หุ่นยนต์ 4 ขา

QUADRAPOD

โดย

นางสาวสุทธิณี สมโภชพิสุทธิ์

นายสุประวิทย์ สามารถ

นางสาวสุพิณญา สุภาแจ่ม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

หุ่นยนต์ 4 ขา

QUADRAPOD

โดย

นางสาวสุทธิณี	สมโภชพิสุทธิ์	47010861
นายสุประวิทย์	สามารถ	47010871
นางสาวสุพิณญา	สุภาแจ่ม	47010878

อาจารย์ที่ปรึกษา

ผศ.ดร.ยุทธนา คิดใจเดียว

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ปริญญานิพนธ์ ปีการศึกษา 2550

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์ 4 ขา

QUADRAPOD

ผู้จัดทำ

1. นางสาวสุทธิณี สมโภชพิสุทธิ์
2. นายสุประวิทย์ สามารถ
3. นางสาวสุพิณญา สุภาแจ่ม

.....อาจารย์ที่ปรึกษา

(ผศ.ดร.ยุทธนา คิดใจเดียว)

หุ่นยนต์ 4 ขา

นางสาว สุทธิณี สมโภชพิสุทธิ์ รหัส 47010861
นาย สุประวิทย์ สามารถ รหัส 47010871
นางสาว สุพิณญา สุภาแจ่ม รหัส 47010878
ผศ.ดร.ยุทธนา คิดใจเดียว อาจารย์ที่ปรึกษา
ปีการศึกษา 2550

บทคัดย่อ

รายงานฉบับนี้นำเสนอการควบคุมหุ่นยนต์เดิน 4 ขา ซึ่งใช้การทำงานของ State Transition Machine และ Image Processing โดยเริ่มจากการประมวลผลภาพที่ได้จากกล้อง หลังจากนั้นจะส่งสัญญาณไปเป็นอินพุทให้กับหุ่นยนต์ เพื่อเป็นทิศทางให้กับหุ่นยนต์ จากนั้นหุ่นจะตรวจสอบสถานะของขาจากการเดินครั้งล่าสุดของแต่ละขา เพื่อตัดสินใจในการเดินสถานะต่อไป ซึ่งสถานะต่อไปของแต่ละขาจะเปลี่ยนเป็นสัญญาณพัลส์ที่มีพัลส์วิธด์ต่างกันมาใช้ในการขับมอเตอร์ให้หุ่นยนต์เคลื่อนที่ไปในทางของภาพที่ได้รับมา

การออกแบบการเดินของหุ่นยนต์ 4 ขานี้สามารถนำไปพัฒนารูปแบบการเดินได้โดยไม่ต้องเปลี่ยนโครงสร้างของโปรแกรม

QUADRAPOD

Ms. Suttinee Sompochpisut ID 47010861

Mr. Suprawit Samart ID 47010871

Ms. Supinya Supajam ID 47010878

Assistant Prof.Dr. Yuttana Kidjaideaw (Advisor)

Educational Year 2007

Abstract

This project presents QUADRAPOD controlling via the State Transition Machine and Image Processing. Firstly, an image is captured from the camera and processed by the computer unit, then the signal is sent to the robot for indicating the direction. Next, the robot checks the previous state of each leg. Finally, the robot sends pulse-width signal to drive motors moving to the specified object.

The design of QUADRAPOD has an advantage of easy implement without the need of rewritten the program structure.

กิตติกรรมประกาศ

รายงานฉบับนี้สามารถลุล่วงไปได้ด้วยดี เพราะได้รับความช่วยเหลือจากหลายบุคคล โดยเฉพาะอย่างยิ่ง ผศ.ดร.ยุทธนา คิดใจเดียว ที่คอยให้คำปรึกษา และคำแนะนำ อีกทั้งอาจารย์ภาควิศวกรรมการวัดคุมที่เอื้อเฟื้อสถานที่และเครื่องมือ ขอขอบพระคุณ คุณพ่อ คุณแม่ที่เป็นกำลังใจและให้การสนับสนุนในทุกๆด้าน โดยเฉพาะทางด้านค่าใช้จ่าย ขอบคุณเพื่อนๆ ทุกคนที่คอยช่วยเหลือในการปฏิบัติงานจนทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

จึงขอขอบคุณมา ณ ที่นี้

ผู้จัดทำ

สารบัญ

บทคัดย่อไทย

บทคัดย่ออังกฤษ

กิตติกรรมประกาศ

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1 บทนำ 1

บทที่ 2 ทฤษฎีการออกแบบโครงสร้าง 3

2.1 เซอร์โวมอเตอร์ 3

2.1.1 หลักการทำงานของเซอร์โวมอเตอร์ 3

2.1.2 ภาคการทำงานของเซอร์โวมอเตอร์ 5

2.2 แบบขา 7

2.2.1 ขาสองข้อต่ออย่างง่าย (Simple Two Link Leg) 8

2.2.2 ขาเพนโทกราฟ์ (Pantograph Leg) 8

2.3 การออกแบบและการสร้าง 11

บทที่ 3 ทฤษฎีและหลักการทำงาน 17

3.1 ทฤษฎีการประมวลผลภาพ 17

3.1.1 การกำหนดสี 17

3.1.2 องค์ประกอบของสี 18

3.1.3 ภาพดิจิตอล 19

3.1.3.1 การแทนภาพด้วยข้อมูลดิจิตอล 19

3.1.3.2 ลักษณะการจัดเก็บข้อมูลดิจิตอล 20

3.1.4 การประมวลผลภาพเชิงตัวเลข 20

3.1.5 สัญญาณข้อมูลภาพจากดิจิตอลวีดีโอ 21

สารบัญ(ต่อ)

3.1.6 ข้อมูลภาพชนิดบิตแมป 21

3.1.7 การสร้างภาพไบนารี 21

3.1.7.1 การหาค่าเทรชโฮลโดยการกำหนดไว้ล่วงหน้า 22

3.1.7.2 การฟิลเตอร์แบบค่ามีเดียน (Median Filtering) 23

3.1.8 Shape Analysis 24

3.2 Moments 24

3.3 ทฤษฎี State Transition Machine 24

3.4 วงจรที่ใช้ในการควบคุมหุ่นยนต์เดิน 4 ขา 43

บทที่ 4 การทดลองและผลการทดลอง 45

4.1 การทดลอง 45

4.2 ผลการทดลอง 45

บทที่ 5 วิเคราะห์และสรุปผลการทดลอง 49

ภาคผนวก

เอกสารอ้างอิง

สารบัญรูป

รูปที่ 1.1	แสดงภาคการทำงานของหุ่นยนต์เดิน 4 ขา	2
รูปที่ 2.1	แสดงการตอบสนองของเซอร์โวมอเตอร์ต่อสัญญาณพัลส์ในความถี่ที่ต่างกัน	4
รูปที่ 2.2	แสดงภาคการทำงานของเซอร์โวมอเตอร์	5
รูปที่ 2.3	วงจรภาคขยายเซอร์โว (SERVO AMP) และภาคขับเซอร์โว (SERVO DRIVE)	6
รูปที่ 2.4	เซอร์โวมอเตอร์พร้อมเฟือง	7
รูปที่ 2.5	ส่วนประกอบของเซอร์โวมอเตอร์	7
รูปที่ 2.6	แสดงลักษณะของขาข้อต่ออย่างง่าย	8
รูปที่ 2.7	แสดงรูปแบบขาของหุ่นแบบ เพนโทรกราฟ	9
รูปที่ 2.8	แสดงการเคลื่อนที่ของขาแบบเพนโทรกราฟ	10
รูปที่ 2.9	แสดงรูปแบบขาของหุ่น	11
รูปที่ 2.10	แผ่นลำตัวหุ่นด้านบน	12
รูปที่ 2.11	แผ่นลำตัวหุ่นด้านล่าง	13
รูปที่ 2.12	แท่นยึดเซอร์โว	15
รูปที่ 2.13	ขาหุ่น	15
รูปที่ 2.14	หุ่นยนต์เดิน 4 ขา	16
รูปที่ 2.15	หุ่นยนต์และวงจรที่ใช้ควบคุมหุ่นยนต์	16
รูปที่ 3.1	กล่องสี RGB	18
รูปที่ 3.2	Moor Machines	25
รูปที่ 3.3	รูปแบบการเดินไปข้างหน้า	26
รูปที่ 3.4	รูปแบบการเลี้ยวซ้าย	26
รูปที่ 3.5	รูปแบบการเลี้ยวขวา	27
รูปที่ 3.6	แสดงสถานะขาหุ่นในสภาวะต่างๆ	27
รูปที่ 3.7	State diagram ของระบบ	37
รูปที่ 3.8	State diagram ของขาที่ 1	38
รูปที่ 3.9	State diagram ของขาที่ 2	39
รูปที่ 3.10	State diagram ของขาที่ 3	40

สารบัญรูป(ต่อ)

รูปที่ 3.11 State diagram ของขาที่ 4 41

รูปที่ 3.12 แสดงการทำงานของคอนโทรลเลอร์เซลล์ที่1 42

รูปที่ 3.13 แสดงวงจรแปลงเป็นแรงดันไฟ 5 V 43

รูปที่ 3.14 แสดงวงจรที่ใช้แปลงสัญญาณ TTL กับ RS-232 43

รูปที่ 3.15 แสดงวงจรไมโครคอนโทลเลอร์ที่ใช้ควบคุมหุ่นยนต์ 44

รูปที่ 4.1 แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งต่างๆ 46

รูปที่ 4.2 แสดงการประมวลผลภาพ ที่ไม่มีการลดNOISE 46

รูปที่ 4.3 แสดงการประมวลผลภาพ ที่มีการลดNOISE 47

รูปที่ 4.4 แสดงการประมวลผลภาพ ที่มีวัตถุสีเดียวกัน 2 อัน 47

รูปที่ 4.5 แสดงการประมวลผลภาพในสภาวะแสงน้อย 48

รูปที่ 4.6 แสดงการประมวลผลภาพในสภาวะแสงมาก 48

สารบัญตาราง

ตารางที่ 3.1 แสดงบิตควบคุม และพัลส์ที่ป้อนให้เซอร์โวมอเตอร์ตัวบน 28

ตารางที่ 3.2 แสดงบิตควบคุม และพัลส์ที่ป้อนให้เซอร์โวมอเตอร์ตัวล่าง 28

ตารางที่ 3.3 แสดงสเตจการเดิน 29

ตารางที่ 3.4 แสดงลักษณะอินพุท และคำสั่งการเดิน 30

ตารางที่ 3.5 ตารางแสดงค่า state การเดินของขาที่ 1 31

ตารางที่ 3.6 ตารางแสดงค่า state การเดินของขาที่ 2 32

ตารางที่ 3.7 ตารางแสดงค่า state การเดินของขาที่ 3 33

ตารางที่ 3.8 ตารางแสดงค่า state การเดินของขาที่ 4 34

บทที่ 1
บทนำ

ปัจจุบันได้มีการสร้างและพัฒนา หุ่นยนต์ในรูปแบบต่างๆ มากยิ่งขึ้น เนื่องจากความต้องการหุ่นยนต์ที่สามารถทำการประมวลผลได้อย่างรวดเร็ว และมีความใกล้เคียงกับสิ่งมีชีวิต จึงได้มีการนำเอาเครือข่ายปัญญาประดิษฐ์มาประยุกต์ใช้งาน อาทิเช่นการประมวลผลสัญญาณ การสังเคราะห์เสียง การจดจำเสียง การจดจำรูปแบบ และระบบควบคุม เป็นต้น

จึงได้มีแนวคิดที่จะสร้างและพัฒนาหุ่นยนต์ ที่สามารถเคลื่อนที่โดยใช้ขา เป็นตัวขับเคลื่อน โดยได้เลือกรูปแบบขาที่ใช้งาน เพื่อใช้ในการพยุงและขับเคลื่อนตัวหุ่นยนต์ และควบคุมหุ่นยนต์ด้วยโปรแกรม โดยใช้หลัก State Transition Machine เพื่อจะสามารถพัฒนาในส่วนของโปรแกรมต่อไปได้โดยง่าย ซึ่งหุ่นยนต์ที่ขับเคลื่อนด้วยขานี้ สามารถที่จะพัฒนาให้เป็นหุ่นยนต์ที่ใช้งานแทนมนุษย์ได้ในจุดที่เป็นอันตรายและเสี่ยง ต่อการทำงาน เช่น หุ่นยนต์ทำความสะอาดกระจกบนตึกสูง หุ่นยนต์ตรวจสอบและตรวจซ่อมโครงสร้างสะพานในที่สูง หุ่นยนต์ตรวจสอบรอยรั่วบนถังก๊าซ ระบบส่งจ่ายก๊าซ หรือ การปิโตรเคมีเป็นต้น

ชื่อโครงงาน

หุ่นยนต์ 4 ขา (QUADRAPOD)

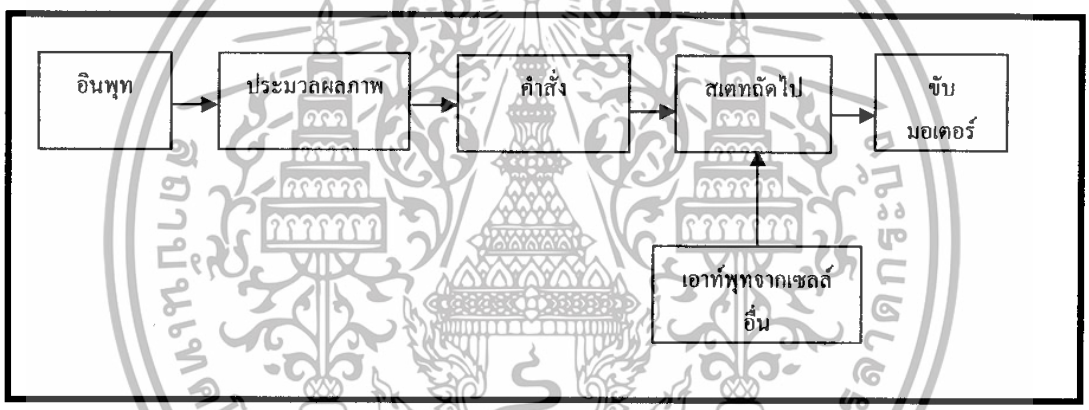
วัตถุประสงค์

- เพื่อการเรียนรู้หลักการทำงานของระบบควบคุมแบบ State Transition Machine
- เพื่อการเรียนรู้การทำงานของไมโครคอนโทรลเลอร์ ซึ่งใช้ภาษาซีเป็นคำสั่งการทำงาน
- เพื่อออกแบบและสร้างหุ่นยนต์ 4 ขาขนาดเล็กซึ่งสามารถติดตามวัตถุที่กำหนดได้ เพื่อเป็นหุ่นยนต์ต้นแบบ ในการพัฒนาหุ่นยนต์ต่อไป
- เพื่อเรียนรู้หลักการทำงานการประมวลผลภาพ

คุณสมบัติ

โครงงานหุ่นยนต์เดิน 4 ขา ในส่วนควบคุมหุ่นยนต์ประกอบด้วย

- ส่วนควบคุมหรือสมองที่ใช้ไมโครคอนโทรลเลอร์เป็นหัวใจหลัก
- ส่วนงานการประมวลผลภาพจากกล้อง เพื่อจับภาพวัตถุ
- มีระบบควบคุมการทำงานเป็นแบบ State Transition Machine
- ส่วนโครงสร้างของหุ่นเป็นพลาสติกอะครีลิค
- ส่วนขับเคลื่อนโดย DC SERVO FP – S3001 จำนวน 8 ตัว ในการควบคุมการเคลื่อนไหวของขาทั้ง 4 ขา
- Block Diagram ของระบบ



รูปที่ 1.1 แสดงภาคการทำงานของหุ่นยนต์เดิน 4 ขา

ขอบเขตของการทำงาน

- สร้างหุ่นยนต์ที่สามารถเดิน 4 ขาได้
- หุ่นยนต์สามารถเดินตามวัตถุที่กำหนดได้

บทที่ 2
ทฤษฎีการออกแบบโครงสร้าง

2.1 เซอร์โวมอเตอร์

2.1.1 หลักการทำงานของเซอร์โวมอเตอร์

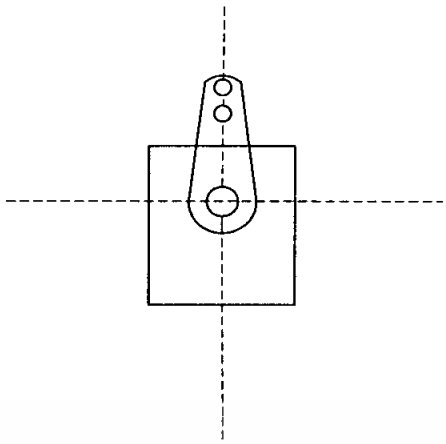
เซอร์โวมอเตอร์ ประกอบด้วย มอเตอร์ความเร็วสูง ภายในมีเฟืองทดรอบ ให้หมุนช้าลง เพื่อจะได้มีกำลังแรงบิดที่สูงขึ้น นอกจากนี้ยังมีวงจรควบคุมมอเตอร์ ซึ่งวงจรนี้จะนำค่าแรงดันเฉลี่ยของพัลส์รูปสี่เหลี่ยม เข้าไปเปรียบเทียบกับค่าแรงดันค่าหนึ่งที่มีอยู่ในวงจร ถ้าค่าต่างกัน วงจรควบคุมจะสั่งให้มอเตอร์หมุนไปตามทิศทาง ซึ่งขึ้นอยู่กับขนาดความกว้างพัลส์ โดยที่แกนเฟืองทดรอบจะถูกพ่วงไปขับแกนของตัวต้านทานปรับค่าได้(Potentiometer) ซึ่งอยู่ในวงจรควบคุมมอเตอร์ ในขณะที่มอเตอร์หมุนตัวต้านทานปรับค่าได้ จะถูกปรับค่าทำให้ค่าแรงดันเปรียบเทียบของวงจรควบคุมมอเตอร์เปลี่ยนไปด้วย จนกระทั่งค่าเฉลี่ยของพัลส์ในวงจรควบคุมมอเตอร์ เท่ากับค่าเฉลี่ยของพัลส์ที่เข้ามา จึงทำให้มอเตอร์หยุดหมุนได้

เซอร์โวมอเตอร์จะมีสายไฟ 3 เส้นคือ สายไฟเลี้ยง สายกราวด์ และสายสัญญาณพัลส์ควบคุม ซึ่งลักษณะของสัญญาณ พัลส์ที่ใช้ควบคุมตำแหน่งของเซอร์โวมอเตอร์ จะเป็นการส่งพัลส์ที่มีความกว้างต่างกัน เพื่อควบคุมให้เซอร์โวมอเตอร์ หมุนไปยังตำแหน่งที่ต้องการ โดยที่มีความกว้างของพัลส์ จะเป็นตัวกำหนดขนาด และทิศทางของการหมุนแกนเซอร์โวมอเตอร์สำหรับคาบเวลา หรือระยะห่างระหว่างพัลส์แต่ละลูก จะเป็นตัวกำหนดแรงบิดของมอเตอร์

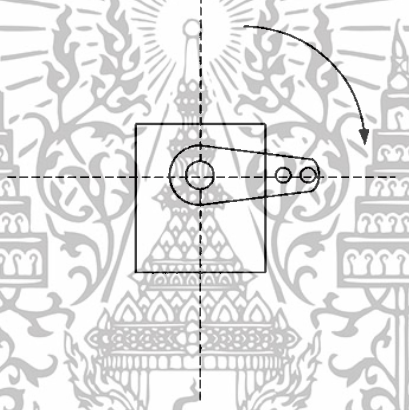
ถ้ากำหนดให้ในสภาวะปกติ เมื่อป้อนพัลส์ สี่เหลี่ยม ที่มีความกว้างขนาด 1.5 ms ให้กับเซอร์โวมอเตอร์ แขนของเซอร์โวมอเตอร์จะอยู่ตำแหน่งกลาง

เมื่อป้อนพัลส์ สี่เหลี่ยม ที่มีความกว้างขนาด 1 ms ให้กับเซอร์โวมอเตอร์ แขนของเซอร์โวมอเตอร์จะหมุนตามเข็มนาฬิกา

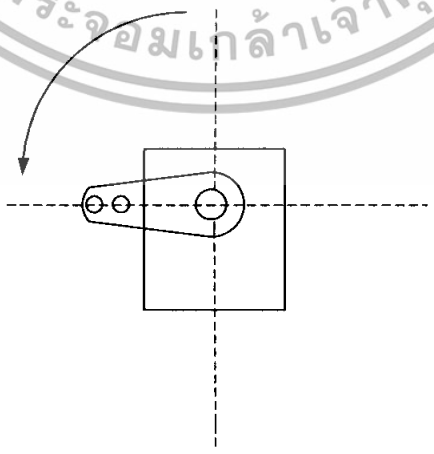
เมื่อป้อนพัลส์ สี่เหลี่ยม ที่มีความกว้างขนาด 2 ms ให้กับเซอร์โวมอเตอร์ แขนของเซอร์โวมอเตอร์จะหมุนทวนเข็มนาฬิกา ตามที่แสดงไว้ในรูปที่ 2.1 ดังต่อไปนี้



