ (Stepping Motor)

Stepping Motor เป็นมอเตอร์ไฟฟ้าชนิดหนึ่ง ซึ่งมี input เป็นกลุ่ม ของ Binary Voltage และ Output การเคลื่อนที่ในเชิงมุม (หมุน) แกนหมุน (Shaft) เป็น Step โดย Resolution ของ Steping Motor อยู่ในช่วงตั้งแต่ 0.1- 30 องศาซึ่งขึ้นอยู่กับโครงสร้างของ Stepping Motor หรือบอกเป็นจำนวน Step ต่อ 1 รอบ Stepping Motor สามารถควบคุมตำแหน่งการหมุนได้ ซึ่งจะมีความละเอียดของมุมในการหมุนและ Step ที่แตกต่างกันออกไป ทั้งนี้ก็ขึ้นอยู่กับ ชนิดของมอเตอร์และลักษณะการส่งสัญญาณไปควบคุมมอเตอร์ ทำให้ Stepping Motor มีความยืดหยุ่นในการนำมาใช้งาน ทำให้สามารถนำมาประยุกต์ใช้งานได้หลายรูปแบบ ไม่ว่าจะเป็นหุ่นยนต์หรือแขนกลต่างๆ เนื่องจากมีความแม่นยำในการควบคุมตำแหน่งสูง



รูปที่ 2.3 โครงสร้างของ Stepping Motor

โครงสร้างของ Stepping Motor มีลักษณะดังรูป ซึ่งประกอบด้วย ขดลวด stator 4 ขด ล้อมรอบแกนหมุน

หลักการทำงาน คือ เมื่อจ่ายกระแสไฟฟ้าให้กับขดลวด Stator Coil a, b, c, d ไม่พร้อมกัน นั่นคือ ถ้าเราจ่ายกระแสให้ a ก่อน โดยไม่จ่ายให้ขดอื่น แล้วตามด้วย b, c และ d เรียงตามลำดับ จะทำให้เกิดสนามแม่เหล็ก หมุนวนในลักษณะทวนเข็มนาฬิกา ซึ่งส่วนของ Rotor ที่เป็นแม่เหล็กถาวร ก็จะหมุนตามสนามแม่เหล็กไปด้วย คือ ทวนเข็มนาฬิกา

ในทำนองเดียวกันถ้าเราจ่ายกระแสให้ขด a, d, c, b, a ..... ก็จะให้ สนามแม่เหล็กหมุนในทิศทางตามเข็มนาฬิกา ซึ่งส่งผลให้ Rotor หมุนตามเข็มนาฬิกา ด้วย การกำหนดความเร็วของ Stepping Motor ทำได้โดยการเปลี่ยนแปลง ความเร็วของ การเปลี่ยนการจ่ายกระแสจากขดลวดขดหนึ่งไปยังอีกขดหนึ่งให้เร็วขึ้น

การกำหนดความเร็วของ Stepping Motor ทำได้โดยการเปลี่ยนแปลง ความเร็วของ การเปลี่ยนการจ่ายกระแสจากขดลวดขดหนึ่งไปยังอีกขดหนึ่งให้เร็วขึ้น

2.7.1 ชนิดของสเต็ปมอเตอร์

สเต็ปมอเตอร์ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

1. แบบแม่เหล็กถาวร(PERMANENT MAGNET_PM)

สเต็ปมอเตอร์แบบ PM จะมีสเตเตอร์ (STATOR) ที่พันขดลวดไว้หลายๆ โพล โดยมีโรเตอร์ (ROTOR) เป็นรูปทรง กระบอกฟันเลื่อย และโรเตอร์ทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรงให้กับขดสเตเตอร์ จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ ทำให้มอเตอร์หมุนมอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด

2. แบบแปรค่ารีลักแตนซ์ (VARIABLE RELUCTANCE- VR)

สเต็ปมอเตอร์แบบVR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้โรเตอร์ทำจากสารเฟอร์โรแมกเนติก กำลังอ่อน มีลักษณะเป็นฟันเลื่อย รูปทรงกระบอกโดยจะมีความสัมพันธ์โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะไปหมุนโรเตอร์ ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆ จุดดังนั้นเมื่อป้อนไฟเข้าขดลวดต่างๆ ในมอเตอร์แตกต่างขดกันไป ก็ทำให้มอเตอร์ หมุนไปตำแหน่งต่างๆ กันโรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

3. แบบผสม(HYBRID-H)

สเต็ปมอเตอร์แบบ H จะเป็นลูกผสมของ VR กับ PM โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR โรเตอร์มีหมวกหุ้ม ปลายซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดีทำให้ได้มุม การหมุนและครั้งน้อยและแม่นยำ ข้อดีก็คือ ให้แรงบิดสูงและมีขนาดกระทัดรัด และให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ

2.7.2 ข้อดีของสเตปปิ้งมอเตอร์เมื่อเปรียบกับมอเตอร์กระแสตรง (DC MOTOR)

1. การควบคุมไม่ต้องอาศัยตัวตรวจจับการหมุน
2. ไม่ต้องใช้แปรงถ่าน ดังนั้นจึงทำให้ไม่มีส่วนที่จะต้องสึกหรอ และปัญหาของการสปาร์ค (ที่เกิดจาก หน้าสัมผัสของแปรงถ่านแหวนตัวนำในโรเตอร์) ที่ทำให้เกิดสัญญาณรบกวน
3. การควบคุมโดยทางวงจรดิจิตอลหรือไมโครโพรเซสเซอร์ ทำได้ง่าย และสะดวก

2.7.3 การควบคุม Stepping Motor แบบ 4 เฟส

ในการควบคุมการทำงานของ Stepping Motor สามารถแบ่งได้ออกเป็น 3 รูปแบบ คือ

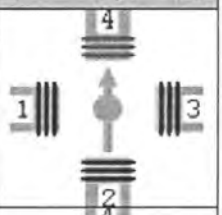
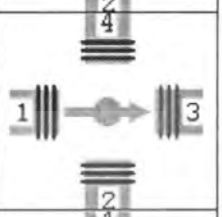
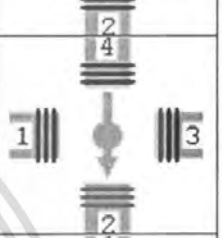
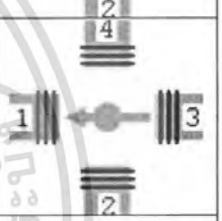
> ควบคุมแบบ Full Step 1 เฟส หรือแบบเวฟ (wave)

> ควบคุมแบบ Full Step 2 เฟส หรือแบบ 2 เฟส

> ควบคุมแบบ Half Step หรือแบบครึ่งสเต็ป

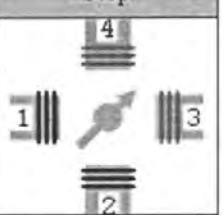
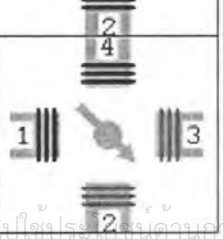
ควบคุมแบบ Full Step 1 เฟส หรือแบบเวฟ (wave)

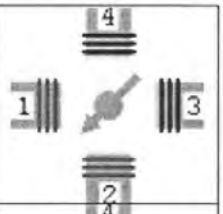
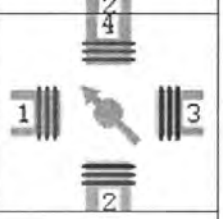
ในการควบคุมการหมุนของ Stepping แบบ 4 เฟสนั้น เราจะต้องกระตุ้นให้มอเตอร์หมุนไปแต่ละ Step โดยจ่ายกระแสไฟฟ้าให้กับ Stepping ทีละเฟสตามลำดับ หลักการคือเริ่มจากจ่ายกระแสให้กับขดลวด Stator เฟสที่ 1 จากนั้นกระตุ้นเฟสที่ 2 และ เฟสที่ 3 ไปเรื่อยๆ ตามลำดับ จากนั้นก็วนกลับมาที่ขดลวด Stator เฟสที่ 1 อีกครั้งและวน Loop ไปเรื่อยๆ ก็จะทำให้ Stepping Motor หมุน และในทางกลับกันถ้าต้องการให้ Stepping Motor หมุนกลับทางก็ต้องกระตุ้นขดลวด Stator เฟส 4 เฟส 3 เฟส 2 และ เฟส 1 ตามลำดับ สามารถเขียนขั้นตอนการทำงานเป็นตารางออกมาได้ดังนี้

Step	เฟส 4	เฟส 3	เฟส 2	เฟส 1	Step
Step 1	ON	OFF	OFF	OFF	
Step 2	OFF	ON	OFF	OFF	
Step 3	OFF	OFF	ON	OFF	
Step 4	OFF	OFF	OFF	ON	
Step 5	ย้อนกลับ Step ที่1				
….	.....				

ควบคุมแบบ Full Step 2 เฟส หรือแบบ 2 เฟส

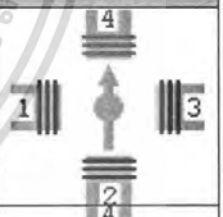
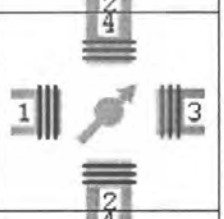
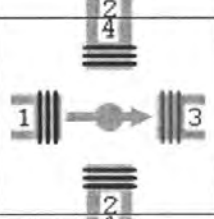
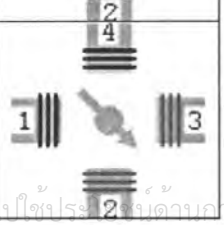
ในการความคุม Stepping Motor แบบ 2 เฟสนั้น เราจะต้องจ่ายกระแสไฟฟ้าเพื่อกระตุ้นขดลวดของมอเตอร์ทีละ 2 เฟส ในเวลาเดี่ยวกันและเรียงกันไปตามลำดับซึ่งได้แสดงดังตารางด้านล่าง โดย Stepping Motor จะหมุนเหมือนกับการควบคุมแบบเวฟ แต่การควบคุมแบบ 2 เฟสจะให้แรงบิดที่สูงกว่าแบบเวฟ

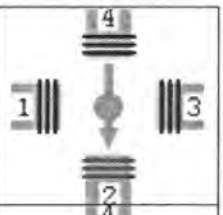
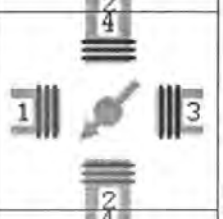
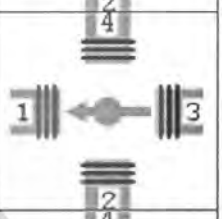
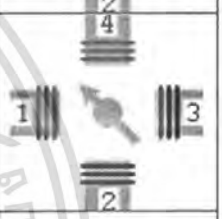
Step	เฟส 4	เฟส 3	เฟส 2	เฟส 1	Step
Step 1	ON	ON	OFF	OFF	
Step 2	OFF	ON	ON	OFF	

Step 3	OFF	OFF	ON	ON	
Step 4	ON	OFF	OFF	ON	
Step 5	ย้อนกลับ Step ที่1				
….	.....				

ควบคุมแบบ Half Step หรือแบบครึ่งสเต็ป่

การควบคุม Stepping Motor แบบครึ่งสเต็ปจะทำให้เราสามารถเพิ่มความละเอียดในการควบคุมการหมุนของ Stepping Motor ได้แม่นยำมากขึ้นซึ่งเป็นการผสมผสานระหว่างการควบคุมแบบเวฟและแบบ Full Step 2 เฟสเข้าด้วยกัน ลักษณะการจ่ายกระแสไฟ เพื่อกระตุ้นขดลวดจะแสดงดังตารางด้านล่าง

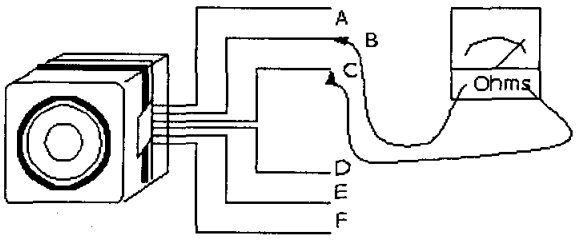
Step	เฟส 4	เฟส 3	เฟส 2	เฟส 1	Step
Step 1	ON	OFF	OFF	OFF	
Step 2	ON	ON	OFF	OFF	
Step 3	OFF	ON	OFF	OFF	
Step 4	OFF	ON	ON	OFF	

Step 5	OFF	OFF	ON	OFF	
Step 6	OFF	OFF	ON	ON	
Step 7	OFF	OFF	OFF	ON	
Step 8	ON	OFF	OFF	ON	
Step 9	ย้อนกลับ Step ที่1				
….	.....				

2.7.4 วิธีการตรวจสอบหาเฟสของขดลวดสเต็ปปิ้งมอเตอร์

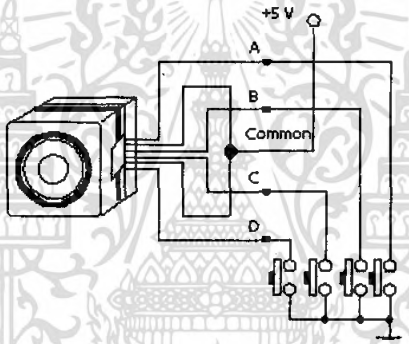
ในขั้นตอนที่ 1 ให้สังเกตว่าสเต็ปปิ้งมอเตอร์ที่นำมาทดลองที่เป็นแบบยูนิโพล่าร์ (Uni-polar stepper motor) จะมีจำนวนสาย 5เส้นหรือ 6 เส้น (ถอดจากดิสก์ไดร์ฟเก่าขนาด 5 นิ้วนำมาใช้งานได้)

ในขั้นตอนที่ 2 ใช้มิเตอร์วัดค่าความต้านทานของเส้นลวดในแต่ละขดดังรูป 6.17 ขั้นตอนการวัด ให้หาสายที่ต่อเป็นจุดร่วมเสียก่อน(common) โดยให้ใช้ มัลติมิเตอร์ตั้งค่าไว้สำหรับการวัดค่าความต้านทาน แต่ละเส้น สังเกตที่ค่าความต้านทาน ถ้าหากเราไม่ได้วัดระหว่าง จุดต่อร่วม(common) กับสายแต่ละเส้น ค่าความต้านทานจะมีค่าเป็น 2 เท่าของการวัดระหว่างจุดต่อร่วมกับสายที่ใช้งาน ตัวอย่างเช่น ถ้าให้จุด B เป็นจุดร่วม หากวัดระหว่างที่จุด A กับจุด B จะมีค่าเท่ากับ 60 Ohm แต่ถ้าวัดระหว่างที่จุด A และจุด C ซึ่งไม่ใช่จุดร่วมก็จะได้ค่าเท่ากับ 120 Ohm หากเป็นแบบที่มีสาย 6 เส้นก็จะมีจุดร่วมสองจุด เพราะมีขดลวดคนละชุดกัน และสายที่เป็นจุดร่วมส่วนใหญ่จะมีสีเหมือนกัน ทำนองเดียวกันหากเป็นแบบที่มีสาย 5 เส้นก็จะมีจุดร่วมเพียงจุดเดียวเท่านั้น



