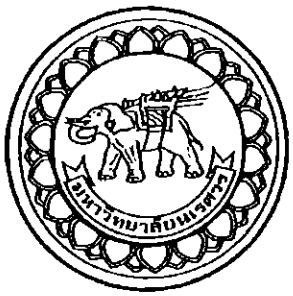


# ระบบควบคุมแบบป้อนกลับและการวาดตำแหน่งเชิงกลสำหรับหุ่นยนต์

**Feedback control and position drawing system for robot**


นายธญา เติมลาภ รหัสนิสิต 49371033

นายภูเบศ ประสาทชัย รหัสนิสิต 49371255




ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2553

# ใบรับรองโครงงานวิศวกรรม

| | | | |
|---|---|---|---|
| **หัวข้อโครงงาน** | ระบบควบคุมแบบป้อนกลับและวาดตำแหน่งเชิงกลสำหรับหุ่นยนต์ | | |
| **ผู้ดำเนินโครงงาน** | นายธญา | เติมลาภ | รหัส 49371033 |
| | นายภูเบศ | ประสาทชัย | รหัส 49371255 |
| **อาจารย์ที่ปรึกษา** | อาจารย์เศรษฐา | ตั้งค้าวานิช | |
| **สาขาวิชา** | วิศวกรรมคอมพิวเตอร์ | | |
| **ภาควิชา** | วิศวกรรมไฟฟ้าและคอมพิวเตอร์ | | |
| **ปีการศึกษา** | 2553 | | |

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการการสอบโครงงานวิศวกรรม

เศรษฐา ตั้งค้าวานิช .....ประธานกรรมการ

( อาจารย์เศรษฐา ตั้งค้าวานิช )

F. Kitt .....กรรมการ

( ดร.พงศ์พันธ์ กิจสนาโยธิน )

สิรภพ คชรัตน์ .....กรรมการ

( อาจารย์สิรภพ คชรัตน์ )

| | | | |
|---|---|---|---|
| **หัวข้อโครงงาน** | ระบบควบคุมแบบป้อนกลับและการวาดตำแหน่งเชิงกลสำหรับหุ่นยนต์ | | |
| **ผู้ดำเนินโครงงาน** | นายธญา | เติมลาภ | รหัส 49371033 |
| | นายภูเบศ | ประสาทชัย | รหัส 49371255 |
| **อาจารย์ที่ปรึกษา** | อาจารย์เศรษฐา | ตั้งค้าวานิช | |
| **สาขาวิชา** | วิศวกรรมคอมพิวเตอร์ | | |
| **ภาควิชา** | วิศวกรรมไฟฟ้าและคอมพิวเตอร์ | | |
| **ปีการศึกษา** | 2553 | | |

.....


## บทคัดย่อ

วัตถุประสงค์ของโครงงานเรื่อง ระบบควบคุมแบบป้อนกลับและการวาดตำแหน่งเชิงกลสำหรับหุ่นยนต์ เป็นการออกแบบและสร้างโปรแกรมเพื่อแสดงภาพจำลองสถานะการเคลื่อนไหวของหุ่นยนต์ตามข้อต่อในส่วนต่างๆรวมถึงการวัดระยะทางการเคลื่อนที่ของหุ่นยนต์ และควบคุมการเคลื่อนที่ของหุ่นยนต์ การจำลองภาพหุ่นยนต์และแสดงผลผ่านทางหน้านั้นมีความจำเป็นต่อผู้ควบคุมเป็นอย่างมาก เนื่องจากหุ่นยนต์ผู้ควบคุมหุ่นยนต์นั้นในสถานการณ์จริงไม่สามารถควบคุมโดยเห็นตัวหุ่นยนต์ได้จึงไม่รู้ว่าในตอนนั้นหุ่นยนต์ของเราก็มีปรับเปลี่ยนรูปร่างของข้อต่อส่วนต่างๆของหุ่นยนต์เป็นอย่างไร เพื่อจะได้วิเคราะห์ได้ว่าเมื่อพบกับส่งกรีดขวางหรืออุปสรรคอยู่ข้างหน้าที่ผู้ควบคุมได้เห็นทางหน้าจอนั้น ผู้ควบคุมหุ่นยนต์ก็จะสามารถปรับเปลี่ยนตำแหน่งข้อต่อของหุ่นยนต์ให้สอดคล้องกับอุปสรรคหรือสิ่งกรีดขวางที่พบได้ โดยในส่วนของการออกแบบระบบนั้นจะมีด้วยกัน 2 ส่วนใหญ่ คือส่วนของการควบคุมการเคลื่อนที่ของหุ่นยนต์ กับการรับค่าการนับรอบการหมุนของเอ็นโค้ดเดอร์เพื่อนำมาเปรียบเทียบการเปลี่ยนแปลงของตำแหน่งหุ่นยนต์