ก. แสดงการตอบสนองของเซอร์โวเมื่อจ่ายพัลส์ขนาด 1.4 ms



ข. แสดงการตอบสนองของเซอร์โวเมื่อจ่ายพัลส์ขนาด 1 ms



ค. แสดงการตอบสนองของเซอร์โวเมื่อจ่ายพัลส์ขนาด 2 ms

รูปที่ 2.1 แสดงการตอบสนองของเซอร์โวมอเตอร์ต่อสัญญาณพัลส์ในความถี่ที่ต่างกัน

ดังนั้นถ้าจ่ายสัญญาณพัลส์ที่มีความกว้างมากหรือน้อยกว่าความกว้างของพัลส์ 1.5 ms ก็จะทำให้เซอร์โวมอเตอร์หมุนต่างทิศกัน ทั้งตามเข็มและทวนเข็มนาฬิกา โดยตำแหน่งของแขนหมุนเซอร์โวมอเตอร์จะเบี่ยงเบนออกจากจุดกึ่งกลางเป็นสัดส่วนกับความกว้างของพัลส์ที่จ่ายให้

2.1.2 ภาคการทำงานของเซอร์โวมอเตอร์

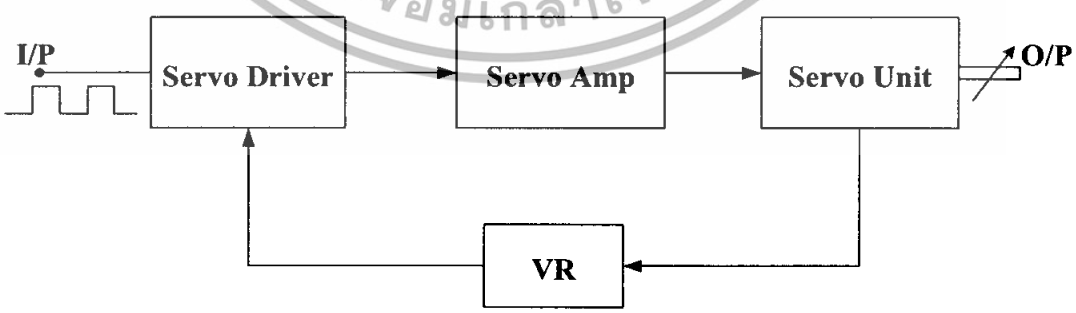
ในเซอร์โวมอเตอร์หนึ่งตัวจะประกอบไปด้วย 3 ภาคการทำงานแต่ละภาคมีหน้าที่และการทำงานดังนี้คือ

ภาคขับเซอร์โว ประกอบด้วย วงจรสร้างสัญญาณพัลส์ และวงจรเปรียบเทียบสัญญาณพัลส์ที่สร้างขึ้น กับสัญญาณพัลส์ I/P ที่รับเข้ามา

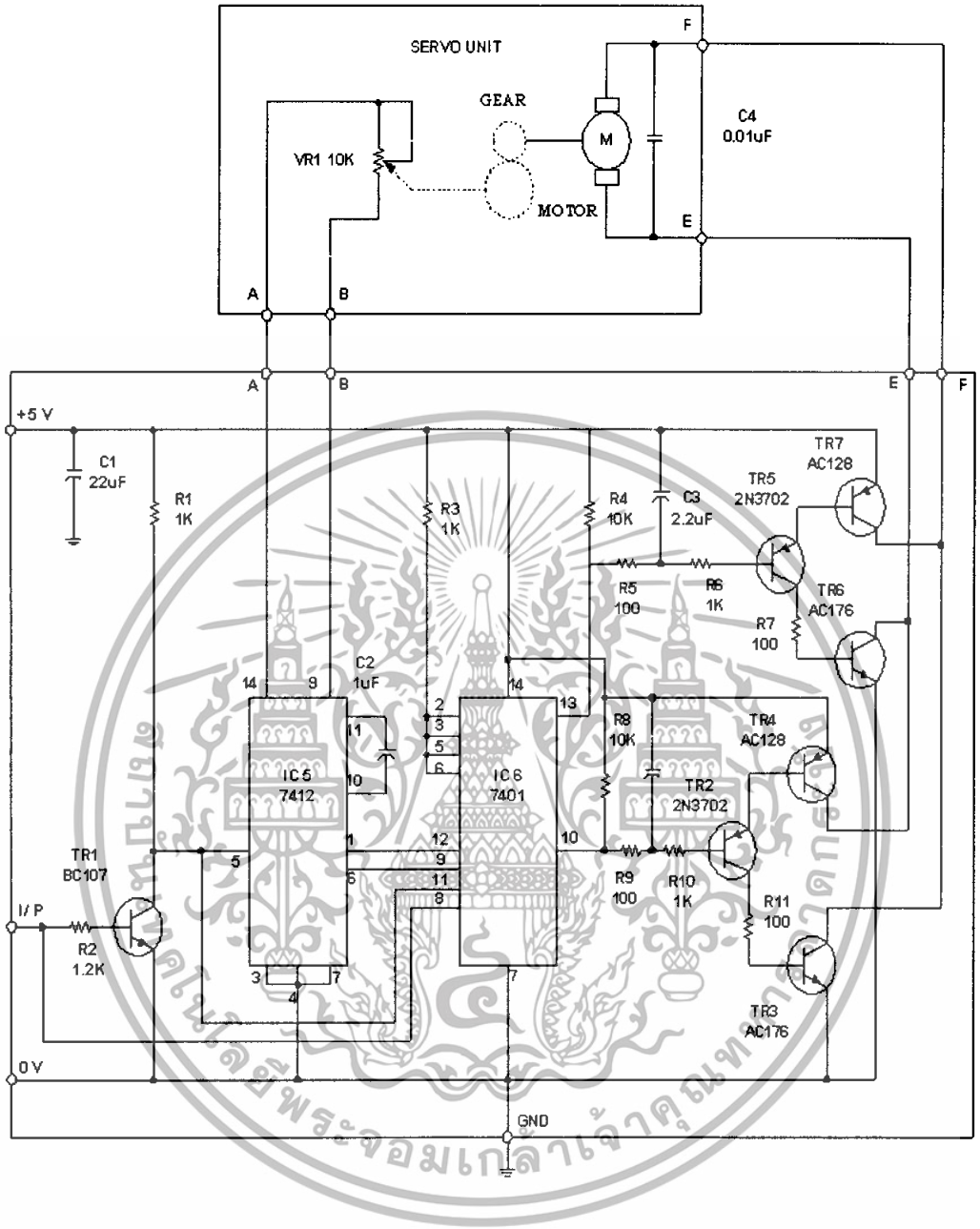
ภาคขยายเซอร์โว ประกอบด้วย วงจร RC Network ที่ช่วยหน่วงสัญญาณให้เซอร์โวสามารถทำงานได้ตลอดช่วงคาบเวลา จนกระทั่งมีสัญญาณลูกต่อไปมา รวมถึงวงจรกลับขั้วแรงดันไฟฟ้าควบคุมทิศทางการหมุนของมอเตอร์

หน่วยเซอร์โว ประกอบด้วย มอเตอร์ความเร็วสูง เฟืองทดรอบ แกนหมุน อุปกรณ์ต่าง ๆ และตัวต้านทานปรับค่าได้ ทำหน้าที่ป้อนกลับตำแหน่ง (Position Feedback)

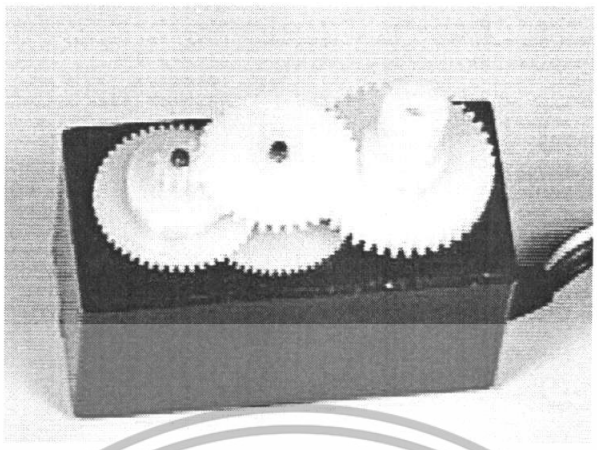
ซึ่งในขณะที่มอเตอร์หมุนตัวต้านทานปรับค่าได้จะเปลี่ยนแปลงค่าซึ่งส่งผลให้ค่าป้อนกลับเปลี่ยนแปลงไปจากนั้นนำค่าป้อนกลับมาปรับและเปรียบเทียบค่าความกว้างของพัลส์ที่ภาคขับเซอร์โว เมื่อขนาดความกว้างของพัลส์มีค่าแรงดันเฉลี่ยเท่ากันมอเตอร์จะหยุดหมุนทันที ซึ่งรูปที่ 2.2 ได้แสดงภาคการทำงานของเซอร์โวมอเตอร์ตามที่ได้กล่าวไว้ในข้างต้น และได้แสดงไว้ดังรูปต่อไปนี้



รูปที่ 2.2 แสดงภาคการทำงานของเซอร์โวมอเตอร์



รูปที่ 2.3 วงจรภาคขยายเซอร์โว (SERVO AMP) และภาคขับเซอร์โว (SERVO DRIVE)



รูปที่ 2.4 เซอร์โวมอเตอร์พร้อมเฟือง

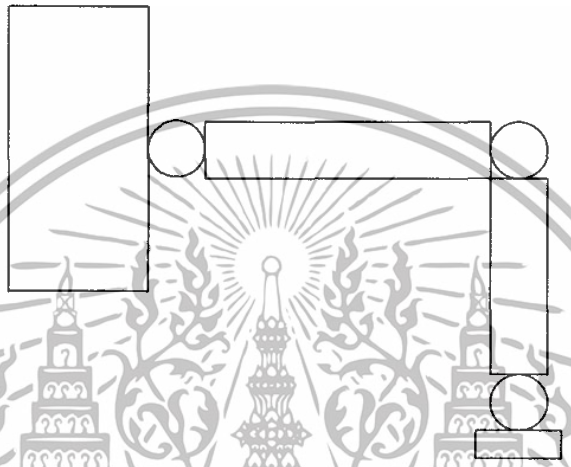
รูปที่ 2.5 ส่วนประกอบของเซอร์โวมอเตอร์

2.2 แบบขา

สำหรับการเลือกลักษณะขาของหุ่นยนต์ สิ่งที่สำคัญต่อการเลือกขาคือ จะต้องเลือกแบบที่มีลักษณะการเคลื่อนไหวมากที่สุด ให้เหมาะสมกับรูปร่างลักษณะโครงสร้างทางกายภาพของตัวหุ่นยนต์ ลักษณะการเดินของขาแต่ละแบบ จะขึ้นอยู่กับข้อจำกัดทางกายภาพของขาด้วย ซึ่งในปัจจุบันมีการออกแบบขาหุ่นยนต์อยู่หลายชนิด แต่ละชนิดจะมีคุณสมบัติเฉพาะแบบ ดังต่อไปนี้

2.2.1 ขาสองข้อต่ออย่างง่าย (Simple Two Link Leg)

ขาชนิดนี้จะมีลักษณะเป็นสองท่อน แต่ละท่อนจะต่อผ่านข้อต่อดังรูปที่ 2.6 ซึ่งสามารถควบคุมลักษณะการเดินได้ โดยการควบคุมของขาแต่ละท่อนซึ่งจะเป็นตัวกำหนดตำแหน่งปลายขาของหุ่นยนต์ ส่วนของขาทั้งหมดจะต่อเข้ากับเดือยที่โคนขาเพื่อใช้ในการก้าวขา และยึด หดขา



รูปที่ 2.6 แสดงลักษณะของขาข้อต่ออย่างง่าย

วิธีการทำงานของข้อต่อ

มีหลายวิธีที่จะทำให้ข้อต่อทำงานได้ โดยใช้ลักษณะการขับเคลื่อนของข้อต่อสำหรับแบบนี้อาจติดตั้งมอเตอร์เข้าที่ข้อต่อโดยตรง หรือใช้โซ่ สายพาน สกรู และส่งกำลังจากมอเตอร์ที่ติดตั้งอยู่ในลำตัวบริเวณโคนขา เพื่อกำหนดมุมที่ข้อต่อในการก้าวเดินของขาหุ่น

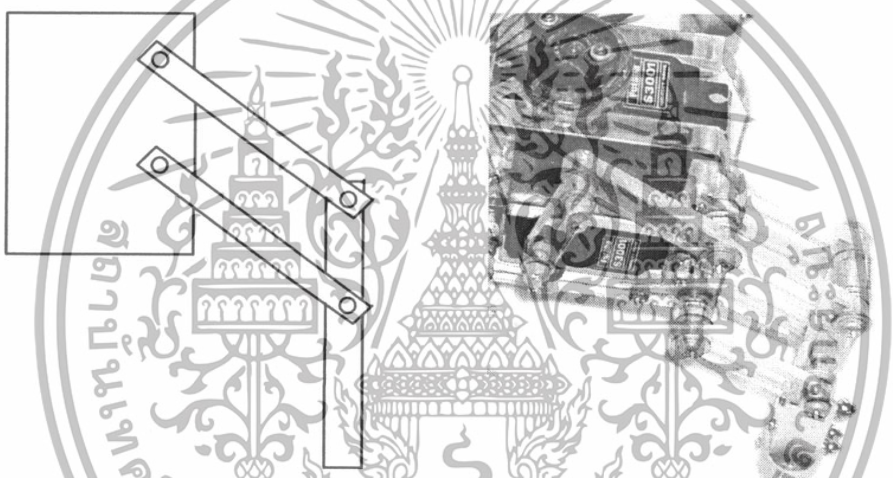
จุดด้อยประการสำคัญของขาแบบนี้คือ เราจำเป็นต้องใช้ตัวขับเคลื่อนอยู่ใกล้กับข้อต่อมากที่สุด การติดตั้งตัวขับเคลื่อนเข้าที่ข้อต่อเข่าทำให้เกิดผลประทบทางไดนามิคต่อขาหุ่น ซึ่งต้องมีการชดเชยโดยใช้ตัวควบคุม ซึ่งจะทำให้ต้องเพิ่มความซับซ้อนให้กับอัลกอริธึมในการเคลื่อนที่ของขา รวมทั้งยังต้องการมอเตอร์ที่มีกำลังสูงที่ข้อต่อส่วนสะโพก เพื่อใช้ในการเคลื่อนขาที่มีมวลมาก ซึ่งเราก็สามารถแก้ปัญหาเหล่านี้ได้โดยการติดตั้งตัวขับเคลื่อนที่ฐานของขาแต่จะเป็นการเพิ่มความซับซ้อนทางแมคคานิกส์

2.2.2 ขาเพนโทกราฟ

ขาเพนโทกราฟ (Pantograpgh Leg) นี้จะประกอบด้วยคานสี่ท่อนขนานกันเป็นรูปสี่เหลี่ยมด้านขนาน ดังรูปที่ 2.7

เป็นแบบที่มีผู้ใช้กันอย่างแพร่หลาย เพราะการควบคุมและระบบทางแมคคานิกส์ของขาทำง่ายซึ่งลดการประมวลที่ซับซ้อนในการควบคุมลงได้

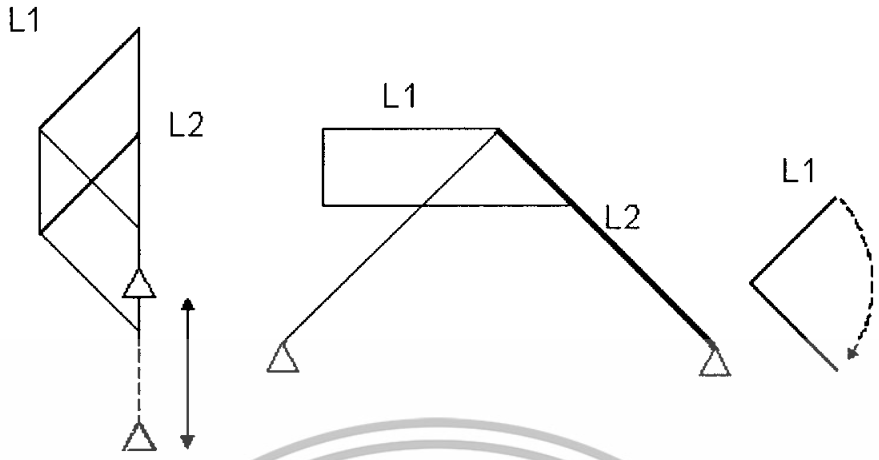
ขอบเขตการเคลื่อนไหวของขา (Workspace) แสดงให้เห็นว่าโครงสร้างแบบนี้ยังมีการคับปลิ้ง (Coupling) ของข้อต่อเกิดขึ้นในการเคลื่อนปลายขา ทำให้ปลายขาเคลื่อนที่เป็นเส้นโค้ง เนื่องจากว่าขาแบบนี้เป็นแบบที่มีลักษณะทางเรขาคณิตอย่างง่าย ๆ ทำให้ผู้ออกแบบมักเลือกขาแบบนี้มาใช้ก่อนแบบอื่น แต่ปัญหาทางแมคคานิกส์ที่พบระหว่างการสร้างขาต้นแบบ ทำให้เราต้องดัดแปลงขาแบบนี้อีกครั้งก่อนที่จะตัดสินใจใช้ขาแบบนี้ให้เป็นขาที่จะนำไปใช้งาน



รูปที่ 2.7 แสดงลักษณะขาของหุ่นแบบ เพนโทรกราฟ

วิธีการทำงานของข้อต่อ

ส่วนประกอบที่ใช้ในการควบคุมเพื่อให้เขาเกิดการเคลื่อนที่อาจใช้ กระบอกสูบ ในระบบของไฮโดรลิก นิวเมตริก หรือมอเตอร์ ซึ่งขึ้นอยู่กับความเหมาะสมและความต้องการที่จะนำไปใช้ ซึ่งไม่ว่าจะเป็นแบบใด ข้อควรพิจารณาที่สำคัญในระหว่างการออกแบบขา คือขนาดของขอบเขตการเคลื่อนไหวของขา (Workspace) ที่ต้องการ คือ การยกขา (Lifting), การยืดขา (Extending) และ การแกว่งขา (Rotating) ดังที่แสดงไว้ในรูปที่ 2.8



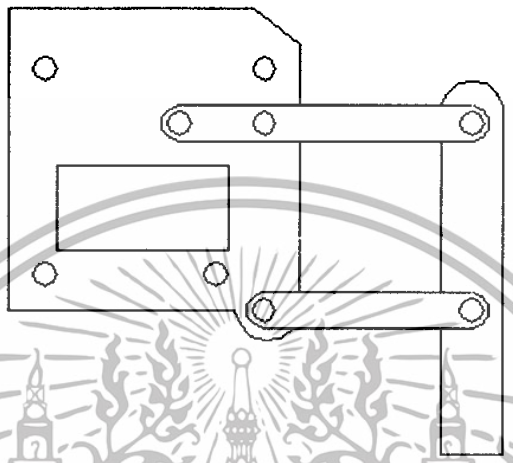
รูปที่ 2.8 แสดงการเคลื่อนที่ของขาแบบเพนโทรกราฟ

ซึ่งมีผลโดยตรงต่อขนาดของขา การเปลี่ยนแปลงขนาดความยาวของแต่ละส่วนของขาจะทำให้ความสูงที่หุ่นยนต์สามารถยกขาได้ หรือระยะที่หุ่นยนต์สามารถยืดขาได้ เปลี่ยนไปซึ่งส่งผลให้ขนาดของการก้าวแต่ละก้าวเปลี่ยนแปลงไปด้วย

ดังนั้นในการออกแบบสิ่งที่สำคัญที่ควรพิจารณาอีกประการหนึ่งก็คือ การพยายามให้ตัวขับเคลื่อน อยู่ติดกับฐานของขา การติดตั้งตัวขับเคลื่อนให้อยู่ที่จุดศูนย์กลางจะทำให้ผลกระทบทางไดนามิคลดลง ทั้งยังลดภาวะทางไดนามิคของแต่ละขาอีกด้วย เพราะไม่ต้องเคลื่อนมวลของตัวขับเคลื่อนที่ติดตั้งอยู่ในตำแหน่งที่ห่างออกไปตามข้อต่อต่าง ๆ อีกทั้งวัสดุที่นำมาใช้ในการสร้างหุ่นยนต์นั้นมีอยู่หลายชนิด ซึ่งราคาก็เป็นปัจจัยอย่างหนึ่ง แต่อย่างไรก็ดีน้ำหนักก็มีบทบาทที่สำคัญกว่า ที่จะต้องนำมาพิจารณาเพื่อประสิทธิภาพโดยรวมของหุ่นยนต์

2.3 การออกแบบและการสร้าง

ในการออกแบบเลือกใช้ขาแบบขาเพนโทกราฟ ตามที่แสดงไว้ในรูปที่ 2.9 เพราะโครงสร้างของขาง่ายต่อการสร้างไม่ซับซ้อนยุ่งยากในระบบเม็กคานิกส์และใช้เซอร์โวมอเตอร์เป็นตัวขับเคลื่อน