รูปที่ 2.4 การใช้มิเตอร์วัดค่าความต้านทาน

ในขั้นตอนที่ 3 หากเป็นแบบที่มีสาย 6 เส้นก็ให้ทำการต่อจุดร่วมเข้าด้วยกันจะได้เป็น 5 เส้น แล้วต่อวงจรตาม รูปหลังจากนั้นให้ทดลองกดสวิตซ์ ที่ต่อเข้ากับแต่ละจุดโดยเริ่มที่ จุด A จุด B จุด C และจุด D แล้วให้สังเกตการหมุนของสเต็ปปิ้งมอเตอร์ว่าหมุนได้ต่อเนื่องหรือไม่ หากมีการกระโดดข้ามสเต็ปก็ให้ทดลองโดยเรียงลำดับการกดสวิตซ์ใหม่ จนหาลำดับของสายได้ถูกต้องคือมอเตอร์เดินตามที่ละสเต็ป อย่างเป็นลำดับ



รูปที่ 2.5 แสดงการต่อวงจรเพื่อทดสอบโดยการสวิตซ์เพื่อหาลำดับ

2.8 ดิจิตอล อิมเมจ โปรเซสซิ่ง (Digital Image Processing)

การประมวลผลภาพด้วยคอมพิวเตอร์นั้น โดยทั่วไปสามารถทำการแบ่งแยกออกเป็น 2 ประเภทใหญ่ๆด้วยกัน คือ การประมวลผลภาพในระดับต่ำ (Low-Level Image Processing) และการประมวลผลภาพในระดับสูง (High-Level Image Processing) ซึ่งสามารถอธิบายความแตกต่างของการประมวลผลภาพทั้ง 2 แบบ ได้ดังนี้

การประมวลผลภาพในระดับต่ำนั้น เป็นการประมวลผลเชิงตัวเลขเกือบทั้งหมดเพื่อหาตัวแปรต่างๆมาอธิบายข้อมูลภาพ และมีจุดประสงค์ที่จะนำตัวแปรเหล่านี้ไปใช้ในการประมวลผลภาพในระดับสูงต่อไป การประมวลผลภาพในระดับต่ำโดยทั่วไปจะประกอบไปด้วยการกำจัดสัญญาณรบกวน , การทำให้ภาพคมชัดขึ้น , การหาขอบเขตของภาพ , การทำเซ็กเม็นท์ภาพ หรือการแบ่งแยกวัตถุภายในภาพ , การจดจำใบหน้าคน เป็นต้น

ในขณะที่ การประมวลผลภาพในระดับสูงคือการทำให้คอมพิวเตอร์รู้จักและเข้าใจภาพได้ เช่น การจดจำรูปแบบของตัวอักษร , แยกแยะวัตถุที่มีรูปร่างแตกต่างไปจากเดิมที่ถูกกำหนดไว้ , การจดจำใบหน้าคน เป็นต้น

ซึ่งโครงงานนี้จัดได้ว่าเป็นการประมวลผลสัญญาณภาพแบบระดับต่ำ เพราะลักษณะของการประมวลผลที่จะทำการแยกระหว่างพื้นฉากหลังกับวัตถุเท่านั้น และตรวจสอบแยกแยะความหมายของภาพที่รับเข้ามาได้

2.8.1 การแบ่งส่วนภาพ (Segmentation)

การแบ่งส่วนภาพนั้นเป็นการกำหนดขอบเขตของวัตถุภายในภาพบริเวณรอบจุดใดๆที่แตกต่างกันให้สามารถเห็นความแตกต่างชัดขึ้เพื่อนำไปประมวลผลขั้นสูงต่อไปได้ง่ายขึ้น ซึ่งสิ่งนี้เป็นส่วนที่จำเป็นต่อการวิเคราะห์ภาพว่าวัตถุในภาพนั้น มีขนาดใหญ่เท่าใด , ส่วนใดเป็นพื้นหลัง และวัดถุทั้งหมดนั้นมีอยู่เท่าไร ดังนั้นการแบ่งส่วนนั้นเป็นพื้นฐานที่จำเป็นสำหรับการชี้และการอธิบายวัดถุที่อยู่ในภาพ ให้สามารถถูกประมวลผลโดยโปรแกรมได้ดีและสะดวกยิ่งขึ้น ซึ่งการแบ่งส่วนนี้สามารถทำได้ง่ายๆโดยการนำข้อมูลภาพที่รับเข้ามานำไปเปรียบเทียบกับค่าคงที่ค่าหนึ่งซึ่งเรียกว่า ค่าเทรซโฮลด์ (Threshold) และภาพที่ได้หลังจากการเปรียบเทียบจะถูกเรียกว่าภาพไบนารี่ (Binary)

2.8.2 การทำเทรซโฮลด์ (Thresholding technique)

การทำเทรซโฮลด์ถือว่าเป็นเทคนิคที่สำคัญในการประมวลผลด้วยภาพในส่วนของการทำเซกเมนท์ภาพ ซึ่งจุดประสงค์ของการทำเซกเมนท์ภาพ คือ การแยกองค์ประกอบขอลภาพไปเป็นส่วนประกอบย่อยๆ ที่มีความสัมพันธ์กันทางกายภาพของภาพนั้น และส่วนประกอบที่ถูกแยกออกนั้นอาจถูกนำไปประมวลผลภาพในส่วนอื่นได้ต่อไป ซึ่งการทำเซกเมนท์ภาพจะมีหลักการทำงานในแนวเดียวกับสายตาของคน คือ สามารถแยกลักษณะเด่นออกมาจากภาพที่มองเห็นได้และเทคนิคการทำเทรสโฮลด์ ซึ่งถือว่าเป็นเทคนิคในการแยกองค์ประกอบของภาพที่ง่ายเทคนิคหนึ่งมีหลักการว่า จุดภาพที่มีคุณสมบัติอยู่ในช่วงใดๆจะถูกจัดเป็นกลุ่มได้โดยที่ระดับความเข้ม นั้นสามารถที่จะแบ่งแยกกลุ่มของจุดภาพออกเป็น 2 กลุ่มได้อย่างชัดเจน คือ กลุ่มของวัตถุ (Object) ซึ่งจะมีลักษณะความเข้มของภาพ g(x,y) ค่อนข้างต่ำ (มืด) กับกลุ่มของส่วนที่เป็นพื้นหลัง (Background) ที่จะมีระดับความเข้มของภาพ g(x,y) ค่อนข้างสูง (สว่าง) ดังภาพ แสดงฮิสโตแกรมของระดับความเข้มของภาพที่ถูกแบ่งออก เป็น 256 ระดับ (ในกรณีของภาพขนาดละเอียดที่ 8 บิต) จะเห็นได้ว่าการที่แยกกลุ่มของข้อมูลออกเป็น 2 กลุ่มอย่างชัดเจนย่อมสามารถทำได้โดยการเลือกค่าของเทรสโฮลด์ที่

มีความเข้มอยู่ระหว่างกลุ่มทั้งสอง บนฮิสโตแกรม ระดับความเข้มของภาพแล้วทำการตรวจสอบแต่ละจุดภาพว่าถ้ามีค่า g(x,y) น้อยกว่าค่าเทรสโฮลด์ ก็ถือว่าเป็นจุดภาพของวัตถุที่แสดงได้ด้วยจุดดำ แต่หากว่าจุด g(x,y) นั้นมีค่ามากกว่าหรือเท่ากับค่าเทรสโฮลด์ก็ถือว่าเป็นจุดภาพใน ส่วนพื้นหลังที่แสดงได้ด้วยจุดขาว ดังนั้นข้อมูลภาพ gthr(x,y) ที่ผ่านการทำเทรสโฮลด์สามารถนิยามได้ดังนี้

IF $g(x,y) \geq T$ gthr(x,y) Object = 255

Else gthr(x,y) Background = 0

ซึ่งในเงื่อนไขข้างต้นนี้เราอนุมานว่า สนใจเฉพาะวัตถุที่สว่างกว่าพื้นหลัง หากว่าเป็นการหาวัตถุที่วางอยู่ในพื้นหลังที่สว่างกว่าวัตถุก็จะแสดงได้ว่า

IF $g(x,y) \leq T$ gthr(x,y) Object = 255

Else gthr(x,y) Background = 0

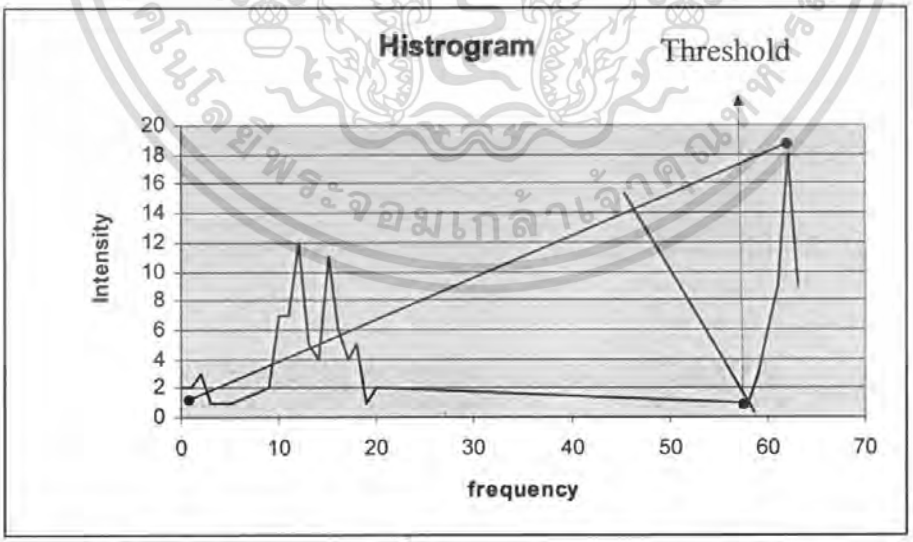
โดยที่ gthr(x,y) คือ ข้อมูลภาพผลลัพธ์เป็นไบนารี

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

T คือ ค่าเทรสโฮลด์ เป็นค่าคงที่ที่มีค่าระหว่าง 0 ถึง L

0 คือ จุดดำ (ส่วนที่เป็นวัตถุ)

255 คือ จุดขาว (ส่วนที่เป็นพื้นหลัง)



รูป 2.6 แสดงฮิสโตแกรมของระดับความเข้มของภาพขนาด 256 ระดับ

เมื่อได้ทำการศึกษาเกี่ยวกับการทำเทรสโฮลด์ไปแล้วภาพที่ผ่านการทำเทรสโฮลด์เราสามารถเรียกอีกอย่างหนึ่งว่าภาพไบนารี่ (Binary Image) ก็ย่อมเกิดคำถามที่ตามมาว่า แล้วจะใช้ค่าเทรสโฮลด์

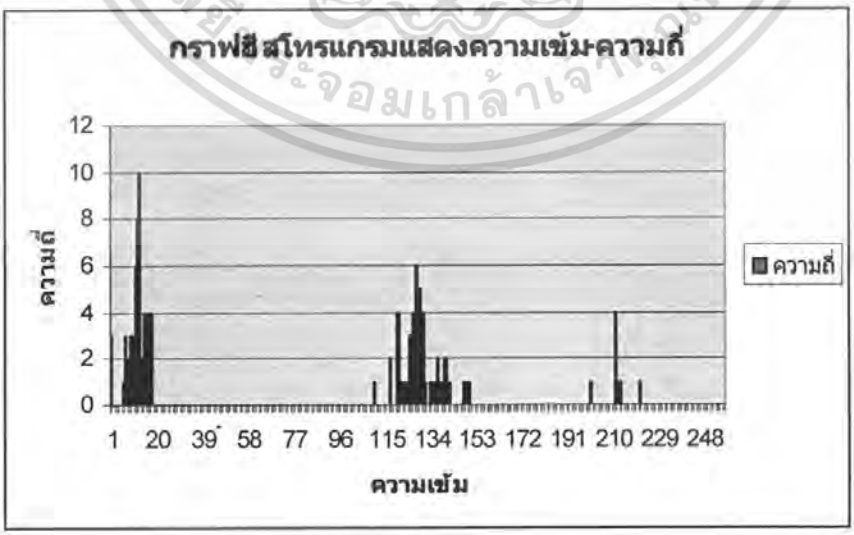
เท่าไหร่ ซึ่งการกำหนดค่าของการทำเทรสโฮลด์นี้สามารถทำได้อยู่ 2 รูปแบบ คือ การกำหนดค่าเทรสโฮลด์แบบคงที่ตลอดทั้งภาพ (Fixed Threshold) และมีอีกวิธี คือ เลือกค่าเทรสโฮลด์จากระดับฮีสโตแกรม (Histogram-derived thresholds)

2.8.2.1เทรสโฮลด์แบบคงที่ตลอดทั้งภาพ (Fixed Threshold)

ซึ่งเป็นทางเลือกหนึ่งที่ถูกใช้ทำเทรสโฮลด์โดยไม่ได้ขึ้นกับข้อมูลภาพที่ได้นำมาประมวลผลโดยการทำเช่นนี้ได้ก็ต่อเมื่อเราทราบแล้วว่าวัตถุที่มีสีเข้มมากๆ ได้ถูกวางไว้บนพื้นหลังที่เป็นสีขาวหรือสีอ่อนกว่าวัตถุโดยที่พื้นหลังจะต้องมีความสว่างที่เท่ากันตลอด ดังนั้นแล้วเราสามารถเลื่อนค่าเทรสโฮลด์เป็น 128 จากระดับความเข้มของภาพขนาด 0 ถึง 255 ได้ซึ่งจะทำให้ผลลัพธ์ที่ได้ถูกต้องมากที่สุด แต่วิธีการนี้จะมีข้อจำกัดตรงที่หากว่าความสว่างของพื้นหลังมีค่าไม่คงที่แล้วจะทำให้เกิดการผิดพลาดในการตรวจจับวัตถุได้

2.8.2.2เทรสโฮลด์จากระดับฮีสโตแกรม (Histogram-derived threshold)

ในกรณีที่ข้องมูลภาพมีความไม่สม่ำเสมอเกิดขึ้นในส่วนของวัตถุ หรือส่วนของพื้นหลังหรือในทั้งสองส่วนซึ่งภาพในลักษณะเช่นนี้ ฮีสโตแกรมระดับความเข้มของภาพที่เกิดขึ้นอาจมีลักษณะดังรูป ดังนั้นในการกำหนดหาค่าเทรสโฮลด์เพื่อจะนำมาช้างานจึงสามารถใช้หลักการของการหาค่าเฉลี่ย (Mean) ระหว่างระดับความสว่างที่ความเข้มสูงสุดกับระดับความมืดที่ความเข้มภาพสูงสุดเช่นกัน โดยสามารถแสดงได้ดังสมการต่อไปนี้



รูป 2.7 กราฟฮีสโตแกรมที่พื้หลังกับวัตถุมีระดับความเข้มภาพไม่แตกต่างกันมาก

$$T = \{h[g(x,y)](Brightness) + h[g(x,y)](Darkness)\} / 2$$

โดยที่ h[g(x,y)](Brightness) คือ ความสว่างของจุดที่มีจำนวนของ พิกเซล มากที่สุด

h[g(x,y)](Darkness) คือ ความมืดของจุดที่มีจำนวนของ พิกเซล มากที่สุด

ซึ่งเมื่อทำการคำนวณเทรสโฮลด์ได้แล้ว ก็สามารถทำการเซกเมนต์ภาพได้โดยนำค่าเทรสโฮลด์ที่ได้มาแทนค่าในสมการก่อนหน้าได้ทันที