โดยส่วนของการควบคุมการเคลื่อนที่นั้นผู้ควบคุมสามารถควบคุมหุ่นยนต์บนคอมพิวเตอร์ผ่านทางอุปกรณ์แบบไร้สายแล้วส่งคำสั่งไปเข้าไมโครคอนโทรลเลอร์เพื่อทำการประมวลผลคำสั่งที่ได้รับมาแล้วไปสั่งการให้มอเตอร์ทำงานตามข้อต่อต่างๆของหุ่นยนต์ โดยที่ตามข้อต่อต่างๆจะมีเอ็นโค้ดเดอร์ติดไว้กับชุดแกนหมุนเพื่อให้เอ็นโค้ดเดอร์มีการคืนค่ากลับมายังไมโครคอนโทรลเลอร์ประมวลผลค่าระยะทางและค่าจำนวนรอบการหมุนแล้วส่งข้อมูลต่อไปยังคอมพิวเตอร์เพื่อที่จะระบุสถานะการเคลื่อนไหวและยะทางการเคลื่อนที่ของหุ่นยนต์ โครงงานนี้จะอธิบายถึงการทำงานของระบบควบคุมแบบป้อนกลับและการวาดตำแหน่งเชิงกลสำหรับหุ่นยนต์ที่พัฒนาขึ้น อุปกรณ์สำคัญที่ใช้ เช่น เอ็นโค้ดเดอร์ รวมไปถึงอุปกรณ์ต่างๆ ที่ใช้ควบคุมการทำงานของหุ่นยนต์ สิ่งที่ได้รับจากโครงงานนี้คือ โปรแกรมที่สามารถระบุสถานะการเคลื่อนไหวและระยะทางการเคลื่อนที่ของหุ่นยนต์ที่ผู้ควบคุมสามารถนำไปใช้ได้กับการควบคุมหุ่นยนต์จริง

| | | | |
|---|---|---|---|
| **Project Title** | Feedback control and position drawing system for robot | | |
| **Name** | Mr. Thaya | Thermlarp | ID. 49371378 |
| | Mr. Poobed | Prasartchai | ID. 49371255 |
| **Project Advisor** | Mr. Settha | Tangkavanit | |
| **Major** | Computer Engineering | | |
| **Department** | Electrical and Computer Engineering | | |
| **Academic Year** | 2010 | | |

.....


## ABSTRACT

The object of project work about supply to feedback control and position drawing system for robot . It was designed typing and created programming for to show to imitate picture the position of movement of Robot follow joint in the other different included distance measuring movement of Robot and they have controller movement of Robot. Imitating picture Robot and show the result pass the face that they need for the controller that real situation can not control by see that body of Robot,So they don't know our Robot having change figure of joint in the other how different part of Robot. By designing part of system having two the big part. It is the part of movement of Robot with receiving value counting round cycle of Encoder for preparing changed the Robot position

By the part of movement controller can control the Robot on computer pass material way wireless type, then send to order into microcontroller for to result to compile order then to order motor have working follow joint of Robot. By that follow have Encoder stick with the cycle axis for give Encoder have returning value to microcontroller result to compile order value distance and amount round cycle then information to computer for will indicate movement and distance movement of Robot. This project work will explain to doing work of Feedback control and position drawing system for Robot that development. The importance material to use for example including different material use control working of Robot. The thing get from this project work is capacity programming to this specify movement of position and distance movement of Robot can use with control Robot.