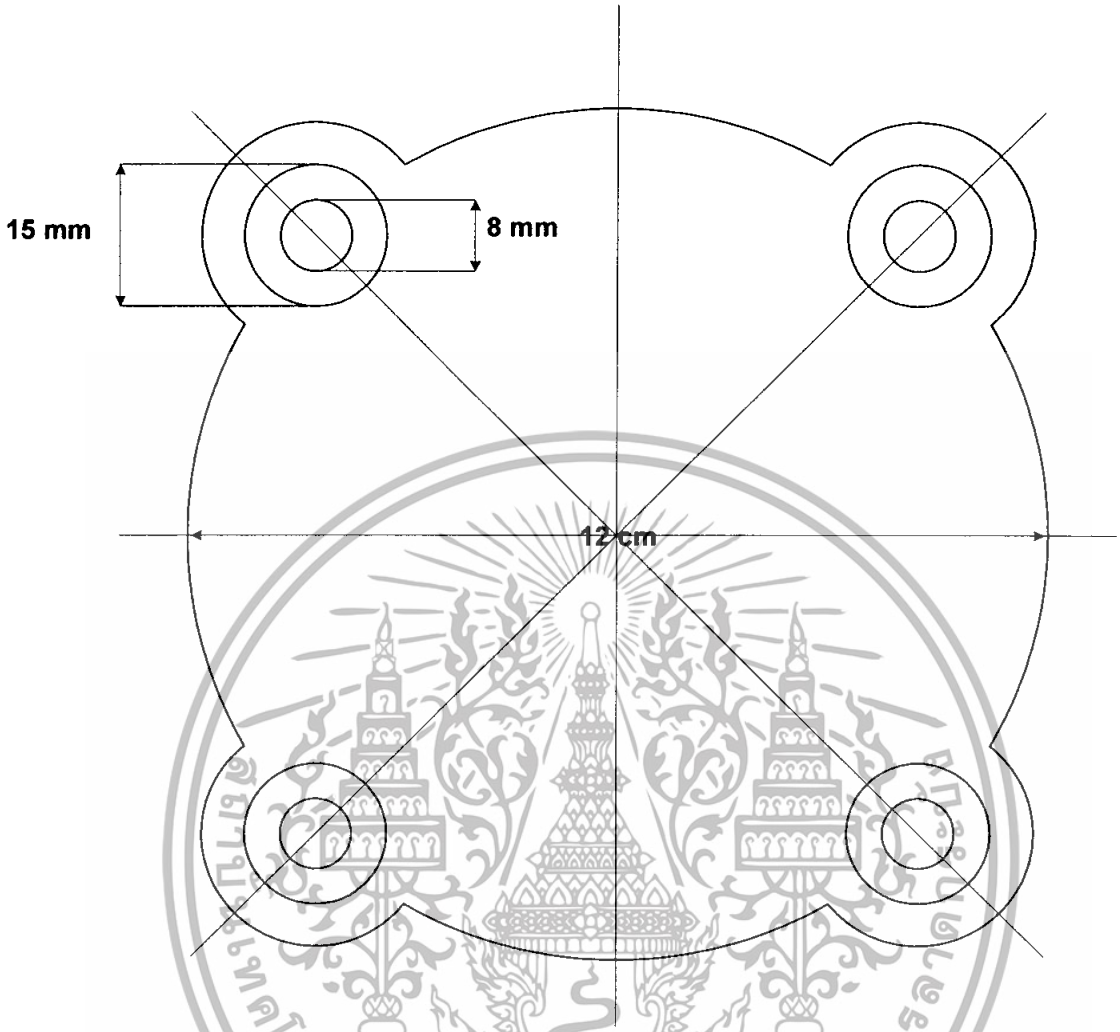
รูปที่ 2.9 แสดงรูปแบบขาของหุ่น

โดยใช้เซอร์โวมอเตอร์ตัวหนึ่งทำหน้าที่ยกขาให้ลอยขึ้นและเซอร์โวมอเตอร์อีกตัวหนึ่งทำการหมุนขาเพื่อเคลื่อนย้ายตำแหน่งเท้าของหุ่นยนต์

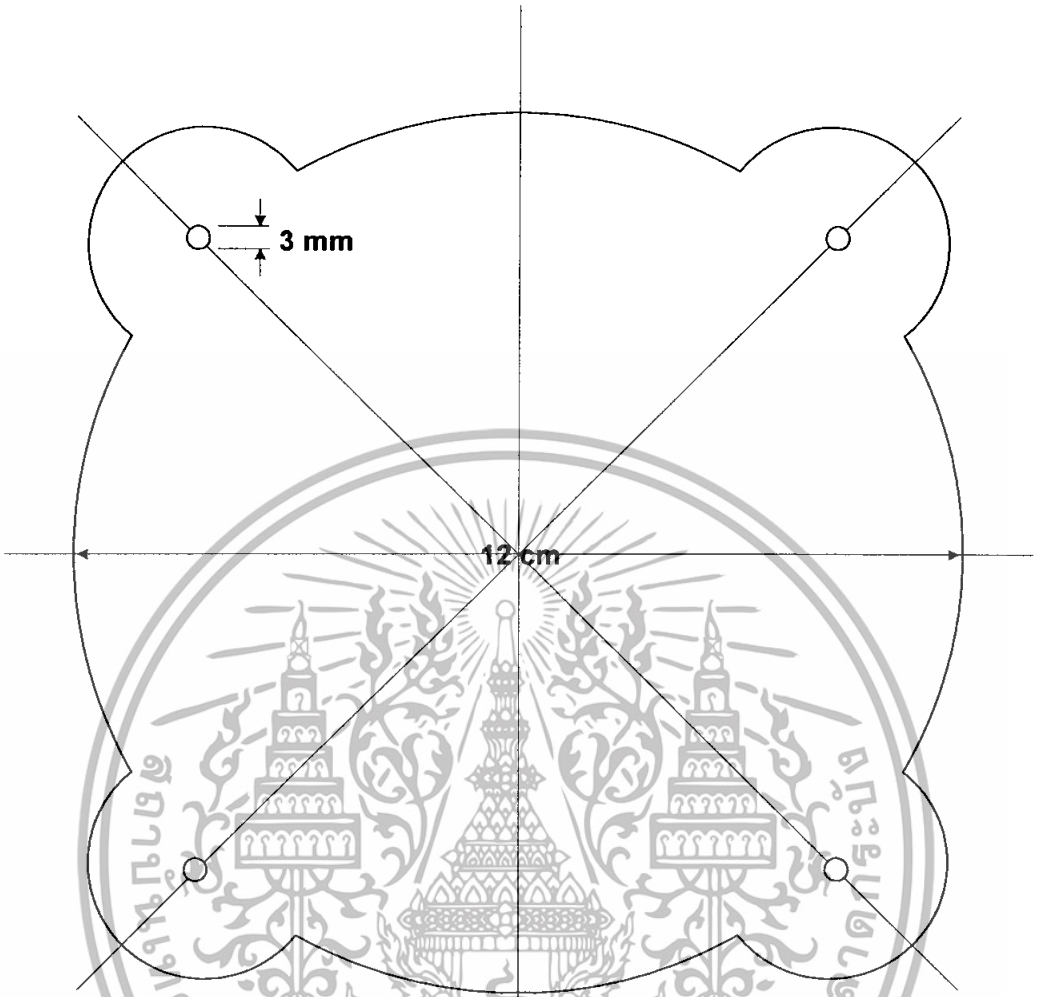
วัสดุที่ใช้สร้างแผ่นพลาสติกซิลิโคนจะนำมาสร้างเป็นตัวหุ่นและขาหุ่น เนื่องจากมีคุณสมบัติเหนียวทนทาน น้ำหนักเบา และตกแต่งง่าย ส่วนจุดหมุนเราใส่ลูกปืนแบริ่งขนาด 8 มิลลิเมตร เพื่อลดแรงเสียดทานของจุดหมุน

แผ่นลำตัวของหุ่นยนต์มีลักษณะเป็นวงกลม ทำหน้าที่ยึดขาทั้งหมดเข้าด้วยกันโดยจัดวางขาให้สมมาตรดังรูปที่ 2.10 ด้านบนของลำตัวมีพื้นที่สำหรับจัดวางอุปกรณ์ควบคุม ลำตัวของหุ่นมีความกว้าง 12 เซนติเมตร (ไม่รวมความยาวของขา)

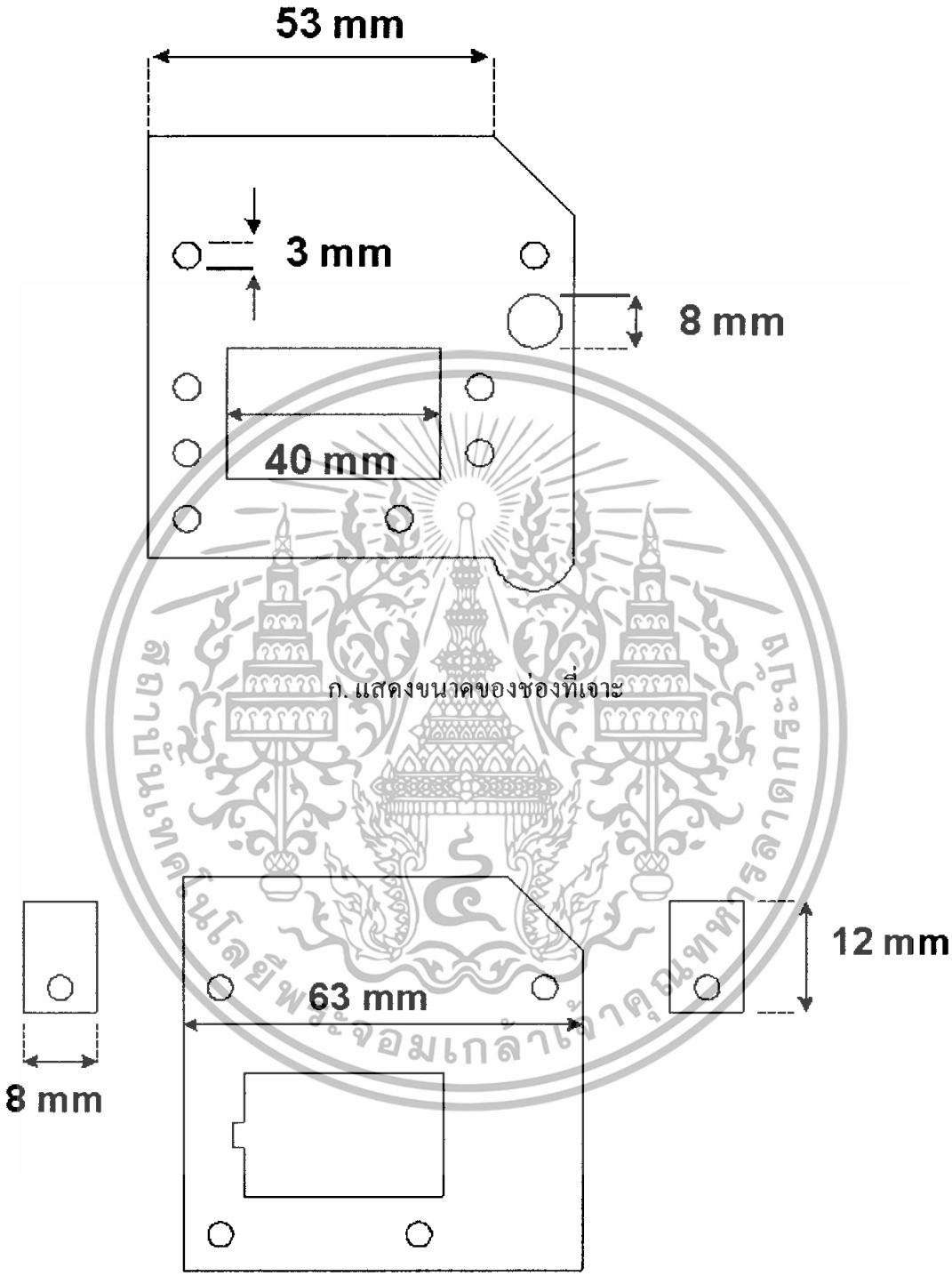
การยึดขาเข้ากับลำตัวหุ่นจะใช้แผ่นยึดลำตัวสองแผ่นประกบด้านบนและด้านล่างเพื่อยึดจุดหมุนส่วนขาให้มีความควบคุมคงที่ ตำแหน่งแต่ละขาวางห่างกัน ที่ 90 องศา ช่องว่างกลางลำตัวระหว่างแผ่นยึดลำตัวทั้งสองสามารถวางแบตเตอรี่ เพื่อจ่ายกำลังงานไฟฟ้าให้กับส่วนควบคุมและส่วนขับเคลื่อนของหุ่นยนต์ได้



รูปที่ 2.10 แผ่นลำตัวหุ่นด้านบน

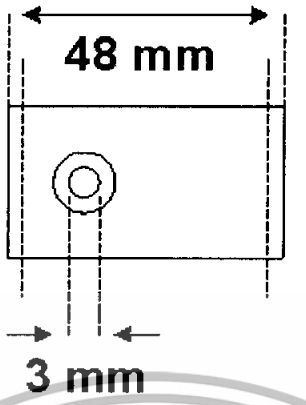


รูปที่ 2.11 แผ่นลำตัวหุ่นด้านล่าง



ก. แสดงขนาดของช่องที่เจาะ

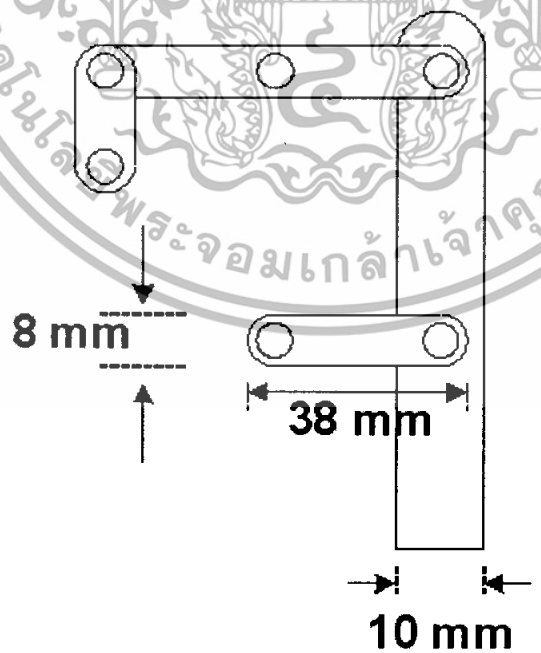
ข. แสดงความกว้างของแท่นยึดเซอร์โว



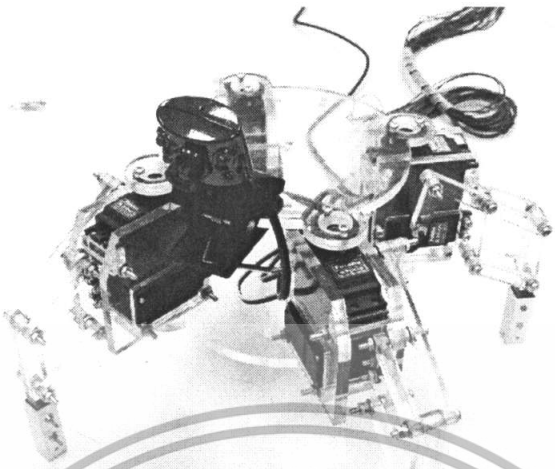
ค. แสดงขนาดของตัวเชื่อมแท่นยึดเซอร์โวทั้งสองด้าน

รูปที่ 2.12 แท่นยึดเซอร์โว

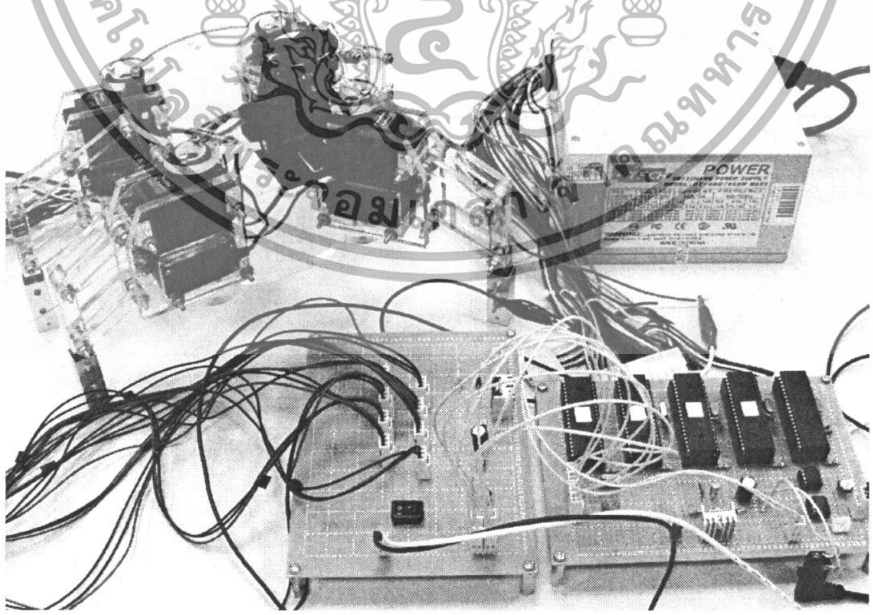
ส่วนประกอบของขาเป็นส่วนที่มีความละเอียดอ่อนที่สุด ประกอบด้วยจุดหมุน 6 จุด ตามที่แสดงไว้ในรูปที่ 2.13 โดยจะใส่ลูกปืนแบริ่งเพื่อลดแรงเสียดทานขณะที่หมุนที่จุดที่มีลักษณะเหมือนคานจะฝังลงไปในชิ้นส่วนของแท่นยึดแกนหมุน



รูปที่ 2.13 ขาหุ่น



รูปที่ 2.14 หุ่นยนต์เดิน 4 ขา



รูปที่ 2.15 หุ่นยนต์และวงจรที่ใช้ควบคุมหุ่นยนต์

บทที่ 3
ทฤษฎีและหลักการทำงาน

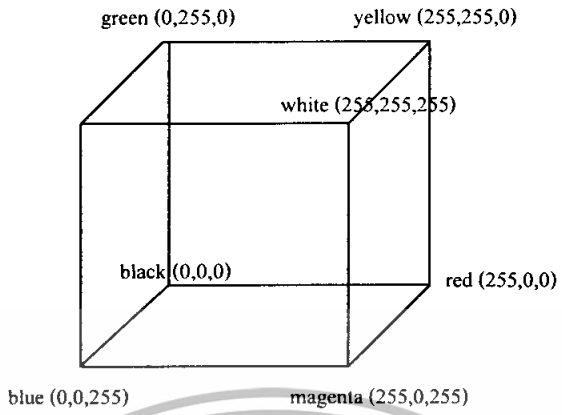
3.1 ทฤษฎีการประมวลผลภาพ

3.1.1 การกำหนดสี

ในคอมพิวเตอร์ สีทุกสีจะประกอบด้วย 3 สีพื้นฐาน คือ สีแดง สีเขียว และสีน้ำเงิน(RGB) ซึ่งสามารถที่จะสังเคราะห์สีอื่นๆได้เกือบทุกสีโดยการผสมสีพื้นฐานทั้งสามนี้ในสัดส่วนต่างๆกัน การผสมสีตั้งอยู่บนพื้นฐานของความเข้มข้นขององค์ประกอบของ RGB ซึ่งเรียกว่า RGB model

โดยแต่ละสีถูกนำเสนอโดยสัดส่วน(Red,Green,Blue) ซึ่งปริมาณของสีพื้นฐานทั้งสาม ในแต่ละสีนิยมนำเสนอด้วยค่าขนาด 1 ไบต์หรือเท่ากับ 8 บิต โดยค่าที่เล็กที่สุดคือค่า 0 สอดคล้องกับการไม่มีสีพื้นฐานสีนั้นเลย และค่าที่ใหญ่ที่สุดคือค่า 255 บ่งบอกถึงความเข้มข้นสูงสุดของสีพื้นฐานสีนั้น อาทิเช่น สัดส่วน (0,0,0) สอดคล้องกับสีดำ เพราะสีพื้นฐานทั้งสามสีไม่มีความเข้มข้นเลย ในขณะที่สัดส่วน (255,255,255) สอดคล้องกับสีขาว ส่วนสีอื่นๆซึ่งเกิดจากการรวมกันนั้นมีได้มากมาย เช่น (255,0,0) จะเป็นสีแดงบริสุทธิ์ (0,255,255) เป็นสีน้ำเงินเข้มและ(0,128,128) เป็นสีน้ำเงินอ่อนขึ้น ดังนั้นการรวมกันที่เป็นไปได้ของสีพื้นฐานในกรณีนี้จะมีขนาด 24 บิต หรือมีค่าได้ 256x256x256 ซึ่งเท่ากับ 16,777,216 ระดับสี จึงนิยมจัดเก็บค่าสีในรูปแบบตัวแปร Long Integer

กระบวนการของการสร้างสีซึ่งมีสีพื้นฐาน 3 สีนั้น ตั้งอยู่บนพื้นฐานของกล่องสี RGB ดังแสดงในรูปที่ 3.1 โดยมุมของกล่องสีนี้จะสอดคล้องกับสีต่างๆดังรูป ปริมาณของสีพื้นฐานทั้งสามสีในสีใดๆจะมีค่าตามแกน x y และ z ตามลำดับ สีที่เป็นส่วนกลับ (Complementary Colors) สามารถคำนวณได้อย่างง่ายๆ โดยการลบค่าของสีจาก 255 ตัวอย่างเช่นสี (0,0,255) เป็นสีน้ำเงินบริสุทธิ์เป็นสีส่วนกลับของสีเหลืองบริสุทธิ์ที่สอดคล้องกับค่า (255,255,0) ซึ่งทั้งสองสีดังกล่าวอยู่ที่มุมตรงข้ามกันของกล่องสี ส่วนมุมอื่นๆก็เช่นเดียวกัน



รูปที่ 3.1 กล่องสี RGB

องค์ประกอบของสี ที่มุมของกล่องสีจะมีความเข้มข้นสูงที่สุด หรือไม่มีความเข้มข้นเลย เท่านั้น เมื่อเคลื่อนที่จากมุมหนึ่งไปยังอีกมุมหนึ่งตามขอบที่เหมือนกัน จะมีเพียงแค่องค์ประกอบเดียวเท่านั้นที่เปลี่ยนแปลง ยกตัวอย่างเช่นเมื่อเคลื่อนที่จากมุมสีเขียวไปยังมุมสีเหลือง องค์ประกอบของสีแดงจะเปลี่ยนแปลงจาก 0 ถึง 255 โดยเมื่อเคลื่อนที่ระหว่างมุมของสีคู่นี้จะได้ทุกโทนสี จากเขียวไปเหลือง