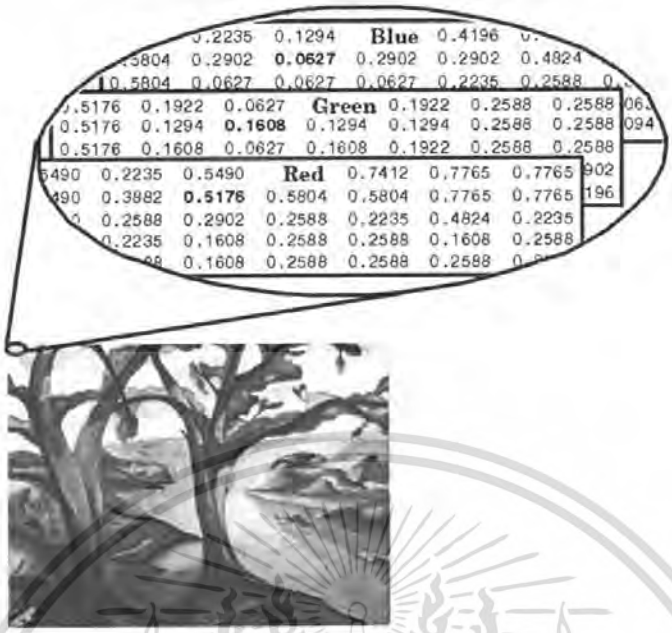
2.8.3 การแยกส่วนสี (Color Segmentation)

ในโครงงานชุดนี้ได้ทำการเลือกใช้สีที่จะนำมาประมวลผลในโปรแกรมอยู่ด้วยกันทั้งหมดอยู่ 3 สีด้วยกัน ซึ่งทั้งหมดจะเป็นแม่สีได้แก่ สีแดง สีน้ำเงิน และสีเขียว (แทนสีของวัตถุ) ซึ่งเหตุผลที่เลือกใช้แม่สีดังกล่าวก็เพราะว่า เราสามารถออกแบบส่วนของโปรแกรมจะใช้ในการกรองสีได้ง่ายกว่าการใช้สีอื่นๆที่ไม่ใช่แม่สี

ความแตกต่างของกลุ่มของจุดสี (Color Pixel) ของแม่มีทั้งสาม จะเห็นว่า เมื่อนำภาพสีที่มีวัตถุแม่สีทั้งสามมาแปลงให้เป็นระดับสีเทาแล้ว จะได้ฮีสโทแกรมที่มีความแตกต่างดังนี้ คือ สีน้ำเงินที่ถูกแปลงให้กลายเป็นสีเทา (Gray Color) แล้วจะมีระดับความสว่างที่ 0 < ระดับความสว่าง > 64 เพราะมีความสว่างน้อยกว่าแม่สีอื่น สีแดงจะอยู่บริเวณ 64 < ระดับความสว่าง > 128 และสีเขียว 128 < ระดับความสว่าง > 255 จะเห็นว่าแม่สีต่างๆจะกระจายเป็นกลุ่มๆอย่างเห็นได้ชัดเจน

2.8.4 การเข้าถึงระดับพิกเซลในภาพ (Pixel Accessing)

ภาพหนึ่งภาพหรือหนึ่งรูปนั้นประกอบไปด้วยจุดสีต่างๆมากมาย ที่มีความแตกต่างกันของระดับสีเต็มไปหมด ซึ่งสีต่างๆที่เรามองเห็นล้วนเกิดจากการผสมกันระหว่างแม่สีทั้งสาม คือ แดง เขียว และน้ำเงิน เมื่อพิจารณาเข้าไปถึงจุดเล็กที่สุดของภาพประกอบไปด้วยจุดสี 3 จุด ซึ่งถูกเรียกว่า พิกเซล (Pixel) จะประกอบไปด้วยแม่สีทั้งสามที่ได้กล่าวไปแล้วซึ่งได้แสดงดังรูป



รูป 2.8 แสดงจุดเล็กที่สุดของภาพที่ประกอบไปด้วยแม่สีแดง เขียว และน้ำเงิน

ซึ่งที่สีสันต่างกันตามความสว่างของแต่ละพิเซลก็จะไม่เหมือนกัน เช่น จุดภาพจุดหนึ่งได้ถูกกำหนดให้แสดงสีแดงเป็นแม่สี (ไม่มีสีอื่นเลยนอกจากสีแดง) ดังนั้นจึงขอสมมุติให้พิกเซลทั้งสามเสมือนว่าเป็นหลอดไฟสามสี คือ หลอดไฟสีแดง หลอดไฟสีเขียว และหลอดไฟสีน้ำเงิน ตามลำดับ เพื่อให้ง่ายต่อความเข้าใจ ในการทำงานของหลอดไฟทั้งสาม ก็จะมีเพียงหลออดไฟสีแดงเท่านั้นที่ติดสว่าง (ความสว่างถูกกำหนดโดยระดับฮีสโตแกรม 0-255 ระดับหรือ 8 บิต) ส่วนที่เหลือก็จะดับอยู่แบบนั้น และถ้าจุดภาพหรือ Dot ดังกล่าวถูกกำหนดให้แสดงขาว หลอดไฟทั้งสามก็จะติดเหมือนกันหมด และความสว่างของสีทั้งสามหลอดก็จะต้องเท่ากัน (ความสว่างเท่ากับ 255 เพราะถ้าต่ำกว่านี้จะเป็นสีเทา)

ดังนั้นการเข้าถึงข้อมูลภาพในระดับพิเซลก็เพื่อทำการอ่านค่าความสว่างของสีต่างๆว่ามีขนาดเท่าไรบ้างเพื่อนำมาเปรียบเทียบกับค่าคงที่ที่จะทำการแยกสี (Color Segmentation) ซึ่งการเปรียบเทียบค่านี้ก็คือการทำทรสโฮลด์สีนั่นเอง ซึ่งจะมีหลักอยู่ว่า จุดของภาพดังกล่าวมีระดับความสว่างของสีต่างๆอยู่ในช่วงของแม่สีใดบ้าง ถ้าสมมติว่าภาพจากกล้องที่รับเข้ามามีความสว่างอยู่ในย่านของสีน้ำเงินพอดี (เช่น ความสว่างของพิกเซล สีน้ำเงิน-สีแดง-สีเขียว = 150-90-65) ซึ่งภาพรวมของจุดสีดังกล่าว คือ สีน้ำเงิน เพียงแต่ว่ายังไม่ใช่สีน้ำเงินที่แท้จริงซึ่งมีค่าเท่ากับ (255-0-0) ดังนั้นจึงต้องทำการปรับความสว่างของพิเซลในจุดังกล่าว (Pixel in Dot) โดยวิธีการทำเทรสโฮลด์ดังที่กล่าวเอาไว้แล้ว

หลังจากที่เราได้ทำการแยกสี (Color Segmentation) ไปแล้ว ลำดับต่อไปก็จะเป็นการตรวจจับหรือตรวจหาชนิดของวัตถุ ว่าอยู่ส่วนใดของภาพ เรียกว่า Object Tracking

2.8.5 การติดตามวัตถุ (Object Tracking)

ระบบที่ใช้การประมวลผลรูปภาพ (Image Processing) เพื่อการคัดแยกหรือตรวจหาวัตถุที่ต้องการจากรูปภาพนั้นโดยส่วนมากแล้วจะต้องมีวิธีการใดวิธีการหนึ่งที่จะใช้ติดตามหรือแยกแยะวัตถุที่ต้องการออกจากวัตถุอื่นๆ ที่ปะปนมากับในรูปออกให้ได้ ซึ่งก็ได้มีผู้คิดค้นวิธีการอยู่หลายแบบด้วยกัน ซึ่งวิธีที่น่าจะเป็นไปได้สำหรับการเขียนโปรแกรมมี 2 แบบ ดังนี้ วิธีแรก การแมทชิ่งกับแม่แบบ (Template Matching with Distance Transforms) หรือ (Model Comparing Method) ซึ่งวิธีการนี้เป็นวิธีที่มีความยืดหยุ่นค่อนข้างสูง ใช้ได้กับรูปภาพที่มีสภาพแสงไม่คงที่ แต่ก็มีความซับซ้อนในการออกแบบและการเขียนโปรแกรมอย่างมาก และใช้เวลาในการทำงานค่อนข้างสูง ซึ่งจะกล่าวให้เห็นถึงหลักการของการทำงานของวิธีแรกอย่างคร่าวๆดังนี้

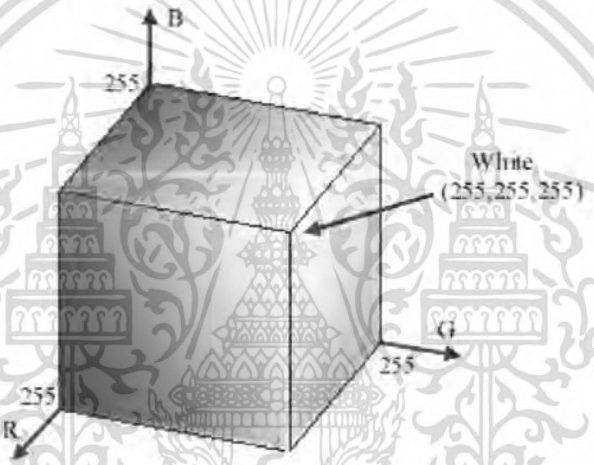
2.8.5.1 การแทรกกิ้งโดยใช้การเทียบกับต้นแบบ (Model Comparing Method)

ในวิธีนี้เราจะต้องมีสิ่งที่สำคัญอันดับแรก คือ ต้นแบบ (Model) ที่จะใช้เปรียบเทียบซึ่งอาจได้จากรูปถ่ายหน้าคนที่ต้องการรทดสอบแล้วนำภาพที่ได้มาทำการตัดทอนส่วนที่ไม่จำเป็นในรูปทิ้งไป (เก็บเอาแต่ลักษณะเด่นของหน้าคนในรูป) หลังจากที่ได้โมเดลมาแล้วก็นำมาเก็บไว้เพื่อรอการใช้งาน ขั้นต่อไปพิจารณารูปภาพที่จะนำมาทำการแทรกกิ้งใบหน้าคน โดยจะเตรียมจากการแยกส่วนภาพ ซึ่งในที่นี้คือการหาขอบภาพ (เพื่อให้ง่ายต่อการเปรียบเทียบ) หรือที่รู้จักกันในชื่อ Edge Detection เมื่อได้ทำการเตรียมรูปภาพที่จะนำไปเปรียบเทียบกับตัวโมเดลที่เก็บไว้แล้วนั้น ขั้นต่อไปก็เป็นส่วนของกการเปรียเทียบระหว่างโมเดลกับรูปภาพ ซึ่งจะมีวิธีการเพื่อใช้หาวิธีเปรียบเทียบอีกขั้นหนึ่งนั้นก็คือ ดิสแทนซ์ ทรานส์ฟอร์ม (Distance Transform) และหนึ่งในวิธีการนั้นคือวิธีของ Hausdorff Transform แต่จะไม่ของกล่าวถึงรายละเอียดในรายงานเล่มนี้เนื่องจากเกินขอบเขต

ในหลักการตลอดหัวข้อ ที่ได้กล่าวมานี้ นอกจากจะสามารถทำการแทรกกิ้งใบหน้าคนได้แล้วถ้าหากว่านำไปทำการจัดการกับโมเดลให้เป็นฐานข้อมูลที่เหมาะสมแล้วเราจะสามารถจดจำใบหน้าหรือรู้จักว่าคนที่กำลังอยู่ในภาพนั้นเป็นคนๆเดียวกันกับที่เก็บไว้ในฐานข้อมูลหรือไม่ได้อีกด้วย เรียกว่า รู้จักวัตถุ (Object Recognition)

2.8.5.2 การแทรกกิ้งโดยการคัดแยกสีหรือกรองสี (Color Filtrations)

เป็นวิธีการที่ถูกในนำมาใช้ในรายงายชุดนี้ เหตุผลก็เพราะสีที่ใช้ในโครงงานมีเพียงแม่สี 3 สี คือ สีแดง สีเขียว และสีน้ำเงิน เท่านั้น จึงทำให้มีความรวดเร็วในการประมวลผลเนื่องจากการแทรกกิ้งแบบนี้จะใช้หลักการกรองเอาสีอื่นที่ไม่ต้องการทิ้งไป ให้เหลือไว้แต่สีของวัตถุที่ต้องการเท่านั้น แต่การกรองสีนั้นหากมีสัญญาณรบกวนที่เป็นสีเดียวกับวัตถุที่ต้องการหาในภาพ ก็จะเกิดการผิดพลาดในการแทรกกิ้งได้ ดังนั้นจะต้องคำนึงถึงสิ่งที่จะแทรกกิ้งด้วยเพื่อตัดปัญหาของสัญญาณรบกวนออกไป จากนั้นจึงหาว่ากลุ่มของสีที่อยู่ในวัตถุที่มีขนาดเท่ากับที่ต้องการหรือไม่ ถ้าใช่ก็จะทำการหาขอบเขตหรือรัศมีของวัตถุดังกล่าว เพื่อหาตำแหน่งศูนย์กลางของวัตถุออกมาเราก็จะได้พิกัดที่จะนำมาใช้งานได้แล้ว



รูป 2.9 แสดงการผสมสีของแม่สีทั้งสาม

รูปแสดงการผสมสีของแม่สีทั้งสามซึ่งกำหนดให้แม่สีดังกล่าวมีระดับความสว่างของแต่ละสีเท่ากับ 255-255-255 = R-G-B (คือระดับที่เป็นไปได้ของเลขฐานสองขนาด 8 บิต * 3 สี) หรือเขียนแทนด้วยสัญญาลักษณ์เลขฐานสิบหกจะได้เป็น FFFFFFh ให้จะเห็นว่าบริเวณที่มีการซ้อนทับกันระหว่างแม่สีจะเกิดสีใหม่ขึ้นมาซึ่งในการแทรกกิ้งโดยการกรองสีจะถือว่าสีนั้นคือสัญญาณรบกวนต้องกำจัดออกไป

