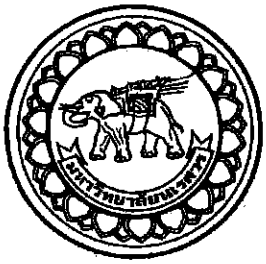
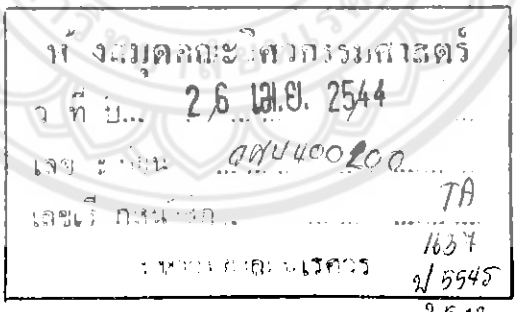


ระบบติดตามวัตถุด้วยกล้องดิจิตอล

Image Tracking System

นางสาวปาริชาติ อรุณอาศิรกุล รหัส 40360364

นายมุนิน ยศถา รหัส 40360455



โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2543

ใบรับรองโครงงานวิจัย

หัวข้อโครงงาน	:	ระบบติดตามวัตถุด้วยกล้องดิจิตอล
Performance	:	Image Tracking System
ผู้ดำเนินโครงงาน	:	นางสาวปาริชาติ อรุณอาศิรกุล รหัส 40360364
		นายมุนิน ยศถา รหัส 40360455
อาจารย์ที่ปรึกษา	:	อ. ปัญญา เหล่าอนันต์ธนา
อาจารย์ที่ปรึกษาร่วม	:	อ. มุฑิตา สงฆ์จันทร์
สาขา	:	วิศวกรรมคอมพิวเตอร์
ภาควิชา	:	วิศวกรรมไฟฟ้าและคอมพิวเตอร์

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิจัย

.....ประธานกรรมการ
(อาจารย์ปัญญา เหล่าอนันต์ธนา)

.....กรรมการ
(อาจารย์มุฑิตา สงฆ์จันทร์)

.....กรรมการ
(อาจารย์พนัส นัถฤทธิ์)

หัวข้อโครงงาน : ระบบติดตามวัตถุด้วยกล้องดิจิตอล
ผู้ดำเนินโครงงาน : นางสาวปาริชาติ อรุณอาศิรกุล รหัส 40360364
นายมุนิน ยศถา รหัส 40360455
อาจารย์ที่ปรึกษา : อ. ปัญญา เหล่าอนันต์ชนา
อาจารย์ที่ปรึกษาร่วม : อ. มุฑิตา สงฆ์จันทร์
สาขา : วิศวกรรมคอมพิวเตอร์
ภาควิชา : วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา : 2543

บทคัดย่อ

ในโครงงานฉบับนี้ได้ศึกษาถึงการใช้กล้องติดตามวัตถุโดยใช้โปรแกรม Delphi5 มาพัฒนาขึ้นเป็นโปรแกรมที่สามารถวิเคราะห์ภาพของวัตถุ โดยการที่จะวิเคราะห์ภาพของวัตถุได้นั้นต้องใช้ความรู้ทางด้าน Image Processing เพื่อใช้วิเคราะห์ภาพหาสี และขอบของภาพ จากนั้นก็หาจุดศูนย์กลางของภาพ หลังจากที่ได้ค่าจุดศูนย์กลางของภาพแล้วก็จะนำมาเปรียบเทียบหาระยะการขจัดที่เกิดขึ้นเมื่อเทียบกับจุดศูนย์กลางของจอภาพ ค่าระยะการขจัดนี้จะถูกส่งไปยังโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ที่พัฒนาขึ้นโดยโปรแกรม C51 มีหน้าที่แปลงเป็นตัวเลข 0 หรือ 1 จำนวน 4 บิต ส่งออกที่พอร์ตRS-232C ไปยังบอร์ดไดเวอร์สเต็ปเปอร์มอเตอร์ซึ่งจะขับตัวสเต็ปเปอร์มอเตอร์ให้หมุนซ้ายหรือขวาจำนวนกี่สเต็ป ซึ่งทำให้กล้องสามารถติดตามวัตถุได้

จากการทำโครงงานนี้สามารถที่จะนำไปประยุกต์ใช้ประโยชน์ได้มากมายทั้งในภาคอุตสาหกรรมโรงงาน และอื่นๆ เช่น ตรวจเช็คการติดฉลากบนขวดยาในขณะที่ขวดยาเลื่อนไปตามสายพาน หรืออาจนำไปประยุกต์ใช้กับแขนกลในการหยิบจับวัตถุที่กำลังเคลื่อนที่ เป็นต้น

Project Title	:	Image Tracking System
Name	:	Ms.Parichat Arunaresirakul ID. 40360364
		Mr.Munin Yostha ID. 40362455
Project Advisor	:	Mr.Panya Loawanuntana
Co- Project Advisor	:	Ms.Muthita Songchan
Field of Study	:	Computer Engineering
Department	:	Electrical and Computer Engineering
Academic Year	:	2000

Abstract

This project adopted in the book is that image processing is the process of an image, typically by a computer, to produce another image, while computer vision is about image acquisition, processing, classification, recognition, and to be all embracing, decision making subsequent to recognition as in.

Single static camera, Movement in scene is detected by analysis of a sequence of images. If either the real size or distance of the object is known then the other can be calculated. Speed can be calculated if a sequence of image is avilable and either the size or distance of the object is known.

The most difficult moving problems are encountered when the vehicle is moving an object it is encountering are also moving, all in different directions. This is well illustrated by the fact that it is often most difficult for a walking human to explain the reasoning behind taking a particular route through a crowded concourse.

กิตติกรรมประกาศ

รายงานโครงงานฉบับนี้สำเร็จลุล่วงโดยได้รับความอนุเคราะห์เป็นอย่างดีจากอาจารย์ปัญญา เหล่าอนันต์ชนา อาจารย์ที่ปรึกษา อาจารย์มุฑิตา สงฆ์จันทร์อาจารย์ที่ปรึกษาร่วม และอาจารย์ผู้ดูแลชีอป ผู้ได้แนะนำหัวข้อ แก้ไข ปรับปรุง ตลอดจนให้กำลังใจแก่ผู้ดำเนินโครงงานจนกระทั่งรายงานโครงงานนี้สำเร็จโดยสมบูรณ์ ผู้ดำเนินโครงงานขอขอบพระคุณทุกท่านมาด้วยความรู้สึกซาบซึ้งยิ่ง

และขอกราบขอบพระคุณ พ่อ แม่ พี่น้องและเพื่อนๆที่ได้ให้การสนับสนุน และช่วยเหลือในด้านต่างๆเกี่ยวกับการเรียน ตลอดจนครูอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ให้ตลอดมา

นางสาวปาริชาติ อรุณอาศิรกุล

นายมุนิน ยศถา

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

ในบทแรกนี้จะเป็นบทนำที่จะกล่าวถึงลักษณะของโครงงานนี้โดยรวม เพื่อให้เข้าใจลักษณะของโครงงานนี้ในเบื้องต้น หากผู้อ่านมีความสนใจต้องการความรู้เพิ่มเติมอีกก็สามารถอ่านต่อในบทอื่นๆได้ ซึ่งจะได้ความรู้และแนวคิดต่างๆที่เกี่ยวข้องกับโครงงานนี้อย่างแน่นอน

1.1 ที่มาและความสำคัญของโครงงาน

โครงงานนี้ได้แนวคิดมาจากความสนใจในการเล่นกล้องดิจิตอลเพื่อเก็บภาพถ่ายแล้วตกแต่งภาพ เพื่อให้เกิดประโยชน์มากขึ้นจึงนำมาประยุกต์เป็นกล้องที่สามารถติดตามวัตถุที่กำลังเคลื่อนที่ได้ โดยโครงงานนี้เป็นการควบคุมกล้องดิจิตอลให้สามารถติดตามวัตถุได้ โดยการควบคุมผ่านทางโปรแกรมคอมพิวเตอร์ที่พัฒนาขึ้นจากโปรแกรม Delphi และควบคุมการเคลื่อนที่ของกล้องด้วยการหมุนของสเต็ปเปอร์มอเตอร์

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการส่งภาพของกล้องดิจิตอลเข้าสู่โปรแกรม Delphi ในคอมพิวเตอร์
2. เพื่อศึกษาการใช้งานโปรแกรม Delphi แบบประยุกต์
3. เพื่อพัฒนากล้องดิจิตอลให้มีความสามารถเพิ่มขึ้น
4. เพื่อพัฒนาโปรแกรมให้ควบคุมสเต็ปเปอร์มอเตอร์ด้วย C51
5. เพื่อให้เป็นแนวทางในการพัฒนาความสามารถของโครงงานต่อไปในอนาคต
6. เพื่อนำโครงงานนี้ไปใช้ให้เกิดประโยชน์ร่วมกับโครงงานอื่นๆได้

1.3 ขอบข่ายของโครงงาน

1. วัตถุที่ใช้เป็นตัวล่อให้กล้องเคลื่อนที่ต้องเป็นลูกปิงปองที่มีสีที่ตัดกับฉาก เช่น ถ้าฉากเป็นพื้นสีดำลูกปิงปองควรเป็นสีที่จางๆหรือสีสว่าง สีที่ใช้อาจเป็นสีขาว สีฟ้า สีเหลือง สีเขียว สีชมพู เป็นต้น หรือถ้าฉากเป็นพื้นสีขาวลูกปิงปองก็ควรเป็นสีเข้มหรือสีมืดๆ
2. มีการควบคุมสภาพแวดล้อมขณะมีการจับภาพ
3. ความเร็วของวัตถุมีขีดจำกัดที่ระดับหนึ่ง ซึ่งขึ้นอยู่กับปัจจัยทางด้านฮาร์ดแวร์

4. การติดตามวัตถุของกล้องจะสามารถติดตามได้ในแนวระดับเท่านั้น

1.4 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 แสดงขั้นตอนการดำเนินงานของโครงงาน

กิจกรรม	เดือน-ปี						
	มี.ค.43	เม.ย.43	พ.ค.43	มิ.ย.43	ก.ค.43	ส.ค.43	ก.ย.43
1. ออกแบบและทำตัวเครื่อง	← →						
2. ศึกษาเรื่อง Image Processing และ สเต็ปเปอร์มอเตอร์	←	—	—	—	→		
3. ออกแบบวงจรควบคุมสเต็ปเปอร์มอเตอร์ และเขียนโปรแกรมควบคุมการทำงานของสเต็ปเปอร์มอเตอร์ด้วย C51		←	—	—	—	→	
4. เขียนโปรแกรมเพื่อวิเคราะห์ภาพจากวัตถุ ด้วย Delphi5				←	—	—	→
5. ทดสอบและแก้ไขการทำงานของระบบ						←	→

1.5 ผลที่คาดว่าจะได้รับ

1. ได้ความรู้เกี่ยวกับการใช้งานกล้องดิจิตอลร่วมกับระบบคอมพิวเตอร์
2. ได้ความรู้เกี่ยวกับการใช้งานโปรแกรม Delphi ทางด้านกราฟฟิก
3. เพิ่มความสามารถของกล้องดิจิตอลธรรมดาตัวหนึ่ง ซึ่งนอกจากจะใช้ถ่ายภาพ ถ่ายวิดีโอ ผ่านทางอินเตอร์เน็ตได้แล้วยังนำมาใช้ในการติดตามวัตถุที่ต้องการได้อีกด้วย
4. ได้ฝึกคิด ฝึกแก้ปัญหาต่างๆที่เกิดขึ้นจากการทำงาน

1.6 งบประมาณที่ต้องใช้

1.	กล้องดิจิตอล	3,300	บาท
2.	บอร์ดวงจรควบคุมสเต็ปเปอร์มอเตอร์	400	บาท

3.	สเต็ปเปอร์มอเตอร์	100	บาท
4.	ฐานวางกล้อง	250	บาท
5.	ฉากหลัง ,ลูกปิงปอง	70	บาท
6.	เอกสารที่ใช้ประกอบ	850	บาท
	รวมเป็นเงินทั้งสิ้น	4,970	บาท

บทที่ 2
หลักการและทฤษฎี

ก่อนที่เริ่มลงมือทำโครงงานได้นั้น เราควรศึกษาหาความรู้ทั้งหมดที่เกี่ยวข้อง เพื่อจะได้เข้าใจโครงงานมากขึ้น และการดำเนินงานของเรานั้นจะง่ายมากขึ้นด้วย

2.1 พื้นฐานทั่วไปของสเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์เป็นมอเตอร์อีกชนิดหนึ่งที่ถูกนำมาใช้งานกันมาก ไม่ว่าจะเป็นส่วนประกอบในหุ่นยนต์ เครื่องจักรกลอัตโนมัติควบคุมด้วยคอมพิวเตอร์ ต่างก็ต้องใช้สเต็ปเปอร์มอเตอร์เป็นตัวขับเคลื่อน ตัวอย่างใกล้ตัวที่มีให้เห็น เช่น ฟลอบปี้ไดรฟ์ ฮาร์ดดิสก์ไดรฟ์ภายในคอมพิวเตอร์ ซึ่งก็ใช้สเต็ปเปอร์มอเตอร์ขนาดเล็กในการขับเคลื่อนหัวอ่านเขียนข้อมูลลงไปในพื้นผิวของแผ่นดิสก์เพื่อให้ตรงกับตำแหน่งของแทร็กที่ถูกต้อง

2.1.1 ชนิดของสเต็ปเปอร์มอเตอร์

การแบ่งชนิดของสเต็ปเปอร์มอเตอร์หากแบ่งตามโครงสร้างพื้นฐานหรือดควววามแตกต่างของรูปแบบโรเตอร์จะแบ่งออกได้ 4 ชนิด แต่ถ้าแบ่งตามวิธีการพันขดลวดบนสเตเตอร์จะแบ่งออกได้ 2 ชนิด อย่างไรก็ตามการแบ่งในลักษณะนี้เพียงยึดหลักที่ว่าหาซื้อง่ายและนิยมนำมาใช้งานกัน

1. แบ่งตามโครงสร้างพื้นฐาน

- ชนิดวาริเอเบิลรีลักแตนซ์ (Variable Reluctance ;VR)

มีโครงสร้างโรเตอร์แบบมัลติทูธ(multi-tooth) ทำจากเหล็กอ่อน จะทราบว่าเป็นมอเตอร์ชนิดนี้โดยการทดสอบได้ง่ายมาก คือใช้มือหมุนเพลามอเตอร์ และสังเกตเห็นว่าหมุนได้ตลอดโดยไม่ติดขัด เพราะโรเตอร์จะไม่เกิดปรากฏการณ์ทางแม่เหล็กแตกต่างจากชนิดPM และชนิดไฮบริด ซึ่งมีสนามแม่เหล็กที่โรเตอร์ ขณะหมุนจะรู้สึกขัดๆเหมือนเป็นฟันเฟือง ชนิดนี้มีจุดด้อยในความถูกต้องของตำแหน่งและทำงานได้ไม่ดีนักเมื่อมีสเต็ปการหมุนสูง

- ชนิดเพอร์มาเนนต์แม็กเน็ต (Permanent Magnet ; PM)

มีโครงสร้างของโรเตอร์แบบเรียบไม่มีซี่ขั้วแม่เหล็ก บนโรเตอร์จะเป็นแบบแม่เหล็กถาวร การควบคุมทำได้โดยป้อนกระแสกระตุ้นที่ขดลวดบนสเตเตอร์ เช่นถ้าเป็นสเตเตอร์แบบ 4 เฟส จะมีขั้วแม่เหล็กอยู่ 4 ขั้ว ซึ่งมีคอยล์พันแยกจากกัน ขั้วแม่เหล็กถาวรบนโรเตอร์จะถูกแรงดึงดูดจากขั้วแม่เหล็กบนสเตเตอร์เมื่อป้อนกระแสไฟฟ้าเข้าสู่ขดลวด โรเตอร์จะอยู่คงที่ที่ขั้วแม่เหล็กบนสเตเตอร์นั้น ถึงแม้ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป ทำให้เกิดเป็นแรงยึดหน่วงขึ้น ชนิดนี้มีข้อดีในความถูกต้องของตำแหน่ง แม้ความเร็วจะมากขึ้นเมื่อเปรียบเทียบกับชนิดอื่น

- ชนิดไฮบริด (Hybrid)

เป็นชนิดที่นิยมใช้งานกันมากที่สุด โดยเฉพาะนำมาใช้กับอุปกรณ์ในเครื่องคอมพิวเตอร์ โครงสร้างได้จากการรวมเอาโครงสร้างของโรเตอร์ชนิดวาริเอเบิลรีลักแตนซ์และชนิดเพอร์มาแนนต์แมกเน็ตมาประกอบเข้าด้วยกัน จึงทำให้เป็นมอเตอร์ชนิดที่มีแรงยึดหน่วงสูง มีแรงบิดดึงและผลักได้ดีซึ่งมีความคงที่และทำงานได้ดีถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง

- ชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต

เป็นสเต็ปเปอร์มอเตอร์แบบใหม่อีกชนิดหนึ่งปรับปรุงมาจากชนิดเพอร์มาเนนต์แมกเน็ต มีโครงสร้างของโรเตอร์เป็นแผ่นยึดติดกับเพลามอเตอร์ มีโมเมนต์ความเฉื่อยต่ำ อัตราเร่งสูง แรงบิดดึงกำลังทางกลและความถูกต้องของตำแหน่งสูงมาก ความเร็วเริ่มหมุนและหยุดสูง สูญเสียพลังงานต่ำ ชนิดนี้มีชื่อเรียกอีกอย่างว่า ดิสก์แมกเน็ตสเต็ปเปอร์มอเตอร์ (disc magnet steppers)

2. แบ่งตามการพันขดลวดบนสเตเตอร์

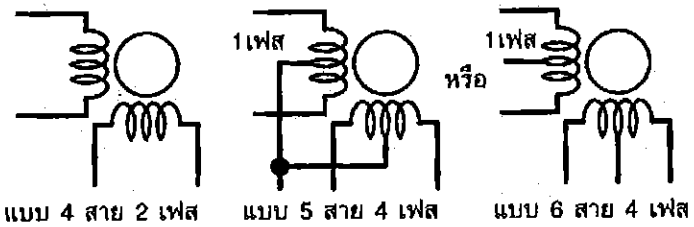
- แบบไบโพลาร์ (Bipolar)

มีการพันขดลวด 1 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ ขั้วแม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้า และทำให้เกิดขั้วแม่เหล็กในทิศทางการไหลและการกลับทิศการไหลของกระแสไฟฟ้า ทำได้โดยการใช้วงจรสวิตชิ่งกลับขั้วไฟฟ้า

- แบบยูนิโพลาร์ (Unipolar)

มีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ ซึ่งแต่ละขดจะทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามกัน

การพิจารณาว่าสเต็ปเปอร์มอเตอร์ตัวใดมีการพันขดลวดแบบใดสังเกตได้ง่าย ถ้าเป็นแบบไบโพลาร์จะมีสายไฟต่อออกจากมอเตอร์เพียง 4 สาย และถ้าเป็นแบบยูนิโพลาร์จะมี 5-6 สายหรือดูได้จากป้ายชื่อที่ติดอยู่กับมอเตอร์ก็ได้



(ก) แบบไบโพลาร์ (ข) แบบยูนิโพลาร์

รูปที่ 2.1 ลักษณะการพันขดลวดบนสเตเตอร์

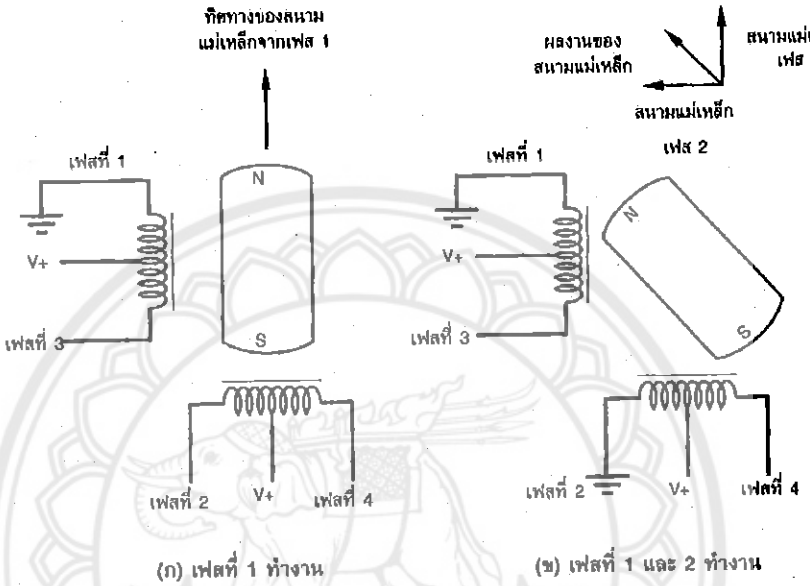
2.1.2 การทำงานของสเต็ปเปอร์มอเตอร์

การทำงานของสเต็ปเปอร์มอเตอร์มีความแตกต่างจากมอเตอร์ทั่วๆไป โดยเมื่อป้อนกำลังไฟฟ้าให้กับตัวมันก็จะหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด แต่มอเตอร์ทั่วไปจะหมุนทันทีและตลอดเวลา สเต็ปเปอร์มอเตอร์สามารถกำหนดตำแหน่งการหมุนด้วยตัวเลขได้อย่างละเอียด โดยการใช้คอมพิวเตอร์เป็นตัวกำหนดและจัดเก็บตัวเลขเหล่านั้นไว้

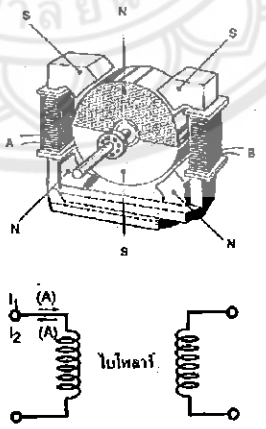
สเต็ปเปอร์มอเตอร์สามารถทำงานในระบบเปิด (Open loop system) แต่วิธีการกำหนดตำแหน่งให้ถูกต้องนั้นจำเป็นต้องมีการป้อนกลับไปยังระบบให้รับรู้ โดยทั่วไปจะใช้การสวิตซ์ติดตั้งไว้ที่ตำแหน่งที่ต้องการตรวจจับ (Limit switch) เมื่อสเต็ปเปอร์มอเตอร์เริ่มหมุนจนกระทั่งถึงตำแหน่งของสวิตซ์ตรวจจับสัญญาณ ก็จะถูกป้อนกลับเข้าสู่ระบบและทราบการทำงานของสเต็ปเปอร์มอเตอร์ได้ตลอดเวลา ซึ่งโดยปกติในวงจรคอนโทรลเลอร์จะมีการกำหนดจุดอ้างอิงไว้ด้วย เพื่อให้เริ่มต้นทำงานและอ้างอิงตำแหน่งได้อย่างถูกต้อง

เพื่อให้เข้าใจมากขึ้นมาดูหลักการทำงานแบบง่ายๆของสเต็ปเปอร์มอเตอร์แบบยูนิโพลาร์ 4 เฟส ตัวโรเตอร์เป็นแม่เหล็กโดยจะเปลี่ยนทิศทางไปตามสนามแม่เหล็ก การให้พลังงานแก่ขดลวดใดขดลวดหนึ่งโรเตอร์ก็จะหมุนไป 90 องศา ดังรูปที่ 2.2 (ก) แต่ถ้าให้ทีเดียว 2 ขดพร้อมกัน โรเตอร์ก็จะหมุนเพียง 45 องศา ดังรูปที่ 2.2 (ข) ซึ่งแบบหลังจะสร้างแรงบิดได้มากว่าแบบแรก สเต็ปเปอร์มอเตอร์จะมีมุมของการเคลื่อนที่แต่ละสเต็ปเป็น 1.8 องศา ดังนั้นที่โรเตอร์จะต้องมีขั้วแม่เหล็ก 50 ขั้ว (90/50 = 1.8) สเต็ปเปอร์มอเตอร์ 4 เฟส ความจริงแล้วเรียกชื่อยังไม่ถูกต้องนัก น่าจะเรียกว่าเป็น

แบบ2 เฟส มากกว่า ถึงแม้ว่าขดลวดจะมี 4 ขดก็ตาม แต่การทำงานของเฟสที่ 3 และ 4 มีค่าเท่ากับเฟสที่ 1หรือ 2 การที่มี 4 ขด ก็เพื่อให้ง่ายต่อการควบคุม เพียงใช้เพาเวอร์ทรานซิสเตอร์เป็นสวิตซ์ 4 ตัวก็ได้

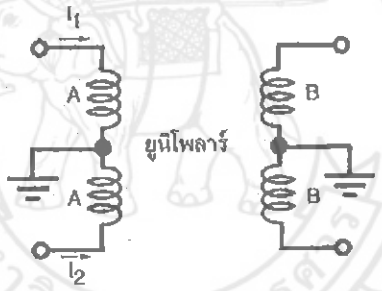


รูปที่ 2.2 ทิศทางการหมุนโรเตอร์ของสเต็ปเปอร์มอเตอร์ 4 เฟส



รูปที่ 2.3 ไบโพลาร์สเต็ปเปอร์มอเตอร์แบบ 2 เฟส
สนามแม่เหล็กจะเปลี่ยนเมื่อกลับทิศทางการไหลของกระแส

ส่วนในรูปที่ 2.3 เป็นการพันขดลวดแบบไบโพลาร์ เมื่อขดลวด A และ B มีกระแสไหลผ่าน สเตเตอร์จะเกิดขั้วแม่เหล็ก เป็นผลให้โรเตอร์ที่มีขั้วแม่เหล็กต่างกับสเตเตอร์ถูกดูด ต่อมาเมื่อกระแสที่ไหลในขดลวด A เปลี่ยนทิศทางกลับ จึงเป็นผลให้ขั้วแม่เหล็กที่แกน A เปลี่ยนจากขั้ว S เป็นขั้ว N จากขั้ว N เป็นขั้ว S สลับกัน โรเตอร์จึงถูกผลักให้หมุนทวนเข็มไป 90 องศา เครื่องหมาย ขีด (-) บนอักษร A, B แทนการกลับขั้ว จะสังเกตได้ว่าเมื่อกลับขั้วแม่เหล็กในแต่ละเฟสจะต้องมีการหยุดกระแสก่อนแล้วกระแสจึงค่อยเปลี่ยนทิศทาง จึงสรุปเป็นสเต็ปได้คือ AB, B AB, A, B, AB, A, AB การทำงานเป็นแบบครึ่งสเต็ปนี้ เป็นผลให้ค่าโมเมนต์มีค่าน้อยกว่าปกติ เพราะมีช่วงเวลาที่กระแสไหลแค่เฟสเดียว ส่วนแบบยูนิโพลาร์ก็คล้ายกับแบบไบโพลาร์โดยติดขดเดียว ในแต่ละเฟสของยูนิโพลาร์จะมีแทปกลางซึ่งจะแบ่งเป็น 2 ขด ดังรูปที่ 2.4 เป็นผลให้ค่าฟลักซ์แม่เหล็กมีค่าน้อยกว่าไบโพลาร์ ดังนั้นเมื่อสนามแม่เหล็กเปลี่ยนแปลงการระแสจะไม่เปลี่ยนทิศทางการไหล สนามแม่เหล็กที่ได้ก็น้อยตาม แรงบิดที่ขึ้นกับสนามแม่เหล็กก็น้อยกว่าด้วย



รูปที่ 2.4 ยูนิโพลาร์สเต็ปเปอร์มอเตอร์
การเปลี่ยนขั้วแม่เหล็กเกิดจากระแสไหลต่างขดกัน
กระแสจะไม่ไหลพร้อมกัน 2 ขดในสเตเตอร์เดียวกัน

2.1.3 การควบคุมการหมุนของสเต็ปเปอร์มอเตอร์

การกระตุ้นเพื่อควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปในการใช้งานจริง ทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้อง แบ่งออกได้ 3 รูปแบบ คือ

1. **แบบเวฟ (Wave)** เป็นการกระตุ้นรูปแบบที่ง่ายที่สุด โดยกระตุ้นขดลวดทีละขดใน

เวลาหนึ่งและเรียงถัดกันไป เช่น ขดที่ 1, 2, 3, 4, 1 หรือ 1, 4 , 3, 2, 1 ขึ้นอยู่กับทิศทางที่ต้องให้หมุน ดังนั้นจึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งเท่านั้นที่ถูกกระตุ้น วงจรกระตุ้นแบบเวฟมีขั้นตอนการทำงานต่างๆ ดังแสดงในตารางที่2.1

ตารางที่ 2.1 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบเวฟ

สเต็ปที่	เฟสที่1	เฟสที่2	เฟสที่3	เฟสที่4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

2. **แบบ 2 เฟส (Two Phase)** เป็นการกระตุ้นอีกรูปแบบหนึ่งซึ่งคล้ายกับแบบเวฟ แต่จ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไปเช่นเดียวกับแบบเวฟ คือ 12, 23, 34, 41, 12 หรือ 14, 43, 32, 21, 14 ขึ้นอยู่กับทิศทางการหมุน การเพิ่มจำนวนของขดลวดที่ถูกกระตุ้นนี้ทำให้เพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดที่ถูกกระตุ้นพร้อมกัน และต่อด้วยแรงดึงจากอีก 2 ขดถัดไป สำหรับข้อเสียของการกระตุ้นแบบนี้ ต้องใช้แหล่งจ่ายกำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆแสดงดังในตารางที่ 2.2

ตารางที่ 2.2 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบ 2 เฟส

สเต็ปที่	เฟสที่1	เฟสที่2	เฟสที่3	เฟสที่4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

3. **แบบครึ่งสเต็ป (Half Step)** เป็นรูปแบบผสมผสานระหว่างการกระตุ้นแบบเวฟและแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกหนึ่งเท่าตัว จะกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ ขดลวด 1, 12, 2, 23, 3, 34, 4, 41, 1 หรือในการหมุนอีกทิศทางหนึ่งจะได้เป็น 1, 14, 4, 43, 3,

32, 2, 21, 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลงและแต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องระวังไว้ว่าเมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2 สเต็ปจึงจะได้เท่ากับ 1 สเต็ปเต็มเหมือนกับในการควบคุม 2 แบบแรก สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้เทียบเท่ากับแบบ 2 เฟส จึงจะเพียงพอ ขั้นตอนการทำงานต่างๆแสดงดังในตารางที่ 2.3

ตารางที่ 2.3 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบครึ่งสเต็ป

สเต็ปที่	เฟสที่1	เฟสที่2	เฟสที่3	เฟสที่4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	-	-	-	ทำงาน

2.2 มาตรฐาน RS-232C

การที่จะทำให้อุปกรณ์จากผู้ผลิตต่างกันสามารถติดต่อสื่อสารกันแบบอะซิงโครนัสได้นั้นจะต้องมีอุปกรณ์ที่เป็นขั้วต่อมาตรฐาน ซึ่งมีมากมายหลายชนิดด้วยกัน โดยมาตรฐานที่ใช้กันอย่างกว้างขวางคือ มาตรฐาน RS-232C ซึ่งกำหนดลักษณะของสัญญาณทางไฟฟ้าที่ถูกใช้ในการเชื่อมต่อแบบอนุกรมโดยตรง มีเพียง 2 ลักษณะคือ Binary 0 หรือแรงดันไฟฟ้าบวก และBinary 1 หรือแรงดันไฟฟ้าลบ

การอินเตอร์เฟส RS-232C มีอยู่ 2แบบด้วยกันคือ โดยพื้นฐานแล้วหากเป็นส่วนที่ใช้ต่อเข้ากับอุปกรณ์ประมวลผลจะถูกเรียกว่า DTE (Date Terminal Equipment) และส่วนที่ติดต่อกับอุปกรณ์สื่อสารจะเรียกว่า DCE (Data Circuitterminating Equipment) โดยทั่วไปจะใช้ขั้วต่อ BD25ตัวผู้สำหรับต่อกับ DTE ส่วน DCE ใช้ขั้วต่อ BD25ตัวเมีย

การทำงานของ RS-232C

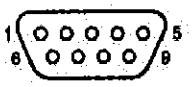
ส่วนการอินเตอร์เฟสทางลอจิกของพอร์ต RS-232C ออกแบบให้การสื่อสารมีความยาวไม่เกิน 50 ฟุตที่ความเร็ว 20,000 บิตต่อวินาที (สามารถใช้งานที่ความยาวสายสื่อสารมากกว่านี้ หรือที่ความเร็วสูงกว่านี้ หรือทั้งสองอย่างได้ แต่ตามมาตรฐานแล้วกำหนดไว้ที่ 50 ฟุต) การสื่อสารจะติดต่อโดยผ่านสายไฟ 25 เส้นซึ่งแยกหน้าที่แตกต่างกันออกไป พอร์ต RS-232C สามารถรองรับการสื่อสารได้ทั้งแบบซิงโครนัสและแบบอะซิงโครนัส หากใช้ในการสื่อสารแบบอะซิงโครนัสแล้วเส้นควบคุมบางเส้นใน 25 เส้นนี้ไม่จำเป็นต้องใช้

ส่วนสำคัญของพอร์ต RS-232C คือ Flow Control หรือส่วนควบคุมการไหลของข้อมูล ตัวอย่างเช่น เครื่องพริ้นเตอร์ส่งสัญญาณออกมาว่า " หยุด บัฟเฟอร์เต็ม รอสักครู่จะแจ้งกลับไปภายหลัง " หลังจากนั้น พริ้นเตอร์ก็จะพิมพ์งานในบัฟเฟอร์ต่อไป แล้วจะแจ้งกลับไปว่า " พร้อมที่จะรับข้อมูลต่อไปแล้ว ส่งข้อมูลเข้ามาได้ " เป็นต้น

RS-232C มีการจัดขา 2 แบบ คือ 9 และ 25 ขา แต่ในโครงงานนี้ได้ใช้แบบ 9 ขา ซึ่งมีการจัดการขาและรายละเอียดฟังก์ชันการทำงานดังนี้