3.1.2 องค์ประกอบของสี

ในการประมวลผลภาพบางประเภท อาจต้องการอ่านค่าขององค์ประกอบสีของแต่ละพิกเซลแล้วแยกองค์ประกอบของ RGB เพื่อนำไปใช้งานแยกกัน ดังนั้นจึงจำเป็นต้องมีฟังก์ชั่นซึ่งทำหน้าที่แยกองค์ประกอบของสีทั้งสามซึ่งถูกเก็บในตัวแปร Long Integer ดังที่กล่าวไว้ในหัวข้อ 3.1.1 ในตัวแปร Long Integer ดังกล่าวจะประกอบด้วย 4 ไบต์ โดยไบต์แรก (The Most significant Byte) เก็บ 0 ต่อมาเป็นองค์ประกอบของสีน้ำเงิน เขียว และแดงตาลำดับ ค่า Long Integer ซึ่งสอดคล้องกับสัดส่วน RGB (63,32,192) มีค่าเป็น 12,591,168 ซึ่งค่านี้ดูไม่เหมือนกับจำนวนเดิมเลย ถ้านำเสนอองค์ประกอบของสีด้วยเลขฐานสิบหก(Hexadecimal Fomat) ซึ่งในตัวอย่างนี้มีค่าเป็น $C02040_h$ จะเป็นเลขฐานสิบหกของ Long Integer 12,591,168 ในเลขฐานสิบ จะเห็นว่าสองหลักสุดท้ายสอดคล้องกับองค์ประกอบสีแดง (40_h = 64) สองหลักถัดมาสอดคล้องกับองค์ประกอบสีเขียว (20_h = 32) และสองหลักที่มีความสำคัญสูงสุดสอดคล้องกับค่าสีน้ำเงิน ($C0_h$ = 192) ด้วยเหตุนี้เลขฐานสิบหกจึงถูกใช้มากในการกำหนดค่าสี

3.1.3 ภาพดิจิตอล

ในการทำจะนำภาพภาพหนึ่งมาประมวลด้วยเครื่องคอมพิวเตอร์นั้น ภาพดังกล่าวจำเป็นจะต้องถูกนำเสนอหรือแทนด้วยตัวเลข แต่ภาพที่ได้มาโดยส่วนมากจะเป็นภาพที่ได้จากตัวรับสัญญาณซึ่งอยู่ในรูปของฟังก์ชั่น f(x,y) ที่ต่อเนื่องในระนาบสองมิติ โดยจะเป็นสัดส่วนกับความสว่างหรือความเข้มของภาพที่ตำแหน่ง(x,y) ซึ่งเรียกว่าระดับสีเท่า(Gray Level) สำหรับภาพขาวดำ และจะเป็นสัดส่วนกับระดับค่า RGB สำหรับภาพสี

3.1.3.1 การแทนภาพด้วยข้อมูลดิจิตอล

ภาพแบบดิจิตอลเป็นภาพที่ถูกแปลงมาจากแบบอนาลอกเพื่อให้อยู่ในรูปของตัวเลข โดยภาพจะถูกแบ่งพื้นที่เป็นสี่เหลี่ยมเล็กๆ ที่เรียกว่าจุดภาพ หรือ พิกเซล(Pixel) ซึ่งถูกระบุตำแหน่งของพิกเซลโดยพิกัด(x,y) และในแต่ละพิกเซลจะระบุข้อมูลระดับสีหรือค่า RGB สำหรับภาพสี และระดับความเข้มสำหรับระดับสีเทา โดยการแปลงข้อมูลแบบดิจิตอลสามารถทำได้ดังขั้นตอนต่อไปนี้

เมื่อนำภาพแบบอนาลอกที่ต้องการประมวลผล มาผ่านส่วนที่เรียกว่าดิจิไตเซอร์(Digitizer) ซึ่งมีหน้าที่ในการเปลี่ยนสัญญาณอนาลอกเป็นสัญญาณดิจิตอล อุปกรณ์ประเภทนี้ได้แก่ กล้องดิจิไตเซอร์ ซึ่งจะทำให้ภาพที่นำเสนอด้วยฟังก์ชั่นที่ต่อเนื่อง f(x,y) ให้กลายเป็นฟังก์ชั่นที่ไม่ต่อเนื่องทั้งระนาบของภาพ ซึ่งเรียกว่าการสุ่ม(Sampling) และผ่านการควอนไตซ์(Quantization) ก็จะได้ข้อมูลที่เป็นดิจิตอล

สมมติว่าสัญญาณภาพต่อเนื่อง f(x,y) ได้ถูกแปลงให้อยู่ฟังก์ชั่นที่ไม่ต่อเนื่องซึ่งจะสามารถนำเสนอในรูปเมติกซ์ และสมมติว่ามีขนาด $N \times N$ ได้ตามสมการ (3.1)

$$f(x,y) = \begin{matrix} f(0,0) & f(0,1) & f(0,2)\ldots f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2)\ldots f(1,N-1) \\ \ldots & \ldots & \ldots \quad \ldots \\ f(N-1,0) & f(N-1,1) & f(N-1,2)\ldots f(N-1,N-1) \end{matrix} \qquad (3.1)$$

โดยฟังก์ชั่น f(x,y) ทางขวาของสมการจะเรียกว่าภาพดิจิตอลและทุกสมาชิกของเมตริกซ์จะเรียกว่าพิกเซลจากกระบวนการสร้างภาพดิจิตอลข้างต้นจะเห็นได้ว่าเราสามารถทราบขนาดของความละเอียดของภาพเป็น $N \times N$ พิกเซล และในกรณีภาพขาวดำขนาดของข้อมูลภาพที่เป็นดิจิตอลจะมีขนาดดังสมการ (3.2)

$$B = N \times N \times M \text{ บิต} \qquad (3.2)$$

เมื่อ B คือขนาดของข้อมูลภาพที่เป็นดิจิตอล (บิต)

M คือจำนวนบิตที่ใช้ในการแทนข้อมูลภาพ 1 พิกเซล สามารถหาได้จากสมการที่ (3.3)

$$G = 2^{M} \quad (3.3)$$

เมื่อ G คือจำนวนระดับของสีเทา (Gray Scale) ที่ต้องการใช้ในการเก็บภาพ

3.1.3.2 ลักษณะการจัดเก็บข้อมูลดิจิตอล

โดยทั่วไปแล้วข้อมูลภาพสีหรือขาวดำจะมีค่าตั้งแต่สองระดับขึ้นไป โดยเฉพาะในกรณีภาพขาวดำ นิยมใช้ระดับความเข้มของจุดภาพระดับของสีเทาเท่ากับ 256 ระดับ ซึ่งทำให้จุดภาพมีค่าอยู่ในช่วง 0-255 โดยใช้พื้นที่เก็บข้อมูล 1 ไบต์ หรือ 8 บิต สำหรับข้อมูล 1 จุดภาพ (M=8) ในกรณีที่ต้องการภาพที่มีความละเอียดของระดับความเข้มหรือระดับสีจำนวนสูงๆ อาจจะต้องการจำนวนบิตสำหรับเก็บข้อมูลมากกว่า 8 บิต อาจเป็น 16 หรือ 24 บิต โดยค่าความเข้มของจุดภาพเท่ากับ 65,536 และ 16,777,216 โดยจะแยกให้ชัดเจนดังนี้

1. ภาพ 2 ระดับคือมีเพียงแต่จุดขาวกับจุดดำเท่านั้น โดยแต่ละจุดภาพเป็นข้อมูลขนาด 1 บิต

2. ภาพ 16 ระดับคือในแต่ละจุดภาพจะมีขนาดของข้อมูล 4 บิต สามารถทำให้แสดงได้ 16 ระดับความเข้มหรือระดับสีขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือภาพขาวดำ

3. ภาพ 256 ระดับคือในแต่ละจุดภาพจะมีขนาดข้อมูลขนาด 8 บิต ซึ่งสามารถทำให้แสดงได้ 256 ระดับความเข้มหรือระดับสี ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือขาวดำ

4. ภาพทรูคัลเลอร์ (True Color) ในแต่ละจุดภาพจะมีขนาดของข้อมูล 24 บิต สามารถทำให้แสดงภาพสีได้เหมือนจริงที่สุดเพราะสามารถแสดงสีได้ถึง 16,777,216 ระดับสี

3.1.4 การประมวลผลภาพเชิงตัวเลข

โดยทั่วไปวิธีการประมวลผลภาพเชิงตัวเลข โดยคอมพิวเตอร์นั้น อาจแบ่งโดยคร่าวๆเป็นสองระดับด้วยกันคือการประมวลผลภาพในระดับต่ำ (Low Level Processing) และการประมวลผลภาพในระดับสูง (High Level Processing) การประมวลผลภาพในระดับต่ำจะเป็นการประมวลผลเชิงตัวเลขเกือบทั้งหมดเพื่อหาตัวแปรต่างๆ มาอธิบายข้อมูลภาพ มีวัตถุประสงค์เพื่อนำตัวแปรเหล่านั้นไปใช้ในการประมวลผลในระดับสูงต่อไป

ส่วนการประมวลผลภาพในระดับสูงเป็นการนำผลลัพธ์ที่ได้จากการประมวลผลภาพในระดับต่ำมาตีความหรือเพื่อส่งข้อมูลให้คอมพิวเตอร์สามารถรู้และเข้าใจได้ ดังนั้นความแตกต่างที่สำคัญของการประมวลผลภาพระดับต่ำและระดับสูงคือข้อมูลที่นำเข้ามาในการประมวลผล โดยการประมวลผลภาพในระดับต่ำจะใช้ความเข้มของจุดโดยตรง ส่วนการประมวลผลระดับสูงนั้นข้อมูลที่นำมาประมวลผลจะถูกแสดงในรูปของสัญลักษณ์ ซึ่งจะแสดงถึงสิ่งต่างๆ ที่อยู่ในภาพ เช่น ขนาด หรือ รูปร่างของวัตถุในภาพ

3.1.5 สัญญาณข้อมูลภาพจากดิจิตอลวีดีโอ

การส่งสัญญาณข้อมูลจากภาพวิดีโอมีลักษณะการส่งที่เป็นลำดับภาพเดี่ยวหรือเฟรม (Frame) ที่ฉายต่อเนื่องกันเช่นระบบวีดีโอ NTSC จะส่งด้วยอัตราเร็ว 30 เฟรมต่อวินาที โดยดิจิตอลวีดีโอแต่ละเฟรมจะเป็นข้อมูลดิจิตอลในลักษณะเมตริกซ์ซึ่งแต่ละจุดจะเรียกว่าพิกเซล มีค่าระดับของความเข้มสี โดยทั่วไปจะใช้ระดับของสีเท่ามีค่าตั้งแต่ 0 – 255 โดย 0 จะแทนความมืดมากที่สุด และ 255 จะแทนความสว่างมากที่สุด

3.1.6 ข้อมูลภาพชนิดบิตแมป

รูปแบบของไฟล์ข้อมูลภาพชนิดบิตแมป (Bitmap) เป็นรูปแบบไฟล์มาตรฐานสำหรับภาพกราฟฟิกบนวินโดว์ซึ่งจะใช้ในการตัดต่อ หรือทำสำเนาภาพต่างๆลงบนคลิปบอร์ด (Clipboard) เมื่อเวลาจัดเก็บไฟล์ที่มีนามสกุล .bmp เป็นฟอร์แมตที่สามารถใช้สร้างเป็นวอลล์เปเปอร์ (Wallpaper) ได้อีกด้วย

โครงสร้างของไฟล์ข้อมูลภาพชนิดบิตแมปจะประกอบด้วย 3 ส่วนคือ

1. ข้อมูลเฮดเดอร์ (Header)
2. ข้อมูลจานสี (Palette)
3. ข้อมูลภาพ (Data)

1. ข้อมูลเฮดเดอร์ คือข้อมูลที่อยู่บริเวณส่วนหัวของไฟล์ ซึ่งประกอบด้วยข้อมูลที่บอกรายละเอียดต่างๆ ของภาพ เช่น ความกว้าง ความยาวของภาพ จำนวนสี จำนวนบิต ความละเอียดของภาพ
2. ข้อมูลจานสีเป็นค่าแม่สี RGB ของภาพ
3. ข้อมูลภาพ เป็นส่วนเก็บข้อมูลสีของแต่ละพิกเซลของภาพ โดยข้อมูลแรกจะเป็นค่าสีของพิกเซลที่อยู่แถวล่างสุดที่ตำแหน่งซ้ายสุด ข้อมูลลำดับต่อไปจะเรียงจากทางขวาจากแถวล่างจนถึงแถวบนสุด

3.1.7 การสร้างภาพไบนารี

การสร้างภาพไบนารี หมายถึง การแปลงข้อมูลภาพที่มีความเข้มหลายระดับ ให้เป็นภาพที่มีความเข้มเพียง 2 ระดับ คือ หนึ่งจุดภาพมีค่าได้สองค่าเท่านั้น โดยเป็นค่า 0 กับค่า 1 ค่าที่เป็น 1 จะหมายถึงจุดภาพที่มีสีดำ และค่าที่เป็น 0 จะหมายถึงจุดภาพที่เป็นสีขาว

การสร้างภาพไบนารีสามารถทำโดยใช้เทคนิคการเปรียบเทียบระหว่างจุดภาพเริ่มต้นกับค่าคงที่ค่าหนึ่ง เรียกว่าค่าเทรชโฮล (Threshold) เทคนิคนี้ใช้กันมากในกรณีที่ข้อมูลภาพมีลักษณะแตกต่างกันระหว่างวัตถุและพื้นหลัง โดยค่าของจุดภาพใดๆ ที่มีค่าน้อยกว่าค่าเทรชโฮลจะถูกกำหนดค่าเป็น 1 (จุดสีดำ) และถ้าค่าของจุดภาพที่มีค่ามากกว่าค่าเทรชโฮลจะถูกเปลี่ยนให้เป็นค่า 0 (จุดสีขาว) ซึ่งการทำงานสามารถแสดงได้ดังสมการที่ (3.4)

$$b(x,y) = 1 ; g(x,y) < Thr$$

$$b(x,y) = 0 ; g(x,y) > Thr \quad (3.4)$$

เมื่อ b(x,y) คือ ข้อมูลภาพผลลพธ์ภาพเป็นไบนารี

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

Thr คือ ค่าเทรชโฮลเป็นค่าคงที่ระหว่าง 0 ถึง L ระดับ

1 คือ จุดภาพที่เป็นสีดำ

0 คือ จุดภาพที่เป็นสีขาว

โดยที่ L คือระดับความเข้มของจุดภาพสูงสุด

ในการสร้างภาพไบนารีโดยใช้เทคนิคเทรชโฮล เพื่อให้ผลลัพธ์ที่ได้มีความคมชัดและเหมาะสม สิ่งที่สำคัญที่สุด คือ ค่าเทรชโฮล เนื่องจากถ้าเลือกค่าเทรชโฮลที่ไม่เหมาะสมภาพที่ได้จะไม่คมชัดและรายละเอียดบางส่วนอาจขาดหายไป ดังนั้นปัญหาการสร้างภาพไบนารีด้วยวิธีเทรชโฮล คือ ทำอย่างไรจึงสามารถคำนวณหาค่าเทรชโฮลที่เหมาะสมได้

3.1.7.1 การหาค่าเทรชโฮลโดยการกำหนดไว้ล่วงหน้า

การหาค่าเทรชโฮลโดยวิธีการกำหนดล่วงหน้า (Preassigned Threshold Value) นี้เป็นวิธีที่ง่ายที่สุด เป็นการกำหนดค่าเทรชโฮลเองจากผู้ใช้ ซึ่งจะขึ้นอยู่กับประสบการณ์ของผู้ใช้ โดยการเลือกค่าคงที่ค่าหนึ่งซึ่งจะเป็นค่าที่อยู่ระหว่างค่าต่ำสุดและค่าสูงสุดของระดับความเข้มของข้อมูลอินพุต

วิธีหนึ่งสำหรับการหาค่าเทรชโฮลล่วงหน้าคือ การหาค่าเทรชโฮลจากค่ากลาง (Midrange Threshold Value) การหาวิธีนี้จะพิจารณาจากค่ากลาง โดยอาศัยการคำนวณพื้นฐานทางสถิติในการหาค่ากลางหรือค่าเฉลี่ย (Mean) มาประยุกต์ ค่าที่ได้เป็นค่ากึ่งกลางระหว่างค่าระดับความเข้มสูงสุดกับค่าต่ำสุดของข้อมูลภาพอินพุต สำหรับการหาค่ากึ่งกลางจะแสดงได้ดังสมการที่ (3.5)

$$Thr = \frac{Max(g(x,y)) + Min(g(x,y))}{2} \quad (3.5)$$

เมื่อ Thr คือ ค่าเทรชโฮล

g(x,y) คือ ข้อมูลภาพอินพุตที่มีระดับความเข้ม ถึง L ระดับ

Max(g(x,y)) คือ ค่าสูงสุดเกรย์สเกลของข้อมูลอินพุต

Min(g(x,y)) คือ ค่าต่ำสุดเกรยสเกลของข้อมูลอินพุต

การหาค่าเทรชโฮลจากค่าเฉลี่ยเลขคณิต หาได้จากสมการที่ (3.6)

$$Thr = \frac{\sum g(x,y)}{N \times N} \tag{3.6}$$

3.1.7.2 การฟิลเตอร์แบบค่ามีเดียน (Median Filtering)

ในกรณีนี้พิกเซลอินพุทจะถูกแทนที่ด้วยค่ามีเดียนของพิกเซลที่อยู่ในหน้าต่างรอบๆพิกเซลนั่นคือ

$$V(m,n) = median\{y(m-k,n-l),(k,l) \in W\} \tag{3.7}$$

โดยที่ W คือหน้าต่าง

อัลกอริทึมสำหรับการฟิลเตอร์แบบมีเดียนจะทำการเรียงค่าพิกเซลในลักษณะที่เพิ่มขึ้นหรือลดลง แล้วทำการเลือกค่าที่อยู่ตรงกลาง โดยทั่วไปแล้วขนาดของหน้าต่างถูกเลือกให้ N_w มีค่าเป็นเลขคี่ ถ้า N_w เป็นเลขคู่ ค่ามีเดียนจะเป็นค่าเฉลี่ยระหว่างสองค่าตรงกลาง โดยทั่วไปแล้วหน้าต่างจะเป็นแบบขนาด 3x3 5x5 7x7 หรือหน้าต่าง 5 จุด

ฟิลเตอร์มีเดียนมีคุณสมบัติต่อไปนี้

1. ฟิลเตอร์แบบไม่เป็นเชิงเส้น(Nonlinear) ดังนั้นสำหรับสองซีเควน์ x(m)และy(m)
   median{x(m)+y(m)}= median{x(m)}+ median{y(m)} (3.8)
2. มีประโยชน์ในการจำกัดเส้นหรือพิกเซลที่แยกออกมาโดดเดี่ยว(Isolated) โดยฟิลเตอร์มีเดียนจะใช้ได้ดีสำหรับสัญญาณรบกวนแบบไบนารี แต่ใช้ไม่ได้ดีสำหรับสัญญาณรับกวนแบบ Gaussian
3. ประสิทธิภาพต่ำเมื่อจำนวนของพิกเซลที่ถูกปนเปื้อนด้วยสัญญาณรบกวนในหน้าต่างมีค่ามากกว่าหรือมีค่าครึ่งหนึ่งของจำนวนพิกเซลในหน้าต่าง

เนื่องจากมีเดียนคือ ค่า (N_w +1)/2 ที่สูงที่สุด (N_w = คี่) การหาค่ามีเดียนต้องการเปรียบเทียบจำนวน (N_w -1)+ (N_w -2)+....+ (N_w -1)/2=3(N_w^2 -1)/8 ค่านี้คือ 30 สำหรับหน้าต่าง 3x3 และ 224 สำหรับหน้าต่างขนาน 5x5 เมื่อใช้เทคนิคการหาที่มีประสิทธิภาพ จำนวนการเปรียบเทียบจะลดลง โดยประมาณเป็น $0.5\, N_w \log_2 N_w$ ถ้าใช้มีเดียนหน้าต่างเคลื่อนที่ (moving window median) จำนวนของคำสั่งจะสามารถลดลงได้อีก ตัวอย่างเช่นถ้า k พิกเซลถูกลบและ k พิกเซลใหม่ถูกเพิ่มเข้ามาในหน้าต่างแล้วค่ามีเดียนใหม่สามารถหาค่าได้ในการเปรียบเทียบที่น้อยกว่า k(N_w +1) ฟิลเตอร์มีเดียนสองมิติในทางปฏิบัติ คือ ฟิลเตอร์มีเดียนที่สามารถแยกได้(separable) กล่าวคือจะทำการฟิลเตอร์มีเดียนหนึ่งมิติสำหรับแถวและคอลัมน์ต่อกัน

3.1.8 Shape Analysis

การคำนวนหาค่าจากภาพที่ได้ถ่ายมานั้น เพื่อใช้ในการคำนวณนั้นสามารถทำได้หลายวิธี เพื่อบอกคุณลักษณะของภาพในรูปของเลขฐานสอง

Area Descriptors เป็นวิธีหนึ่งที่สามารถบอกพื้นที่ของกลุ่มภาพ โดยแปลงกลุ่มภาพนั้นเป็นในรูปแบบเลขฐานสอง ซึ่งวิธีนี้มีความแม่นยำค่อนข้างสูงแต่ มีความยากในการคำนวณโดยเฉพาะถ้ารูปมีขนาดใหญ่