บทที่ 3

การออกแบบและการสร้าง

ซอฟต์แวร์ที่ใช้ในโครงงาน

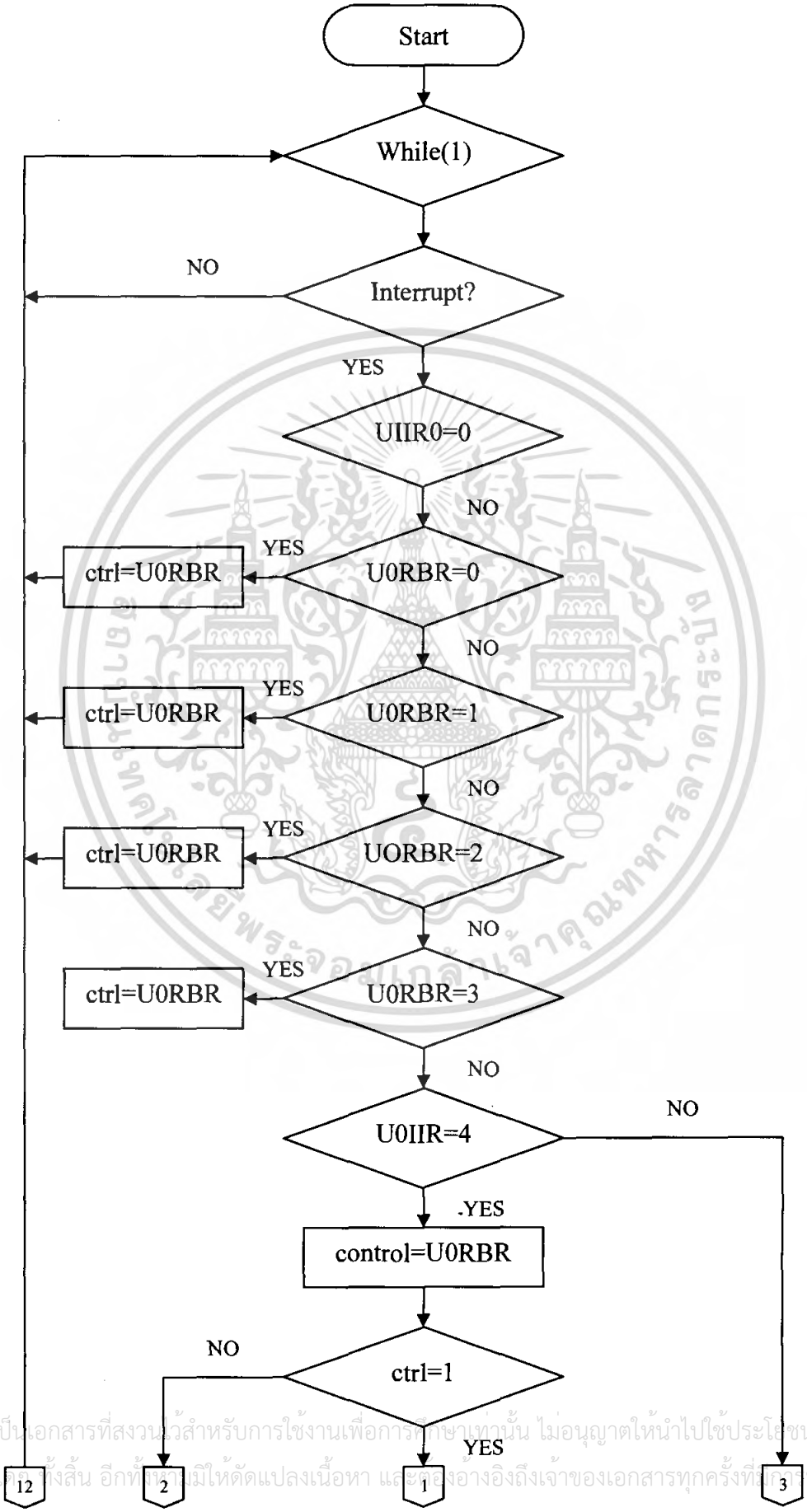
ส่วนของซอฟต์แวร์คอมไพเลอร์เพื่อแปลภาษาซีเป็นภาษาเครื่องของ LPC2138 จะใช้ชุดพัฒนา RealView Microcontroller Development Kit Version 3.02a โดยใช้คอมไพเลอร์ CARM ของบริษัท Keil เอง เป็นรุ่น Evaluation ที่อนุญาตให้ดาวน์โหลดมาใช้ทดสอบไม่เกิน 16kB และห้ามนำโคดโปรแกรมที่ได้ไปใช้ในทางการค้า หลังจากคอมไพล์ได้เป็นไฟล์ภาษาเครื่อง ทำการโปรแกรมลงชิปไมโครคอนโทรลเลอร์ด้วยโปรแกรม LPC2000 Flash Utility ของบริษัท Philips ซึ่งอนุญาตให้ใช้งานได้ฟรี

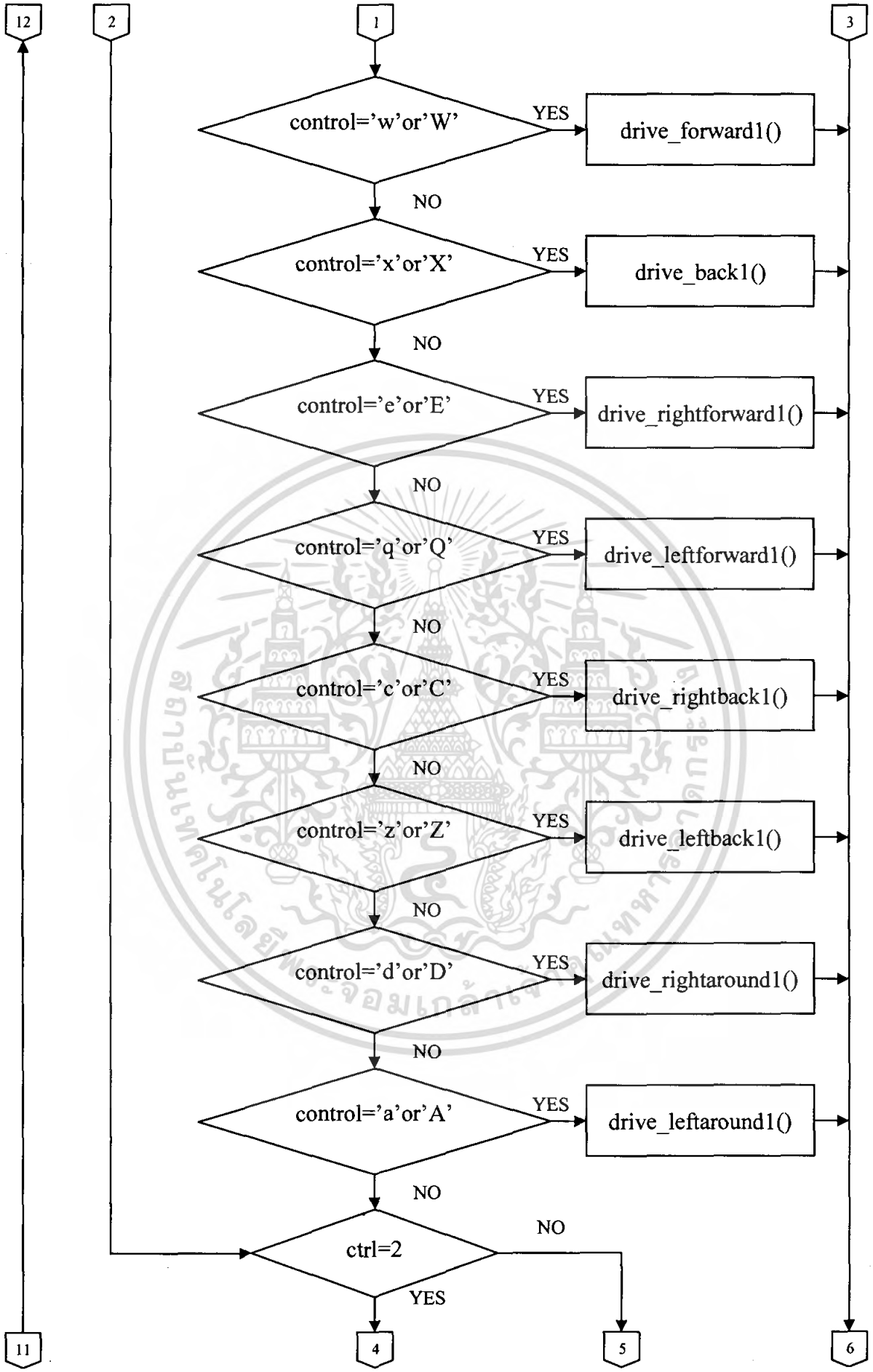
3.1 ซอฟต์แวร์ Realview Microcontroller Development Kit V3.02a

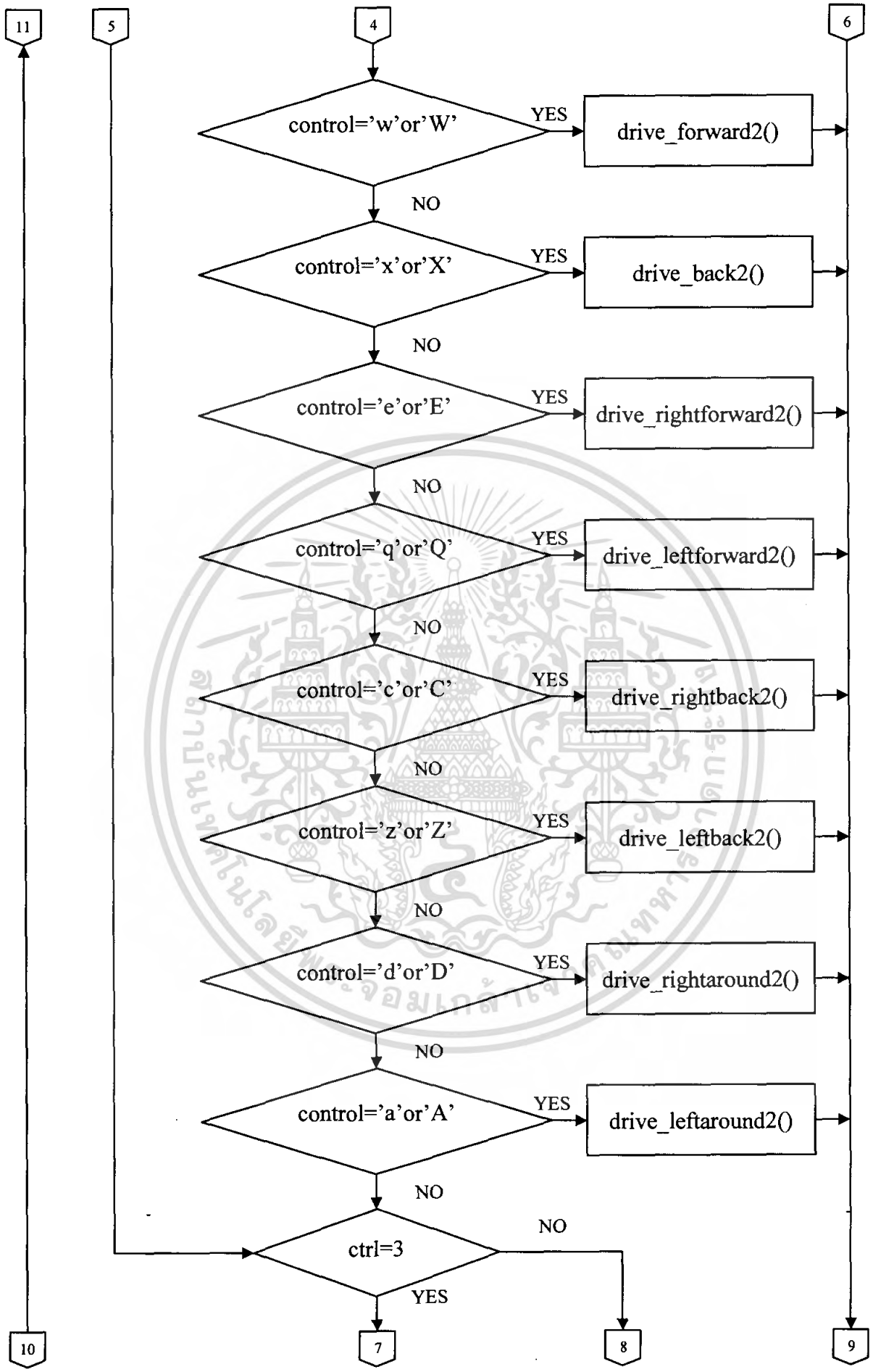
ก่อนอื่นต้องทำการดาวน์โหลดซอฟต์แวร์ก่อน โดยให้เข้าไปที่เว็บไซต์ของบริษัท Keil โดยตรงที่ http://www.keil.com ให้พิจารณาในส่วน Sotfware Downloads คลิกเลือกหัวข้อ Evaluation Software ก่อนที่จะดาวน์โหลดซอฟแวร์จะต้องลงทะเบียนก่อน จึงจะดาวน์โหลดได้ ในที่นี้จะได้ไฟล์ชื่อ rvmdk302a.exe ดับเบิลคลิกไฟล์เพื่อติดตั้งโปรแกรม โดยโปรแกรมจะแสดงหน้าต่างตามรูปที่3.1

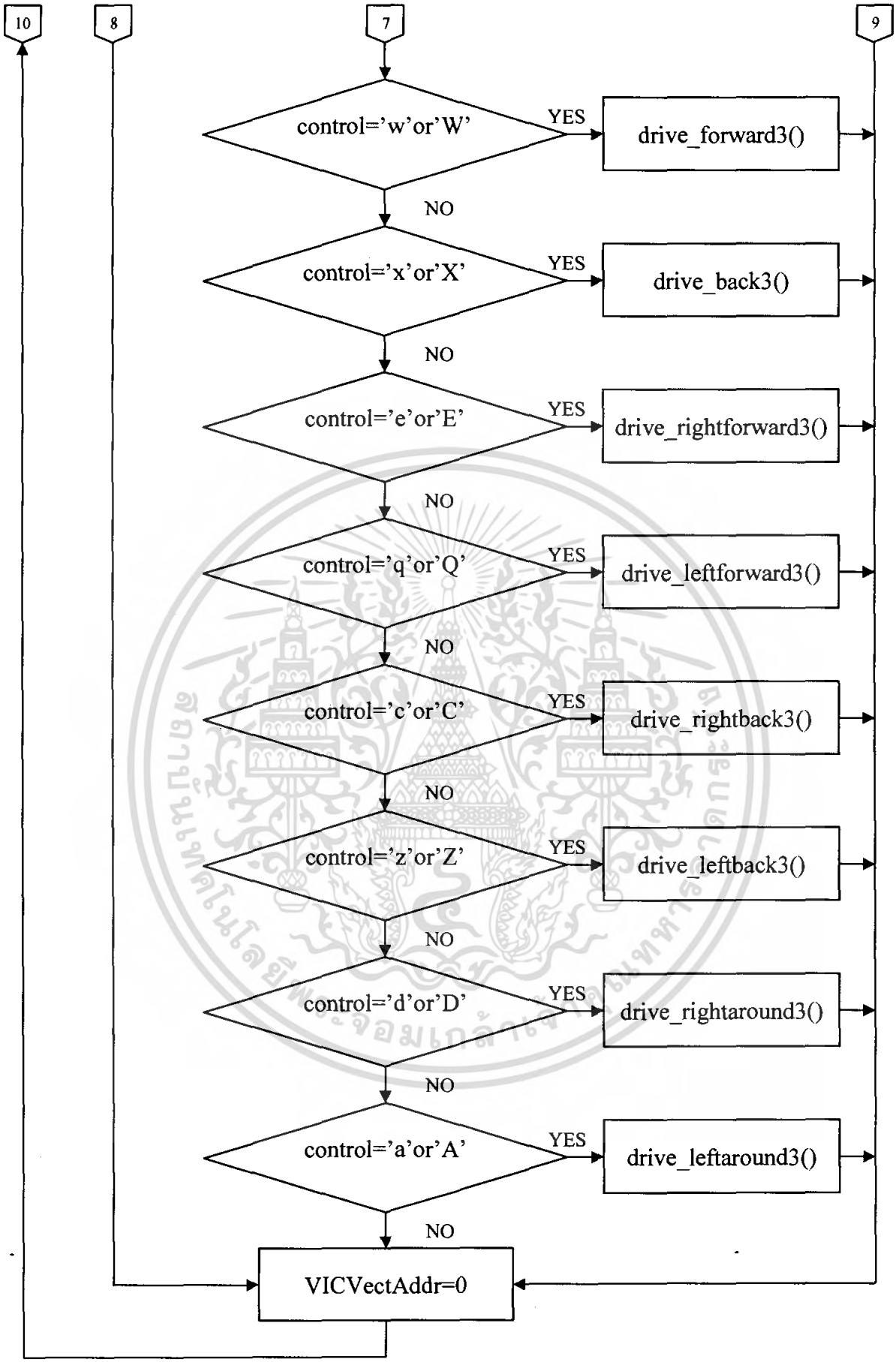
รูปที่ 3.1 หน้าต่างขั้นตอนแรกของการติดตั้งซอฟต์แวร์ Realview Microcontroller Development Kit V3.02a