ตารางที่ 2.4 รายละเอียดฟังก์ชันการทำงานของคอนเน็กเตอร์ของพอร์ต RS-232C แบบ 9 ขา

ขาที่	ลักษณะการทำงาน	คำอธิบาย
1	Input	Data Carrier detect (DCD)
2	Input	Receive Data (RX)
3	Output	Transmit Data (TX)
4	Output	Data Terminal Ready (DTR)
5	GND	Signal Ground
6	Input	Data Set Ready (DSR)
7	Output	Request To Send (RTS)
8	Input	Clear To Send (CTS)
9	Input	Ring Indicator (RI)



รูปที่ 2.5 แสดงการจัดการขาคอนเน็กเตอร์ของพอร์ต RS-232C แบบ 9 ขา

2.3 ET-EM PLUS (ET-EPROM EMULATOR)

ET-EM PLUS คือ บอร์ดวงจรที่ถูกสร้างขึ้นมาเพื่อใช้แทนส่วนของตัว EPROM หรือ RAM ซึ่งจะมีความง่ายและสะดวกกว่าการใช้ EPROM หรือ RAM จริงๆ ทำให้เหมาะกับการพัฒนาระบบไมโครต่างๆหรือในการแก้ไขเปลี่ยนแปลงโปรแกรมการทำงานของเครื่องเป็นไปโดยง่าย

เราสามารถเขียนข้อมูลเข้าไปในส่วนของ ET-EM PLUS ได้โดยตรงซึ่งต่างกับการที่เราต้องนำ EPROM นั้นออกมาเขียนเปลี่ยนแปลงแล้วจึงนำกลับเข้าไปใส่ในวงจรใหม่

งานที่นำไปใช้

ET-EM PLUS สามารนำไปใช้งานได้หลายรูปแบบขึ้นอยู่กับผู้ใช้จะนำไปใช้งาน ยกตัวอย่างเช่น

1. ROM MONITOR ใช้ในการเขียนโปรแกรมในส่วนของ MONITOR PROGRAM ของเครื่องไมโครต่างๆ เพราะในการเริ่มต้นพัฒนาระบบไมโครอะไรก็ตามนั้นยังไม่มีส่วนของโปรแกรมในการเขียนอ่านข้อมูลจากภายนอก ซึ่งถ้านำ ET-EM PLUS มาใช้ก็จะสะดวกกว่าใช้ EPROM เป็น MONITOR จริงๆมาก

2. CHARACTER GENERATOR เราสามารถใช้ ET-EM PLUS ในการทำเป็น CHARACTER GENERATOR ในการเปลี่ยนแปลงหรือเพิ่มเติมตัวอักษรเข้าไป เช่น ทำภาษาไทยในเครื่องพิมพ์ ซึ่งเราสามารถทดสอบเขียนแล้วทดสอบทำงานได้โดยตรง

3. BIOS เป็น ROM BIOS ในเครื่อง PC ใช้ทดสอบหรือแก้ไขตัวโปรแกรม BIOS ของเครื่องใช้หรือใช้ทดสอบเขียนโปรแกรมตรวจสอบการทำงานเป็นส่วนๆ ของเครื่องก็ได้ ฯลฯ

คุณสมบัติของ ET-EM PLUS

1. สามารถส่งผ่านข้อมูลจากเครื่อง PC ได้ทาง PRINTER PORT ทำให้ง่ายในการจะหาเครื่อง PC มาใช้ เพราะเครื่อง PC ต่างๆนั้นจะมีส่วนของ PRINTER PORT อยู่แล้ว

2. ใช้กับไฟล์ได้หลายรูปแบบ เช่น BINARY FILE , INTEL HEX FILE , MOTOLORA FILE (S FORMAT)

3. สามารถเปลี่ยนแปลงข้อมูลใน ET-EM PLUS ทั้งหมดหรือบางไบต์ก็ได้โดยไม่จำเป็นต้องโหลดไฟล์ใหม่หมด

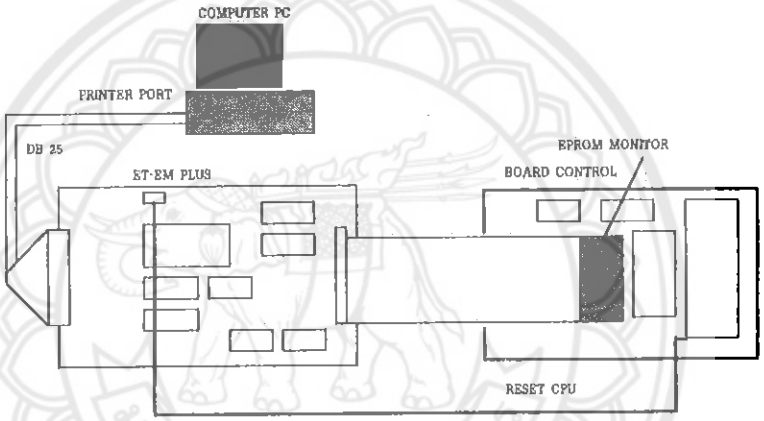
4. สามารถตั้ง OFFSET คือค่าตำแหน่งที่อยู่ของข้อมูลจากไฟล์ได้โดยตรงอย่างอิสระ

5. สามารถเลือกรูปแบบการส่งข้อมูลจากไฟล์ได้หลายแบบ เช่น ส่งเต็มข้อมูล, ส่งเฉพาะไบต์คู่ , ส่งข้อมูลเฉพาะไบต์คี่ ซึ่งทำให้เราสามารถนำ ET-EM PLUS 2 ตัวมาต่อกับ PRINTER PORT 2 ชุด ใช้พัฒนาระบบเครื่องที่จำเป็นต้องใช้ข้อมูล 16 บิต

6. สามารถต่อกับ PRINTER PORT ได้ทั้ง LPT1 , LPT2 หรือ MONO CARD ได้โดยอิสระต่อกัน

7. มีความเร็วในการส่งผ่านข้อมูลสูงมาก

8. สามารถ RESET CPU ได้เมื่อโหลดข้อมูลแล้ว โดยมีแบบ RESET HIGH และ RESET LOW



รูปที่ 2.6 แสดงการเชื่อมต่อ ET-EM PLUS เข้ากับ PRINTER PORT ของเครื่อง PC

2.4 CP-SB31 (SINGLE BOARD 31 ON PC)

CP-SB31 ถูกสร้างขึ้นเพื่อใช้งานควบคุม ซึ่งตรงกับหน้าที่หลักของ CPU ในตระกูล MCS51 คือเป็นไมโครคอนโทรลเลอร์ โครงสร้างทางกายภาพของบอร์ด CP-SB31 มีดังนี้

ลักษณะของบอร์ด CP-SB31

- CPU 8031 (ON BOARD) หรือ 8032 , 8052 , 8751

MEMORY

- มี SOCKET ขนาด 28 PIN 2 ตัว สามารถใส่หน่วยความจำได้สูงสุด 96 KB

I/O

- 8x3 บิต INPUT /OUTPUT (8255)

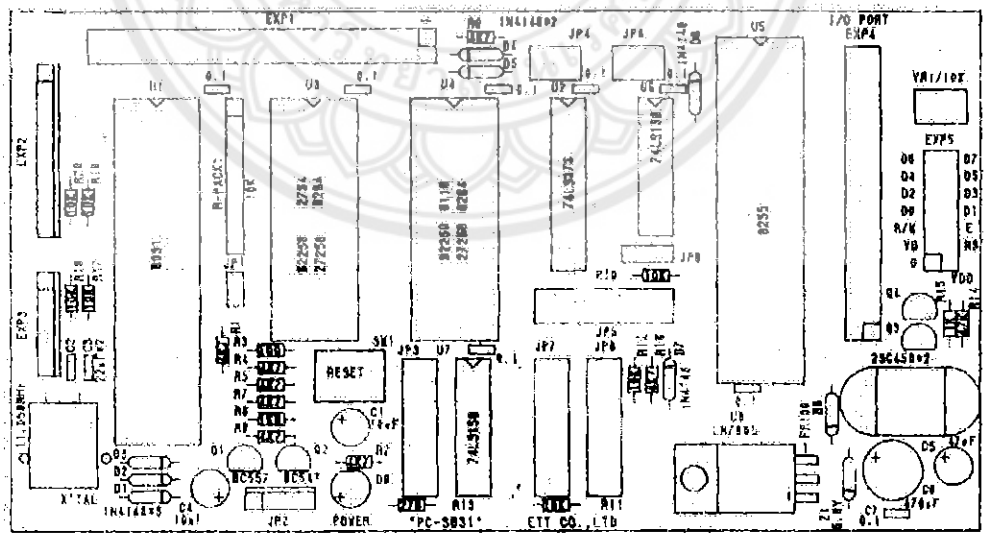
- 8x1 บิต INPUT /OUTPUT (PORT1)
- 1 SERAIL PORT (RS-232)

POWER

- 10 VDC POWER SUPPLY JACK
- 5 VDC (REGULATE) 7805 ON BOARD

คุณลักษณะพิเศษของ CP-SB31

1. หน่วยความจำสามารถเลือกได้ทั้งขนาด , ตำแหน่ง และลักษณะการทำงาน (DATA MEMORY , CODE MEMORY , CODE&DATA MEMORY)
2. สามารถพัฒนาโปรแกรมได้ทั้งภาษาแอสเซมบลี้ (ร่วมกับ SB31-DEBUGGER) หรือภาษาเบสิค (เพื่อใช้ 8052 AH-BASIC) หรือ ET EPROM EMULATOR ก็ได้
3. ต่อกับ LCD ได้ทันที โดยไม่ต้องใช้ I/O พอร์ต
4. มี I/O พอร์ต ขนาด 8 บิต ถึง 4 พอร์ต
5. ต่อร่วมกับอุปกรณ์สนับสนุนของบริษัทอีทีที ได้ทันที เช่น SSRAC , RTC , 72IO , ET-AD ฯลฯ



รูปที่ 2.7 แสดงส่วนประกอบของ CP-SB31

2.5 พื้นฐานทั่วไปเกี่ยวกับรูปภาพ

2.5.1 ชนิดของรูปภาพทั่วไป

รูปภาพสามารถแบ่งตามวิธีการจัดเก็บได้เป็น 2 ชนิดใหญ่ๆดังต่อไปนี้

รูปภาพแบบเวคเตอร์ (Vector Graphic) เป็นรูปภาพที่ไม่ขึ้นกับความละเอียดของภาพ เนื่องจากภาพชนิดนี้ถูกสร้างขึ้นจากสมการเส้นต่างๆ ไม่ว่าจะเป็นเส้นตรง เส้นโค้ง เมื่อเราทำการย่อขยายรูปภาพแบบนี้ คอมพิวเตอร์จะทำการคำนวณรูปภาพใหม่ทำให้ภาพคมชัดเสมอ ตัวอย่างของรูปภาพแบบนี้ที่เห็นได้ชัด คือ รูปภาพที่สร้างจากโปรแกรม Adobe Illustrator ,CorelDRAW, MacroMedia Freehand เป็นต้น

รูปภาพแบบบิตแมป (Bitmap Image) เป็นรูปภาพที่เกิดจากจุดเล็กๆประกอบกันขึ้นมาจนเห็นเป็นภาพขึ้นมา คุณภาพของรูปภาพชนิดนี้จะขึ้นอยู่กับความละเอียด หากภาพมีความละเอียดมากก็ยิ่งชัดเจนมากขึ้น เมื่อเราทำการย่อขยายรูปภาพ คอมพิวเตอร์จะทำการขยายภาพขึ้นด้วยความละเอียดที่มีอยู่ทำให้ภาพที่ได้มีลักษณะหยาบ ตัวอย่างของรูปภาพแบบนี้ คือ รูปภาพที่สร้างจากโปรแกรมAdobe Photoshop , CorelPHOTO Paint เป็นต้น

2.5.2 ความละเอียดของภาพ

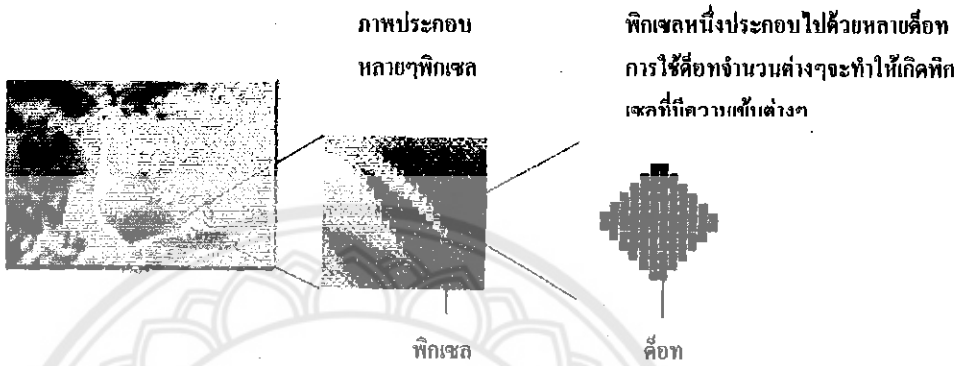
ความละเอียดของภาพ เป็นสิ่งที่บอกถึงคุณภาพของภาพนั้น หน่วยที่เรานิยมจะใช้บอกถึงความละเอียดของภาพคือ พิกเซลต่อนิ้ว (Pixel / Inch) ค่านี้บอกให้เราทราบว่าภาพมีจำนวนพิกเซลกี่พิกเซลในหนึ่งนิ้ว และคุณยังสามารถคำนวณหาจำนวนจุดทั้งหมดของภาพได้อีกด้วย ตัวอย่างเช่น ภาพขนาด 1*1 นิ้วที่มีความละเอียดเท่ากับ 8 พิกเซลต่อนิ้ว ภาพนี้จะมีพิกเซลทั้งหมดเท่ากับ 64 พิกเซล

การเลือกใช้ภาพที่มีความละเอียดสูงๆนั้นเป็นสิ่งที่ดี แต่การใช้ความละเอียดมากกว่าอุปกรณ์แสดงผล เราจะไม่สามารถใช้ประโยชน์จากความละเอียดที่เพิ่มขึ้นมานั้น อีกทั้งยังทำให้เครื่องคอมพิวเตอร์ทำงานช้าลงอีกด้วย วิธีการเลือกความละเอียดที่ถูกต้องคือ เลือกความละเอียดตามอุปกรณ์แสดงผลที่คุณใช้งาน

2.5.3 พิกเซล (Pixel) และดอท(Dot)

พิกเซล คือ จุดเล็กที่สุดของภาพ พิกเซลหนึ่งสามารถแสดงได้หลายสี ส่วนดอทจะเป็น

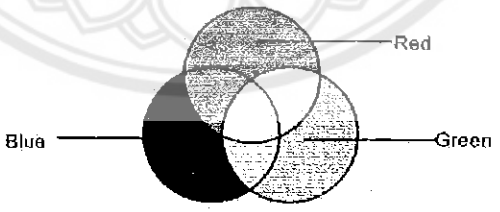
จุดเล็กที่สุดที่ใช้ในกระบวนการพิมพ์ภาพ การสร้างพิกเซลขึ้นมาหนึ่งพิกเซลจะต้องใช้ดือทหลายดือท เพื่อทำให้เกิดภาพความเข้มและสีต่างๆกัน หน่วยดือทต่อนิ้ว(dpi) จะใช้บอกความละเอียดของเครื่องพิมพ์ ส่วนหน่วย พิกเซลต่อนิ้ว(ppi) จะใช้บอกความละเอียดของเครื่องสแกน และจอภาพ



รูปที่ 2.8 แสดงลักษณะของพิกเซลและดือท

2.5.4 โหมดสีแบบ RGB

RGB Color เป็นโหมดที่ใช้แชนแนลสีจำนวน 3 สีคือ แดง , เขียว , น้ำเงิน โดยแต่ละสีจะมีการไล่ลำดับสีได้ถึง 256 ระดับ เมื่อรวมกันทั้ง 3 สีจะสามารถแสดงสีได้สูงถึง 16.7 ล้านสี (2 ยกกำลัง 24 บิต) โหมดนี้เป็นโหมดที่เหมาะสำหรับใช้ในการตบแต่งสีเพราะสามารถแทนสีได้มาก และยังเป็นโหมดสีเดียวกับที่ใช้ในจอมอนิเตอร์อีก



รูปที่ 2.9 โหมดสีแบบ RGB

2.6 ส่วนประกอบที่สำคัญของโครงงานมีดังนี้

1. กล้องดิจิตอล ทำหน้าที่เหมือนเซนเซอร์คอยจับภาพของวัตถุที่เคลื่อนที่

2. สเต็ปเปอร์มอเตอร์ ทำหน้าที่เป็นตัวขับเคลื่อนให้กล้องหมุนซ้ายขวาได้
3. ไดเวอร์สเต็ปเปอร์มอเตอร์ ใช้ควบคุมตัวสเต็ปเปอร์มอเตอร์ให้หมุนได้ตามข้อมูลที่ส่งมาจากคอมพิวเตอร์
4. คอมพิวเตอร์ เป็นตัวประมวลผลข้อมูลต่างๆที่เกี่ยวข้อง
5. ไมโครคอนโทลเลอร์8051 เป็นตัวเชื่อมต่อระหว่างโปรแกรมของไดเวอร์สเต็ปเปอร์มอเตอร์ กับโปรแกรมที่ใช้หาระยะของจุดศูนย์กลาง ณ เวลานั้นให้ทำงานประสานกัน