ในการคำนวณหาค่า Area descriptors โดยที่ R แทนค่าพื้นที่ของรูปภาพ I (k, j) โดยที่ R เป็นกลุ่มค่าของแต่ละพิกเซล แม้ว่ารูปที่คิดนั้นอาจมีรูหรือช่องว่างในรูปภาพโดยสมมุติให้ภาพ I (k, j) นั้นเป็นรูปภาพแบบฐานสองโดยที่ ค่าของภาพที่จับได้นั้นคือ Foreground มีค่าเป็น 1 และส่วนที่เหลืออยู่คือ Background มีค่าเป็น 0

3.2 Moments

เป็นกระบวนการหาค่าของโมแมนต์ โดยสมมุติให้ x แทนค่าแถว และ y แทนค่าของหลักของภาพฐานสอง โดยเราสามารถคำนวณหาค่าของโมเมนต์ได้ดังนี้

$$M_{kj} = \sum_{(x,y)\in R} x^k y^j \qquad k \geq 0, j \geq 0 \qquad (3.9)$$

Low-Order Moments

เป็นกระบวนการที่นำทฤษฎีของโมเมนต์ มาหาค่าของจุดศูนย์กลางของวัตถุ (x_c,y_c)โดยสามารถหาค่าจุดศูนย์กลางของวัตถุได้ดังนี้

$$x_c = \frac{m_{10}}{m_{00}} \qquad (3.10)$$

$$y_c = \frac{m_{01}}{m_{00}} \qquad (3.11)$$

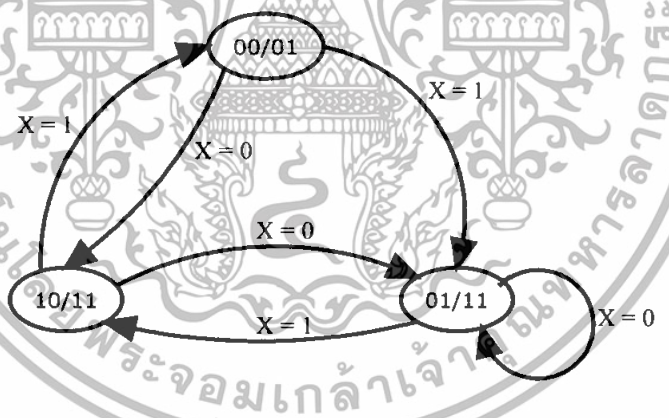
3.3 ทฤษฎี State Transition Machine

หุ่นยนต์แมลง 4 ขา ที่ควบคุมด้วยระบบโครงข่ายประสาทเทียมนี้แบ่งการทำงานออกเป็น 4 เซลล์ ซึ่งแต่ละเซลล์จะทำงานแทนขาแต่ละขาของหุ่นและทำงานเป็นอิสระต่อกัน โดยที่ทุกๆเซลล์ได้รับอินพุทค่าเดียวกัน จากนั้นแต่ละเซลล์จะนำค่าอินพุทและค่าเอาท์พุทของเซลล์อื่นๆเข้ามาแล้วทำการเปรียบเทียบกับค่าในตารางซึ่งเป็นค่าที่เซลล์แต่ละเซลล์ได้เรียนรู้แล้ว (learned)

ถ้าเท่ากันก็จะส่งค่าเอาท์พุทออกมา ถ้าค่าที่ได้รับมาไม่เท่ากันก็จะส่งค่าที่ใกล้เคียงที่สุดออกมาในกรณีนี้หุ่นจะเดินไม่ถูกสเตท แต่หลังจากที่แต่ละเซลล์ให้เอาท์พุทออกมาแล้วก็จะทำให้ได้ค่าใหม่เป็นอินพุทซึ่งตอนนี้นี่เองที่ทำให้เมื่อนำค่าที่ได้มาเทียบตารางก็จะเป็นค่าที่ตรงกัน

การทำงานของหุ่นจะเป็นแบบ state transition machine คือทำงานทีละ state เป็นจังหวะที่ต่อเนื่องกันไปและใช้ state ที่แล้วมาเป็นตัวอ้างอิง กล่าวคือค่าเอาท์พุทที่ออกมาจะได้มาจากค่าเอาท์พุทของ state ขณะนั้นโดยพิจารณาว่าขณะนี้ตำแหน่งของขาอยู่ที่ไหน(ลักษณะ state เป็นอย่างไร) จากนั้นรับค่าอินพุทเข้ามาแล้วพิจารณาว่าตำแหน่งของขาเป็นอย่างนี้ รับอินพุทมาอย่างนี้จะให้เอาท์พุทอย่างไร ซึ่งในโครงงานนี้จะใช้ การเขียน State Diagram แบบ Moore Machines ซึ่งจะประกอบไปด้วย รูปทรงกลมแทนสเตทต่างๆและเส้นเชื่อมต่อระหว่างสเตทที่มีลูกศรชี้ที่สเตทถัดไปบนเส้นจะมีอินพุทกำกับไว้

ตัวอย่าง Moore Machines

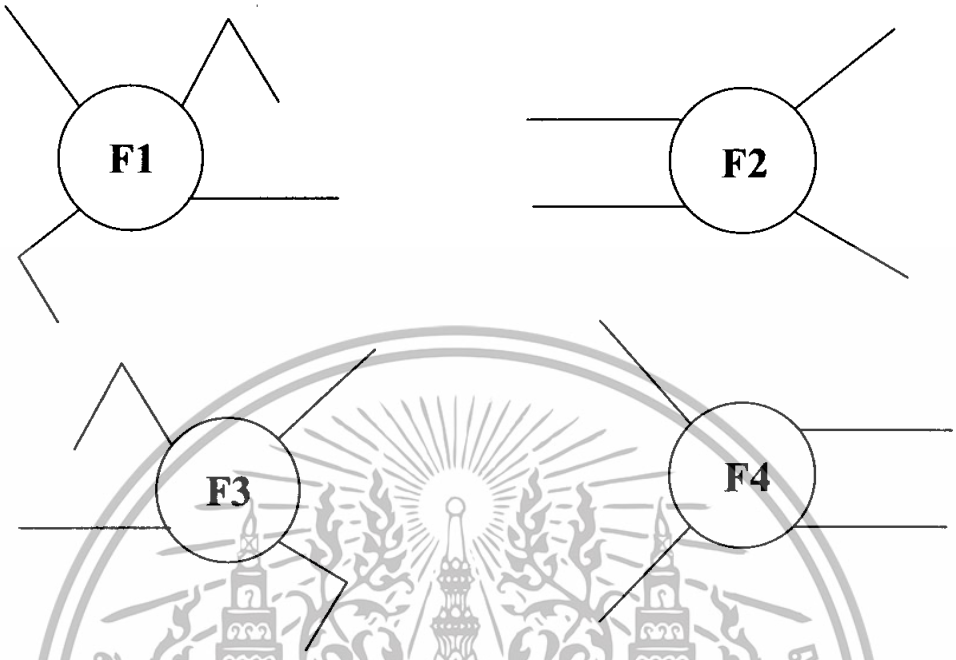


รูปที่ 3.2 Moor Machines

Moore Machines จะมีการเพิ่มเอาท์พุทเข้าไปในแต่ละสเตท หรือก็คือ เมื่อสเตทปัจจุบัน(PS)ได้รับอินพุทใดๆเข้ามาจะทำให้เกิดเอาท์พุทค่าหนึ่งๆ แล้วจึงเปลี่ยนสเตทไปที่สเตทถัดไป(NS) ตัวเลขที่อยู่ในวงกลมคือ สเตท/เอาท์พุท ตัวเลขที่อยู่บนเส้นคือ อินพุท

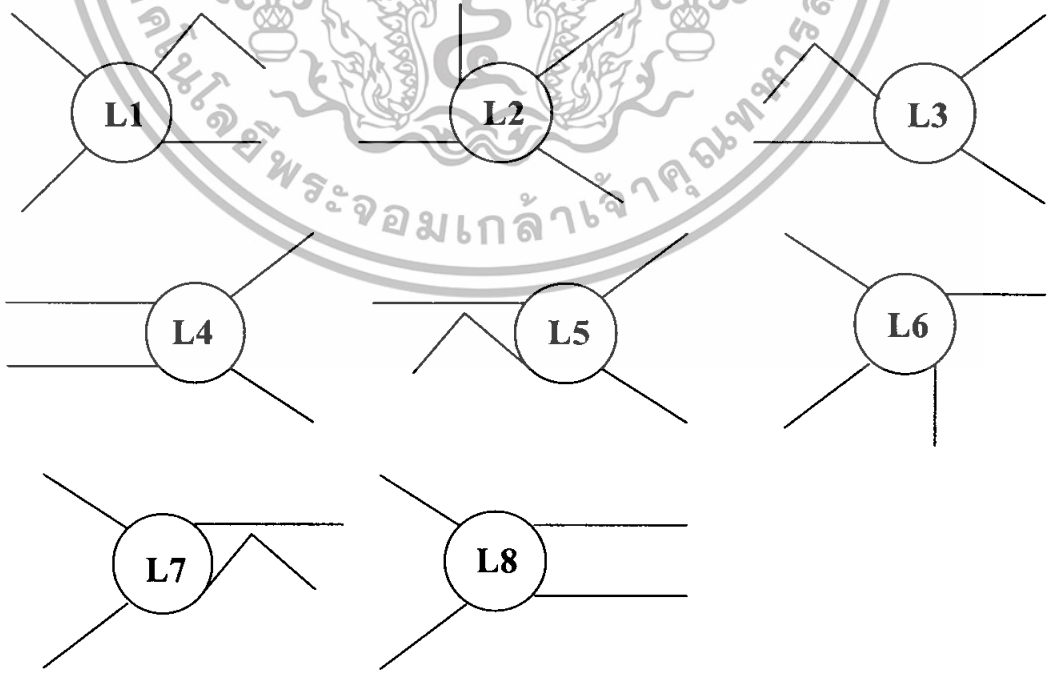
จากหลักการนี้เริ่มต้นต้องมีการคิดรูปแบบการเดินของหุ่นออกมาเป็นสเตทการเดินแบบต่างๆ เช่น การเดินหน้า เลี้ยวซ้าย เลี้ยวขวา เป็นต้นแล้วเราก็จะพบว่าในการเดินแบบต่างๆของหุ่น มีขาใดที่ทำงานบ้างแล้วนำมาเขียนแสดงเป็นตารางการทำงาน

การเดินไปข้างหน้า



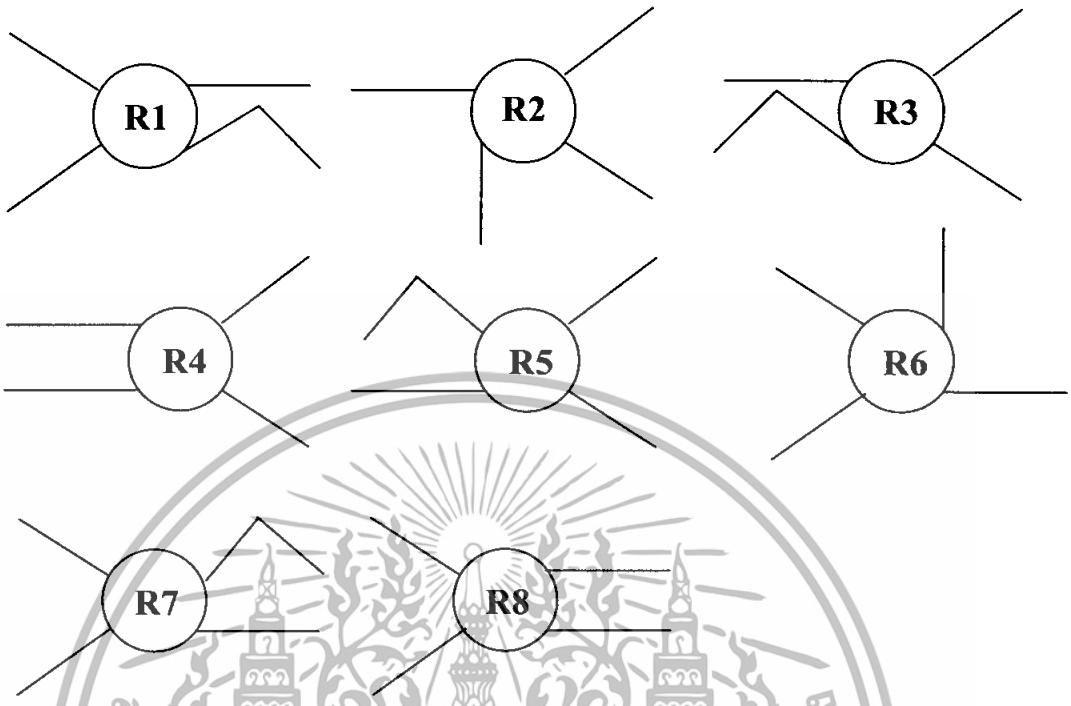
รูปที่ 3.3 รูปแบบการเดินไปข้างหน้า

การเลี้ยวซ้าย



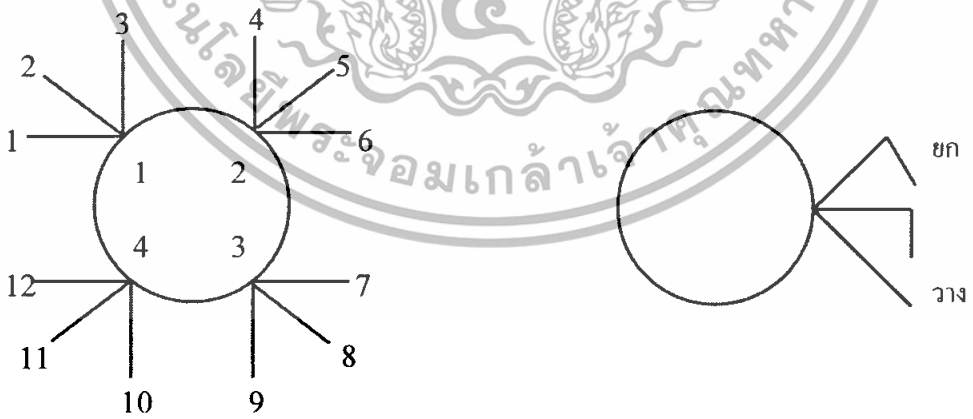
รูปที่ 3.4 รูปแบบการเลี้ยวซ้าย

การเลี้ยวขวา



รูปที่ 3.5 รูปแบบการเลี้ยวขวา

และเมื่อเราเขียนสเตทการเดินทุกแบบแล้วจะได้ลักษณะการก้าวขาและการยกขาของขาแต่ละขาออกมา ดังนี้



รูปที่ 3.6 แสดงสถานะขาหุ่นในสภาวะต่างๆ

เนื่องจากการก้าวขาแต่ละขาจะมีตำแหน่งในการก้าวขาอยู่ 3 ตำแหน่งและการยกขาอยู่ 3 ตำแหน่งนั่นแสดงว่าต้องใช้บิตควบคุมทั้งหมด 4 บิต (มอเตอร์ตัวบน 2 บิต มอเตอร์ตัวล่าง 2 บิต) และจากทฤษฎีของเซอร์โวมอร์เตอร์ เราสามารถกำหนดให้แต่ละตำแหน่งมีบิต และการป้อนพัลส์ให้กับมอร์เตอร์ดังนี้

เซอร์โวตัวบน

ขา	ตำแหน่ง	บิต	t_{on}	t_{off}
1	1	10	0.6 ms	19.4 ms
	2	00	1.2 ms	18.8 ms
	3	01	1.5 ms	18.5 ms
2	4	10	0.9 ms	19.1 ms
	5	00	1.2 ms	18.8 ms
	6	01	1.8 ms	18.2 ms
3	7	10	0.6 ms	19.4 ms
	8	00	1.2 ms	18.8 ms
	9	01	1.5 ms	18.5 ms
4	10	10	0.9 ms	19.1 ms
	11	00	1.2 ms	18.8 ms
	12	01	1.8 ms	18.2 ms

ตารางที่ 3.1 แสดงบิตควบคุม และพัลส์ที่ป้อนให้เซอร์โวมอเตอร์ตัวบน

เซอร์โวตัวล่าง

ขา	ตำแหน่ง	บิต	t_{on}	t_{off}
1	ยก	10	0.3 ms	19.7 ms
	วาง	01	2.4 ms	17.6 ms
2	ยก	01	2.4 ms	17.6 ms
	วาง	10	0.3 ms	19.7 ms
3	ยก	10	0.3 ms	19.7 ms
	วาง	01	2.4 ms	17.6 ms
4	ยก	01	2.4 ms	17.6 ms
	วาง	10	0.3 ms	19.7 ms

ตารางที่ 3.2 แสดงบิตควบคุม และพัลส์ที่ป้อนให้เซอร์โวมอเตอร์ตัวล่าง

จากลักษณะการเดินและบิตควบคุมที่ได้กำหนดไว้นั้น เราจะนำมาเขียนเป็น ตารางสเตจการเดินได้ดังนี้

Stage	ขา 1		ขา 2		ขา 3		ขา 4	
	บน	ล่าง	บน	ล่าง	บน	ล่าง	บน	ล่าง
F1	00	01	01	01	10	01	00	00
F2	10	01	00	10	00	01	01	10
F3	10	10	00	10	00	00	01	10
F4	00	01	01	10	10	01	00	10
L1	00	01	01	01	10	01	00	10
L2	01	01	00	10	00	01	01	10
L3	01	10	00	10	00	01	01	10
L4	10	01	00	10	00	01	01	10
L5	10	01	00	10	00	01	01	01
L6	00	01	01	10	01	01	00	10
L7	00	01	01	10	01	10	00	10
L8	00	01	01	10	10	01	00	10
R1	00	01	01	10	10	10	00	10
R2	10	01	00	10	00	01	10	10
R3	10	01	00	10	00	01	10	01
R4	10	01	00	10	00	01	01	10
R5	10	10	00	10	00	01	01	10
R6	00	01	10	10	10	01	00	10
R7	00	01	10	01	10	01	00	10
R8	00	01	01	10	10	01	00	10

ตารางที่ 3.3 แสดงสเตทการเดิน

หลังจากที่เราได้ลักษณะการเดินและค่าบิตควบคุมต่างๆมาแล้วขั้นตอนที่สำคัญต่อไปก็คือการนำค่าเอาท์พุทจากเซลล์อื่นๆมาเก็บในตารางเพื่อเป็นตัวชี้ค่าเอาท์พุทของสเตท ต่างๆซึ่งตารางนี้นี่เองที่เรียกว่าตารางสเตทซึ่งเปรียบเสมือนสมองของนิวรอนแต่ละเซลล์

การจัดเก็บค่าเอาท์พุทจากเซลล์อื่นในแต่ละเซลล์จะแตกต่างกันเล็กน้อยโดยมีลักษณะดังนี้

Cell self+3	Cell self+2	Cell self+1	Self	Input	Output

เมื่อ

Self : คือค่าเอาท์พุทของเซลล์นั้นๆที่เป็นบิตควบคุมในเวลา t-1

Cell self + n : คือค่าเอาท์พุทของเซลล์อื่นที่เป็นบิตควบคุมในเวลา t-1

Input : คือค่าที่รับจากภายนอกเช่นพวกเซนเซอร์ต่างๆ

Output : คือค่าเอาท์พุทของเซลล์นั้นๆที่เป็นบิตควบคุมในเวลา t

การเก็บค่าก็ทำได้โดยอ่านค่าจากพอร์ตเข้ามาค่าของ Cell self+2 และ Cell self+1 จะถูกกำหนดให้เข้ามาที่พอร์ต 1 ส่วนค่า Cell selft+3 จะเข้ามาที่พอร์ต 2 แล้วค่าที่ได้จะถูกเก็บตามลำดับข้างบน ดังนั้นจากตารางแสดงเสตจการเดินที่ 3.1 เมื่อแทนค่าบิตควบคุมแล้วมาสร้างตารางstate การเดินที่รับค่าของอินพุท เอาท์พุทของแต่ละเซลล์ และอินพุทกำหนดได้ดังนี้

stage	Command
เดินหน้า	0000
เลี้ยวซ้าย	0001
เลี้ยวขวา	0010

ตารางที่ 3.4 แสดงลักษณะอินพุท และคำสั่งการเดิน

ดังนั้นจะสามารถเขียนสเตจการเดินของแต่ละขาได้ดังนี้

output	cell4	cell3	cell2	self	command	stage
1001	0000	1001	0101	0001	0000	F
1010	0110	0001	0010	1001	0000	
0001	0110	0000	0010	1010	0000	
0001	0010	1001	0110	0001	0000	
0101	0010	1001	0101	0001	0001	L
0110	0110	0001	0010	0101	0001	
1001	0110	0001	0010	0110	0001	
1001	0110	0001	0010	1001	0001	
0001	0101	0001	0010	1001	0001	
0001	0010	0101	0110	0001	0001	
0001	0010	0110	0110	0001	0001	
0001	0010	1001	0110	0001	0001	
1001	0010	1010	0110	0001	0010	R
1001	1010	0001	0010	1001	0010	
1001	1001	0001	0010	1001	0010	
1010	0110	0001	0010	1001	0010	
0001	0110	0001	0010	1010	0010	
0001	0010	1001	1010	0001	0010	
0001	0010	1001	1001	0001	0010	
0001	0010	1001	0110	0001	0010	