3.2 Flow Chart โปรแกรมขับ Stepping Motor ที่ใช้กับ ARM7 LPC2138

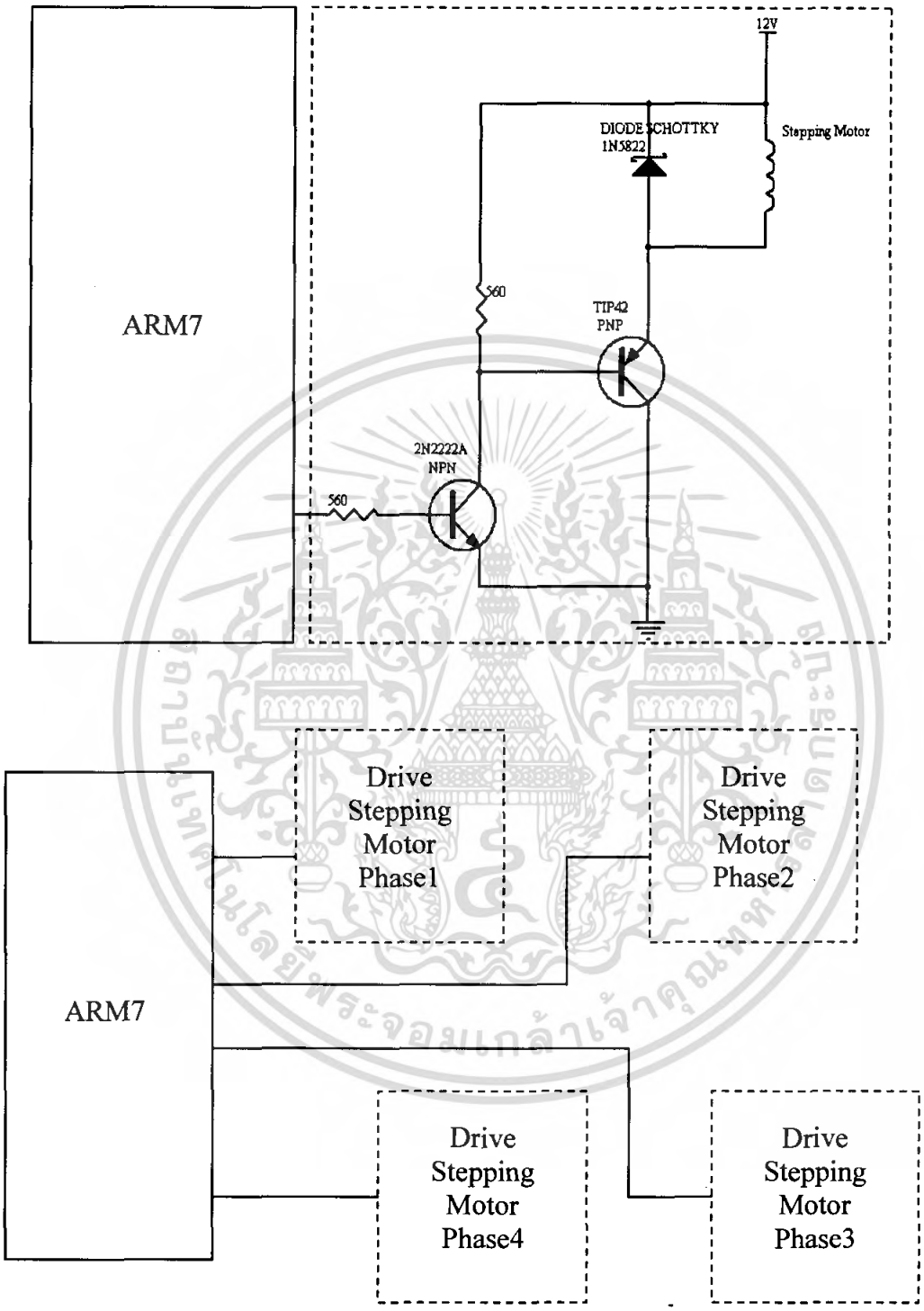








3.3 วงจรขยายสัญญาณขับ Stepping Motor



รูปที่ 3.2 วงจรขยายสัญญาณขับ Stepping Motor

3.4 โปรแกรมภายในเครื่องคอมพิวเตอร์

โปรแกรมที่ใช้งานเพื่อทำการติดต่อและควบคุมหุ่นนั้นเป็นโปรแกรมในลักษณะของกราฟฟิกส์ ยูสเซอร์ อินเตอร์เฟสซ์ (Graphics User Interfaces) คือ โปรแกรมที่ใช้การแสดงผลแบบรูปภาพเพื่อให้ผู้ใช้งานสามารถทำงานได้โดยไม่ต้องยุ่งกับคำสั่งจำนวนมาก และโปรแกรมที่ใช้ในการสร้างโครงงานนี้ คือ บอร์แลนด์ ซี++ บิลล์เดอร์ 6.0 (BCB 6.0) ซึ่งเป็นโปรแกรมที่ใช้ภาษาซีเป็นพื้นฐานในการเขียนโปรแกรม ซอฟต์แวร์ ซี++ บิวล์เดอร์ นั้นจัดเป็นเครื่องมือเขียมโปรแกรมชนิด วิชัวล์ โปรแกรมมิ่ง (Visaul Programming) เช่น เดียวกับโปรแกรมเดลฟาย, วิชัวล์ เบสิก, วิชัวล์ ซี++ เป็นต้น

ส่วนที่เป็นหัวใจหลักของโครงงานนี้ คือ การประมวลผลภาพหรือ (Image Processing) เราจะใช้ Intel OpenCV Library ซึ่งเป็นไลบรารี่สำหรับงานด้านอิมเมจโปรเซสซิ่ง (Image Processing) ซึ่งพัฒนาขึ้นโดยบริษัท Intel และเผยแพร่ให้ใช้งานได้ไม่เสียค่าลิขสิทธิ์ ใช้ในการประมวลสัญญาณภาพที่รับเข้ามาจากกล้องวิดีโอ แล้วนำผลที่ได้จากกระบวนการไปทำการควบคุมหุ่นยนต์โดยผ่านทางพอร์ตอนุกรม (Serial Port) ของเครื่องคอมพิวเตอร์อีกต่อหนึ่ง โดยจะมีการออกแบบโปรแกรมไว้ดังนี้

3.4.1 การรับภาพจากกล้องวิดีโอ

เราใช้กล้องเว็บแคม ในการรับภาพเข้าสู่คอมพิวเตอร์ ซึ่งสามารถปรับความละเอียดของภาพได้ โดยขนาดของภาพอยู่ที่ 480k พิกเซล คำสั่งที่ใช้ในการรับภาพจากกล้องเว็บแคมคือ cvCapture (เป็นคำสั่งของ OpenCV Library) โดยภาพที่รับเข้ามานี้จะถูกจัดเก็บในรูปแบบสี RGB

3.4.2 การทำงานของโปรแกรม

ในการเขียนโปรแกรมจะทำการแยกส่วนของโปรแกรมที่ใช้งาน โดยหลังจากที่รับภาพจากกล้องก็จะทำกระบวนการภาพซึ่งแบ่งเป็น

Colors Segmentation

เพื่อปรับภาพที่ได้ให้เหมาะสมก่อนนำไปคัดแยกสี เหตุที่ต้องทำก็เพราะในภาพที่รับเข้ามาจากกล้องวิดีโอนั้น ยังมีปัญหาจากแสงที่ไม่สม่ำเสมอ ที่มาตกกระทบกับวัตถุทำให้สีของวัตถุที่กล้องจับได้นั้นเกิดการผิดเพี้ยน เช่นความมืดของสีที่บริเวณของขอบภาพ สีของวัตถุที่ไม่เท่ากันเป็นต้น ดังนั้นการทำเซกเมนต์ดังกล่าวก็เพื่อให้เกิดความชัดเจนของสีทั้งหมดออกมาเท่ากัน ตลอดทั้งภาพ (มีความสว่างเท่ากันทั้งภาพ)

Color Filtration and Object Positioning

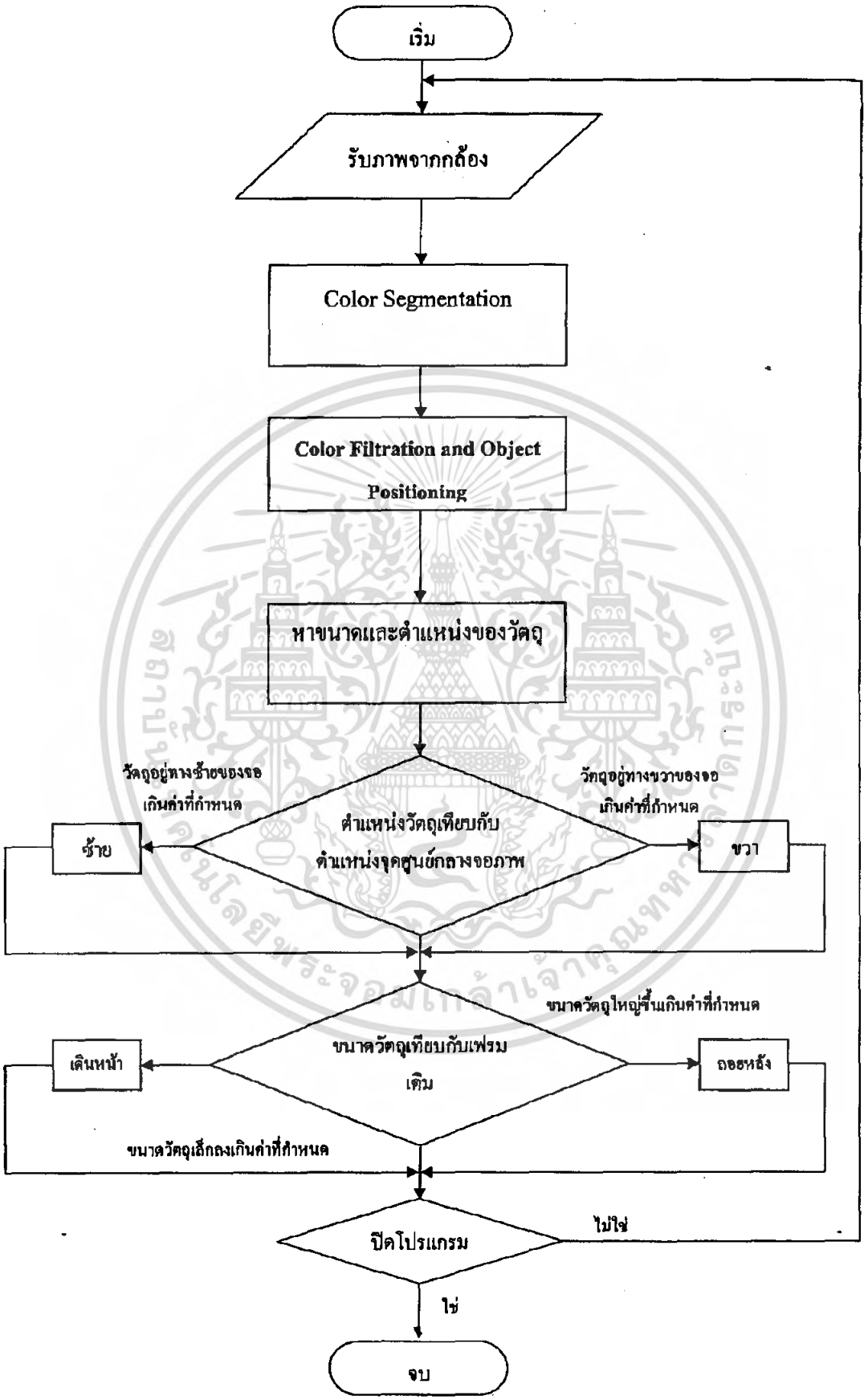
หลังจากที่เราได้ภาพที่เหมาะสมที่จะใช้ในการคัดแยกสีของวัตถุแล้ว เราก็จะทำการกรองสี โดยใช้การเทรชโฮลด์(Threshold) ซึ่งในที่นี้จะทำการกรองเอาสีของวัตถุที่เราสนใจ จากนั้นทำการหาตำแหน่งของวัตถุในภาพว่าอยู่ ณ ตำแหน่งใดในภาพโดยใช้การอ้างอิงกับจำนวนพิกเซลในภาพนั้น

Object Tracking Processing

โปรแกรมสำหรับติดตามวัตถุ ซึ่งวัตถุที่จะติดตามนั้น คือ วัตถุที่มีสีแตกต่างจากสิ่งแวดล้อม โดยจากการที่เราได้ทำการ Colors Segmentation และ Color Filtration and Object Positioning แล้ว เราจะทำการหาตำแหน่งและขนาดของวัตถุ โดยหาจากตำแหน่งค่าพิกเซล เราจะได้จุดตำแหน่งพิกัดของวัตถุ จากนั้นเราจะทำการเปรียบเทียบพิกัดนั้นกับจุดศูนย์กลางของจอแสดงผลภาพ ถ้าพิกัดของวัตถุอยู่ทางด้านซ้ายของพิกัดกลางจอภาพเกินค่าที่กำหนดก็จะออกคำสั่งให้เลี้ยวซ้าย ถ้าพิกัดของวัตถุอยู่ทางด้านขวาของพิกัดกลางจอภาพเกินค่าที่กำหนดก็จะออกคำสั่งให้เลี้ยวขวา ต่อมาเราจะเทียบขนาดวัตถุจากเฟรมเดิม ถ้าวัตถุมีขนาดใหญ่ขึ้นเกินค่าที่กำหนดก็จะออกคำสั่งให้ถอยหลัง ถ้าวัตถุมีขนาดเล็กลงเกินค่าที่กำหนดก็จะออกคำสั่งให้เดินหน้า ที่ต้องมีการกำหนดค่าค่าหนึ่งไว้เป็นค่าเปรียบเทียบเพราะว่า เพื่อลดปัญหาการเคลื่อนที่กระตุกของหุ่นยนต์เนื่องจากวัตถุที่เราต้องการติดตามอาจเคลื่อนที่อยู่ตลอดเวลา กล่าวคือถ้าวัตถุอยู่ในตำแหน่งที่ไม่ต่างไปจากเดิมมากนักคือไม่เกินค่าที่กำหนดก็จะไม่มีการออกคำสั่ง แต่ถ้าเกินกว่าค่าที่กำหนดก็จะมีการออกคำสั่งนั้นเอง

เงื่อนไขของวิธีนี้ คือต้องควบคุมแสงรบกวนจากภายนอก ปัญหาจากแสงรบกวนภายนอกในระบบที่ต้องใช้การประมวลผลภาพนั้น จำเป็นอย่างยิ่งที่จะต้องมีแสงที่กล้องจับเข้ามานั้นเพียงพอและสม่ำเสมอ ไม่มากเกินไปจนเกิดการสะท้อนแสง ไม่เช่นนั้นแล้วภาพเมื่อนำไปประมวลผลแล้วจะทำให้การทำงานของระบบผิดพลาดได้ แก้ปัญหาโดยการจัดสภาพแสงและสิ่งแวดล้อมให้เหมาะสมเพื่อให้การทำงานถูกต้องมากที่สุด