2.7 วิธีการเลือกอุปกรณ์ที่ใช้ให้เหมาะสมกับงาน

การเลือกใช้อุปกรณ์ที่เหมาะสม จะทำให้ได้ภาพของวัตถุที่กำลังเคลื่อนที่มีลักษณะใกล้เคียงกับวัตถุจริง ดังนั้นเราควรคำนึงถึงปัจจัยเหล่านี้

1. ความเร็วของวัตถุต้องสัมพันธ์กับความเร็วของกล้องดิจิตอล
2. การเคลื่อนที่ของวัตถุ อาจทำให้เกิดข้อผิดพลาดของภาพที่จับได้ อันมีสาเหตุมาจากสิ่งแวดล้อมรอบๆวัตถุ
3. การวิเคราะห์การเคลื่อนที่ระหว่างจุดสองจุดของภาพที่ต่างกัน ควรกำหนดการตอบสนองในเวลาตามลำดับ
4. พื้นที่ในการค้นหาตำแหน่งของวัตถุนั้นควรเป็นพื้นที่เล็กๆ เนื่องจากการค้นหา ณ เวลาจริงค่อนข้างจะทำได้ยากและใช้เวลามาก

2.8 สิ่งแวดล้อมที่มีผลต่อการจับภาพของวัตถุ

จากการวิเคราะห์ภาพของวัตถุโดยวิธีการแยกสีเพื่อหาขอบของวัตถุนั้น จำเป็นต้องมีการกำหนดสิ่งแวดล้อมต่าง ๆ เพื่อให้เกิดความผิดพลาดน้อยที่สุด โดยการกำหนดสีของฉากและวัตถุให้มีสีตัดกันซึ่งจะทำให้ได้ขอบของภาพที่ชัดเจนยิ่งขึ้น และมีการให้แสงในมุมที่เหมาะสมกับตำแหน่งของวัตถุ เพื่อไม่ให้เกิดภาพที่เพี้ยนไปจากภาพจริงของวัตถุมากนัก

2.9 หลักการทำงานของโปรแกรมที่ใช้ในการวิเคราะห์ภาพของวัตถุ

ในส่วนโปรแกรมการวิเคราะห์ภาพนั้นเราได้ใช้ความรู้ทางด้านอิมเมจเข้ามาช่วยในการวิเคราะห์หาขอบของภาพมาประยุกต์กับการเขียนโปรแกรมทางด้านกราฟฟิกของ Delphi โดยมีขั้น

ตอนการทำงาน ดังนี้

1. เริ่มต้นการทำงานโดยรับอินพุตจากกล้องดิจิตอล ซึ่งเป็นภาพของวัตถุ
2. จับภาพของวัตถุ แล้วทำการประมวลผลภาพที่ได้โดยแปลงภาพบิตแมปที่ได้ให้เป็นภาพในโหมดสีดำ-ขาว ซึ่งจะทำให้เกิดขอบของภาพชัดเจนขึ้น
3. ทำการหาตำแหน่งของขอบ และคำนวณหาจุดศูนย์กลาง โดยใช้วิธีหาตำแหน่งของขอบทางด้านซ้าย และตำแหน่งของขอบทางด้านขวาแล้วนำมาบวกกันหาร 2 ก็จะได้จุดศูนย์กลางของภาพ (เนื่องจากโครงงานนี้ติดตามวัตถุเพียงแกนเดียวเท่านั้น จึงใช้วิธีการคำนวณแบบนี้ ซึ่งถ้าหากต้องการพัฒนาเป็นหลายแกนก็ไม่ยากสำหรับผู้ที่สนใจ) ซึ่งจะถูกเก็บเป็นค่าจุดศูนย์กลางเก่า
4. ทำซ้ำในข้อ 2 อีก และเก็บค่าจุดศูนย์กลางที่ได้เป็นค่าของจุดศูนย์กลางใหม่
5. ทำการคำนวณหาการขจัดที่ได้โดยนำ (ค่าของจุดศูนย์กลางใหม่ – ค่าจุดศูนย์กลางเก่า) จะได้ค่าค่าหนึ่ง ซึ่งถ้าค่าที่ได้มีค่าเป็นบวกแสดงว่าวัตถุเคลื่อนที่ไปทางขวา แต่ถ้าค่าที่ได้เป็นค่าลบแสดงว่าวัตถุเคลื่อนที่ไปทางซ้าย แล้วเปลี่ยนค่าจุดศูนย์กลางใหม่ เป็นจุดศูนย์กลางเก่า
6. ส่งค่าที่ได้ไปที่โปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ แล้วทำซ้ำในข้อที่ 4 เรื่อยไปจนกว่าจะยกเลิกการทำงาน

บทที่3
วิธีการดำเนินงาน

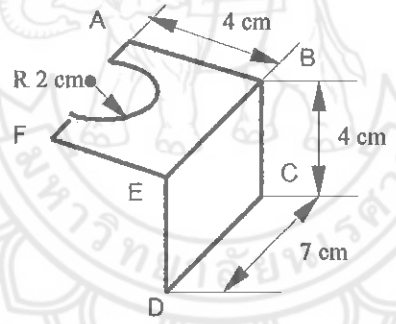
จากบทนำเราได้วางแผนการดำเนินงานไว้มาในบทนี้จะเป็นวิธีการดำเนินงานทั้งหมดที่ทำในโครงงานนี้ ซึ่งจะกล่าวละเอียดมาขึ้นกว่าเดิม และทำให้เข้าใจโครงงานนี้มากขึ้นด้วย

3.1 ทำฐานวางกล้องด้วยอลูมิเนียม

1. ทำการออกแบบโดยคำนึงถึงการใช้งานที่เหมาะสม กล่าวคือเราต้องการฐานที่มั่นคงสามารถต่อกับตัวกล้องได้อย่างเหมาะสม และสมดุล เมื่อกล้องหมุนด้วยแรงที่เกิดจากมอเตอร์

2. ขั้นตอนการทำฐานกล้อง

- นำอลูมิเนียมฉากมาตัดเป็นชิ้น 4 ชิ้น โดยแต่ละชิ้นมีขนาดเป็นดังรูป

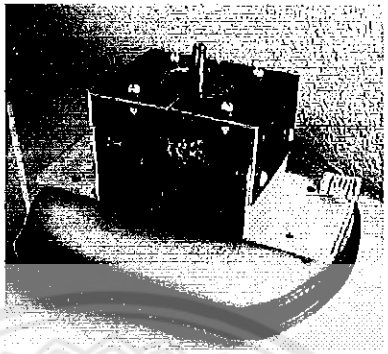


รูปที่ 3.1 แสดงชิ้นส่วนของอลูมิเนียมที่จะนำมาประกอบเป็นฐาน

- หลังจากตัดอลูมิเนียมได้ตามแบบแล้วก็ทำการตกแต่งลบคมตามขอบ และมุมรอบๆชิ้นงาน ทุกแผ่น

- ทำการเจาะรู ที่แผ่นอลูมิเนียมทั้ง 4 มุม(A, C, D, F) โดยรูที่มุมAจะมีระยะห่างขอบAF เท่ากับ 2.5 ซม. และห่างจากด้านAB 1 ซม. รูที่มุม F ก็เช่นกันห่างจากด้าน AF เท่ากับ 2.5 ซม. และห่างจาด้าน FE 1 ซม. รูที่มุมC ห่างจากด้าน BC และ CD ด้านละ 1 ซม. ส่วนรูที่มุม D ก็เช่นกับมุม C โดยห่างจากด้าน ED และ CD ด้านละ 1 ซม. โดยขนาดของรูมีเส้นผ่านศูนย์กลาง 0.5 ซม.

- นำน็อต (ตัวผู้และตัวเมีย) ยาว 1.5 ซม. จำนวน 8 ตัว มาใช้ในการประกอบตัวฐานกล้องให้ติดกัน โดยตัวฐานวางกล้องนี้จะติดกับตัวสเต็ปเปอร์มอเตอร์ด้วย ดังรูปที่3.2



รูปที่ 3.2 ฐานวางกล้อง

3. ทำการกลึงเหล็กสวมบริเวณหัวมอเตอร์ เพื่อเป็นขากล้องให้มีความยาว 4 ซม. และเจาะรูให้ทะลุทั้งสองด้านโดยรูนี้สูงขึ้นมา 0.5 ซม.จากปลายเหล็กสวม ประโยชน์ของรูนี้ใช้เป็นตัวล็อคขากล้องกับตัวมอเตอร์ และที่ปลายอีกด้านหนึ่งทำเป็นเกลียวนอกสูง 2 ซม. และสวมน็อตตัวเมียไว้ด้วยเพื่อใช้ล็อคตัวกล้องให้ติดกับขากล้อง (ข้อควรระวังคือ หากความสูงของขากล้องสูงเกินไปอาจทำให้ฐานกล้องล้มได้ง่ายเมื่อเกิดแรงดึงจากมอเตอร์)

3.2 เลือกสเต็ปเปอร์มอเตอร์

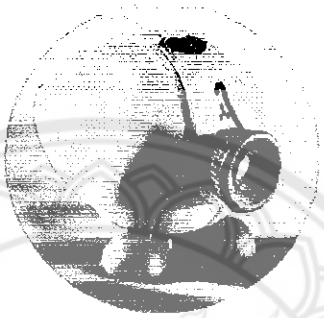
การเลือกใช้สเต็ปเปอร์มอเตอร์ควรเลือกให้เหมาะสมกับงาน และหาซื้อได้ง่าย โดยในโครงงานนี้ต้องการความละเอียดและไม่ต้องการแรงดึงมากนัก เราจึงเลือกใช้สเต็ปเปอร์มอเตอร์แบบยูนิโพลาร์ แบบ 6สาย มี E = 12V 1.8 องศา/STEP Iw = 0.42 A

3.3 กล้องดิจิตอลที่ใช้

กล้องที่ใช้เป็นกล้องดิจิตอลยี่ห้อ Philip รุ่น Philip VGA Digital Camera (Vesta) Version 4.10.0.2222 เป็น USB port สามารถถ่ายวิดีโอ และถ่ายภาพนิ่งได้ ซึ่งเหมาะกับการนำมาใช้ในโครงงานนี้ สามารถปรับความละเอียดได้สูงถึง 640x480 ทั้งยังสามารถนำมาใช้งานต่างๆได้มากมายด้วย Application ที่แถมมากับDriverกล้อง เช่น Video e-mail/ Video capture , Snapshot images , Video conferencing , Editing and special effects เป็นต้น และยังมีประโยชน์อื่นๆที่น่าสนใจที่จะซื้อ

มาไว้ใช้งานส่วนตัวได้อีกด้วย

เราสามารถต่อกล้องที่เป็น USB Port บนคอมพิวเตอร์ได้ โดยไม่ต้องปิดเครื่องคอมพิวเตอร์ก่อนหรือเรียกอีกอย่างว่า " hot – swappable " ซึ่งหมายความว่าเราไม่ต้องทำการ restart Window 98 ทุกครั้งที่ต่ออุปกรณ์ที่เป็น USB port โดย USB driver จะค้นหา driver ของกล้องที่เป็นฮาร์ดแวร์ตัวใหม่นั้นภายในเวลาเพียงเล็กน้อย 2-3 วินาที โดยไม่ต้อง restart ใหม่



รูปที่ 3.3 Philip VGA Digital Camera (Vesta)

3.4 ทำฉาก

1. เตรียมกล่องการดาษแข็งๆ 1 ใบขนาดใหญ่ (ขนาดใหญ่เท่าที่จะหาได้) ในการทดลองนี้ใช้ขนาดกว้าง 37 ซม. ยาว 55 ซม. ลึก 37 ซม. และเตรียมกระดาษแข็งสีดำ 1 แผ่นขนาดกว้างยาวกว่ากล้องประมาณ 10-20 ซม.

2. ตัดส่วนที่เป็นก้นกล่องทิ้งเหลือขอบของด้านที่เป็นก้นกล่องไว้ประมาณด้านละ 5 ซม. เพื่อใช้เป็นตัวยึดให้กระดาษติดกับกล่องและทำให้กระดาษแข็งแข็งแรงขึ้นด้วย

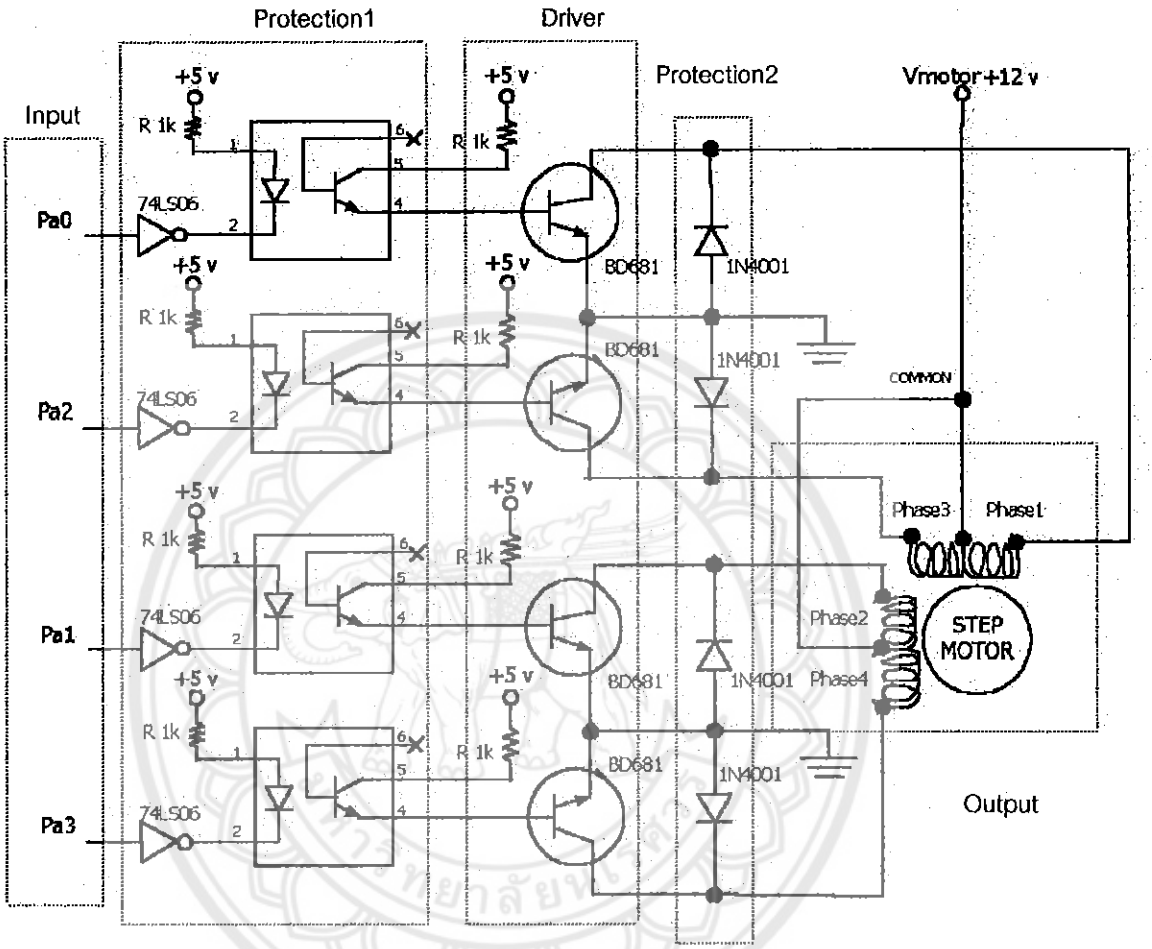
3. ทำการแปะกระดาษแข็งสีดำเข้ากับก้นกล่อง โดยแปะด้านในกล่องและหันด้านที่มีสีดำไว้ด้านในกล่องเช่นกัน

4. ทำการวางกล้องภายในกล่อง โดยวางกล่องแบบตะแคงนอน และเปิดฝากล่องไว้ วางกล้องให้ห่างจากฉากสีดำในที่ระยะหนึ่งเท่าใดก็ได้ ในการวางกล้องควรมีการยึดฐานให้ติดกับกล่องให้แน่นด้วยมิฉะนั้นจะทำให้กล้องเคลื่อนย้ายได้ แล้วจะทำได้ระยะต่างๆที่ผิดพลาดได้