ตารางที่ 3.5 ตารางแสดงค่าสเตทการเดินของขาที่ 1

output	cell4	cell3	cell1	self	command	stage
0010	0000	1001	0001	0101	0000	F
0010	0110	0001	1001	0010	0000	
0110	0110	0000	1010	0010	0000	
0101	0010	1001	0001	0110	0000	
0010	0010	1001	0001	0101	0001	L
0010	0110	0001	0101	0010	0001	
0010	0110	0001	0110	0010	0001	
0010	0110	0001	1001	0010	0001	
0110	0101	0001	1001	0010	0001	
0110	0010	0101	0001	0110	0001	
0110	0010	0110	0001	0110	0001	
0101	0010	1001	0001	0110	0001	
0010	0010	1010	0001	0110	0010	R
0010	1010	0001	1001	0010	0010	
0010	1001	0001	1001	0010	0010	
0010	0110	0001	1001	0010	0010	
1010	0110	0001	1010	0010	0010	
1001	0010	1001	0001	1010	0010	
0110	0010	1001	0001	1001	0010	
0110	0010	1001	0001	0110	0010	

ตารางที่ 3.6 ตารางแสดงค่าสเตทการเดินของขาที่ 2

output	cell4	cell2	cell1	self	command	stage
0001	0000	0101	0001	1001	0000	F
0000	0110	0010	1001	0001	0000	
1001	0110	0010	1010	0000	0000	
1001	0010	0110	0001	1001	0000	
0001	0010	0101	0001	1001	0001	L
0001	0110	0010	0101	0001	0001	
0001	0110	0010	0110	0001	0001	
0001	0110	0010	1001	0001	0001	
0101	0101	0010	1001	0001	0001	
0110	0010	0110	0001	0101	0001	
1001	0010	0110	0001	0110	0001	
1001	0010	0110	0001	1001	0001	
0001	0010	0110	0001	1010	0010	R
0001	1010	0010	1001	0001	0010	
0001	1001	0010	1001	0001	0010	
0001	0110	0010	1001	0001	0010	
1001	0110	0010	1010	0001	0010	
1001	0010	1010	0001	1001	0010	
1001	0010	1001	0001	1001	0010	
1010	0010	0110	0001	1001	0010	

ตารางที่ 3.7 ตารางแสดงค่าสเตทการเดินของขาที่ 3

output	cell3	cell2	Cell1	self	command	stage
0110	1001	0101	0001	0000	0000	F
0110	0001	0010	1001	0110	0000	
0010	0000	0010	1010	0110	0000	
0000	1001	0110	0001	0010	0000	
0110	1001	0101	0001	0010	0001	L
0110	0001	0010	0101	0110	0001	
0110	0001	0010	0110	0110	0001	
0101	0001	0010	1001	0110	0001	
0010	0001	0010	1001	0101	0001	
0010	0101	0110	0001	0010	0001	
0010	0110	0110	0001	0010	0001	
0010	1001	0110	0001	0010	0001	
1010	1010	0110	0001	0010	0010	R
1001	0001	0010	1001	1010	0010	
0110	0001	0010	1001	1001	0010	
0110	0001	0010	1001	0110	0010	
0010	0001	0010	1010	0110	0010	
0010	1001	1010	0001	0010	0010	
0010	1001	1001	0001	0010	0010	
0010	1001	0110	0001	0010	0010	

ตารางที่ 3.8 ตารางแสดงค่าสเตทการเดินของขาที่ 4

การทำงานของโปรแกรม

1 เริ่มต้นจะรับค่าอินพุทกล้อง แล้วทำการประมวลผลภาพว่าวัตถุอยู่ตำแหน่งใด เพื่อกำหนดทิศทางการเดินให้กับหุ่นยนต์ และรับอินพุทจากเซนเซอร์อินฟาเรดเพื่อจับระยะวัตถุ จากนั้นก็ตีความหมายของของค่าอินพุทตามตาราง ค่าที่ตีความหมายแล้วนี้จะมีค่าเป็นเลขไบนารี 4 บิต ซึ่งก็คือค่า command ที่อยู่ในตารางที่ 3.4 นั่นเอง

2 หลังจากนั้นก็เก็บค่าเอาท์พุทที่มาจากเซลล์ต่างๆรวมทั้งค่าอินพุทเข้ามาเก็บไว้ในตัวแปรตัวแปรหนึ่งที่ชื่อ input เรียงตามลำดับซึ่งก็คือค่า cell4, cell3, cell2, self นั่นเอง ตัวอย่างเช่น

สมมุติว่าค่าเอาท์พุทจากเซลล์ 4 เท่ากับ 1101 อ่านค่าเอาท์พุทจากเซลล์ 3 ได้เท่ากับ 1011 อ่านค่าเอาท์พุทจากเซลล์ 2 ได้เท่ากับ 1011 อ่านค่าเอาท์พุทจากเซลล์ 1(ตัวเองหรือ self) ได้เท่ากับ 0111 อ่านค่าอินพุท ได้เท่ากับ 0101 เรียงค่าโดย

(เซลล์ 4 x 0XFFFF)+(เซลล์ 3 X 0XFFF)+(เซลล์ 2 X 0XFF)+(self X 0XF)+ค่าอินพุท ซึ่งได้เท่ากับ '11011011101101110101' แล้วนำค่าที่ได้เก็บในตัวแปร input

3 ทำการวนลูปเปิดค่าในตารางเรียนรู้ที่อยู่ในตัวคอนโทรลตั้งแต่ค่าแรกวนไปเรื่อยๆ จนกว่าค่าที่ได้จะเท่ากับค่าในตัวแปร input หรือจนกว่าจะถึงค่าสุดท้ายโดยให้ pointer เป็นตัวชี้ค่าที่อยู่ในตารางออกมาแล้วตัดค่า output ออกไปก่อนที่จะนำมาเทียบกับตัวแปร input เช่น เปิดตารางได้ค่า ' 01011101101110110110101 ' ตัดค่า output ออกโดย นำมาแอนกับค่า 0XFFFFF แล้วจะได้ '11011011101101110101' จากนั้นเก็บค่าในตัวแปร compare

4 เปรียบเทียบค่า input กับค่า compare ที่เปิดได้ในตารางว่าตรงกันหรือไม่ถ้าเท่ากันก็ให้ค่าเอาท์พุทที่ได้ออกมาโดยเอาค่าในตำแหน่งที่ pointer ชี้อยู่มาตัดให้เหลือแต่ค่า output ดังนี้ เปิดตารางได้ค่า ' 01011101101110110110101 ' นำมาแอนกับค่า 0XF00000 แล้วหารด้วยค่า 0XFFFFF จะเหลือค่าเอาท์พุทคือ ' 0101 '

5 นำค่าเอาท์พุทที่ได้มาตีความว่าควรขับมอเตอร์ตัวใดก่อน ตัวใดหลัง ขับด้วยค่าพัลส์เท่าใดเช่นค่า เอาท์พุทมีค่า " 0101 " หมายความว่ามอเตอร์ตัวบนอยู่ที่ตำแหน่ง 01 ซึ่งต้องจ่ายพัลส์ที่มีคาบเวลา t_{on} เท่ากับ 0.6 ms ทุกๆ 20 ms มอเตอร์ตัวล่างอยู่ที่ตำแหน่ง 01 เช่นกันซึ่งต้องจ่ายพัลส์ที่มีคาบเวลา t_{on} เท่ากับ 0.3 ms ทุกๆ 20 ms

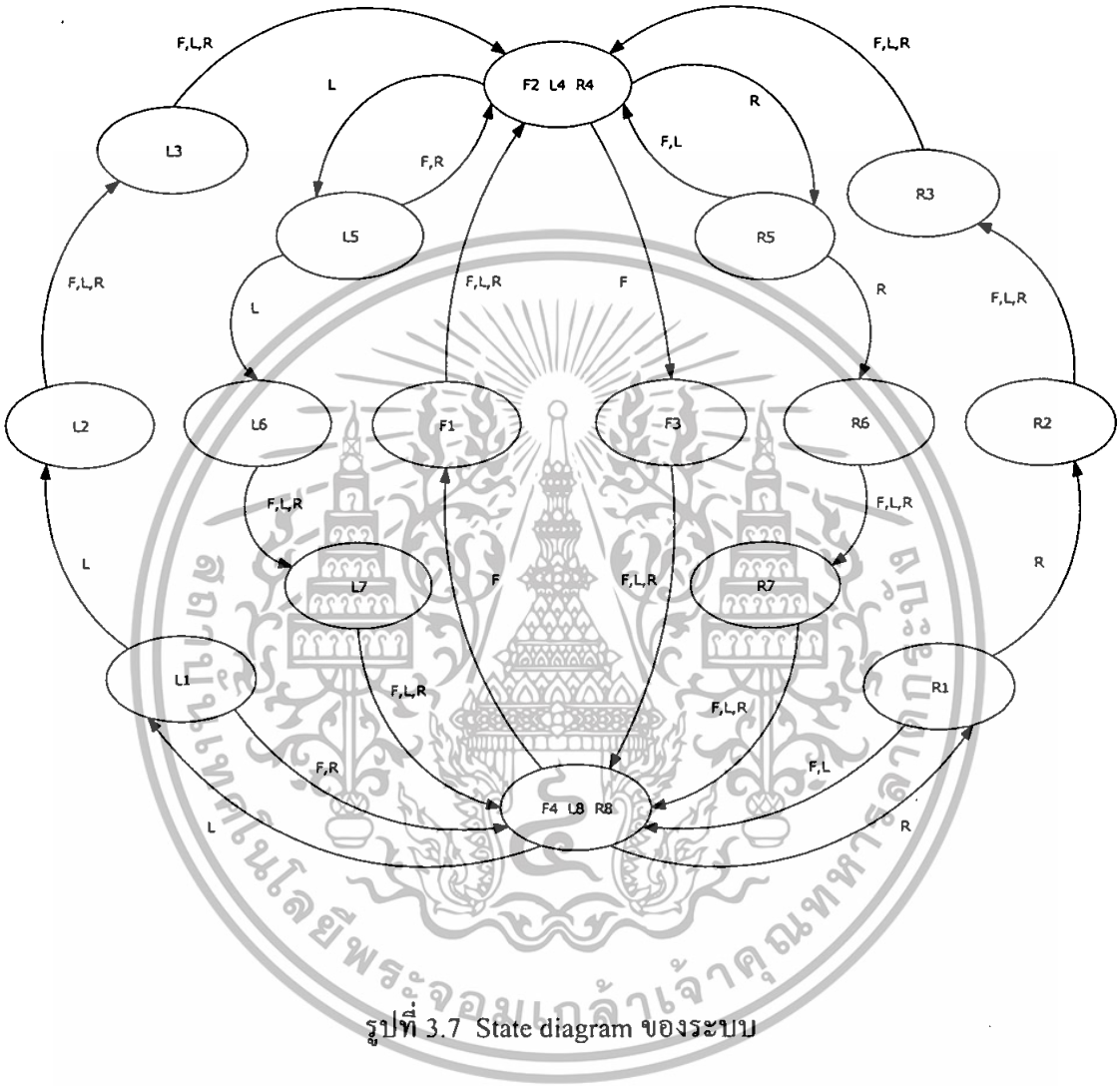
6 ทำการขับมอเตอร์โดย

- เซตบิตมอเตอร์ตัวบนและมอเตอร์ตัวล่างให้เป็น 1
- หน่วงเวลาด้วยค่า t_{on} ที่น้อยที่สุด

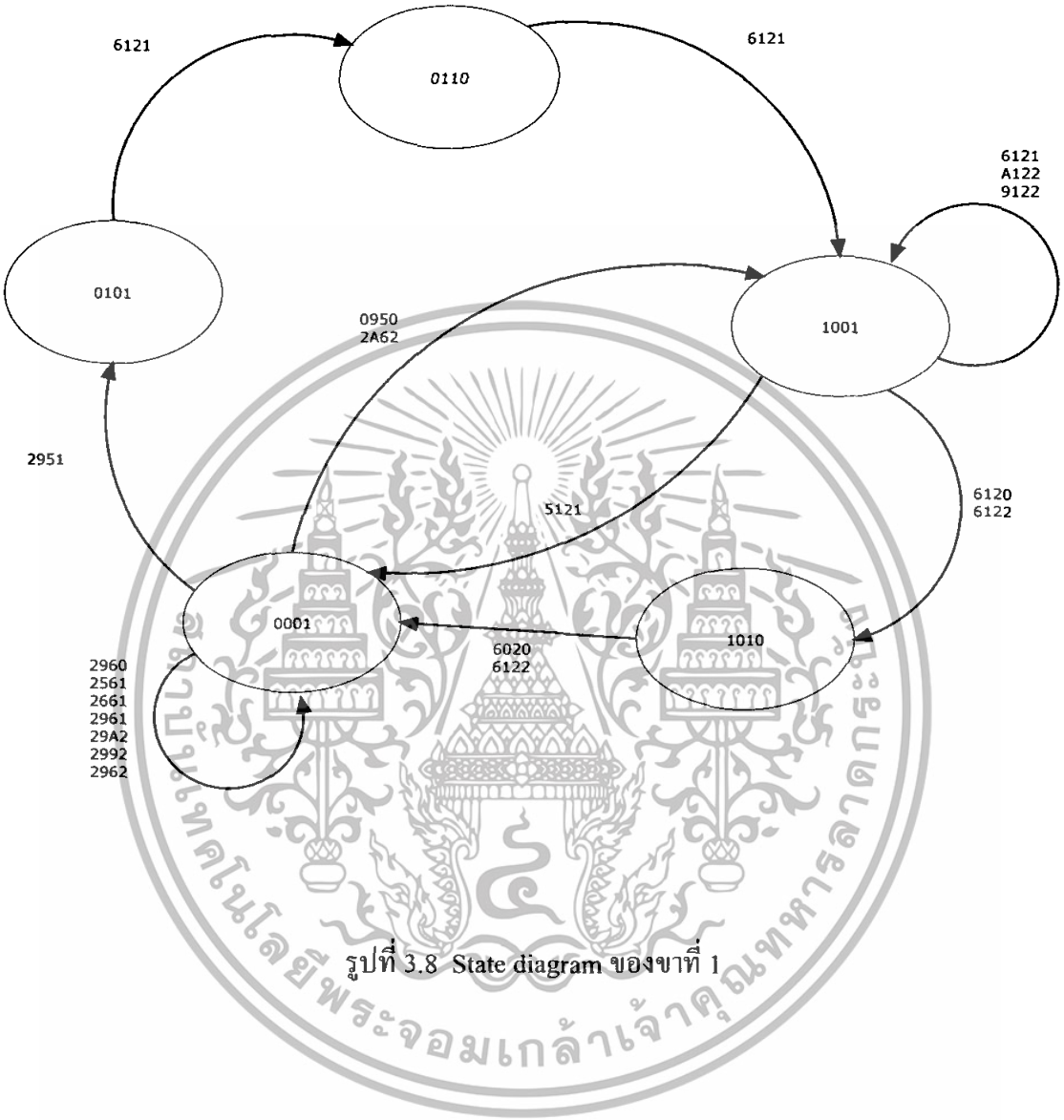
- เซตบิตมอเตอร์ตัวที่มีคาบเวลา t_{on} น้อยกว่าให้เป็น 0
- หน่วงเวลาด้วยค่า t_{on} ที่เท่ากับ(ค่า t_{on} ที่มากกว่า - ค่า t_{on} ที่น้อยกว่า)
- เซตบิตมอเตอร์ตัวที่เหลือให้เป็น 0
- หน่วงเวลาด้วยค่า t_{on} ที่เท่ากับ(20 ms - ค่า t_{on} ที่มากที่สุด)

7 ทำตามขั้นตอนที่ 1 – 6 ซ้ำไปเรื่อยๆ โดยรับค่าอินพุทเข้ามาแล้วเปิดตารางหุ่นยนต์จะทำงานโดยอัตโนมัติไปเรื่อยๆ