# กิตติกรรมประกาศ

ทางคณะผู้จัดทำโครงงาน "ระบบควบคุมแบบป้อนกลับและการวาดตำแหน่งเชิงกลสำหรับหุ่นยนต์" ขอขอบคุณ อาจารย์เศรษฐา ตั้งค้าวานิช ที่ให้ความช่วยเหลือในโครงงานนี้ให้สามารถดำเนินการไปได้ด้วยดี โดยช่วยให้คำแนะนำปรึกษาเกี่ยวกับโครงงานตลอด ทั้งให้ความเอื้อเฟื้อสถานที่ในการทำงานและอุปกรณ์เครื่องมือต่างๆ อีกทั้งอาจารย์ทุกท่านในภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ที่ให้คำแนะนำและช่วยเหลือในครั้งนี้

นายธญา เดิมลาภ

นายภูเบศ ประสาทชัย

# สารบัญ

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญตาราง

# สารบัญรูป

รูปที่ หน้า

# สารบัญรูป (ต่อ)

รูปที่ หน้า

# สารบัญรูป (ต่อ)

# บทที่ 1
# บทนำ

## 1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจากหุ่นยนต์กู้ภัยที่ได้มีการพัฒนาขึ้นนั้นยังขาดในส่วนของการระบุตำแหน่งภาพของตัวหุ่นยนต์ในระยะไกลที่ผู้บังคับหุ่นยนต์ไม่สามารถดูตัวหุ่นยนต์ในระยะนั้นๆได้ เช่น ตำแหน่งแขนกล ตำแหน่งคอ ตำแหน่งข้อศอก ซึ่งทุกตำแหน่งที่ได้กล่าวมานั้นมีความสำคัญต่อการเคลื่อนที่ในสถานการณ์ต่างๆของหุ่นยนต์ มีส่วนสำคัญในการค้นหาผู้ประสบภัย

ดังนั้น โครงงานนี้จึงจัดทำขึ้นมา เพื่อสามารถระบุตำแหน่งของข้อต่อในส่วนต่างๆของหุ่นยนต์ได้ เพื่อให้การทำงานสะดวกและยังช่วยให้ผู้บังคับนั้นเห็นภาพของหุ่นยนต์ในกรณีที่ผู้บังคับไม่สามารถเข้าไปบังคับหุ่นยนต์ในระยะใกล้ได้ เพื่อที่ว่าผู้บังคับจะได้ตัดสินใจได้ว่า เมื่อผู้บังคับได้เห็นสภาพแวดล้อมและสิ่งกีดขวางต่างๆที่อยู่ข้างหน้าของหุ่นยนต์จากกล้องแล้วจะสามารถตัดสินใจได้ว่าควรที่จะปรับเปลี่ยนรูปแบบของข้อต่อในส่วนต่างๆในรูปแบบใดที่จะสามารถทำให้หุ่นยนต์เคลื่อนที่ผ่านอุปสรรคหรือสิ่งกีดขวางข้างหน้าของหุ่นยนต์นั้นได้ๆ

## 1.2 จุดประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาการทำงานของไมโครคอนโทรลเลอร์

1.2.2 เพื่อใช้ความรู้เชิงวิศวกรรมทางด้านอิเล็กทรอนิกส์

1.2.3 เพื่อศึกษาการรับส่งข้อมูลแบบไร้สาย

1.2.4 เพื่อศึกษาเกี่ยวกับ Encoder

1.2.5 เพื่อศึกษาและออกแบบโปรแกรมการแสดงภาพในระบบ 2D แบบรับ input ภายนอก

## 1.3 ขอบเขตของโครงงาน

1.3.1 สามารถจำลองภาพของหุ่นยนต์ออกมาเป็นแบบ 2D

1.3.2 สามารถรับ-ส่งข้อมูลที่ได้จากการคำนวณค่าของ Encoder แล้วนำค่าที่ได้ส่งผ่านการสื่อสารแบบไร้สายระยะไกล

1.3.3 รับค่าจาก Encoder มาเปลี่ยนแปลงรูปสถานะของหุ่นยนต์ตามข้อต่อต่างๆได้

## 1.4 ผลคาดว่าที่จะได้รับ

1.4.1 สามารถจำลองภาพหุ่นยนต์เป็นรูป 2D ได้

1.4.2 สามารถรับ-ส่งข้อมูลที่ได้จาก Encoder ผ่านทางการสื่อสารแบบไร้ระยะไกลได้

1.4.3 สามารถคำนวณค่าที่ได้จาก Encoder ได้

1.4.4 สามารถรับค่าจาก Encoder แล้วประมวลผลเชื่อมต่อกับภาพจำลองหุ่นยนต์เพื่อเป็นสถานะของหุ่นยนต์ตามข้อต่อต่างๆได้