3.5 Flow Chart Object Tracking Processing

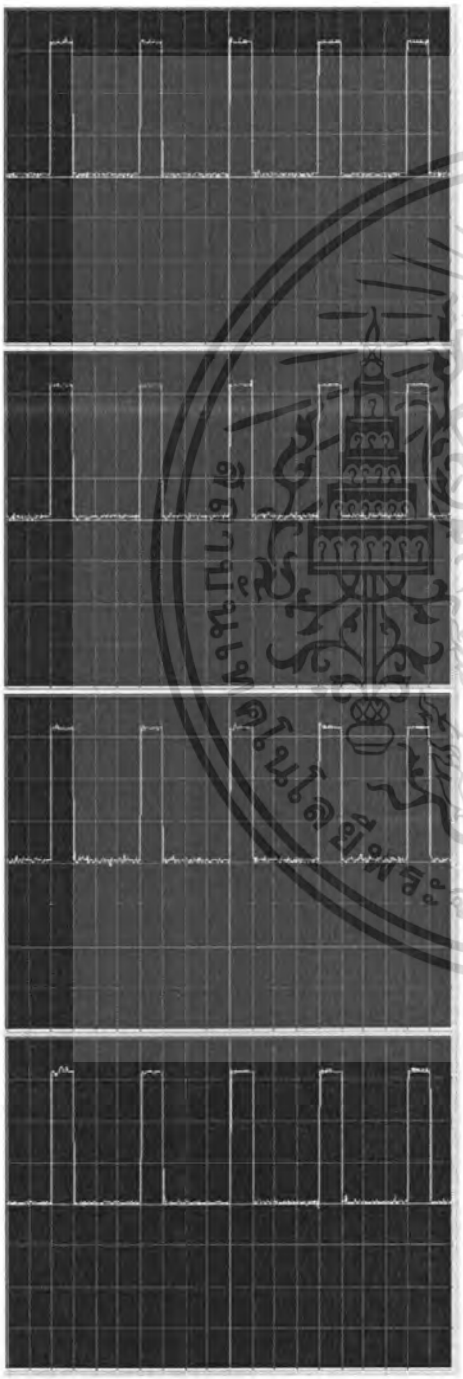


บทที่ 4

การทดลองและผลการทดลอง

-ในส่วนของการเคลื่อนที่ของRobot เราได้โปรแกรมระดับความเร็วไว้ 3 ระดับ

4.1 ที่ความเร็วระดับที่ 1



กราฟสัญญาณ P0.16

Risetime	1.600ms
Vmax	3.32V
Freq	4.999Hz
Duty Cycle	25.00%
Falltime	1.580ms

กราฟสัญญาณ P0.17

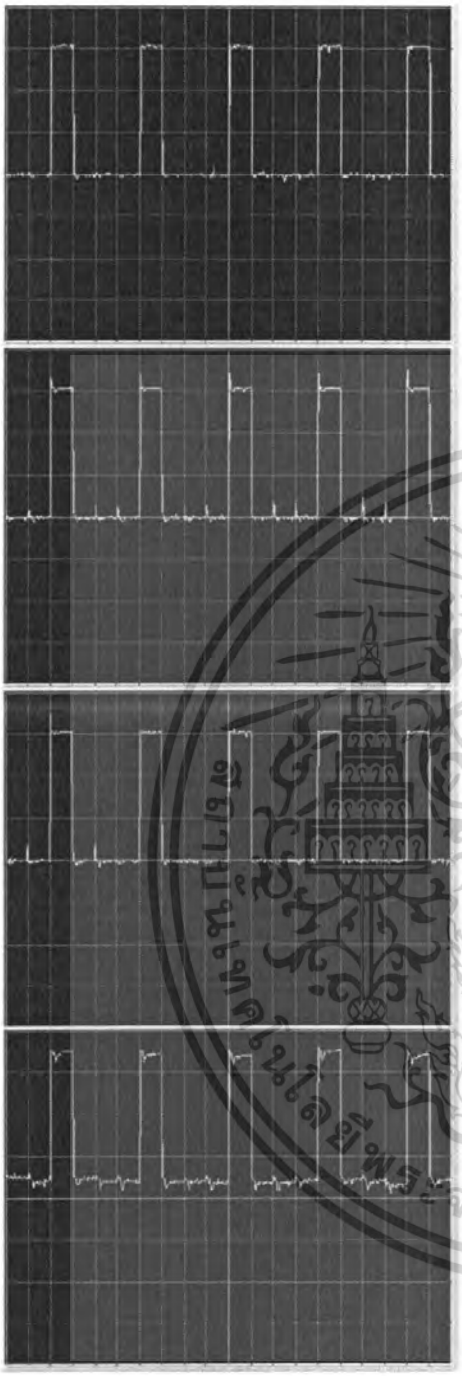
Risetime	1.580ms
Vmax	3.32V
Freq	5.000Hz
Duty Cycle	24.99%
Falltime	1.600ms

กราฟสัญญาณ P0.18

Risetime	1.580ms
Vmax	3.36V
Freq	4.999Hz
Duty Cycle	24.99%
Falltime	1.600ms

กราฟสัญญาณ P0.19

Risetime	1.621ms
Vmax	3.24V
Freq	4.999Hz
Duty Cycle	25.00%
Falltime	1.600ms



กราฟสัญญาณ P0.20

Risetime	1.559ms
Vmax	3.12V
Freq	5.001Hz
Duty Cycle	25.01%
Falltime	1.579ms

กราฟสัญญาณ P0.21

Risetime	1.449ms
Vmax	3.44V
Freq	5.000Hz
Duty Cycle	25.11%
Falltime	1.863ms

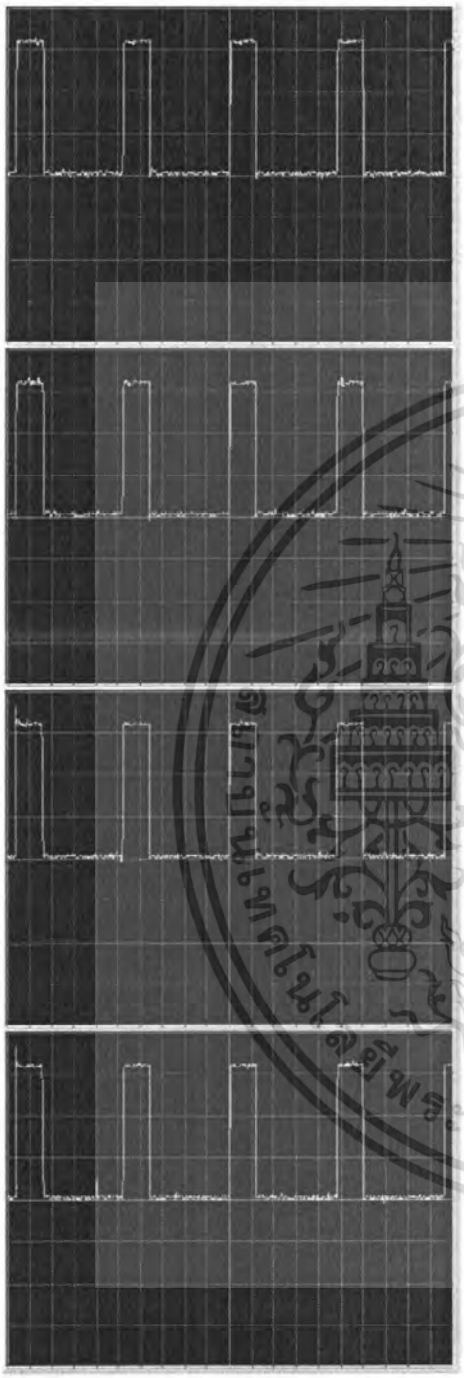
กราฟสัญญาณ P0.22

Risetime	1.621ms
Vmax	3.12V
Freq	4.999Hz
Duty Cycle	25.02%
Falltime	1.643ms

กราฟสัญญาณ P0.23

Risetime	1.519ms
Vmax	3.60V
Freq	4.999Hz
Duty Cycle	25.06%
Falltime	1.667ms

4.2 ที่ความเร็วระดับที่ 2



กราฟสัญญาณ P0.16

Risetime	810.3us
Vmax	3.28V
Freq	8.333Hz
Duty Cycle	24.98%
Falltime	760.9us

กราฟสัญญาณ P0.17

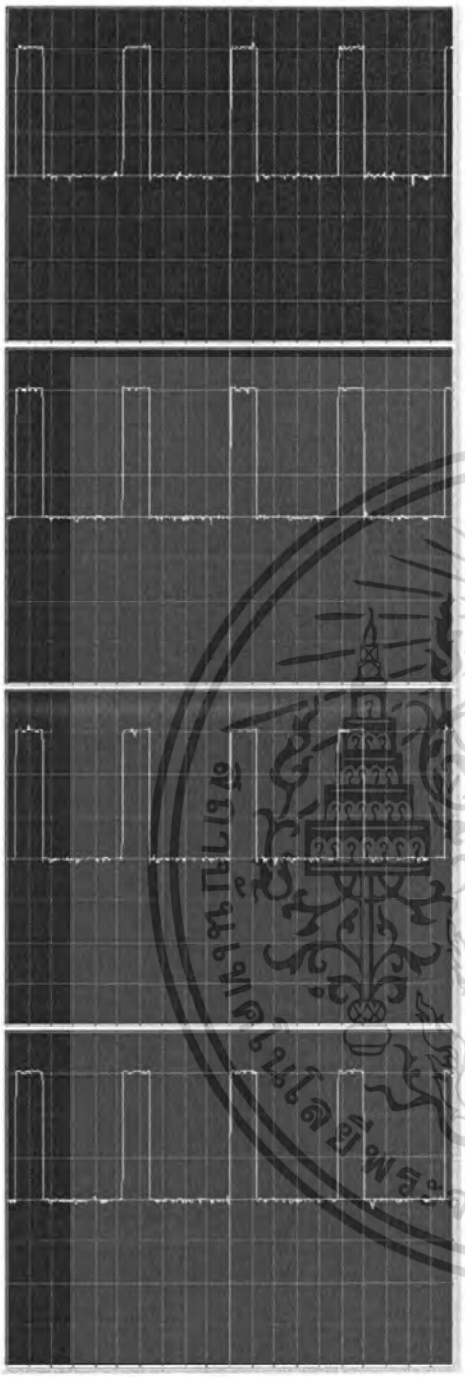
Risetime	831.5us
Vmax	3.28V
Freq	8.333Hz
Duty Cycle	24.96%
Falltime	761.4us

กราฟสัญญาณ P0.18

Risetime	780.0us
Vmax	3.28V
Freq	8.334Hz
Duty Cycle	25.00%
Falltime	779.9us

กราฟสัญญาณ P0.19

Risetime	810.2us
Vmax	3.28V
Freq	8.333Hz
Duty Cycle	24.99%
Falltime	800.0us



กราฟสัญญาณ P0.20

Risetime	800.0us
Vmax	3.16V
Freq	8.334Hz
Duty Cycle	24.98%
Falltime	760.0us

กราฟสัญญาณ P0.21

Risetime	769.6us
Vmax	3.12V
Freq	8.333Hz
Duty Cycle	25.00%
Falltime	789.6us

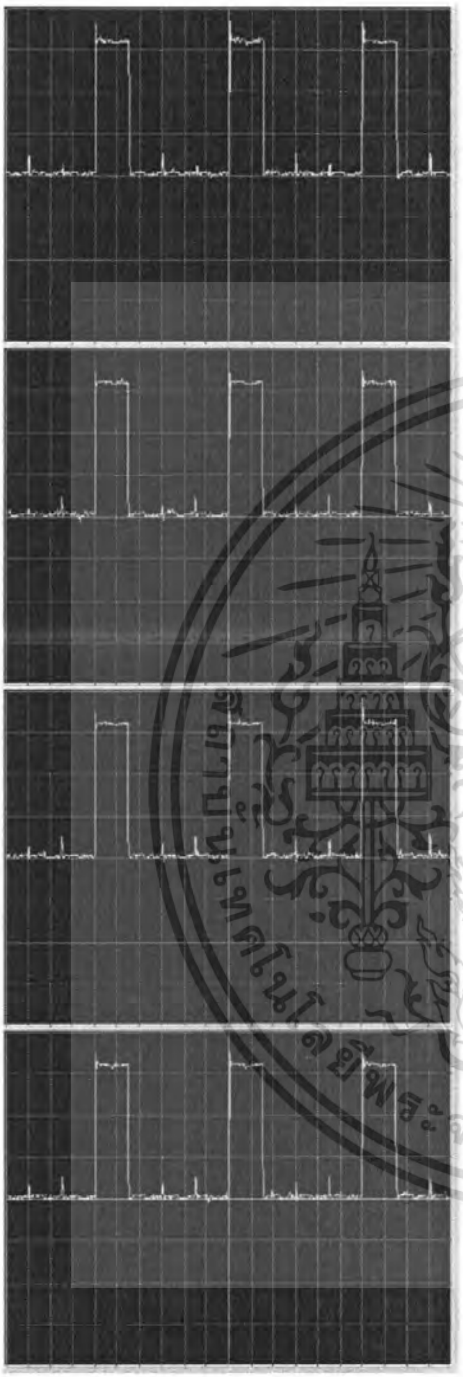
กราฟสัญญาณ P0.22

Risetime	810.5us
Vmax	3.12V
Freq	8.334Hz
Duty Cycle	24.99%
Falltime	770.0us

กราฟสัญญาณ P0.23

Risetime	821.3us
Vmax	3.08V
Freq	8.332Hz
Duty Cycle	25.01%
Falltime	810.5us

4.3 ที่ความเร็วระดับที่ 3



กราฟสัญญาณ P0.16

Risetime	280.8us
Vmax	3.68V
Frequency	16.67Hz
DutyCycle	25.40%
Falltime	337.0us

กราฟสัญญาณ P0.17

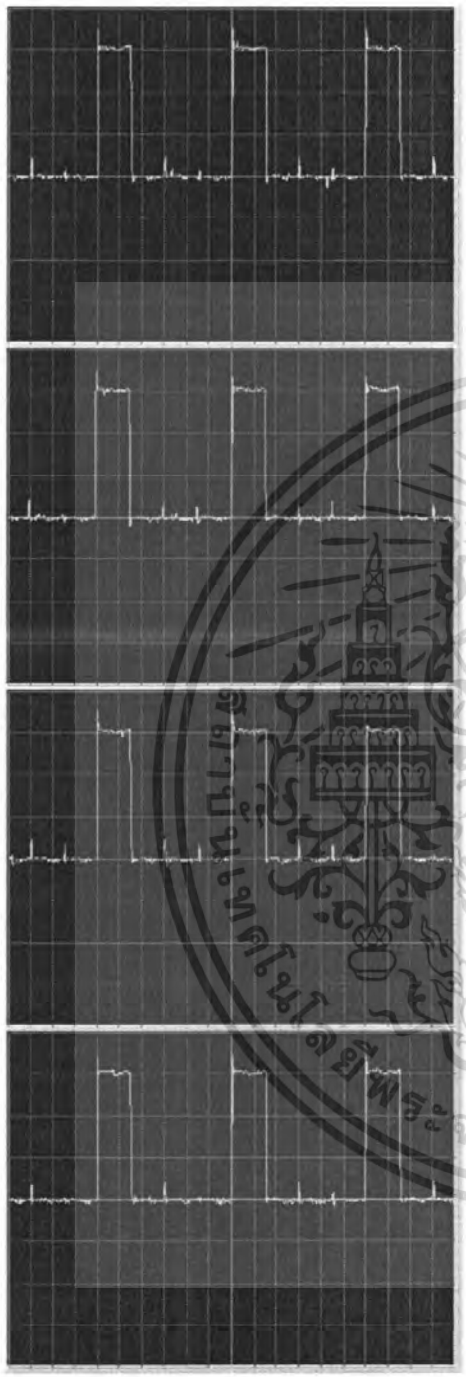
Risetime	300.9us
Vmax	3.44V
Frequency	16.67Hz
DutyCycle	24.74%
Falltime	429.4us