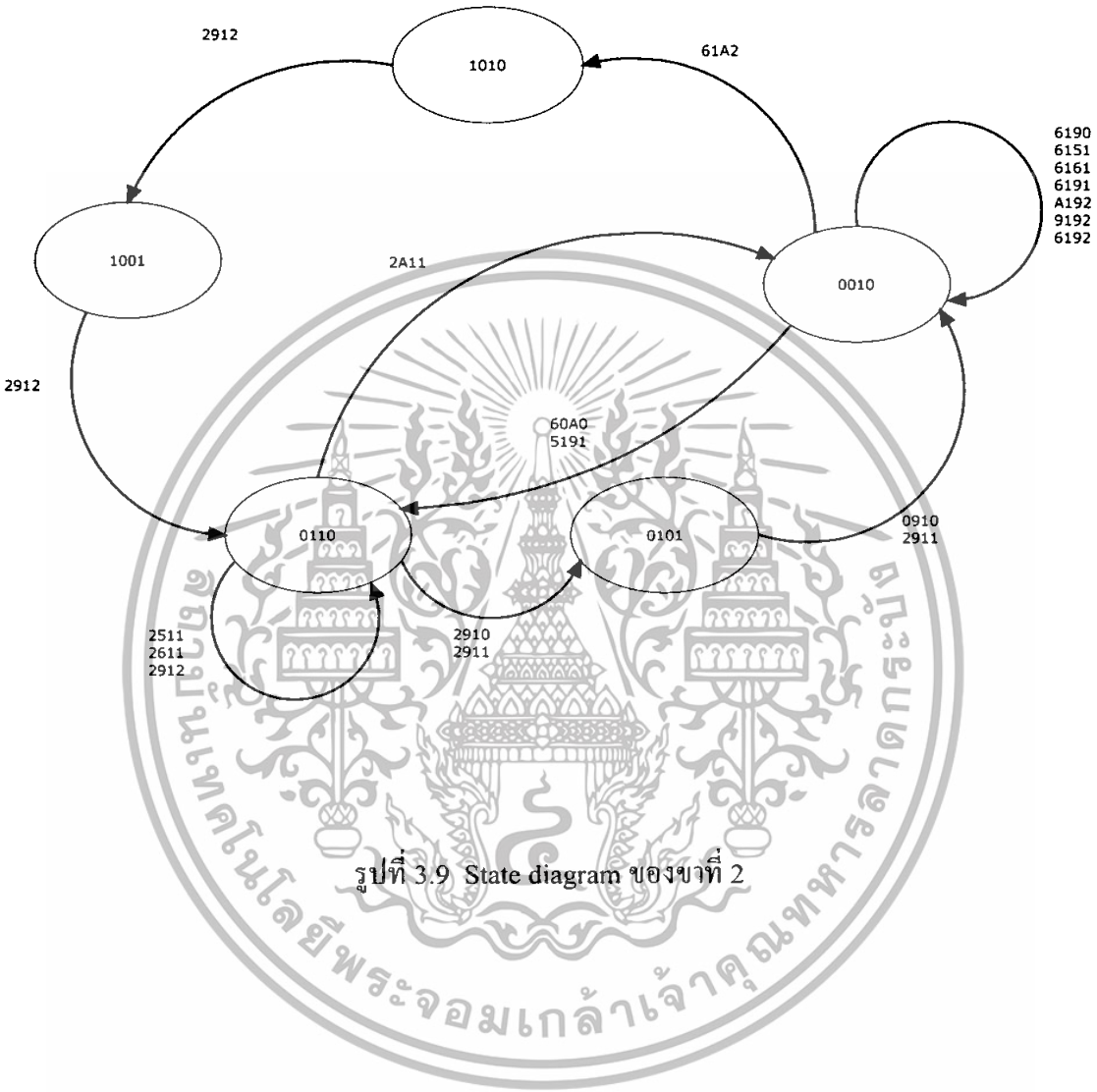
State Diagram



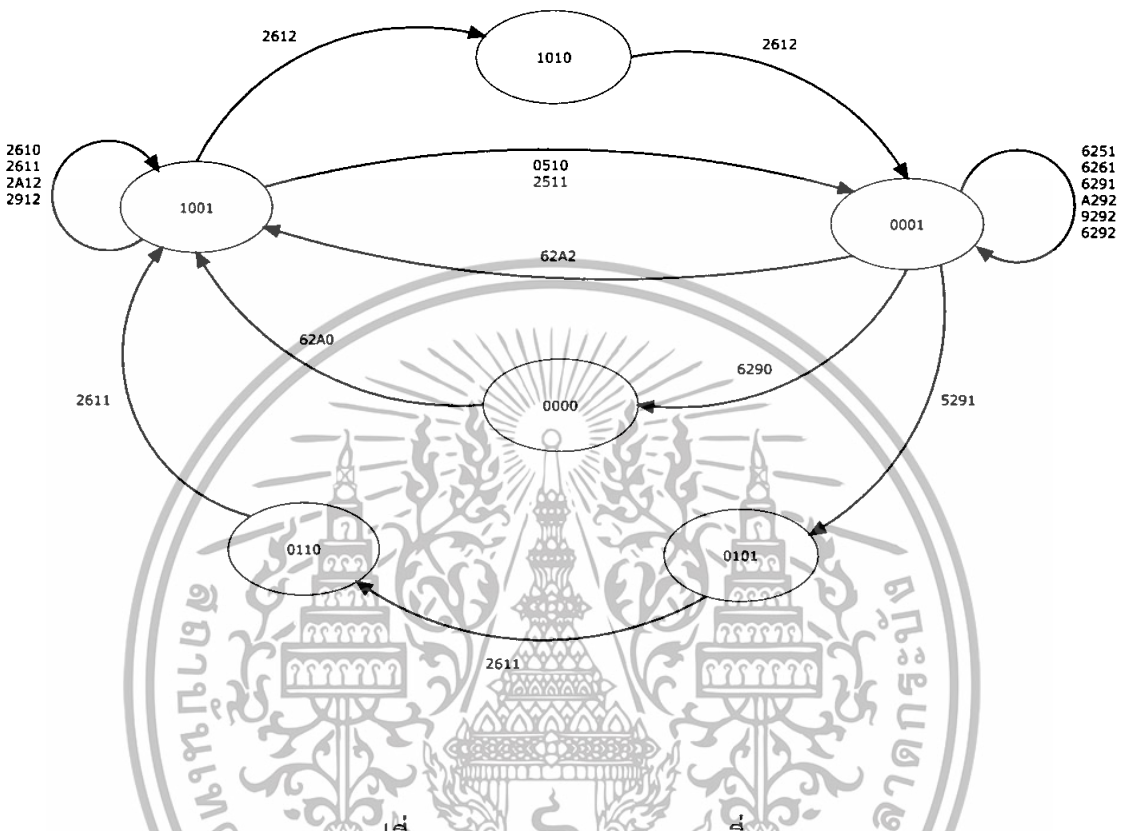
รูปที่ 3.7 State diagram ของระบบ



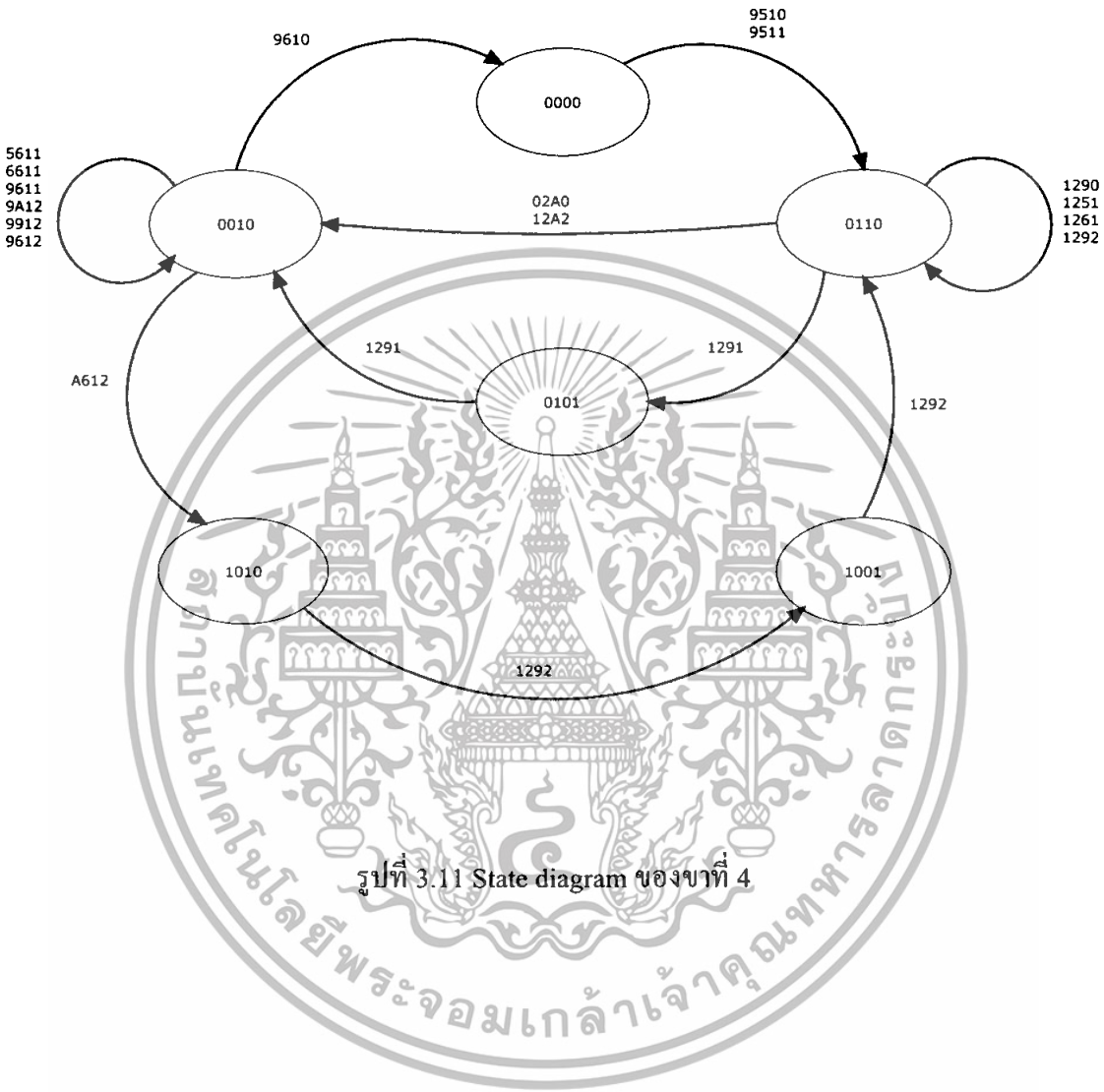
รูปที่ 3.8 State diagram ของขาที่ 1



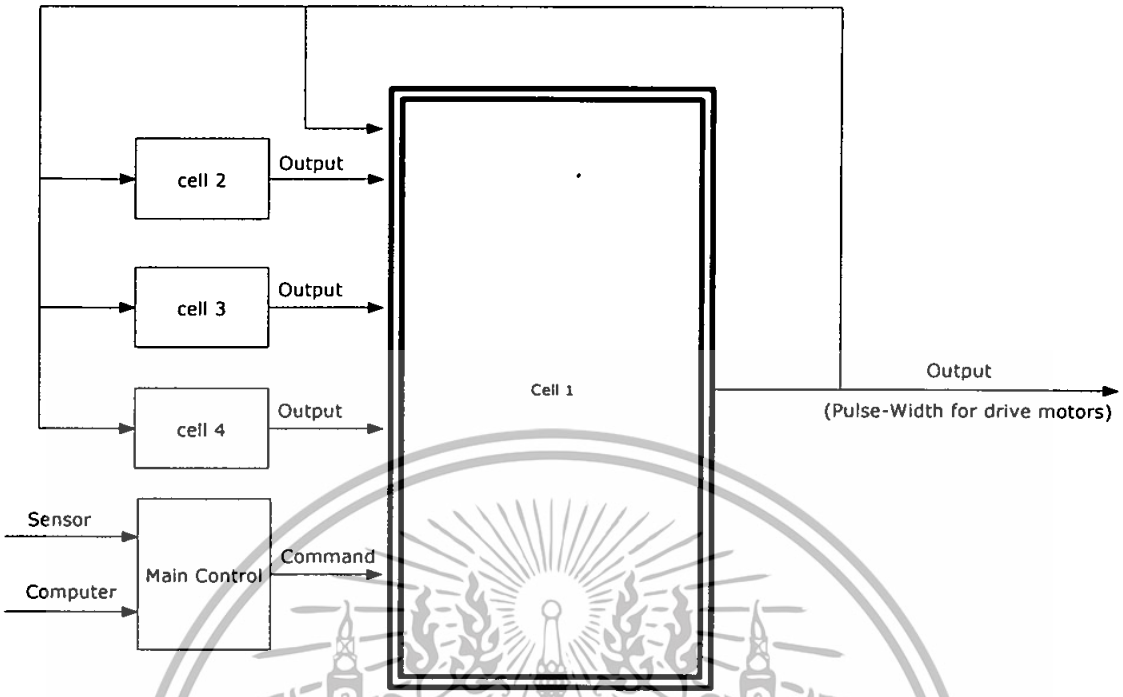
รูปที่ 3.9 State diagram ของขาที่ 2



รูปที่ 3.10 State diagram ของขาที่ 3



รูปที่ 3.11 State diagram ของขาที่ 4

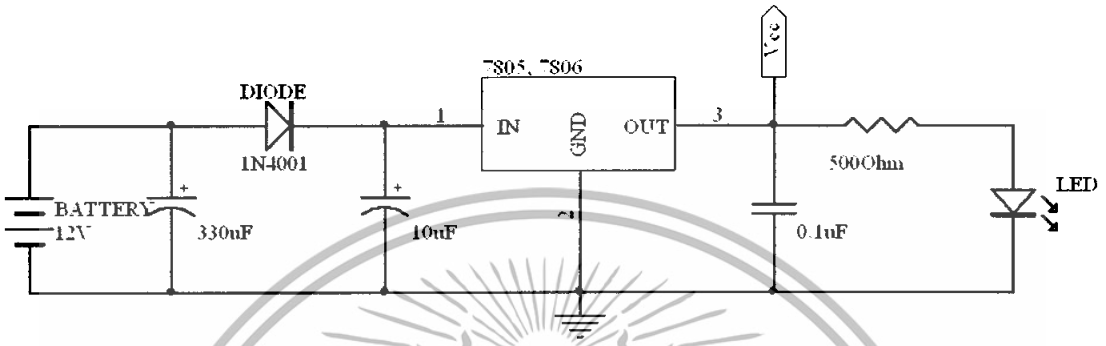


รูปที่ 3.12 แสดงการทำงานของคอนโทรลเลอร์เซลล์ที่ 1

3.4 วงจรที่ใช้ในการควบคุมหุ่นยนต์เดิน 4 ขา

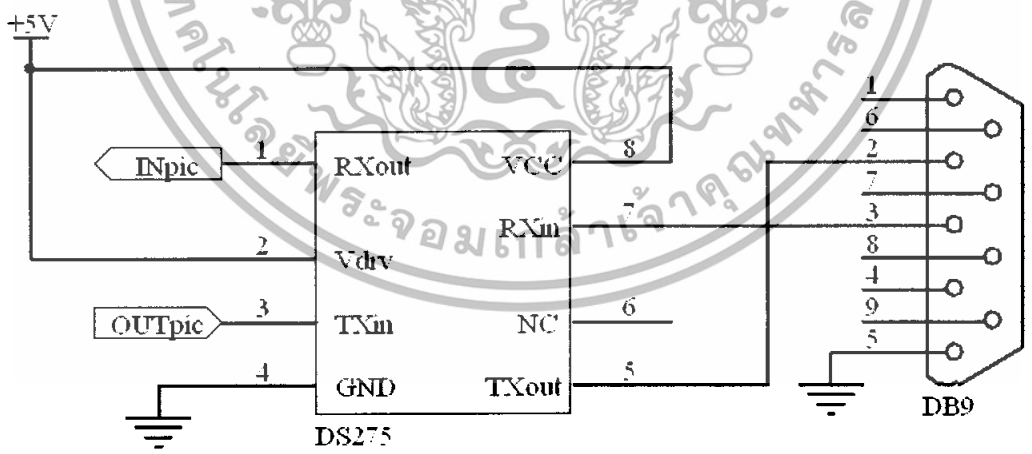
วงจรทั้งหมดที่ใช้ในการควบคุมหุ่นยนต์เดิน 4 ขานั้น สามารถแยกออกเป็นวงจรย่อยๆได้ดังนี้

1. วงจรแปลงแรงดันไฟ 5V, 6V



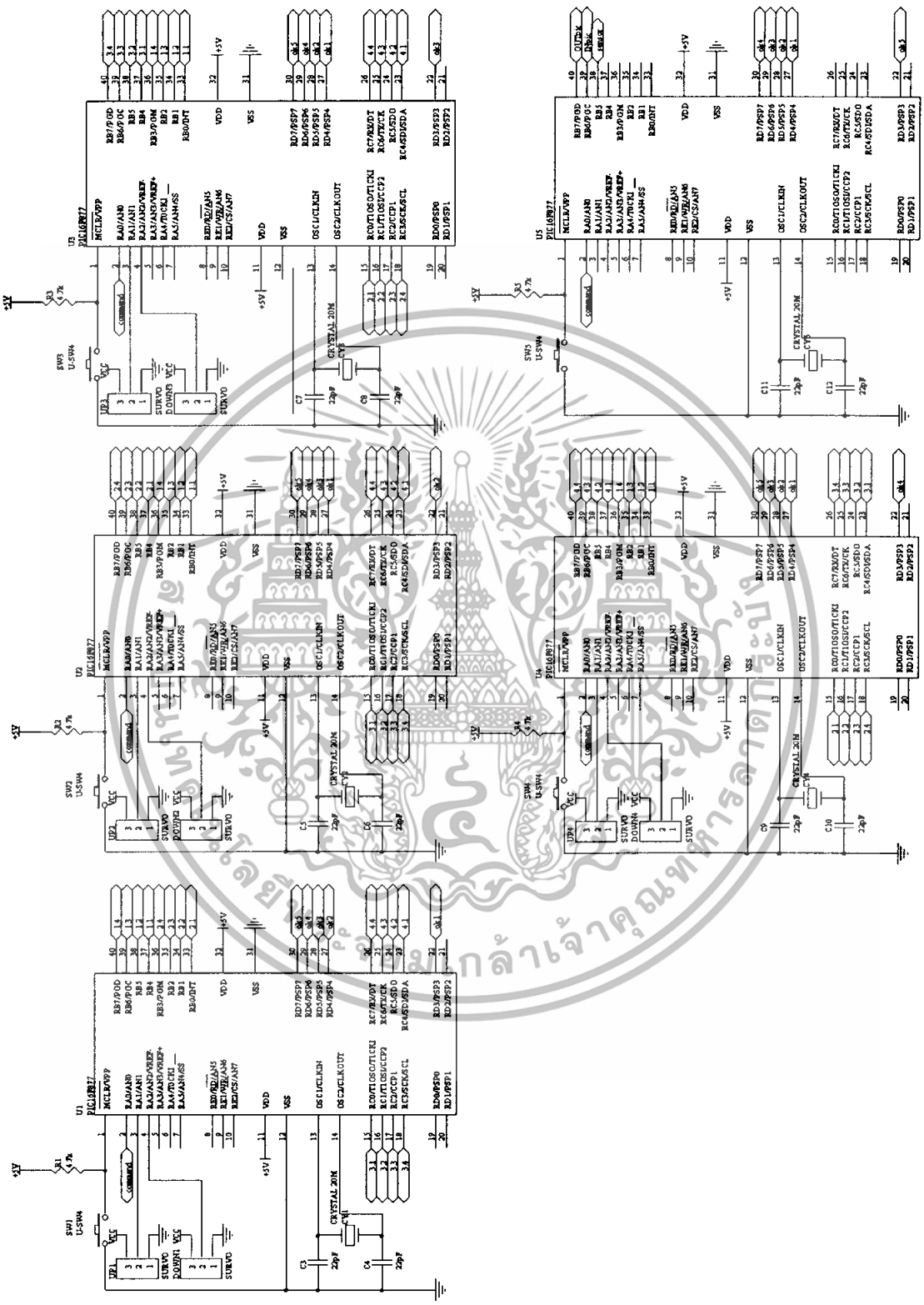
รูปที่ 3.13 แสดงวงจรแปลงเป็นแรงดันไฟ 5 V

2. วงจรที่ใช้แปลงสัญญาณจากระดับ TTL ไปเป็นระดับของ RS-232 และในทำนองเดียวกันก็รับระดับสัญญาณจาก RS-232 เพื่อแปลงเป็นระดับสัญญาณจากระดับ TTL ให้กับไมโครคอนโทรลเลอร์ได้



รูปที่ 3.14 แสดงวงจรที่ใช้แปลงสัญญาณ TTL กับ RS-232

3. วงจรไมโครคอนโทลเลอร์ที่ใช้ควบคุมหุ่นยนต์



รูปที่ 3.15 แสดงวงจรไมโครคอนโทลเลอร์ที่ใช้ควบคุมหุ่นยนต์

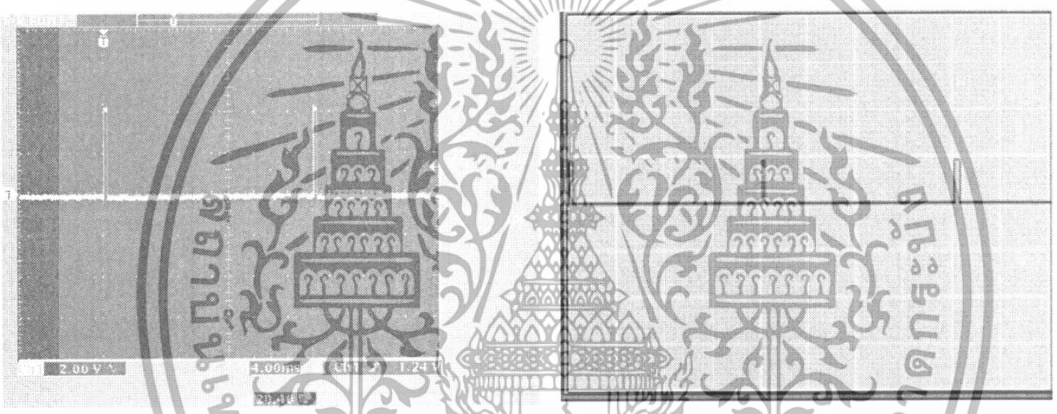
บทที่ 4

การทดลองและผลการทดลอง

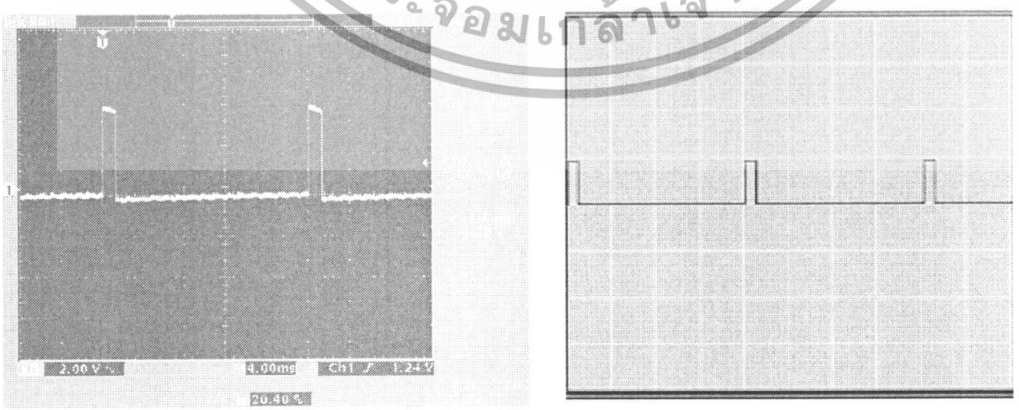
4.1 การทดลอง

- ทดสอบความกว้างของพัลส์ที่ตำแหน่งต่างๆของมอเตอร์
- ทดสอบการประมวลภาพวัตถุจากกล้องที่สภาวะต่างๆ