3.5 ทำวงจรควบคุมสเต็ปเปอร์มอเตอร์

1. ทำการออกแบบวงจรขับกำลังมอเตอร์ โดยใช้ความรู้ทางอิเล็กทรอนิกส์ โดยแบ่งเป็น

ส่วนอินพุต เอาต์พุต ส่วนขับสเต็ปเปอร์มอเตอร์ ส่วนโปรเทคชั่น



รูปที่ 3.4 วงจรควบคุมสเต็ปเปอร์มอเตอร์

- ส่วน Input จะรับอินพุตมาจากโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์มาที่ พอร์ต Pa0-Pa3 โดยค่าที่รับมาจะเป็นเลข 0 หรือ 1 จำนวน 4 บิต
- ส่วน Protection1 เป็นส่วนที่ไว้แยกสัญญาณทางไฟฟ้าระหว่างบอร์ดไดเวอร์กับเครื่องคอมพิวเตอร์และ ET Board เพื่อป้องกันความเสียหายที่อาจเกิดขึ้นได้กับอุปกรณ์ที่สำคัญและมีราคาแพง หรือยากต่อการแก้ไข
- ส่วน Driver หรือตัวขับสเต็ปเปอร์มอเตอร์ ใช้ทรานซิสเตอร์เป็นตัวสวิตซ์ให้สเต็ป

เปอร์มอเตอร์หมุนได้ตามข้อมูลอินพุตที่ป้อนเข้ามา เช่นเมื่อมีอินพุตป้อนเข้ามาที่ Pa0 เป็น +5 V จะทำให้ทรานซิสเตอร์ ON มีผลทำให้กระแสไหลครบวงจร เกิดความต่างศักย์ขึ้นระหว่างขา COMMON กับ Phase1 เป็น 12 V

- ส่วน Protection2 เป็นมีไว้ป้องกันการเกิดกระแสไหลย้อนกลับจากการเปลี่ยนแปลงโวลต์แบบทันทีทันใดของขดลวดภายในสเต็ปเปอร์มอเตอร์
- ส่วน Output จะเป็นสายสัญญาณทั้ง 6 เส้นซึ่งจะมีค่าความต่างศักย์ค่าหนึ่งเมื่อเทียบกับขา COMMON

ในการทดสอบการทำงานนั้นจะพบว่า เมื่อเรากระตุ้นเฟส 1, 2, 3, 4 เรียงกันแล้วจะทำให้สเต็ปเปอร์มอเตอร์หมุนทวนเข็มไป และเมื่อเรากระตุ้นเฟส 4, 3, 2, 1 เรียงกันแล้วจะทำให้สเต็ปเปอร์มอเตอร์หมุนตามเข็ม

2. จัดซื้ออุปกรณ์ต่างๆ ดังนี้

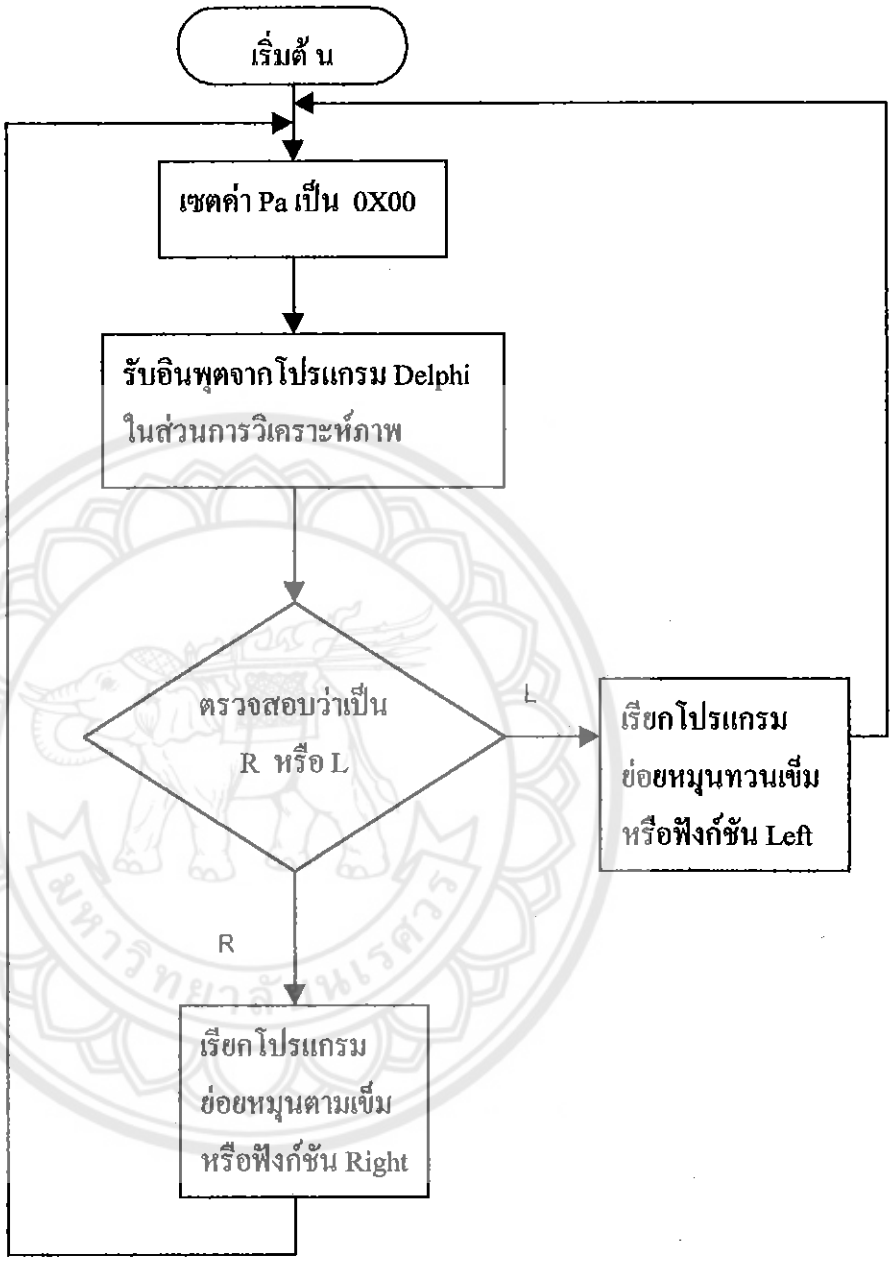
IC 74LS06	1	ตัว
OPTO ISOLATOR (4N25)	4	ตัว
ตัวต้านทาน 1 กิโลโอห์ม(R 1 K)	8	ตัว
ทรานซิสเตอร์เบอร์ BD681	4	ตัว
ไดโอดเบอร์ 1N4007	4	ตัว
LED (ใช้แสดงผลการทำงาน)	4	ตัว
CONNECTER 6 pin	1	ตัว
CONNECTER 16 pin	1	ตัว

3. ต่อวงจรควบคุมสเต็ปเปอร์มอเตอร์ และทดสอบการทำงานของวงจร ตามข้อที่1

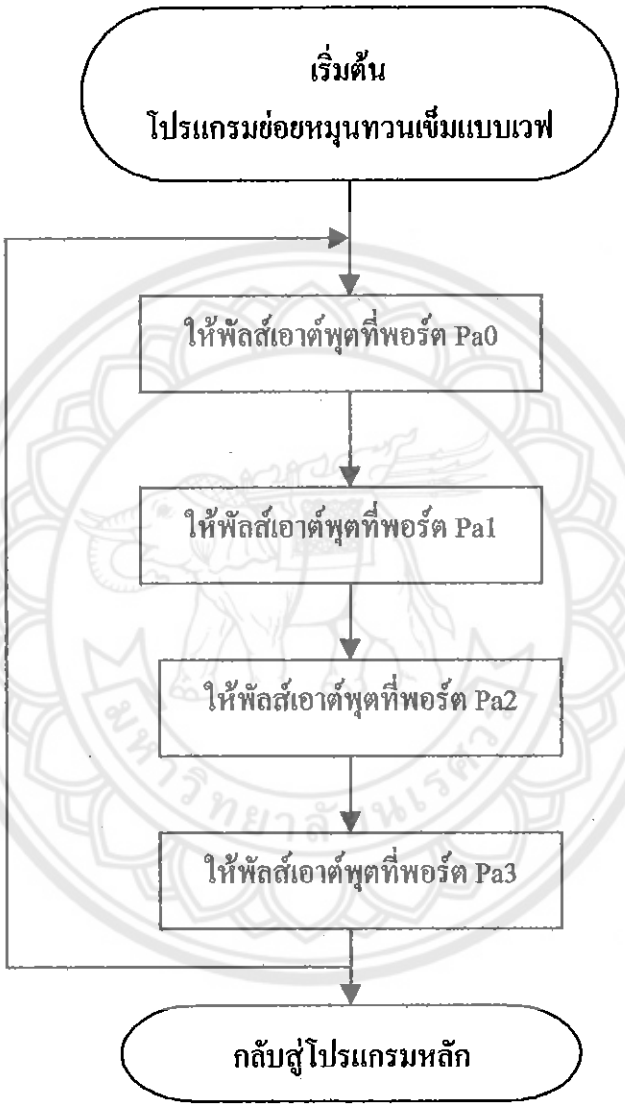
3.6 โปรแกรมควบคุมสเต็ปเปอร์มอเตอร์

ลักษณะการเขียนโปรแกรมควบคุมการหมุนของสเต็ปเปอร์มอเตอร์ที่ใช้ในโครงงานนี้เป็นแบบเวฟ(Wave) ซึ่งเป็นการกระตุ้นแบบที่ง่ายที่สุดแบบหนึ่ง

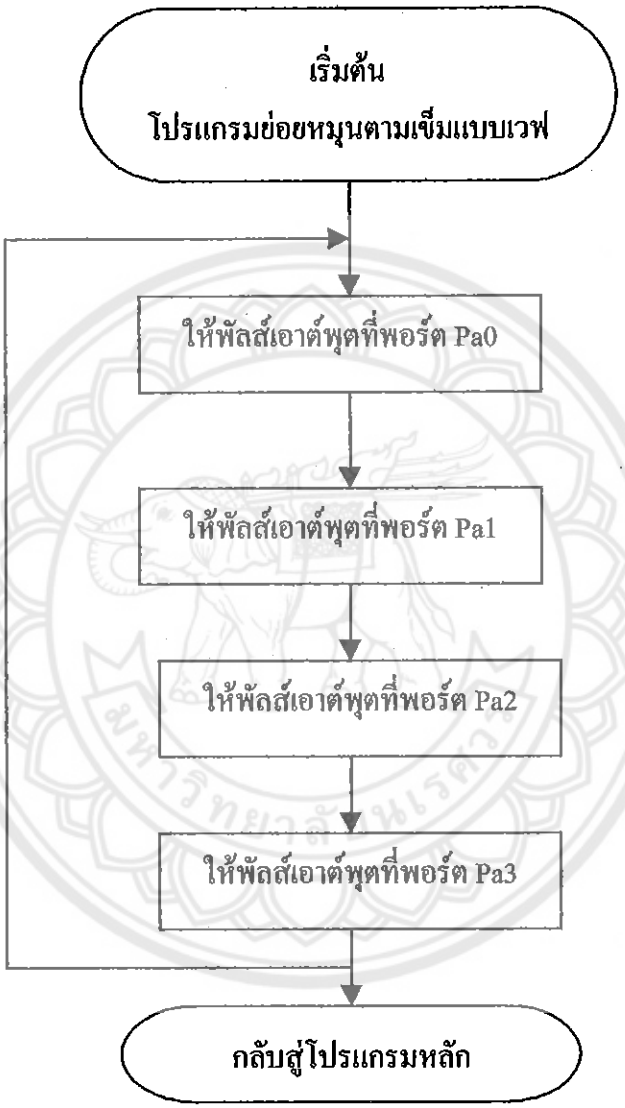
เราสามารถเขียนโฟลว์ชาร์ตการทำงานของโปรแกรมนี้ได้ดังนี้



รูปที่ 3.5 โฟลว์ชาร์ตโปรแกรมหลักของโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์



รูปที่ 3.6 โฟลว์ชาร์ตโปรแกรมย่อยหมุนทวนเข็มแบบเวฟ



รูปที่ 3.7 โฟลว์ชาร์ตโปรแกรมย่อยหมุนตามเข็มแบบเวฟ

หลังจากที่เราได้ดูโฟลว์ชาร์ตการทำงานของโปรแกรมไปแล้ว ก็พอจะเข้าใจการทำงานของโปรแกรมมากขึ้น ต่อมาจะเป็น Source Code ของโปรแกรมนี้

```
#include <reg51.h>
#include <string.h>
#include <absacc.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>

#define Pa    XBYTE [0xE0E0]  /*      //ship 255 port A              */
#define Pb    XBYTE [0xE0E1]  /*      //ship 255 port B              */
#define Pc    XBYTE [0xE0E2]  /*      //ship 255 port C              */
#define Pcon  XBYTE [0XE0E3]  /*      //ship 255 register control    */
#define LCD_CONTROL  XBYTE[0xE0C0]
#define LCD_BUSY_FLAG XBYTE[0xE0C1]
#define LCD_WRITEDATA XBYTE[0xE0C2]
#define LCD_READDATA  XBYTE[0xE0C3]

#define StartStep 10

void delay(unsigned int time)
{
  unsigned int i;
  for(i=0;i < time;i++)
     {
     }
}
```

```
void wait_for_lcd_ready (void)
{
  while ((LCD_BUSY_FLAG & 0x80) != 0)
        {
        }
      //wait util buy flag = 0 (lcd ready)
}

void specified_lcd(void)
{
  //for lcd more than 2 line
  //and 5x7 dot per cursor font
  LCD_CONTROL=0x38;                    //function set bit5=1
  wait_for_lcd_ready ();
}

void clear_lcd(void)
{
      LCD_CONTROL=0x01;
      //bit0=1 clear lcd
      //and cursor goto top left
   wait_for_lcd_ready ();
}
void cursor_on(void)
{
  LCD_CONTROL = 0x0f;          //bit2=1 lcd monitor on
  wait_for_lcd_ready ();
}
```

```
void cursor_off(void)
{
  LCD_CONTROL = 0x0c;
  wait_for_lcd_ready ();
}

void entry_mode_set(void)
{
  LCD_CONTROL = 0x02;
//bit0=0 after put data shift cursor to right
//bit1=1 after read/write dd ram address increase1

        wait_for_lcd_ready ();
}

void init_lcd(void)
{
        specified_lcd();
        cursor_on();
        clear_lcd();
        entry_mode_set();
}

void WChar(char ch)
{
  LCD_WRITEDATA = ch;
  wait_for_lcd_ready();
}
```

```
void gotoxy(unsigned char row, unsigned char col)
{
  if (row == 1) LCD_CONTROL = 0x80+col-1; else
  if (row == 2) LCD_CONTROL = 0xc0+col-1; else
  if (row == 3) LCD_CONTROL = 0x90+col-1; else
  if (row == 4) LCD_CONTROL = 0xd0+col-1;
  wait_for_lcd_ready ();
}

void WCharxy(unsigned char row,unsigned char col,unsigned char ch)
{
  gotoxy(row,col);
  WChar(ch);
}

void string_to_lcd (unsigned char row,unsigned char col,char *st)
{
  data unsigned char mcol,colfinish,len;

        //clear_lcd();
  len = strlen(st);
  colfinish=col+len-1;
  if (colfinish > 16)  colfinish=16;

  for (mcol=col; mcol<=colfinish; mcol++)
        {
        WCharxy(row,mcol,st[mcol-col]);
        }
```

```
}

void string_to_lcd32 (char *st)
{
  data unsigned char mcol,colfinish,len,i;
  xdata char display_lcd[32];

  for (i=0;i<=31;i++)
        {
        display_lcd[i]=' ';
        }
  len = strlen(st);

  if (len>32) len=32;
  for (i=0;i<=len-1;i++)
        {
        display_lcd[i]=st[i];
        }

  for (mcol=1; mcol<=16; mcol++)
        {
        WCharxy(1,mcol,display_lcd[mcol-1]);
        }
  for (mcol=1; mcol<=16; mcol++)
        {
        WCharxy(2,mcol,display_lcd[(mcol+16)-1]);
        }
}
```

```
void string_to_lcd32RotateLeft (char *st)
{
  data unsigned char mcol,colfinish,len,i,temp;
  xdata char display_lcd[32];

  for (i=0;i<=31;i++)
          {
           display_lcd[i]=' ';
          }
  len = strlen(st);

  if (len>32) len=32;
  for (i=0;i<=len-1;i++)
          {
          display_lcd[i]=st[i];
          }

  while (1)
    {
    string_to_lcd32(display_lcd);
    temp=display_lcd[0];
    for (i=1;i<=31;i++)
      {
      display_lcd[i-1]=display_lcd[i];
      }
    display_lcd[31]=temp;
    delay(10000);
    }
```

```
}

void XStop(void)
 {
  while(1)
        {
        Pa=0x00;
        }
 }


void Lift(int XStepNo)/*Move forward on x axis*/
{
   unsigned int Tround,XSpeed,i=0;
        unsigned int round;
        char stepx[2];
        round=1;
        Tround=XStepNo/4;
        XSpeed=800;
        Pa=0x30;
        //delay(XSpeed);
        stepx[0]='D';
        while(i<XStepNo)
    {
         //fade in
         if(round<StartStep)
//fade out
        XSpeed=XSpeed-50;
```

```
//of motor
        else if((round>=StartStep)&&(round<=(Tround-StartStep))) //rotation
                XSpeed=XSpeed+0;
        else XSpeed=XSpeed+50;
           Pa=0x01;
        delay(XSpeed);
                i++;
                if(i<XStepNo)
                  {
                        Pa=0x02;
                   delay(XSpeed);
                        i++;
                        if(i<XStepNo)
                          {
                                Pa=0x04;
                delay(XSpeed);
                                i++;
                                if (i<XStepNo)
                {
                   Pa=0x08;
                  delay(XSpeed);
                    i++;
                                        round=round+1;
                                          }else
                                                {
                                                 Pa=0x00;
                                                 delay(XSpeed);
                                                 stepx[0]='C';
```

```
                                        }/*end if step4*/
                                }else
                                        {
                                        Pa=0x00;
                                        delay(XSpeed);
                                        stepx[0]='B';
                                        }/*end if step3*/
                        }else
                                {
                                        Pa=0x00;
                                        delay(XSpeed);
                                        stepx[0]='A';
                                        }/*end if step2*/
                }/*end while*/
                Pa=0x00;
                //delay(XSpeed);
                //string_to_lcd(1,1,stepx);
}/*end of Lift function */

void Right(int XStepNo)/*Move backward on x axis*/
{
    unsigned int XSpeed,round,Tround,i=0;
            char stepx[2];
            round=1;
            Tround=XStepNo/4;
            XSpeed=800;
            Pa=0x30;
            //delay(XSpeed);
```

```
stepx[0]='D';
while(i<XStepNo)
 {
 if(round<StartStep)
        XSpeed=XSpeed-50;
        else if((round>=StartStep)&&(round<=(Tround-StartStep)))
                XSpeed=XSpeed+0;
                else XSpeed=XSpeed+50;
          Pa=0x08;
          delay(XSpeed);
          i++;
          if(i<XStepNo)
                {
                 Pa=0x04;
                 delay(XSpeed);
                    i++;
                   if(i<XStepNo)
                        {
                                Pa=0x02;
                              delay(XSpeed);
                                i++;
                            if (i<XStepNo)
                              {
                              Pa=0x01;
                              delay(XSpeed);
                              i++;
                                      round=round+1;
                                      }else
```

```
                                                    {
                                                    Pa=0x00;
                                                    delay(XSpeed);
                                                    stepx[0]='C';
                                                    }/*end if step4*/
                                        }else
                                                    {
                                                Pa=0x00;
                                                    delay(XSpeed);
                                                    stepx[0]='B';
                                                    }/*end if step3*/
                                    }else
                                                    {
                                                    Pa=0x00;
                                                    delay(XSpeed);
                                                    stepx[0]='A';
                                                }/*end if step2*/
                        }/*end of while*/
                    Pa=0x00;
                    //delay(XSpeed);
                    //string_to_lcd(2,1,stepx);
                }/*end of Right function*/
                void init_serial(void)
                {
                    SCON = 0x52;
                    PCON = 0x00;
                    TMOD = 0x20;
                    TH1 = 0xFD;
```

```
   TR1 = 1;
// TI = 1;
// RI = 0;
}

void main(void)
{
   unsigned char mBUF,mess[10],Strdat[20],*pmess,Xvalue[15],*pXvalue;
         unsigned char Rstr[10],Lstr[10];
         unsigned int i,j,k,ir,il,checkend;
         unsigned int Rint,Lint; //integer value

         delay(2000);
          Pcon=0x80;

         init_lcd();
         init_serial();

         Pa=0x00;//Preset Port A
         Pb=0x00;//Preset Port B
         Pc=0x00;//Preset Port C
         strcpy(Strdat," ");//Clear Array
         strcpy(Rstr," ");
         strcpy(Lstr," ");
         Rint=0;
         Lint=0;
         i=0;
         j=0;
```