## 1.5 แผนการดำเนินงาน

ตารางที่ 1.1 ตารางแผนดำเนินงาน

| รายการ | พ.ศ. 2552 | | | | | | | พ.ศ. 2553 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | มิ.ย. | ก.ค. | ส.ค. | ก.ย. | ต.ค. | พ.ย. | ธ.ค. | ม.ค. | ก.พ. | มี.ค. |
| 1.5.1 ศึกษา Encoder ในรูปแบบต่างๆ | ← | → | | | | | | | | |
| 1.5.2 ศึกษาระบบไมโครคอนโทรลเลอร์ | ← | → | | | | | | | | |
| 1.5.3 จัดหาวัสดุอุปกรณ์ | ← | → | | | | | | | | |
| 1.5.4 จัดหางบประมาณ | ← | → | | | | | | | | |
| 1.5.5 หาเครื่องมือและสถานที่การทำโครงงาน | ← | → | | | | | | | | |
| 1.5.6 จัดหาและปรับปรุงหุ่นยนต์กู้ภัย | | ← | → | | | | | | | |
| 1.5.7 ทำการประกอบชุดวงจรไมโครคอนโทรลเลอร์ | | | ← | → | | | | | | |
| 1.5.8 เขียนโปรแกรมรับค่าสัญญาณพัลซ์ของ Encoder | | | ← | → | | | | | | |
| 1.5.9 เขียนภาพจำลองหุ่นยนต์แบบ 3D | | | ← | → | | | | | | |
| 1.5.10 นำระบบ2ส่วนมาเข้าร่วมกันทดสอบและปรับปรุงการ | | | | | | | ← | → | | |

## 1.6 งบประมาณที่ใช้

| | | |
|---|---|---|
| อุปกรณ์อิเล็กทรอนิกส์ | | 420 บาท |
| ไมโครคอนโทรลเลอร์ | | 180 บาท |
| Encoder | | 1,400 บาท |
| | รวมทั้งหมด | 2,000 บาท |

# บทที่ 2
# หลักการและทฤษฎีที่เกี่ยวข้อง

ในการจัดทำโครงงานนี้ประกอบไปด้วยหลักการและทฤษฎีต่างๆ เพื่อให้ผู้บังคับหุ่นยนต์นั้นรู้สภาพหุ่นยนต์และเพื่อจะได้วิเคราะห์อุปสรรคข้างหน้าที่จะเจอได้ว่าเราควรเปลี่ยนแปลงรูปร่างตรงข้อต่อส่วนไหนของหุ่นยนต์ เพื่อให้ผ่านอุปสรรคที่อยู่ตรงข้างหน้านั้นไปได้ โดยจะใช้หลักการร่วมกับส่วนควบคุมหุ่นยนต์ เข้ารวมกับการใช้ Encoder เพื่อนับรอบการหมุนของมอเตอร์แล้วนำข้อมูลสัญญาณที่ได้รับจาก Encoder นั้นๆมาส่งผ่านการสื่อสารระบบไร้สายมายังผู้บังคับ โดยค่าที่ได้รับมานั้นจะนำมาเป็นอินพุตในการเปลี่ยนแปลงรูปร่างของหุ่นยนต์ให้ผู้บังคับหุ่นยนต์นั้นเห็นถึงการเปลี่ยนแปลงรูปร่าง เมื่อเวลาที่ผู้บังคับหุ่นยนต์กดปุ่มบังคับให้มอเตอร์หมุน