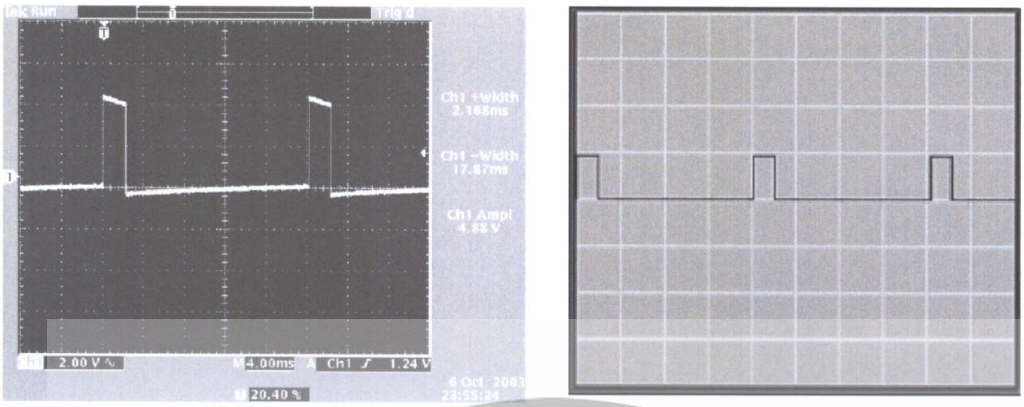
4.2 ผลการทดลอง



ก. แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งขวาสุด

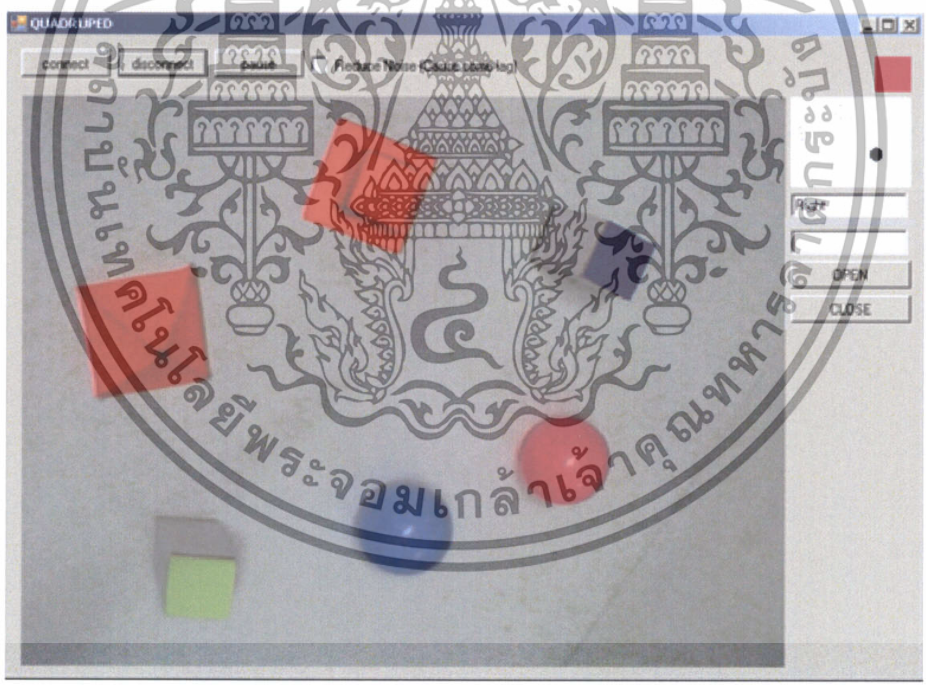


ข. แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งตรงกลาง

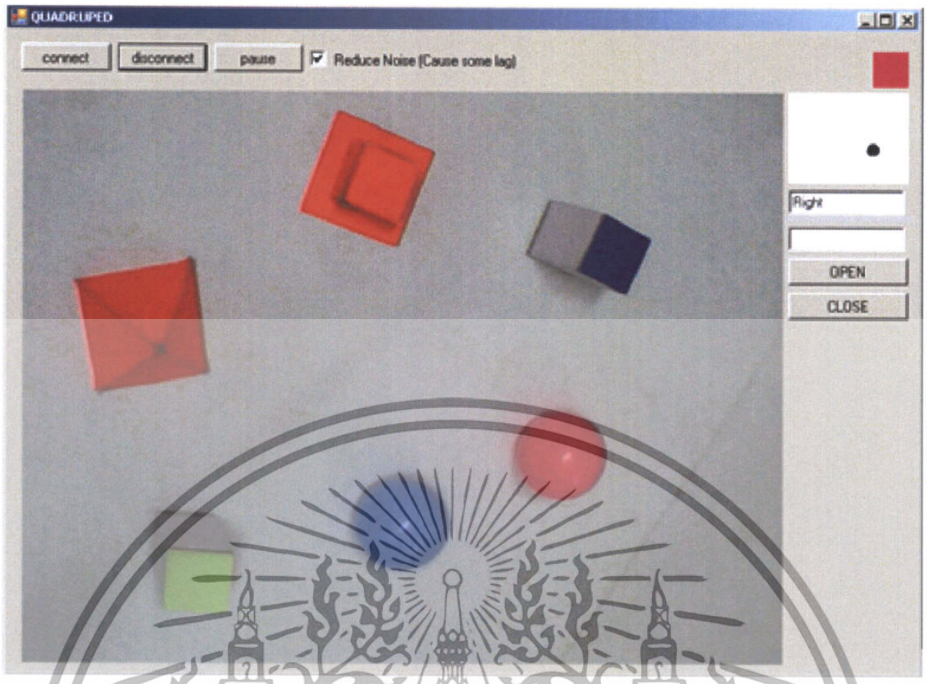


ค. แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งซ้ายสุด

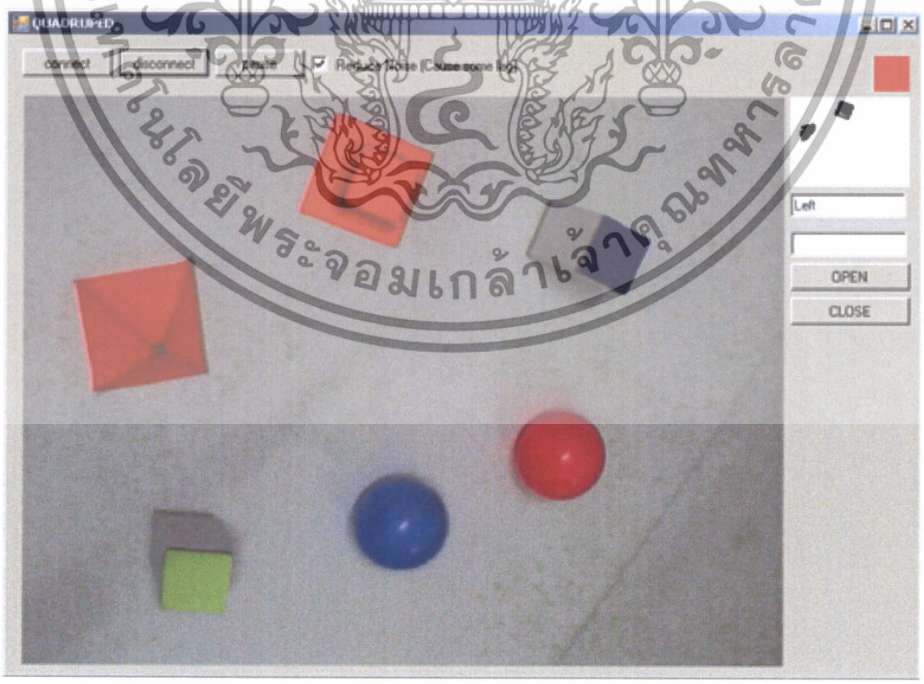
รูปที่ 4.1 แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งต่างๆ



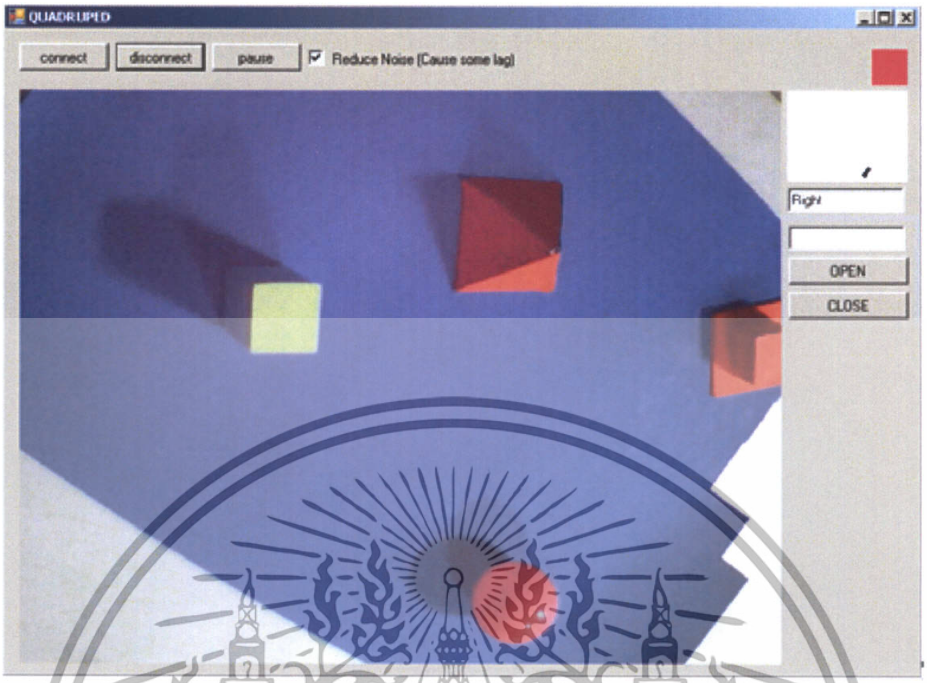
รูปที่ 4.2 แสดงการประมวลผลภาพ ที่ไม่มีการลดNOISE



รูปที่ 4.3 แสดงการประมวลผลภาพ ที่มีการลดNOISE



รูปที่ 4.4 แสดงการประมวลผลภาพ ที่มีวัตถุสีเดียวกัน 2 อัน



รูปที่ 4.5 แสดงการประมวลผลภาพในสภาวะแสงน้อย

รูปที่ 4.6 แสดงการประมวลผลภาพในสภาวะแสงมาก

บทที่ 5
วิเคราะห์และสรุปผลการทดลอง

จากผลการทดลองที่ได้จะเห็นว่าตำแหน่งของเซอร์โวแต่ละตัวเปลี่ยนไปตามความกว้างของพัลส์ที่ป้อนเข้าไป ซึ่งเปลี่ยนตำแหน่งได้ตั้งแต่ 0-180 องศา โดยที่ตำแหน่งที่ 0 องศาจะใช้ความกว้างพัลส์ประมาณ 0.3 ms ที่ตำแหน่ง 90 องศา จะใช้ความกว้างพัลส์ประมาณ 1.2 ms และ ที่ตำแหน่ง 180 องศา จะใช้ความกว้างพัลส์ประมาณ 2.1 ms นั่นแสดงว่าสามารถเลือกใช้ตำแหน่งเซอร์โวมอเตอร์ต่างๆได้จากการเทียบอัตราส่วนความกว้างของพัลส์

ในส่วนของโปรแกรมที่ใช้ในการประมวลผลภาพสามารถใช้งานได้ดีในสภาวะที่เหมาะสม ซึ่งโปรแกรมนี้สามารถกำหนดสีของวัตถุและบอกตำแหน่งของวัตถุนั้นได้ มีส่วนของ Reduce Noise ที่ช่วยทำให้ภาพที่ประมวลได้คมชัดมากขึ้น แต่จากการทดสอบด้วยวัตถุสีเดียวกัน 2 ชิ้น ค่าจุดศูนย์กลางที่ได้จากการคำนวณนั้นอาจทำให้บอกตำแหน่งของวัตถุไม่ได้หรือคลาดเคลื่อน

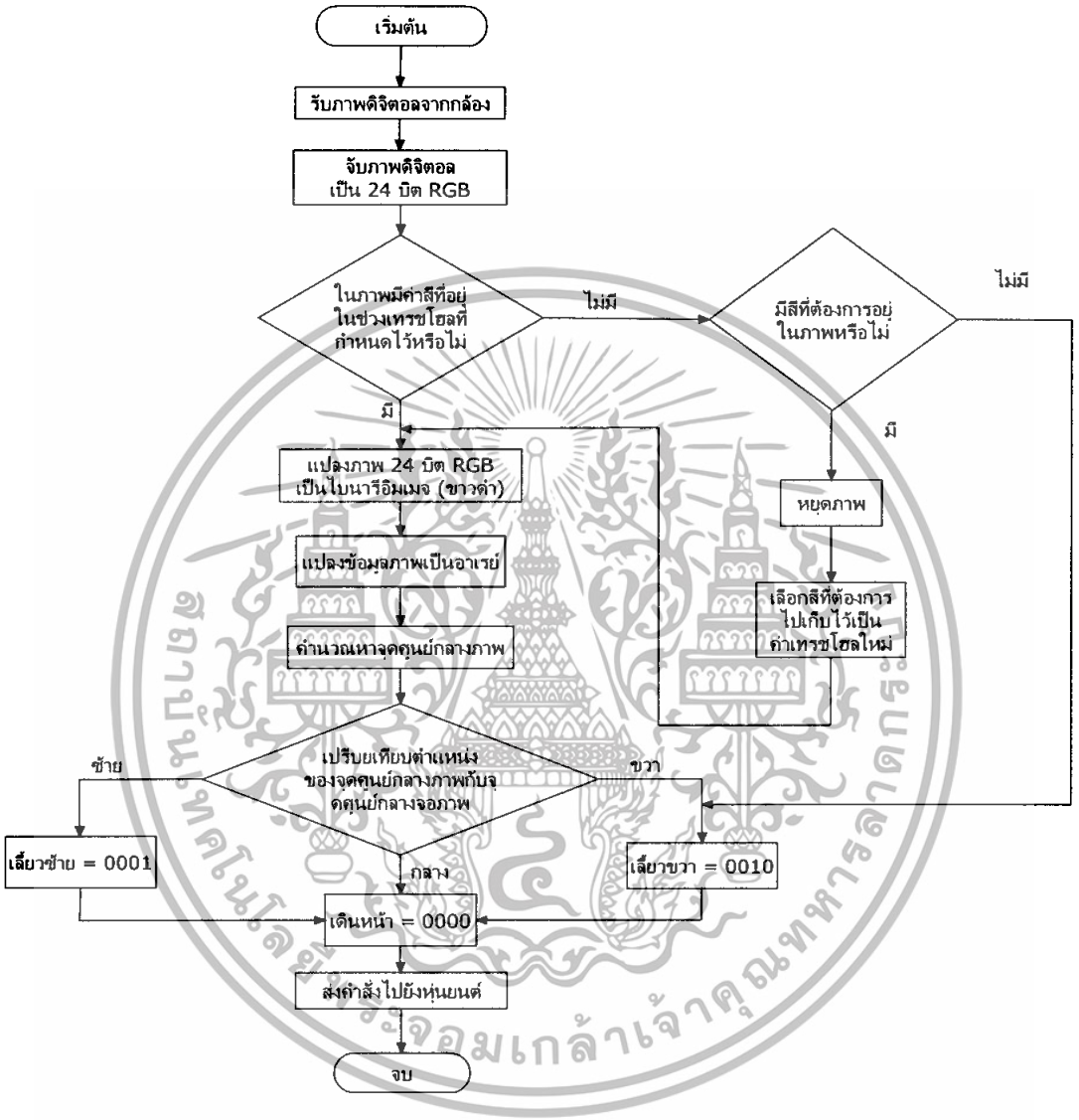
เมื่อทดสอบโปรแกรมในสภาวะที่แสงสว่างมากเกินไปหรือแสงน้อยมากๆจะได้ผลไม่ดีเท่าที่ต้องการ เนื่องจากสีที่ได้จากกล้องจะผิดเพี้ยนออกไป แม้กล้องที่ใช้จะมีการชดเชยแสงก็ตาม และในบางครั้งผิววัตถุที่มีความเงาจะทำให้ประมวลได้ยากขึ้น

เอกสารอ้างอิง

1. รศ.ดร.ชูชาติ ปิณฑวิรุจน์, “การประมวลผลภาพดิจิตอลด้วย Matlab”, ครั้งที่1, แผนกตำรา คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,2550.
2. นางสาวณภัทรพร หริ่งรอด, นายณรงค์พล ทองเหลือง, “โครงข่ายประสาทเทียมควบคุม-หุ่นยนต์”, ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2547
3. นาย เชิดศักดิ์ ประชากิตติกุล, นางสาว ณวดี เอี่ยมเจริญ, “หุ่นยนต์จำเส้นทางสู่สิ่งมีชีวิต”, ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2547

ภาคผนวก

หลักการประมวลผลภาพ



Flow Chart แสดงหลักการประมวลผลภาพ

โปรแกรมหลักที่ควบคุมหุ่นยนต์ทั้ง 4 เซลล์

```
#include <16F877a.h>
#device *=16
#use delay(clock=20000000)
#fuses HS,NOWDT
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
#use fast_io(a)
#byte port_a = 0x05
#byte PIR1 = 0x0C
#byte RCREG = 0x1A
#bit RCIF = PIR1.5
void sendoutput(char com)
{
  switch(com)
  {
    case '0':
      port_a = 0;
      break;
    case 'b':
      port_a = 1;
      break;
    case '2':
      port_a = 2;
      break;
    case 'd':
      port_a = 3;
      break;
  }
}
char command() {
  char temp;
  while(!RCIF);
  temp = RCREG;
  return temp;
}
void main() {
  port_b_pullups(TRUE);
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_CLOCK_DIV_2);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE); setup_counters(RTCC_INTERNAL,RTCC_DIV_8);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  set_tris_a(0x00);
  while(1)
  {
    sendoutput(command());
  }
}
```

โปรแกรมควบคุมเซลล์ที่1

```
#include <source_head.c>
unsigned int16 state[]={0x0951,0x6129,0x602A,0x2961,
                0x2951,0x6125,0x6126,0x6129,0x5129,0x2561,0x2661,0x2961,
                0x2A61,0xA129,0x9129,0x6129,0x612A,0x29A1,0x2991,0x2961};
unsigned int8 state2[]={0x90,0xA0,0x10,0x10,
                0x51,0x61,0x91,0x91,0x11,0x11,0x11,0x11,
                0x92,0x92,0x92,0xA2,0x12,0x12,0x12,0x12};

int top_step[]={12,18,8}; //00,01,10
int down_step[]={12,24,3}; //00,01,10
int self=0x01;
#include <source.c>
```

โปรแกรมควบคุมเซลล์ที่ 2

```
#include <source_head.c>
unsigned int16 state[]={0x0915,0x6192,0x60A2,0x2916,
                0x2915,0x6152,0x6162,0x6192,0x5192,0x2516,0x2616,0x2916,
                0x2A16,0xA192,0x9192,0x6192,0x61A2,0x291A,0x2919,0x2916};
unsigned int8 state2[]={0x20,0x20,0x60,0x50,
                0x21,0x21,0x21,0x21,0x61,0x61,0x61,0x51,
                0x22,0x22,0x22,0x22,0xA2,0x92,0x62,0x62};

int top_step[]={12,16,8}; //00,01,10
int down_step[]={12,24,3}; //00,01,10
int self=0x06;
#include <source.c>
```

โปรแกรมควบคุมเซลล์ที่ 3

```
#include <source_head.c>
unsigned int16 state[]={0x0519,0x6291,0x62A0,0x2619,
                0x2519,0x6251,0x6261,0x6291,0x5291,0x2615,0x2616,0x2619,
                0x261A,0xA291,0x9291,0x6291,0x62A1,0x2A19,0x2919,0x2619};
unsigned int8 state2[]={0x10,0x00,0x90,0x90,
                0x11,0x11,0x11,0x11,0x51,0x61,0x91,0x91,
                0x12,0x12,0x12,0x12,0x92,0x92,0x92,0xA2};

int top_step[]={13,16,8}; //00,01,10
int down_step[]={12,24,3}; //00,01,10
int self=0x09;
#include <source.c>
```

โปรแกรมควบคุมเซลล์ที่ 4

```
#include <source_head.c>
unsigned int16 state[]={0x9510,0x1296,0x02A6,0x9612,
                0x9512,0x1256,0x1266,0x1296,0x1295,0x5612,0x6612,0x9612,
                0xA612,0x129A,0x1299,0x1296,0x12A6,0x9A12,0x9912,0x9612};
unsigned int8 state2[]={0x60,0x60,0x20,0x00,
                0x61,0x61,0x61,0x51,0x21,0x21,0x21,0x21,
                0xA2,0x92,0x62,0x62,0x22,0x22,0x22,0x22};

int top_step[]={13,17,8}; //00,01,10
int down_step[]={12,24,3}; //00,01,10
int self=0x02;
#include <source.c>
```

Source_head.c

```
#include <16F877a.h>
#device *=16
#use
delay(clock=20000000,RESTART_WDT)
#fuses HS,NOWDT
#use fast_io(a)
#use fast_io(b)
#use fast_io(c)
#use fast_io(d)

#byte port_a = 0x05
#byte port_b = 0x06
#byte port_c = 0x07
#byte port_d = 0x08
#bit top_survo = port_a.1
#bit d_survo = port_a.2
#bit acc = port_d.3
#bit Hcom = port_a.3
#bit Lcom = port_a.0
```

Source.c

```
int32 datain=0;
int command=3,index=0;

#int_TIMER0
TIMER0_isr() {
  restart_wdt();
}

void send_output()
{
  int temp=0,cacc=0;

  temp = self;
  temp <<= 4;
  port_b = temp&0xF0;

  acc = 1;
  while(cacc!=0x70)
  {
    cacc = port_d;
    cacc &= 0b01110000;
  }

  command = 0;
  command = Hcom;
  command <<= 1;
  command += Lcom;

  acc = 0;
}

void generate(int t1,int t2)
{
  int time,i,tmin,tmax;
  int1 flag;

  if(t1>t2)
  {
    tmax = t1;
    tmin = t2;
    flag = 1;
  }
  else
  {
    tmax = t2;
    tmin = t1;
    flag = 0;
  }

  for(i=0;i<50;i++)
  {
    top_survo = 1;
    d_survo = 1;

    for(time=0;time<tmin;time++)
      delay_us(100);

    if(flag==1)
      d_survo = 0;
    else
      top_survo = 0;

    for(time=0;time<(tmax-tmin);time++)
      delay_us(100);

    top_survo = 0;
    d_survo = 0;

    for(time=0;time<(200-tmax);time++)
      delay_us(100);
  }
}

void prepare_drive()
{
  unsigned int8 out,outtop,outdown,ton1,ton2;

  out = state2[index];
  out &= 0xF0;
  out >>= 4;
  self = out&0x0F;

  outtop = self & 0b00001100;
  outdown = self & 0b00000011;
  outtop >>= 2;

  switch(outtop)
  {
    case 0:
      ton1 = top_step[0];
      break;
    case 1: //FORESTEP
      ton1 = top_step[1];
      break;
    case 2: //BACKSTEP
      ton1 = top_step[2];
      break;
```

```
  }

  switch(outdown)
  {
    case 0:
      ton2 = down_step[0];
      break;
    case 1:
      ton2 = down_step[1];
      break;
    case 2:
      ton2 = down_step[2];
      break;
  }

  //wait_drive();
  generate(ton1,ton2);
}

void find_state()
{
  unsigned int i,j,k,temp,temp2,min=40;
  unsigned int32 data;

  if(command!=3)
  {
    if(command==0)
    {
      j=0;
      k=3;
    }
    else if(command==1)
    {
      j=4;
      k=11;
    }
    else
    {
      j=12;
      k=19;
    }

    for(i=j;i<=k;i++)
    {
      data = state[i];
      data <<= 4;
      temp = state2[i];
      temp &=0x0f;
      data += temp;

      data ^= datain;
      temp2=0;

      while(data>0)
      {
        temp = data%2;
        if(temp == 1)
          temp2++;
        data /= 2;
      }
      if(temp2<min)
      {
        min=temp2;
        index=i;
      }
    }
  }
}

void save_input()
{
  int tcell=0;
  datain=0;

  datain = port_c;
  datain <<= 4;

  tcell = port_b;
  tcell &= 0x0F;
  datain += tcell;
  datain <<= 4;

  datain += self;
  datain <<= 4;

  datain += command;
}

void main() {

  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_CLOCK_DIV_2);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE);

setup_counters(RTCC_INTERNAL,RTCC_DI
V_8);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  enable_interrupts(INT_TIMER0);
  enable_interrupts(global);

  set_tris_a(0b00001001);
  set_tris_b(0x0F);
  set_tris_c(0xFF);
  set_tris_d(0xF0);
  send_output();

  while(TRUE)
  {
    save_input();
    find_state();
    prepare_drive();
    send_output();
  }
```