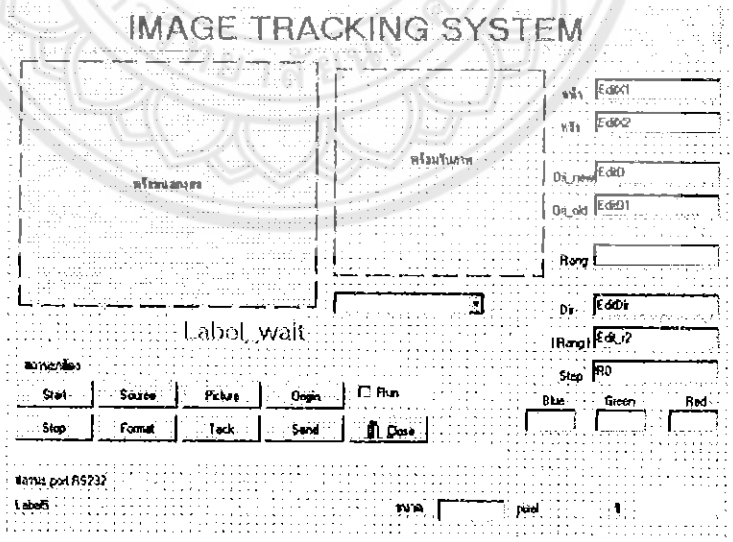
```
    k=0;
    ir=0;
    il=0;
    clear_lcd();
    while(1)
{
      while(RI!=1)  //have data?
  {
              }
RI = 0; //get data
          //mBUF=SBUF;
          mess[0]=SBUF; //transfer data from sbuf register
          if(mess[0]==13)
            WChar('X');
          if(mess[0]!=10)//Is it 's not end of Line data?
             {
            /*check  R,L and 0-9*/
                  if(mess[0]>='0' && mess[0]<='9'||mess[0]=='R'||mess
[0]=='L')//|| mess[0]=='E')
                     {
                          Strdat[k]=mess[0]; //get a string of a point
                          k=k+1;
                     }
                else //if(mess[0]!='E')
                  {
                          Strdat[k]=" ";
                pmess=Strdat; //Pointer point to the string
                          if(*pmess=='R')//check R
```

```
                pmess=pmess+1;
            while(*pmess>='0' && *pmess<='9')
                {
                    Rstr[ir]=*pmess; //get R value
in string
                    pmess=pmess+1;
                    ir=ir+1;
                }
                if(*pmess=='L')//check L
                    pmess=pmess+1;
            while(*pmess>='0'&& *pmess<='9')
                {
                    Lstr[il]=*pmess;//check L
value in string
                    pmess=pmess+1;
                    il=il+1;
                }
                Rstr[ir]=0;
                Lstr[il]=0;
                Rint=atoi(Rstr);//get R value in integer
                Lint=atoi(Lstr);//get L value in integer
                Right(Rint);
                Lift(Lint);

            delay(500);
            delay(500);

        printf("R%dL%d\n",Rint,Lint); //Sent Interface Program
```

```
                    k=0;
                    ir=0;
                    il=0;
                   }
              }
         }
}
```

3.7 โปรแกรมวิเคราะห์ภาพของวัตถุ

มีหน้าที่ในการวิเคราะห์ภาพของวัตถุ โดยรับอินพุตภาพมาจากกล้องดิจิตอล แล้วเข้าสู่กระบวนการวิเคราะห์ภาพ ซึ่งเราจะวิเคราะห์สี วิเคราะห์ขอบ คำนวณหาจุดศูนย์กลางของภาพ และหาระยะขจัด แล้วส่งเอาต์พุตออกไปยังส่วนโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ เพื่อให้ง่ายต่อการเข้าใจลองดูลักษณะของโฟลว์ชาร์ตของโปรแกรมนี้ดังรูปที่ 3.8 และตัว Source code ของโปรแกรม ซึ่งในรายงานโครงงานนี้จะนำเสนอเพียงUnitที่สำคัญ คือ Unit1 เป็นส่วนโปรแกรมหลัก ทำหน้าที่ในการวิเคราะห์ภาพ และVideoCap เป็นโมดูลที่เขียนฟังก์ชันการทำงานที่โปรแกรมหลักสามารถเรียกใช้ได้



รูปที่ 3.8 แสดงหน้าจอของโปรแกรมการติดตามวัตถุในโปรแกรมDelphi5

รูปที่ 3.9 โฟลว์ชาร์ตของโปรแกรมส่วนวิเคราะห์ภาพ

```
unit Unit1;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, ExtCtrls,VideoCap, ComCtrls, Async32, Buttons;

type
 TForm1 = class(TForm)
  Label1: TLabel;
  Image3: TImage;
  Label3: TLabel;
  Label4: TLabel;
  Label6: TLabel;
  Panel1: TPanel;
  videoarea: TPanel;
  Image1: TImage;
  Button1: TButton;
  Button2: TButton;
  Button3: TButton;
  Button4: TButton;
  Button5: TButton;
  Button6: TButton;
  ProgressBar1: TProgressBar;
  Edit1: TEdit;
  EditR: TEdit;
  Button7: TButton;
```

Comm1: TComm;

Panel2: TPanel;

Label7: TLabel;

ColorDialog1: TColorDialog;

EditSize: TEdit;

Label24: TLabel;

Label27: TLabel;

EditG: TEdit;

EditB: TEdit;

ShapeShowColor: TShape;

Label25: TLabel;

Label26: TLabel;

Label28: TLabel;

EditGain: TEdit;

BitBtn1: TBitBtn;

EditX1: TEdit;

EditX2: TEdit;

EditO: TEdit;

EditO1: TEdit;

Edit_r1: TEdit;

EditDir: TEdit;

Edit_r2: TEdit;

Label29: TLabel;

Label30: TLabel;

Label31: TLabel;

Label32: TLabel;