กราฟสัญญาณ P0.18

Risetime	290.2us
Vmax	3.48V
Frequency	16.67Hz
DutyCycle	24.75%
Falltime	436.9us

กราฟสัญญาณ P0.19

Risetime	304.7us
Vmax	3.48V
Frequency	16.67Hz
DutyCycle	24.76%
Falltime	476.0us



กราฟสัญญาณ P0.20

Risetime	279.5us
Vmax	3.48V
Frequency	16.67Hz
DutyCycle	25.40%
Falltime	337.7us

กราฟสัญญาณ P0.21

Risetime	296.8us
Vmax	3.32V
Frequency	16.55Hz
DutyCycle	24.59%
Falltime	328.5us

กราฟสัญญาณ P0.22

Risetime	286.5us
Vmax	3.48V
Frequency	16.67Hz
DutyCycle	25.44%
Falltime	362.2us

กราฟสัญญาณ P0.23

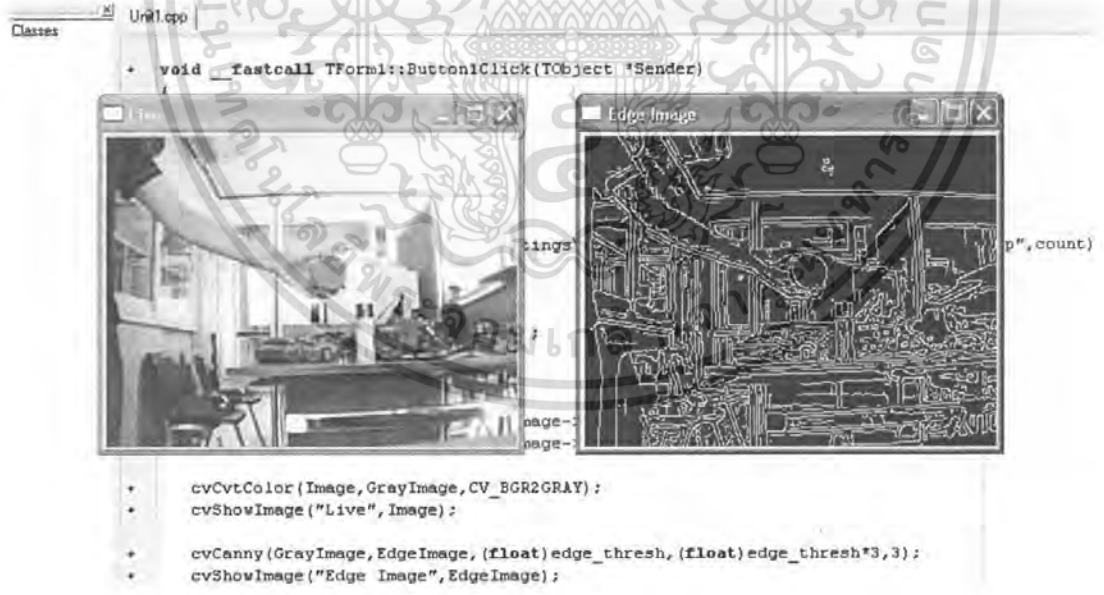
Risetime	282.7us
Vmax	3.44V
Frequency	16.67Hz
DutyCycle	25.39%
Falltime	333.1us

4.4 การทดลองส่วนโปรแกรมการประมวลผลภาพ

ในกาออกแบบโปรแกรมเกี่ยวกับส่วนประมวลผลภาพในโครงงานเล่มนี้ได้ทำการค้นคว้าข้อมูลเกี่ยวกับ อิมเมจ โปรเซสซิ่ง และการใช้งานโปรแกรม บอร์แลนด์ ซี++ บิวล์เดอร์ ทั้งจากหนังสือและเวปไซต์ที่เกี่ยวข้องจำนวนมาก เพื่อหาวิธีการออกแบบการทำงานของโปรแกรมและอัลกอริทึม (Algorithm) ของการเขียนซอร์ฟแวร์ แล้วได้นำมาปรับแต่งและพัฒนาให้เหมาะสมกับจุดประสงค์ของโครงงาน ดังนั้นโปรแกรมที่สร้างขึ้นมาเพื่อทดลองในบทนี้มีทั้งที่ถูกใช้และไม่ได้นำมาใช้ในโปรแกรมแต่ก็ถือว่าเกี่ยวข้องกับโครงงานทั้งสิ้น

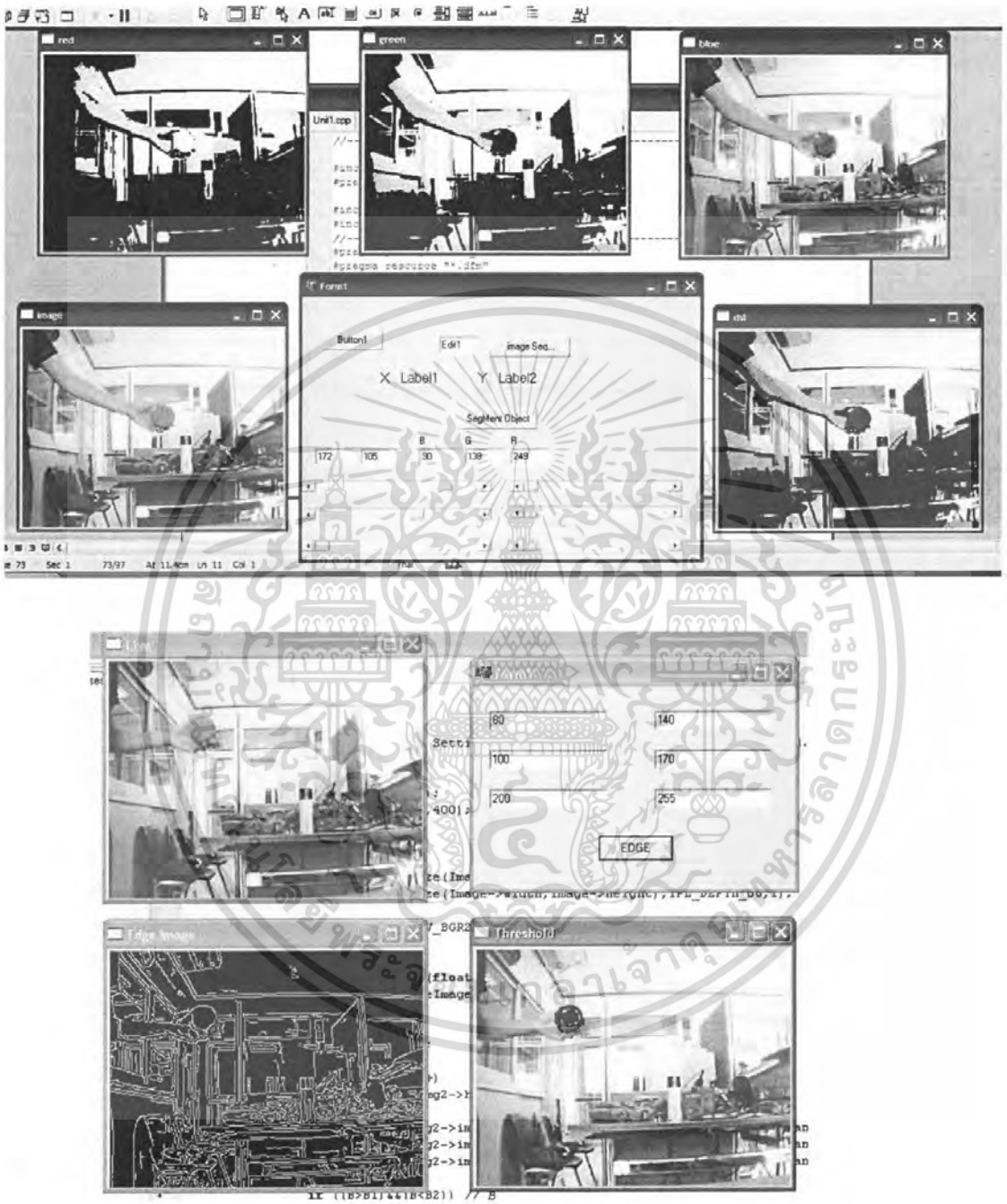
4.4.1 การทดลองหาขอบภาพ (Edge Detection)

ในอิมเมจโปรเซสซิ่งที่ต้องการให้โปรแกรมสามารถตรวจจับวัตถุ (Object Tracking) และหรือรู้จักวัตถุ (Object Recognition) จำเป็นอย่างยิ่งที่จะต้องใช้การหาขอบภาพเพื่อใช้หาลักษณะเด่นที่จำเป็นที่อยู่ในภาพก่อนการประมวลผล เพื่อเปรียบเทียบกับฐานข้อมูลหรือภาพต้นแบบ (Model Picture) แต่เนื่องจากว่าการตรวจจับในลักษณะนี้ จะต้องใช้การออกแบบโปรแกรมที่ซับซ้อนมากจนเกินความรู้ที่มีอยู่



รูปที่ 4.1 การทดลองหาขอบภาพ (Edge Detection)

4.4.2 การแยกสี (Color Segmentation)



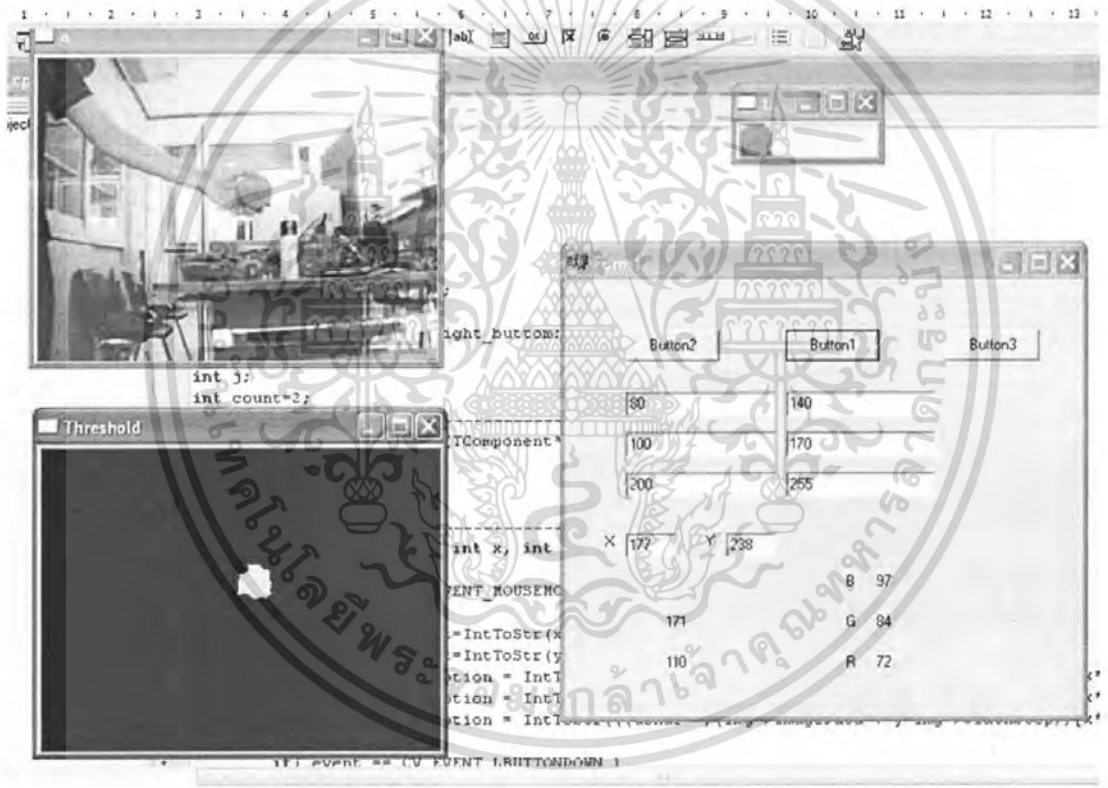
รูปที่ 4.2 การแยกสี (Colors Segmentation)

ก่อนที่จะทำการนำภาพไปแทรกกิ้งวัตถุ จะต้องปรับให้ภาพมีความเหมาะสมก่อนจะนำไปประมวลผลซึ่งขั้นตอนแรกก็คือการแยกส่วนสี เพื่อให้ภาพมีความแตกต่างกันระหว่างวัตถุแต่ละ

กลุ่มและวัตถุกับฉากหลังอย่างชัดเจนมากขึ้น โดยการกำหนดค่าเทรสโฮลด์สีนั้น แต่ละสีก็จะใช้ค่าเทรสโฮลด์ที่ต่างกันเพื่อให้เกิดความสมบูรณ์ของแต่ละสีให้มากที่สุด ซึ่งแสดงการแยกส่วนสีแดง สีเขียว และสีน้ำเงิน ตามลำดับ

4.4.3 การกรองสีและหาตำแหน่งวัตถุ (Color Filtration and Object Positioning)

หลังจากที่ได้ทำ Color Segmentation ไปแล้วและได้ปรับค่าเทรสโฮลด์ให้กับแต่ละสีอย่างเหมาะสมแล้ว ขั้นตอนต่อไปก็จะนำผลที่ได้มากรองเอาเฉพาะที่ต้องการมาประมวลผล เพื่อเก็บตำแหน่งศูนย์กลางของวัตถุสีเขียว เพื่อใช้ในการแทรกคิ้งในโปรแกรมย่อยลำดับต่อไป



รูปที่ 4.3 การกรองสีและหาตำแหน่งวัตถุ (Color Filtration and Object Positioning)

4.4.4 ผลการทดลองการกรองสีและการหาตำแหน่งของภาพ

ในการกรองสีในภาพแต่ละสีนั้นสามารถทำการกรองเฉพาะสีที่ต้องการออกมาได้ค่อนข้างได้ผลเป็นที่หน้าพอใจ แม้ว่าจะมีสัญญาณรบกวนหลุดออกมาปะปนกับภาพแต่ก็ไม่เป็นปัญหาในการหาตำแหน่งวัตถุของโปรแกรมแต่อย่างใด

รูปที่ 4.4 แสดงการหาตำแหน่งวัตถุ (Object Positioning)

บทที่ 5
สรุปและวิจารณ์ผล

5.1 สรุปผลการทดสอบ

ผลการทดสอบการขับสเต็ปปิ้งมอเตอร์

ในการทดสอบเราทำการวัดสัญญาณที่เอาท์พุทที่ได้จากไมโครคอนโทรลเลอร์อาร์ม 7 (Microcontroller ARM7) ซึ่งได้ทำการโปรแกรมไว้ โดยให้ พอร์ต P0.16-P0.23 เป็นพอร์ตสัญญาณเอาท์พุทที่นำไปขับสเต็ปปิ้งมอเตอร์ (Stepping Motor) ต่อไป โดยผ่านวงจรขยายกระแสด้วยทรานซิสเตอร์ที่นำมาต่อแบบดาร์ลิงตัน และแบ่งความเร็วออกเป็น 3 ระดับ

โปรแกรมที่เขียนได้กำหนดให้ เมื่อมีการกดปุ่มที่กำหนดจากคีย์บอร์ดคอมพิวเตอร์จะทำให้มีสัญญาณเอาท์พุตออก พอร์ต P0.16-P0.23 ตาทที่ได้กำหนด โดยที่ P0.16-P0.19 ใช้ขับขับสเต็ปปิ้งมอเตอร์ซ้ายและ p0.20- p0.23 ใช้ขับขับสเต็ปปิ้งมอเตอร์ ขวา ซึ่งในการขับสเต็ปปิ้งมอเตอร์จะส่งสัญญาณเข้าที่แต่ละเฟสโดยการไล่ลำดับเฟส ถ้าส่งข้ามลำดับจะทำให้มอเตอร์หมุนไม่ตรงตามที่ต้องการ ดังนั้นจึงทำการโปรแกรมให้สัญญาณเอาท์พุทออกจากพอร์ต P0.16-P0.23 เป็นPluse ในการทำงาน 1 รอบ จะส่งสัญญาณ 4 Pluse เรียงลำดับกัน ในการขับสเต็ปปิ้งมอเตอร์แต่ละตัว