Label33: TLabel;

```
Label34: TLabel;
Label35: TLabel;

Resolution: TComboBox;

Label_wait: TLabel;
Button9: TButton;
Label2: TLabel;
Label5: TLabel;
TimRun: TTimer;
chkTimer: TCheckBox;
procedure Button1Click(Sender: TObject);
procedure CAPStatus(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button7Click(Sender: TObject);

procedure Comm1RxChar(Sender: TObject; Count: Integer);
procedure FormCreate(Sender: TObject);
procedure EditGainKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure ShapeShowColorMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure ResolutionChange(Sender: TObject);
```

```
    procedure Button9Click(Sender: TObject);
    procedure TimRunTimer(Sender: TObject);
    procedure chkTimerClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  iSelcolor : Integer;
  i,j,iCount,iColor,iBlckHight,iBlckWidth : Integer;
  gain:byte;
  picture:integer;
  Origin, Origin_old:integer;
  Round:integer;
  start : integer;
  pstep : integer;

implementation

{$R *.DFM}

procedure TForm1.CAPStatus(Sender: TObject);
begin
        Panel1.Color := clBtnFace;
```

```
        Panel1.Refresh;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
 MyCapStatusProc : TCapStatusProc;
// E: Exception;
begin
                // Start CAP -  Video
        CapSetVideoArea( VideoArea );
 CapSetInfoLabel( Label1 );
 MyCapStatusProc := CAPStatus;
        CapSetStatusProcedure( MyCapStatusProc );
        if CapOpenDriver then
 begin
                CapSetCapSec( 15 * 3 );
                CapShow;
        end;

 Comm1.DeviceName := 'Com1';
 try
  Comm1.Open;
  Label27.Caption := 'Device ready: ' + Comm1.DeviceName;
  Label27.Caption := GetProviderSubtypeName(Comm1.ProviderSubtype) + ' พร้อม';
 except
  on E: ECommError do
    Edit1.Text := E.Message;
```

```
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
   Comm1.Close;
   Label27.Caption := 'Port RS232 ปิดแล้ว';
          CapCloseDriver;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
          CapDlgVSource;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
          CapDlgVFormat;
end;
procedure TForm1.Button5Click(Sender: TObject);
var
//  iCount : Integer;
  SingleImageFileName : string;
begin
//     iHighcolor := iSelcolor + 10000;
   ProgressBar1.Max := Image3.Picture.Height;
   Label24.Visible := False;
                    // Save Video as Bitmap to file in TEMP-Path
```

```
  SingleImageFileName := 'Image1.bmp';
  CapSetSingleImageFileName( SingleImageFileName );
        CapGrabSingleFrame;
        CapSetVideoLive;
  Image3.Picture.LoadFromFile('Image1.bmp');
end;

procedure TForm1.Button6Click(Sender: TObject);
var
        SingleImageFileName : string;

   bitmap:tbitmap;
   r,g,b:byte;
   col:dword;
   sumR,sumG,sumB,tmp:double;
   i,j:integer;
   Wsize:longint;

   Row1, Col1, X1, Y1, Z:integer;
   Row2, Col2, X2, Y2, W:integer;
   Rang, Rang1, step : integer;
        S: string;
        Count: Integer;
begin
        start := 1;

   refresh();
```

```
// Send Picture
   //    iHighcolor := iSelcolor + 10000;
  ProgressBar1.Max := Image3.Picture.Height;
  Label24.Visible := False;
                  // Save Video as Bitmap to file in TEMP-Path
  SingleImageFileName := 'Image1.bmp';
  CapSetSingleImageFileName( SingleImageFileName );
        CapGrabSingleFrame;
        CapSetVideoLive;
  Image3.Picture.LoadFromFile('Image1.bmp');
// End Send Picture
// Analise Picture
        X1 := 0; X2 := 0; W := 0; Z := 0;

   bitmap:=tbitmap.Create;
   bitmap.LoadFromFile('Image1.bmp');
   WSize:=0;
   sumR:=0;SumG:=0;SumB:=0;

   for i:=0 to bitmap.Width-1 do
   begin
      for j:=0 to bitmap.Height-1 do
      begin
         col:=bitmap.Canvas.Pixels[i,j];
         r:=lo(col);
         g:=hi(col);
         b:=hiword(col);
         sumR:=sumR+r;
```

```
        sumG:=sumG+g;
        sumB:=sumB+b;
        r:=((30*r)+(59*g)+(11*b))div 100;
        if r>Gain then
        begin
           inc(Wsize);
           bitmap.Canvas.Pixels[i,j]:=$ffffff;
        end else bitmap.Canvas.Pixels[i,j]:=0;
    end;
end;

tmp:=(bitmap.Height+1)*(bitmap.Width+1);
sumR:=sumR/tmp;
sumG:=sumG/tmp;
sumB:=sumB/tmp;
editR.text:=format('%.0f',[SumR]);
editR.Refresh;
editG.text:=format('%.0f',[SumG]);
editG.Refresh;
editB.text:=format('%.0f',[SumB]);
editB.Refresh;
r:=strtoint(editR.text);
g:=strtoint(editG.text);
b:=strtoint(editB.text);
shapeshowcolor.Brush.Color:=rgb(r,g,b);
editsize.Text:=inttostr(Wsize);
editsize.Refresh;
image3.Picture.Bitmap:=bitmap;
```

```
//   if (Wsize = 0) then
//   begin
//         start := 0;
//         break;
//   end;

// Determine Rang and Dir
   for Col1 := 1 to bitmap.width-1 do
   begin
         for Row1 := 1 to bitmap.height-1 do
         begin
                X1 := Col1; Y1 := Row1;
            if (image3.Canvas.Pixels[X1,Y1] = clWhite)then
            begin
                                    EditX1.Text := inttostr(X1);
               editX1.Refresh;
               W := 1;
               break;
            end;
            if (W=1)then break;
         end;
      if (W=1)then break;
   end;

   for Col2 := bitmap.width-1 downto 1 do
   begin
         for Row2 := 1 to bitmap.height-1 do
         begin
```

```
            X2 := Col2; Y2 := Row2;
        if (image3.Canvas.Pixels[X2,Y2] = clWhite)then
        begin
          EditX2.Text := inttostr(X2);
          editX2.Refresh;
          Z := 1;
          break;
        end;
```

```
unit VideoCap;

interface

uses Windows, Dialogs, Controls, SysUtils, StdCtrls, MMSystem, AviCap;

const
      MAXVIDDRIVERS = 10;
      MS_FOR_15FPS  = 66;
      MS_FOR_20FPS  = 50;
      MS_FOR_30FPS  = 33;
      MS_FOR_25FPS  = 40;          // rate in msec

type
 TCapStatusProc = procedure(Sender: TObject) of object;

var
 ghCapWnd          : THandle;
```

```
gCapVideoArea          : TWinControl;
gCapVideoDriverName    : string;
//gdwCapNofMaxVideoFrame  : DWord;
gdwCapNofMaxVideoFrame  : Integer;
gCapVideoFileName      : string;
gCapSingleImageFileName : string;
gCapVideoInfoLabel     : TLabel;
gCapStatusProcedure    : TCapStatusProc;

        procedure CapSetVideoArea( Container: TWinControl );
        procedure CapSetVideoFileName( FileName : string );
        procedure CapSetSingleImageFileName( FileName : string );
        procedure CapSetInfoLabel( InfoLabel : TLabel );
        procedure CapSetStatusProcedure( StatusProc : TCapStatusProc );

        function  CapOpenDriver : Boolean;
function  CapInitDriver( Index : Integer ): Boolean;
        procedure CapCloseDriver;
procedure CapShow;
        procedure CapSetCapSec( NofMaxVideoFrame : Integer );
procedure CapStart;
procedure CapStop;
function  CapHasDlgVFormat  : Boolean;
function  CapHasDlgVDisplay : Boolean;
function  CapHasDlgVSource  : Boolean;
procedure CapDlgVFormat;
procedure CapDlgVDisplay;
procedure CapDlgVSource;
```

```
 procedure CapSetVideoOverlay;
 procedure CapSetVideoLive;
 procedure CapGrabSingleFrame;

implementation

(*------------------------------------------------------------*)
(*--- C A P - V I D E O  D R I V E R ---*)
(*------------------------------------------------------------*)

(*------------------------------------------------------------*)
procedure CapSetVideoArea( Container: TWinControl );
begin
 gCapVideoArea  := Container;
end;

(*------------------------------------------------------------*)
procedure CapSetVideoFileName( FileName : string );
begin
        gCapVideoFileName := FileName;
end;

(*------------------------------------------------------------*)
procedure CapSetSingleImageFileName( FileName : string );
begin
        gCapSingleImageFileName := FileName;
end;
```

```
(*------------------------------------------------------------------*)
procedure CapSetInfoLabel( InfoLabel : TLabel );
begin
        gCapVideoInfoLabel := InfoLabel;
end;

(*------------------------------------------------------------------*)
procedure CapSetStatusProcedure( StatusProc : TCapStatusProc );
begin
        gCapStatusProcedure := StatusProc;
end;

(*------------------------------------------------------------------*)
(* -- Video For Windows Status Callback Function --- *)
(*------------------------------------------------------------------*)
function StatusCallbackProc(hWnd : HWND; nID : Integer; lpsz : LongInt): LongInt; stdcall;
var
        TmpStr    : string;
 dwVideoNum :  Integer;
begin
        // hWnd:         Application main window handle
        // nID:          Status code for the current status
        // lpStatusText:  Status text string for the current status

        TmpStr := StrPas(PChar(lpsz));
        gCapVideoInfoLabel.Caption := TmpStr;
        gCapVideoInfoLabel.Refresh;
```

```
        if nID = IDS_CAP_STAT_VIDEOCURRENT then
        begin
                dwVideoNum := StrToInt( Copy(TmpStr, 0, Pos(' ', TmpStr)-1));
                if dwVideoNum >= gdwCapNofMaxVideoFrame then
        begin
                        capCaptureAbort(ghCapWnd);
                        if @gCapStatusProcedure <> nil then gCapStatusProcedure(nil);
                end;
        end;
        Result := 1;
end;


(*------------------------------------------------------------*)
function CapOpenDriver : Boolean;
var
 Retc          : LongInt;
        DriverIndex     : Integer;
 DriverStarted   : boolean;
        achDeviceName    : array [0..80] of Char;
 achDeviceVersion : array [0..100] of Char;
        achFileName     : array [0..255] of Char;
begin
        Result := FALSE;
        if gCapVideoArea = nil then exit;

        Result     := TRUE;
```

```
                // Create the Video Capture Window
        ghCapWnd := capCreateCaptureWindow( PChar('KruwoSoft'),
        WS_CHILD or WS_VISIBLE, 0, 0,
        gCapVideoArea.Width, gCapVideoArea.Height,
        gCapVideoArea.Handle, 0);
if ghCapWnd <> 0 then
begin
                        // Install Status-Callback-Function
        retc := capSetCallbackOnStatus(ghCapWnd, LongInt(0));
 if retc <> 0 then
 begin
                retc := capSetCallbackOnStatus(ghCapWnd, LongInt(@StatusCallbackProc));
          if retc <> 0 then
          begin
                                // Open Installed Video Driver
   DriverIndex := 0;
     repeat
                                DriverStarted := CapInitDriver( DriverIndex );
    if NOT DriverStarted then DriverIndex := DriverIndex + 1;
                            until (DriverStarted = TRUE) OR (DriverIndex >=
MAXVIDDRIVERS);

                                // Keep Name of Video Driver
                            if capGetDriverDescription( DriverIndex,
                    achDeviceName,  80,
                        achDeviceVersion, 100 ) then
                            begin
                                gCapVideoDriverName := string(achDeviceName);
```

```
                                    end;

                                                    // Set Capture FileName
                                StrPCopy(achFileName, gCapVideoFileName);
                                                    retc := capFileSetCaptureFile(ghCapWnd, LongInt
((@achFileName));
                                if retc = 0 then
    begin
        showmessage(gCapVideoDriverName+': Error in capFileSetCaptureFile');
    end;
    exit;
   end;
                            end;
        end;
        Result := FALSE;
        CapCloseDriver;
 ghCapWnd := 0;
end;

(*-------------------------------------------------------------*)
function CapInitDriver( Index : Integer ): Boolean;
var
 Retc          : LongInt;
        CapParms       : TCAPTUREPARMS;
begin

        Result := FALSE;
        if ghCapWnd = 0 then exit;
```

```
                    // Connect to Video Capture Driver
  if capDriverConnect(ghCapWnd, Index) <> 0 then
  begin
    retc := capCaptureGetSetup(ghCapWnd, LongInt(@CapParms), sizeof(TCAPTUREPARMS));
                    if retc <> 0 then
    begin
//                          CapParms.dwRequestMicroSecPerFrame := 40000;    // 25 FPS
Requested capture rate
//      CapParms.dwRequestMicroSecPerFrame := 100000;    // 10 FPS Requested capture rate
                          CapParms.dwRequestMicroSecPerFrame := 66667;    // 15 FPS
Requested capture rate
                        CapParms.fLimitEnabled    := FALSE;
                    CapParms.fCaptureAudio    := FALSE;              // NO Audio
                    CapParms.fMCIControl      := FALSE;
                        CapParms.fYield           := TRUE;
                    CapParms.vKeyAbort        := VK_ESCAPE;
                    CapParms.fAbortLeftMouse  := FALSE;
                    CapParms.fAbortRightMouse := FALSE;

                        retc := capCaptureSetSetup(ghCapWnd, LongInt(@CapParms), sizeof
(TCAPTUREPARMS));
     if retc = 0 then exit;
    end;
                    Result := TRUE;
          end;
end;
```

```
(*-------------------------------------------------------------*)
procedure CapCloseDriver;
begin
        if ghCapWnd <> 0 then
 begin
          capSetCallbackOnStatus(ghCapWnd, LongInt(0));
          capDriverDisconnect( ghCapWnd );
                DestroyWindow( ghCapWnd ) ;
                ghCapWnd := 0;
        end;
end;

(*-------------------------------------------------------------*)
procedure CapShow;
begin
        if ghCapWnd = 0 then exit;

        // Start Video overlay by default
        capPreviewScale(ghCapWnd, 1);
 capPreviewRate(ghCapWnd, MS_FOR_25FPS);
        capOverlay(ghCapWnd, 0);
        capPreview(ghCapWnd, 1);
end;

(*-------------------------------------------------------------*)
procedure CapSetCapSec( NofMaxVideoFrame : Integer );
begin
        //gdwCapNofMaxVideoFrame := DWord( NofMaxVideoFrame );
```

```
  gdwCapNofMaxVideoFrame := NofMaxVideoFrame ;
end;

(*-------------------------------------------------------------*)
procedure CapStart;
begin
        if ghCapWnd = 0 then exit;
                // Start video capture to file
        capCaptureSequence( ghCapWnd );
end;

(*-------------------------------------------------------------*)
procedure CapStop;
begin
        if ghCapWnd = 0 then exit;
                // Stop video capture to file
        capCaptureAbort(ghCapWnd);
end;

(*-------------------------------------------------------------*)
function  CapHasDlgVFormat  : Boolean;
var
 CDrvCaps : TCapDriverCaps;
begin
        Result := TRUE;
        if ghCapWnd = 0 then exit;

        capDriverGetCaps(ghCapWnd, LongInt(@CDrvCaps), sizeof(TCapDriverCaps));
```

```
 Result := CDrvCaps.fHasDlgVideoFormat;
end;

(*------------------------------------------------------------*)
function  CapHasDlgVDisplay : Boolean;
var
 CDrvCaps : TCapDriverCaps;
begin
        Result := TRUE;
        if ghCapWnd = 0 then exit;

        capDriverGetCaps(ghCapWnd, LongInt(@CDrvCaps), sizeof(TCapDriverCaps));
 Result := CDrvCaps.fHasDlgVideoDisplay;
end;

(*------------------------------------------------------------*)
function  CapHasDlgVSource  : Boolean;
var
 CDrvCaps : TCapDriverCaps;
begin
        Result := TRUE;
        if ghCapWnd = 0 then exit;

        capDriverGetCaps(ghCapWnd, LongInt(@CDrvCaps), sizeof(TCapDriverCaps));
 Result := CDrvCaps.fHasDlgVideoSource;
end;

(*------------------------------------------------------------*)
```

```
procedure CapDlgVFormat;
begin
        if ghCapWnd = 0 then exit;

        capDlgVideoFormat(ghCapWnd);
end;

(*---------------------------------------------------------------*)
procedure CapDlgVDisplay;
begin
        if ghCapWnd = 0 then exit;

        capDlgVideoDisplay(ghCapWnd);
end;

(*---------------------------------------------------------------*)
procedure CapDlgVSource;
begin
        if ghCapWnd = 0 then exit;

        capDlgVideoSource(ghCapWnd);
end;

(*---------------------------------------------------------------*)
procedure CapSetVideoOverlay;
begin
        if ghCapWnd = 0 then exit;
```

```
        capPreview(ghCapWnd, 0);
        capOverlay(ghCapWnd, 1);
end;

(*-------------------------------------------------------------*)
procedure CapSetVideoLive;
begin
        if ghCapWnd = 0 then exit;

        capOverlay(ghCapWnd, 0);
        capPreviewScale(ghCapWnd, 1);
 capPreviewRate(ghCapWnd, MS_FOR_25FPS);
        capPreview(ghCapWnd, 1);
end;

(*-------------------------------------------------------------*)
procedure CapGrabSingleFrame;
var
        achSingleFileName  : array [0..255] of Char;
begin
        if ghCapWnd = 0 then exit;

        capGrabFrame(ghCapWnd);
        StrPCopy(achSingleFileName, gCapSingleImageFileName);
        capFileSaveDIB(ghCapWnd, LongInt(@achSingleFileName));
end;

initialization
```

```
        ghCapWnd              := 0;
    gCapVideoArea         := nil;
    gCapVideoDriverName   := 'No Driver';
        gdwCapNofMaxVideoFrame := 0;
    gCapVideoFileName     := 'Video.avi';
    gCapSingleImageFileName := 'Image.bmp';
        gCapVideoInfoLabel    := nil;
    gCapStatusProcedure   := nil;
end.
```

บทที่ 4
ผลการทดลองและการวิเคราะห์

ในขั้นตอนการทดลองนั้นแบ่งเป็นการทดลองบอร์ดไดเวอร์กับโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ การทดลองในส่วนโปรแกรมวิเคราะห์ภาพ และการทดลองการรับส่งข้อมูลระหว่าง Delphi5 กับ C51

4.1 การทดลองบอร์ดไดเวอร์กับโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์

วัตถุประสงค์ของการทดลอง

1. เพื่อพัฒนาวงจรไดเวอร์สเต็ปเปอร์มอเตอร์ให้เหมาะสมกับการใช้งาน
2. เพื่อพัฒนาโปรแกรมสำหรับควบคุมสเต็ปเปอร์มอเตอร์ด้วย C51

การทดลองที่1 : ทดสอบความถูกต้องของวงจรไดเวอร์สเต็ปเปอร์มอเตอร์

ก่อนที่เราจะทำการต่อบอร์ดไดเวอร์ เราได้ทดลองต่อวงจรเพียงส่วนเดียวจากทั้งหมดสี่ส่วนก่อน พบว่าได้ผล จึงเริ่มทำการต่อวงจรทั้งหมดแล้วต่อเอาต์พุตเข้ากับLED 4 ตัว พบว่าได้ผล เราจึงต่อบอร์ดไดเวอร์เข้ากับตัวสเต็ปเปอร์มอเตอร์ แล้วลองป้อนอินพุตเข้าไป พบว่าได้เอาต์พุตออกมาตรงตามความต้องการ

การทดลองที่2 : ทดสอบโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์

ต่อมาเราเขียนโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ด้วย C51 โดยตัวโปรแกรมนี้จะมีหน้าที่รับข้อมูลจาก Delphi5 มาวิเคราะห์ว่าต้องสั่งให้มอเตอร์หมุนไปซ้ายหรือขวากี่สเต็ป หรืออาจสั่งให้หยุดก็ได้ แล้วส่งค่าไปที่บอร์ดไดเวอร์สเต็ปเปอร์มอเตอร์ผ่านพอร์ต RS-232 โดยแบ่งเป็นการทดลองย่อยดังนี้

การทดลองที่ 2.1 : ทดสอบความละเอียดของการหมุนของสเต็ปเปอร์มอเตอร์

- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนตามเข็มไป 1 สเต็ป หลายครั้ง พบว่าเคลื่อนไป 1 สเต็ปแรกเท่านั้น

- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนตามเข็มไป 2 สเต็ป หลายครั้ง พบว่าเคลื่อนไป 2 สเต็ปแรกเท่านั้น
- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนตามเข็มไป 3 สเต็ป หลายครั้ง พบว่าเคลื่อนไป 3 สเต็ปแรกเท่านั้น
- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนตามเข็มไป 4 สเต็ป หลายครั้ง พบว่าเคลื่อนไปได้ทุกครั้งที่ส่งค่าให้สเต็ปเปอร์มอเตอร์

การทดลองที่ 2.2 : ทดสอบความละเอียดของการหมุนของสเต็ปเปอร์มอเตอร์

- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนทวนเข็มไป 1 สเต็ป หลายครั้ง พบว่าเคลื่อนไป 1 สเต็ปแรกเท่านั้น
- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนทวนเข็มไป 2 สเต็ป หลายครั้ง พบว่าเคลื่อนไป 2 สเต็ปแรกเท่านั้น
- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนทวนเข็มไป 3 สเต็ป หลายครั้ง พบว่าเคลื่อนไป 3 สเต็ปแรกเท่านั้น
- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนทวนเข็มไป 4 สเต็ป หลายครั้ง พบว่าเคลื่อนไปได้ทุกครั้งที่ส่งค่าให้สเต็ปเปอร์มอเตอร์

การทดลองที่ 2.3 : ทดสอบความถูกต้องของจำนวนสเต็ปการหมุนของสเต็ปเปอร์มอเตอร์

- ก่อนการทดลองให้ทำเครื่องบอกตำแหน่งไว้เป็นจุดเริ่มต้น
- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนตามเข็มไป 200 สเต็ป พบว่าหมุนไปได้ระยะหนึ่งแล้วหยุด (ทำเครื่องหมายบอกตำแหน่งไว้)
- ทดลองส่งค่าให้สเต็ปเปอร์มอเตอร์หมุนทวนเข็มไป 200 สเต็ป พบว่าหมุนไปได้ระยะหนึ่งแล้วหยุด (สังเกตุเครื่องหมายบอกตำแหน่ง) พบว่าตรงกับตำแหน่งเริ่มต้น
- ทำการทดลองซ้ำ และลองเปลี่ยนค่าของสเต็ปการหมุน พบว่าได้ผลเช่นกัน

การทดลองที่ 2.4 : ทดสอบความสามารถของโปรแกรมในการสั่งให้มอเตอร์หยุดหมุน
ทดลองสั่งให้สเต็ปเปอร์มอเตอร์หยุดหมุน พบว่าสามารถหยุดได้

การทดลองที่ 2.5 : ทดสอบโปรแกรมที่สั่งให้มอเตอร์หมุนวนต่อเนื่องกัน

ทดลองเขียนโปรแกรมแบบวนลูปสั่งให้สเต็ปเปอร์มอเตอร์หมุนตามเข็ม 100 สเต็ป ,หมุนทวนเข็ม 200 สเต็ป, หยุดหมุน(delay (2000)), หมุนตามเข็ม 300 สเต็ป ,หมุนทวนเข็ม 100 สเต็ป วนไปเรื่อยๆ พบว่าได้ผล

โปรแกรมส่วนควบคุมสเต็ปเปอร์มอเตอร์ไม่สามารถสั่งให้สเต็ปเปอร์มอเตอร์หมุนได้ถ้าจำนวนสเต็ปต่ำกว่า 3 สเต็ปติดกันได้ แต่สามารถสั่งให้สเต็ปเปอร์มอเตอร์หมุนไปซ้ายหรือขวาได้ โดย 1 รอบของการหมุนมี 200 สเต็ป

4.2 การทดลองในส่วนโปรแกรมวิเคราะห์ภาพ

วัตถุประสงค์ของการทดลอง

1. เพื่อศึกษาการส่งภาพของกล้องดิจิตอลเข้าสู่โปรแกรม Delphi
2. เพื่อศึกษาการใช้งานโปรแกรมแบบประยุกต์
3. เพื่อพัฒนากล้องให้มีความสามารถเพิ่มขึ้น

การทดลองที่ 3 : ทดสอบโปรแกรมในการรับภาพจากกล้องดิจิตอล

เป็นการทดสอบการเขียนโปรแกรมให้โปรแกรมสามารถรับภาพจากกล้องได้โดย แสดงผลที่ Image1 และบันทึกไฟล์เป็น Image1.bmp

ผลการทดลองเป็นดังนี้ โปรแกรมสามารถรับภาพจากกล้องได้ และบันทึกเป็นไฟล์ bitmap ได้สำเร็จ

การทดลองที่ 4 : ทดสอบโปรแกรมการวิเคราะห์สี

เป็นการทดสอบการเขียนโปรแกรมการวิเคราะห์สีของวัตถุ โดยให้แสดงเป็นภาพแบบขาว-ดำ ที่ความละเอียดต่างๆ โดยวัตถุที่ใช้วิเคราะห์สี คือลูกปิงปองสีพื้นต่างๆ , ฉากที่ใช้มีสีดำ

ตารางที่ 4.1 : การทดลองที่ 4 ทดสอบโปรแกรมการวิเคราะห์สี

ค่าความละเอียดที่ใช้ในการทดลอง	สี									
	ขาว	ดำ	ส้ม	ฟ้า	ชมพู	ม่วง	เขียวอ่อน	เหลือง	น้ำเงิน	แดง
128x96	ข	ด	ข	ข	ข	ด	ข	ข	ด	ด
160x120	ข	ด	ข	ข	ข	ด	ข	ข	ด	ด

176x144	ข	ด	ข	ข	ข	ด	ข	ข	ด	ด
240x176	ข	ด	ข	ข	ข	ด	ข	ข	ด	ด
320x240	ข	ด	ข	ข	ข	ด	ข	ข	ด	ด
352x288	ข	ด	ข	ข	ข	ด	ข	ข	ด	ด
640x480	ข	ด	ข	ข	ข	ด	ข	ข	ด	ด

หมายเหตุ 1. ในการทดลองมีการให้แสงที่คงที่เท่ากันตลอดทุกความละเอียด

2. ข หมายถึง ผลลัพธ์ที่ได้จากการวิเคราะห์สี มีสีขาว

ด หมายถึง ผลลัพธ์ที่ได้จากการวิเคราะห์สี มีสีดำ

ผลการทดลองเป็นดังนี้

โปรแกรมสามารถแยกความแตกต่างของสีได้ โดยสามารถแปลงสีที่มีสีอ่อน เช่น สีขาว สีเหลือง สีฟ้า สีชมพู สีเขียวอ่อน ให้เป็นสีขาวได้ และแปลงสีที่มีสีเข้ม เช่น สีดำ สีแดง สีม่วง สีน้ำเงิน ให้เป็นสีดำได้

การทดลองที่ 5 : ทดสอบโปรแกรมการวิเคราะห์หาขอบ

เป็นการทดสอบการเขียนโปรแกรมการวิเคราะห์หาขอบของวัตถุที่ความละเอียดต่างๆ วัตถุที่ใช้มีสีเหมือนกันแต่มีรูปทรงต่างกัน

ตารางที่ 4.2 : การทดลองที่5.1 ทดสอบโปรแกรมการวิเคราะห์หาขอบของรูปทรงต่างๆ

ค่าความละเอียดที่ใช้ในการทดลอง	รูปทรง			
	วงกลม	สี่เหลี่ยม	สามเหลี่ยม	หลายเหลี่ยม
128x96	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
160x120	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
176x144	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
240x176	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
320x240	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
352x288	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
640x480	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้

ผลการทดลองเป็นดังนี้

โปรแกรมสามารถบอกความแตกต่างของรูปทรงได้ โดยสามารถหาขอบของรูปได้หลายลักษณะ เช่น วงกลม สามเหลี่ยม สี่เหลี่ยม เป็นต้น

ตารางที่ 4.3 : การทดลองที่5.2 ทดสอบโปรแกรมการวิเคราะห์หาขอบของลายต่างๆ

ค่าความละเอียดที่ใช้ในการทดลอง	ลักษณะลาย			
	ตามขวาง	ตามยาว	ตัวหนังสือ	จุด
128x96	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
160x120	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
176x144	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
240x176	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
320x240	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
352x288	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้
640x480	หาขอบได้	หาขอบได้	หาขอบได้	หาขอบได้

ผลการทดลองเป็นดังนี้

โปรแกรมสามารถบอกความแตกต่างของลายได้ โดยสามารถหาขอบของภาพได้หลายลาย เช่น ลายจุด ลายตามขวาง ลายตามยาว เป็นต้น

การทดลองที่6 : ทดสอบโปรแกรมการหาจุดศูนย์กลางของภาพ

หาจุดศูนย์กลางของภาพที่ความละเอียดต่างๆ โดยลูกปิงปองที่ใช้เป็นลูกเดียวกันและอยู่ที่ตำแหน่งเดิม โดยให้แสงคงที่ตลอดเวลา จากผลการทดลองจะเห็นว่าค่าจุดศูนย์กลางของวัตถุมีค่าไม่เท่ากันที่ความละเอียดต่างกัน

4.3 การทดลองการรับส่งข้อมูลระหว่าง Delphi5 กับ C51

วัตถุประสงค์ของการทดลอง

เพื่อให้โปรแกรม Delphi 5 และ C51 สามารถติดต่อสื่อสารกันได้

การทดลองที่7 : ทดสอบการส่งค่าจากโปรแกรม Delphi ไปยัง C51

เป็นการทดสอบโปรแกรมในการส่งค่าการขจัดที่ได้ (แปลงเป็นจำนวนสเต็ปแล้ว) ไปยังโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์เพื่อสั่งให้มอเตอร์หมุนไปซ้ายหรือขวากี่สเต็ป หรือหยุดหมุน

ผลการทดลองเป็นดังนี้

1. ในขั้นแรกเราได้ทดลองส่งค่าออกทาง LCD ก่อนพบว่าได้ผล
2. ต่อมาเราจึงทดลองจริง พบว่าได้ผลเช่นกัน

การทดลองที่8 : เป็นการทดสอบโปรแกรมว่าสามารถทำงานแบบอัตโนมัติได้หรือไม่

ผลการทดลองพบว่าโปรแกรมสามารถทำงานแบบอัตโนมัติได้

บทที่ 5
บทสรุป

จากผลการทดลองในบทที่ 4 นั้น เราจะมาสรุปและวิจารณ์กันในบทนี้ พร้อมกับเสนอแนะความรู้ต่างๆที่จะเป็นประโยชน์ต่อไปสำหรับผู้ที่สนใจหรือต้องการศึกษาหาความรู้

5.1 สรุปและวิจารณ์ผลการทดลอง

- การทดลองบอร์ดไดเวอร์กับโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์

สรุปการทดลองที่ 1 จากการทดลองเราสามารถสรุปได้ว่าบอร์ดไดเวอร์สเต็ปเปอร์มอเตอร์สามารถนำมาใช้งานได้ตรงตามความต้องการของโครงงาน

สรุปการทดลองที่ 2 .1 จากการทดลองจะพบว่าเมื่อเราสั่งให้โปรแกรมส่งค่าสเต็ปการหมุนเป็น 1 สเต็ปหลายครั้งติดต่อกันจะมีผลทำให้สเต็ปมอเตอร์หมุนเพียงสเต็ปแรกสเต็ปเดียวเท่านั้น เนื่องมาจากการเขียนโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ ของเราเป็นแบบ Wave หากต้องการให้สเต็ปเปอร์มอเตอร์หมุนได้ละเอียดมากขึ้นควรเขียนโปรแกรมแบบที่สามารถสั่งงานได้ทีละสเต็ปหรือใช้ไมโครสเต็ป เป็นต้น

สรุปการทดลองที่ 2 .2 จากการทดลองจะพบว่าเมื่อเราสั่งให้โปรแกรมส่งค่าสเต็ปการหมุนเป็น 1 สเต็ปหลายครั้งติดต่อกันจะมีผลทำให้สเต็ปเปอร์มอเตอร์หมุนเพียงสเต็ปแรกสเต็ปเดียวเท่านั้น เนื่องมาจากการเขียนโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ ของเราเป็นแบบ Wave หากต้องการให้สเต็ปเปอร์มอเตอร์หมุนได้ละเอียดมากขึ้นควรเขียนโปรแกรมแบบที่สามารถสั่งงานได้ทีละสเต็ปหรือใช้ไมโครสเต็ป เป็นต้น

สรุปการทดลองที่ 2 .3 จากการทดลองนี้เราต้องการทดสอบการทำงานของโปรแกรมว่าสามารถทำให้สเต็ปเปอร์มอเตอร์หมุนได้ตามสเต็ปที่ต้องการหรือไม่ ซึ่งจากผลการทดลองพบว่าสเต็ปเปอร์มอเตอร์สามารถหมุนได้ตามสเต็ปที่ต้องการ

สรุปการทดลองที่ 2.4 จากผลการทดลองทำให้เราสามารถสั่งให้สเต็ปเปอร์มอเตอร์หยุดหมุนได้ และสามารถสั่งหยุดผ่านทางโปรแกรม Delphi5 ได้ด้วย

สรุปการทดลองที่ 2.5 จากการทดลองนี้เราต้องการทดสอบการทำงานของโปรแกรมว่า

สามารถทำให้สเต็ปเปอร์มอเตอร์หมุนได้ตามสเต็ปที่ต้องการหรือไม่ ซึ่งจากผลการทดลองพบว่าสเต็ปเปอร์มอเตอร์สามารถหมุนได้ตามสเต็ปที่ต้องการ

- **การทดลองในส่วนโปรแกรมวิเคราะห์ภาพ**

สรุปการทดลองที่ 3 โปรแกรมสามารถรับภาพจากกล้องได้ และสามารถปรับค่าความละเอียดของกล้องได้ จับภาพแล้วบันทึกเป็นไฟล์ Image1.bmp ได้

สรุปการทดลองที่ 4 โปรแกรมสามารถวิเคราะห์สีของวัตถุได้ และสามารถแปลงภาพสีให้เป็นภาพแบบขาว-ดำได้ ทุกค่าของความละเอียด

สรุปการทดลองที่ 5 โปรแกรมสามารถหาขอบของภาพจากวัตถุได้ไม่ว่าวัตถุจะมีรูปทรงอย่างไรมีสีแบบไหนมีลายด้วยรึเปล่า ที่สำคัญคือสีของวัตถุต้องตัดกับสีของฉาก และฉากควรเป็นสีพื้นๆเท่านั้น

สรุปการทดลองที่ 6 โปรแกรมสามารถหาจุดศูนย์กลางของภาพได้ทุกค่าของความละเอียด แต่ค่าจุดศูนย์กลางที่ความละเอียดต่างกันจะมีค่าไม่เท่ากัน เนื่องจากจำนวนพิกเซลในภาพ 1 ภาพมีการเปลี่ยนแปลง ซึ่งจำนวนพิกเซลในภาพมีความสัมพันธ์กับค่าความละเอียดที่เราเปลี่ยนแปลงด้วย

- **การทดลองการรับส่งข้อมูลระหว่าง Delphi กับ C51**

สรุปการทดลองที่ 7 โปรแกรม Delphi สามารถส่งค่าการขจัดไปยัง โปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ได้

สรุปการทดลองที่ 8 โปรแกรมสามารถติดตามวัตถุได้อย่างอัตโนมัติ และสามารถสั่งให้หยุดการติดตามได้จากหน้าจอ Delphi

- **สรุปผลการทดลองของโครงงานนี้**

1. เข้าใจการส่งภาพของกล้องดิจิตอลเข้าสู่โปรแกรม Delphi ในคอมพิวเตอร์
2. เข้าใจการใช้งานโปรแกรม Delphi แบบประยุกต์
3. สามารถพัฒนากล้องให้มีความสามารถเพิ่มขึ้นได้
4. สามารถพัฒนาโปรแกรมให้ควบคุมสเต็ปเปอร์มอเตอร์ด้วย C51 ได้
5. สามารถนำโครงงานนี้ไปเป็นแนวทางในการพัฒนาต่อได้ในอนาคต
6. สามารถนำโครงงานนี้ไปใช้ประโยชน์ต่างๆหรือร่วมกับโครงงานอื่นได้

5.2 ปัญหาและวิธีแก้ปัญหาที่พบในโครงงาน

1. ความไม่ละเอียดของโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ เราไม่สามารถเขียนโปรแกรมให้รันต่ำกว่า 3 สเต็ปติดต่อกันได้

วิธีแก้ : ควรหาวิธีเขียนโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ให้สามารถรัต่ำกว่า 3 สเต็ปได้ หรือใช้ไมโครสเต็ปเข้าช่วย

2. ความล่าช้าในกระบวนการวิเคราะห์ผลของโปรแกรมวิเคราะห์ภาพ เนื่องจากโปรแกรมวิเคราะห์ภาพทำการคำนวณล่าช้าเกินไป ทำให้ส่งค่าไปในส่วนควบคุมมอเตอร์ได้ช้า ซึ่งส่งผลให้การตอบสนองไม่ดีเท่าที่ควร

วิธีแก้ : ควรออกแบบการเขียนโปรแกรมส่วนวิเคราะห์ภาพให้เร็วกว่านี้

3. ปัญหาด้านเทคนิค การทำส่วนฮาร์ดแวร์ให้มีประสิทธิภาพนั้นยาก

วิธีแก้ : ควรค้นคว้าหาความรู้ด้านเครื่องกลเพิ่มเติม

4. ระยะเวลาในการทำงานไม่เพียงพอต่อการทำงาน

วิธีแก้ : ควรมีการวางแผนการทำงานล่วงหน้าแต่เนิ่นๆ และเริ่มทำงานทันทีที่พร้อม

5. ความรู้ที่ไม่ค่อยเพียงพอ

วิธีแก้ : ควรศึกษาหาความรู้เพิ่มเติมในเรื่องที่เกี่ยวข้องกับโครงงานนั้นๆ

6. ปัญหาทางด้านการเงิน

วิธีแก้ : ควรเก็บสะสมเงินไว้ล่วงหน้าเยอะๆ

5.3 ข้อเสนอแนะเพื่อการทำโครงงานต่อไป

1. ควรจะพัฒนาความสามารถของกล้องให้เพิ่มมากขึ้น
2. ควรจะนำไปพัฒนาให้เกิดประโยชน์และสามารถใช้กับงานจริงได้ เช่น
   - สามารถนำไปติดกับจรวดนำวิถีให้สามารถติดตามเป้าหมายได้
   - สามารถนำไปติดตั้งกับกล้องถ่ายวิดีโอ เพื่อให้การบันทึกภาพนั้นไม่เกิดการสั่น
3. ควรพัฒนาApplication ให้มากขึ้นกว่าเดิม เช่น
   - สามารถแยกแยะวัตถุได้อย่างถูกต้องเมื่อมีวัตถุที่เป็นตัวล่ออยู่ด้วย
   - สามารถแยกแยะรูปทรงของวัตถุได้
   - สามารถติดตามวัตถุได้หลายมิติ

บรรณานุกรม

1. Archilla,L.E.., **"Lear Com 8051 Assembler User's Reference Manual,"** Lear Com Company,1988.
2. Ayala, K.J.,**"The 8051 Microcontroller Architecture, Programing and Applications,"** West Publishing Company,1991.
3. Coffron, W.J., **"Z80 Application,"** Sybex Inc., 1983.
4. Coffron, W.J., **"The IBM PC connection,"** Sybex Inc., 1985.
5. Houghton, W.G., **"Mastering Digital Device Control,"** Sybex Inc., 1987.
6. Intel Corporation., **"Microcontroller HandBook,"** Intel Corporation,1992.
7. Kohn, K.P., **"Multicomputer Communication By Serial Bus,"** Seimens Components XVIII, No 6, 1983
8. Lichti, W.P., **"Introduction to microprocessor / controller Design,"** Lessen Technical Press, 1991
9. Adrian Low , **"INTRODUCTORY COMPUTER VISION AND IMAGE PROCESSING"** , 1ST EDITION , McGRAW-HILL , BOOK COMPANY ., ENGLAND : 1991.

ประวัติผู้ดำเนินโครงงาน

ชื่อ-นามสกุล	นางสาวปาริชาติ อรุณอาศิรกุล
วันเดือนปีเกิด	9 ตุลาคม 2521
สถานที่เกิด	จังหวัดพิษณุโลก
ที่อยู่	176/7 ถนนพิชัยสงคราม ตำบลในเมือง อำเภอเมือง จังหวัดพิษณุโลก 65000
ประวัติการศึกษา	พุทธศักราช 2536 จบการศึกษาระดับมัธยมศึกษาตอนต้นจาก โรงเรียนเฉลิมขวัญสตรี จังหวัดพิษณุโลก พุทธศักราช 2539 จบการศึกษาระดับมัธยมศึกษาตอนปลายจาก โรงเรียนเฉลิมขวัญสตรี จังหวัดพิษณุโลก

ชื่อ-นามสกุล	นายมุนิน ยศถา
วันเดือนปีเกิด	14 ตุลาคม 2523
สถานที่เกิด	จังหวัดอ่างทอง
ที่อยู่	28/16 หมู่ 4 ตำบลบ้านหลุม
	อำเภอเมือง จังหวัดสุโขทัย 64000
ประวัติการศึกษา	พุทธศักราช 2537 จบการศึกษาระดับมัธยมศึกษาตอนต้นจาก
	โรงเรียนสุโขทัยวิทยาคม จังหวัดสุโขทัย
	พุทธศักราช 2539 จบการศึกษาเทียบเท่าระดับมัธยมศึกษาตอนปลายจาก
	การศึกษานอกโรงเรียน อำเภอเมือง จังหวัดสุโขทัย