จากผลการทดลองที่ได้จะเห็นว่า

1. ที่ความเร็วทั้ง 3 ระดับ จะให้สัญญาณออกจากพอร์ต P0.16-P0.23 ประมาณ 3.3 Volt ซึ่งเป็นไปตามความสามารถของไมโครคอนโทรลเลอร์อาร์ม 7 นี้

2. ความถี่ของสัญญาณทั้ง 3 ระดับ มีค่าต่ากัน โดยที่ speed 1 มีความถี่ของสัญญาณต่ำสุด และ speed 2 และ speed 3 มีความถี่ของสัญญาณสูงขึ้นตามลำดับซึ่งเป็นไปตามโปรแกรมที่ได้ออกแบบไว้

3. ค่า Duty Cycle ของสัญญาณที่ระดับความเร็วทั้ง 3 ระดับ มีค่าประมาณ25% เนื่องจากในการทำงาน 1 รอบ เราให้สัญญาณออกมี 4 Pluse ความถี่เท่าๆกัน ซึ่ง Duty Cycle วัดจาก ช่วงเวลาที่สัญญาณ ON ต่อช่วงเวลาทั้งหมดในหนึ่งรอบ (X 100%)

ผลการทดสอบการประมวลผลภาพ

จากการทดลองเขียนโปรแกรมประมวลผลภาพที่ได้ทดลองในบทที่ 4 นั้นเพื่อต้องการแสดงให้เห็นว่ามีขั้นตอนในการออกแบบโปรแกรมและเกิดผลอย่างไรบ้างเมื่อโปรแกรมทำงาน ซึ่งในการเขียนโปรแกรมนั้นจำเป็นอย่างยิ่งท่ะต้องพิสูจน์ว่าโปรแกรมที่ได้สามารถทำงานได้ตามที่ต้องการ ซึ่งเป็นส่วนที่สำคัญในการพัฒนาต่างๆต่อไป

ในการทดลองในบทที่ 4 ที่ผ่านมานั้น การหาขอบภาพ (Edge Detection) นั้นไม่ได้นำมาใช้งานในโครงงาน เนื่องจากว่าไม่จำเป็นต้องใช้ในการติดตามวัตถุแบบกรองสี แต่ได้นำมาแสดงไว้เพื่อเป็นแนวคิดในการพัฒนาต่อไป

ขั้นตอนการทดลองส่วนของโปรแกรมย่อยต่างๆที่ได้นำมาใช้งานในโปรแกรมประมวลผลภาพซึ่งก็ถือว่าตัวโปรแกรมนั้นสามารถทำงานได้เป็นอย่างดีไม่ว่าจะเป็นส่วนของโปรแกรมการแยกสี โปรแกรมกรองสีและหาตำแหน่งวัตถุ และสุดท้ายโปรแกรมควบคุมหุ่นและการตรวจจับวัตถุ ซึ่งผลที่ได้จากการทดลองออกมานับว่าอยู่ในเกณฑ์ที่ดีมาก

5.2 ปัญหาและแนวทางการแก้ปัญหา

1. สเต็ปปิ้งมอเตอร์ (Stepping Motor) มอเตอร์ที่ใช้ Load R ประมาณ 5-6 Ohm และเราใช้แหล่งจ่าย 12 Volt ดังนั้นสเต็ปปิ้งมอเตอร์ (Stepping Motor) จะดึงกระแส ประมาณ 2 A เราจึงทำการต่อวงจรขยายกระแสแบบ Darlington เพิ่มในวงจร และใช้ Heat Sink ในการระบายความร้อนให้ทรานซิสเตอร์เพื่อป้องกันทรานซิสเตอร์เสียหาย

2. ในตอนที่ทรานซิสเตอร์ On จะทำให้มีกระแสไหลผ่านสเต็ปปิ้งมอเตอร์ ซึ่งสเต็ปปิ้งมอเตอร์จะเสมือนเป็นตัวเหนี่ยวนำ เมื่อมีกระแสไหลผ่านในตอนแรกจะสะสมพลังงาน และเมื่อทรานซิสเตอร์ Off สเต็ปปิ้งมอเตอร์ จะทำหน้าที่เสมือนแหล่งจ่ายกระแส ดังนั้นหากไม่มีการป้องกันจะทำให้ ทรานซิสเตอร์เกิดความเสียหายจาก Back EMF ได้ ดังนั้นจึงต้องมีการต่อ Diode เพิ่มที่แต่ละเฟของมอเตอร์

3. ปัญหาการเขียนโปรแกรม เนื่องจาก Digital Image Processing เป็นกระบวนการที่เพิ่งเคยศึกษา จึงทำให้ต้องใช้เวลาในการศึกษาเป็นเวลานาน และตำราหนังสือที่เกี่ยวข้องก็หาค่อนข้างยากเฉพาะทาง

4. ปัญหาจากแสงรบกวนภายนอก ในระบบที่ต้องใช้การประมวลผลภาพนั้น จำเป็นอย่างยิ่งที่จะต้องมีแสงที่กล้องจับเข้ามานั้นเพียงพอและสม่ำเสมอ ไม่มากเกินไปจนเกิดการสะท้อนแสง ไม่เช่นนั้นแล้วภาพเมื่อนำไปประมวลผลแล้วจะทำให้การทำงานของระบบผิดพลาดได้ แก้ปัญหาโดยการจัดสภาพแสงและสิ่งแวดล้อมให้เหมาะสมเพื่อให้การทำงานถูกต้องมากที่สุด

5.3 ประโยชน์ที่ได้รับ

จากการศึกษาโครงงานนี้นับว่ามีประโยชน์อย่างยิ่ง ทั้งในด้าน

1. **การใช้งานARM7** เนื่องจากปัจจุบันไมโครคอนโทรลเลอร์ได้ถูกนำมาใช้งานอย่างแพร่หลายในชีวิตประจำวัน ดังนั้นการศึกษาโครงสร้าง การทำงาน และการใช้งานARM7 จึงเป็นประโยชน์อย่างยิ่งในการนำไปประยุกต์ใช้งานในอนาคตได้

2. **Stepping Motor** ความรู้พื้นฐานด้านการทำงานของStepping Motor ทำให้สามารถนำไปประยุกต์ ใช้ในงานประเภทต่างๆได้ ซึ่งจะเป็นประโยชน์ในอนาคตต่อไป

3. **Digital Image processing** เนื่องจาก งานด้านการประกอบวงจรทางอิเล็กทรอนิกส์ในปัจจุบัน เครื่องจักรกลระบบอัตโนมัติภายในโรงงานอุตสาหกรรม ต้องมีความสามารถ ความละเอียดและความแม่นยำที่สูงมาก และเนื่องจากการประมวลผลด้วยภาพถ่าย (Digital Image Processing) จะมีความยืดหยุ่นสูงกว่าในการปรับรูปแบบการใช้งาน และมีราคาถูกกว่าเมื่อเทียบกับวิธีอื่น ดังนั้นการศึกษาเรื่อง Digital Image processing จึงนับว่าเป็นประโยชน์อย่างยิ่งในการทำงานในอนาคต

หนังสืออ้างอิง

1. รศ.ดร.ชูชาติ ปิณฑวิรุจณ์, "การประมาวลผลภาพดิจิตอลด้วย Matlab DIGITAL IMAGE PROCESSING USING MATLAB ",แผนกตำรา คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ,2550,ภาคภนวก ค.
2. อรพิน ประวัติบริสุทธิ์, " คู่มือเรียนภาษาซี ",PROVISION กรุงเทพฯ,2547
3. โอภาส ศิริครรชิตถาวร, "เรียนรู้และพัฒนา ไมโครคอนโทรลเลอร์ARM7 LPC2148 ด้วยภาษาซี " ,วชิรินทร์สาส์น รัชดา กรุงเทพฯ, 2549
4. http://www.wara.com/modules.php?name=News&file=article&sid=259

ภาคผนวก

โปรแกรมส่วนของ Digital Image Processing

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CPort"
#pragma resource "*.dfm"
TForm1 *Form1;
static CvCapture*capture = NULL;
static IplImage *image,*red,*green,*blue,*dst,*image1,*image2,*image3,*image4 ;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
Cn=48;
}
//---------------------------------------------------------------------------
void mousemove1(int event, int x, int y, int flags, void* param)
{
    if(event==CV_EVENT_LBUTTONDOWN)
    {
     //Form1->Edit2->Text = IntToStr(x);
     //Form1->Edit3->Text = IntToStr(y);
     Form1->B  =((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels + 0]; // B
     Form1->G  =((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels + 1]; // G
     Form1->R  =((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels + 2]; // R
     Form1->Edit1->Text = IntToStr(Form1->B);
     Form1->Edit2->Text = IntToStr(Form1->G);
     Form1->Edit3->Text = IntToStr(Form1->R);
    }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Theshold()
{
if (p1==1)
{
    cvSplit(image, blue, green, red, NULL);
    cvThreshold( red, red, ScrollBar1->Position,255, CV_THRESH_TOZERO );
    cvThreshold( green, green, ScrollBar2->Position,255, CV_THRESH_TOZERO );
    cvThreshold( blue, blue, ScrollBar3->Position,255, CV_THRESH_TOZERO );
    cvMerge(blue, green, red, NULL, dst);
    cvShowImage("dst",dst);
}
```

```
int B;
int G;
int R;
if (p2==1)
{
for(int j=0;j<(dst->width);j++)
{
        for(int i=0;i<(dst->height);i++)
        {
        B=((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 0]; // B
        G=((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 1]; // G
        R=((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 2]; // R
        if ((B>=B1)&&(B<=B2)) // B
        {
            if((G>=G1)&&(G<=G2)) // G
            {
                if((R>=R1)&&(R<=R2)) // R
                {
                (((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 0])=255; // B
                (((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 1])=255; // G
                (((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 2])=255; // R
                }
                else
                {
                ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 0]=0; // B
                ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 1]=0; // G
                ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 2]=0; // R
                }
            }
            else
            {
            ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 0]=0; // B
            ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 1]=0; // G
            ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 2]=0; // R
            }
        }
        else
        {
        ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 0]=0; // B
        ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 1]=0; // G
        ((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 2]=0; // R
        }
        } //for
}
        cvShowImage("threshold",dst);
```

```
i_max=0,i_min=dst->height,j_max=0,j_min=dst->width;
for(int j=1;j<dst->width;j++)
{
        for(int i=1;i<dst->height;i++)
        {
        if ((((uchar *)(dst->imageData + i*dst->widthStep))[j*dst->nChannels + 0])==255) // B)
        {
                if ( i_min > i)
                { i_min = i ;}

                if ( i_max < i)
                { i_max = i ;}

                if ( j_min > j)
                { j_min = j ;}

                if ( j_max < j)
                { j_max = j ;}
        }
    }
}

cvRectangle (image,cvPoint(j_min,i_min),cvPoint(j_max,i_max),CV_RGB(0,255,0));
cvShowImage("Mask",image);
}
if (p4==1)
{
int a,b;
a=((j_max-j_min)/2)+j_min; // cen.width rec
b=((i_max-i_min)/2)+i_min; // cen.height rec
if(i_max=0)
{
Cn=48;
}
else
{
if ( dst->width/2 - a < -20)  // Right
{
Edit10->Text =IntToStr(55);
Cn=55;
}
else
{
    if ( dst->width/2 - a > 20)  // Left
    {Edit10->Text =IntToStr(57);
    Cn=57;
```

```
    }
    else
    {
        if ( dst->height/2 - b < -10)    // Forward
        {Edit10->Text =IntToStr(56);
        Cn=56;}

        else
        {
            if ( dst->height/2 - b > 10)    // Backward
            {Edit10->Text =IntToStr(50);
            Cn=50;}
            else
            {

                if ((i_max-i_min)-(imo-ino)>40)    // Backward
                            {
                                    if ((j_max-j_min)-(jmo-jno)>40)
                                    {
                Form1->Edit10->Text =IntToStr(50);
                Cn=50;
                io=1;
                }
                            }

                if ((imo-ino)-(i_max-i_min)>40)    // Forward
                {
                    if ((jmo-jno)-(j_max-j_min)>40)
                    {
                    Form1->Edit10->Text =IntToStr(56);
                    Cn=56;
                    io=1;
                    }
                }

                if (io!=1)
                {
                Form1->Edit10->Text =IntToStr(48);
                Cn=48;
                io=0;
                }
            }
        }
    }
}
}
}//p4
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
capture = cvCaptureFromCAM(-1);
cvNamedWindow("image",1);
cvMoveWindow("image",10,400);
ScrollBar1->Max = 255;
      ScrollBar1->Min = 0;
      ScrollBar2->Max = 255;
      ScrollBar2->Min = 0;
      ScrollBar3->Max = 255;
      ScrollBar3->Min = 0;
if(capture)
{
     for(;;)
     {

     if(!cvGrabFrame(capture))
               break;
      image = cvRetrieveFrame(capture);
      cvShowImage("image",image);
      Theshold();

      if(cvWaitKey(1)>=0)
          break;
     }
     cvReleaseCapture(&capture);
     cvDestroyWindow("image");
     cvDestroyWindow("mask");
     cvDestroyWindow("threshold");
     cvDestroyWindow("merge");
     cvReleaseImage(&image);
     cvReleaseImage(&red);
     cvReleaseImage(&blue);
     cvReleaseImage(&green);
     cvReleaseImage(&dst);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar1Change(TObject *Sender)
{
Theshold();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar2Change(TObject *Sender)
{
```

```
Theshold();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar3Change(TObject *Sender)
{
Theshold();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
p1=1;
      red   = cvCreateImage(cvSize(image->width,image->height), IPL_DEPTH_8U, 1);
      green = cvCreateImage(cvSize(image->width,image->height), IPL_DEPTH_8U, 1);
      blue  = cvCreateImage(cvSize(image->width,image->height), IPL_DEPTH_8U, 1);
      dst   = cvCreateImage(cvSize(image->width,image->height), IPL_DEPTH_8U, 3);

      cvNamedWindow("dst",1);
      cvMoveWindow("dst",360,400);
      cvSetMouseCallback("dst",mousemove1,0);

      cvNamedWindow("threshold",1);
      cvMoveWindow("threshold",710,400);

      cvNamedWindow("Mask",0);
      cvMoveWindow("Mask",360,50);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
R1=StrToInt(Form1->Edit8->Text);
R2=StrToInt(Form1->Edit9->Text);
G1=StrToInt(Form1->Edit6->Text);
G2=StrToInt(Form1->Edit7->Text);
B1=StrToInt(Form1->Edit4->Text);
B2=StrToInt(Form1->Edit5->Text);
p2=1;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
//--------------------Confix Value Pixel--------------------//
imo=i_max;
ino=i_min;
jmo=j_max;
jno=j_min;
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
p4=1;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
      ComPort1->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{
         Timer1->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button8Click(TObject *Sender)
{
     Timer1->Enabled = false;
     ComPort1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComPort1RxChar(TObject *Sender, int Count)
{
    int RxBytes;
            int BytesReceived;
            unsigned char *RxBuffer;
            RxBytes = ComPort1->InputCount();
            RxBuffer = new unsigned char [RxBytes];
            BytesReceived = ComPort1->Read(RxBuffer, RxBytes);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{   //Sleep(10);
    unsigned char cTmp;
    cTmp = Cn;
    ComPort1->Write(&cTmp,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button9Click(TObject *Sender)
{
  TTestThread* MyThread =new TTestThread(false);
}
//---------------------------------------------------------------------------
```