## 2.1 มอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงนั้นถูกใช้กันอย่างกว้างขวาง โดยมอเตอร์กระแสไฟฟ้านั้นจะมีค่าสัมประสิทธิ์ ความสัมพันธ์ระหว่างความเร็วรอบมอเตอร์กับแรงบิด สามารถเลือกมอเตอร์ไฟฟ้ากระแสตรงที่เหมาะกับการใช้งานในรูปแบบต่างๆได้เกือบทุกรูปแบบ สำหรับการใช้งานของทั้งมอเตอร์และการสร้างใหม่ (Regeneration) ในทิศทางและการหมุน การทำงานอย่างต่อเนื่องของ DC Motors โดยทั่วไปจะอยู่ในช่วงความเร็ว 8 ต่อ 1 รวมทั้ง การลดภาระหรือการลดความเร็วในระยะเวลาสั้น ๆ จะอยู่ในช่วงไร้ขอบเขต (ควบคุมการลดความเร็วลงถึงศูนย์รอบ ต่อนาทีได้อย่างราบเรียบนุ่มนวล) มักจะใช้มอเตอร์ไฟฟ้ากระแสตรง เมื่อมันต้องจ่ายแรงบิดที่จะทำให้มอเตอร์หมุนมากกว่าแรงบิดขณะใช้งานปกติ 3 เท่าหรือ มากกว่า และในสถานการณ์ฉุกเฉิน มอเตอร์ไฟฟ้ากระแสตรงสามารถที่จะจ่ายแรงบิดได้มากกว่า 5 เท่าของแรงบิดใช้งานปกติ โดย ปราศจากการหยุดกลางคัน (Stalling) (ต้นกำลังสามารถจ่ายกำลังให้ได้)

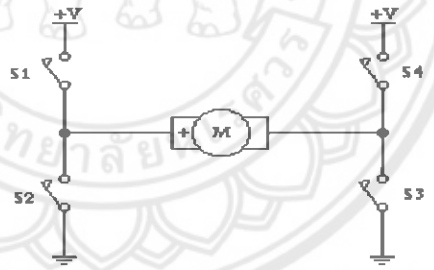
### 2.1.1 การควบคุมมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรง สามารถที่จะควบคุมความเร็วจนถึงศูนย์รอบต่อนาทีได้อย่างไม่มีอุปสรรคโดยการเร่งในทิศทางตรงกันข้ามอย่างทันทีทันใด โดยไม่ต้องสับเปลี่ยนวงจรกำลังและมอเตอร์ไฟฟ้ากระแสตรง จะตอบสนองการเปลี่ยนแปลงของสัญญาณควบคุมได้อย่างรวดเร็ว เนื่องจากมีอัตราแรงบิดต่อความเฉื่อยสูงขดลวดสนามแม่เหล็กมอเตอร์ไฟฟ้ากระแสตรง โดยทั่วไปจะแบ่งโดยแยกประเภทของสนามแม่เหล็กของมอเตอร์ ได้แก่ ลวดขนาน ( Shunt-wound ) ขดลวดอนุกรม ( Series-wound ) ขดลวดแบบผสม ( Compound-wound ) นอกจากนี้ ยังมีแบบแม่เหล็ก

ถาวรและแบบไม่มีแปรงถ่าน (Brushless) ใช้งานอยู่บ้างเหมือนกัน ปกติจะเป็นมอเตอร์ที่มีกำลังม้าต่ำ ๆ มอเตอร์อาจจะแบ่งประเภทเป็นแบบใช้งานต่อเนื่องหรือใช้งานเป็นช่วงๆ มอเตอร์ที่ใช้งานต่อเนื่อง สามารถที่จะทำงานโดยไม่ต้องมีเวลาหยุดพักเลย ได้การควบคุมความเร็วของมอเตอร์กระแสตรงมีหลายวิธีด้วยกัน ซึ่งอาจจะใช้วิธีการควบคุมแบบพื้นฐานทั่วไป เช่นการควบคุมด้วยวิธีการใช้ตัวต้านทานปรับค่า โดยต่ออนุกรมกับมอเตอร์ หรือใช้วิธีการการควบคุมโดยการเปลี่ยนค่าของระดับแรงดันที่ป้อนให้กับมอเตอร์แต่การควบคุม ในวิธีดังกล่าวถึงแม้ว่าจะควบคุมความเร็วมอเตอร์ให้คงที่ได้ แต่ที่ความเร็วต่ำจะส่งผลให้แรงบิดต่ำไปด้วย ดังนั้นเราจึงเลือกใช้วิธีการควบคุมโดยการจ่ายกระแสไฟให้กับมอเตอร์เป็นช่วงๆ โดยอาศัยกระแสไฟที่ป้อนให้กับมอเตอร์ให้เป็นค่าเฉลี่ยที่เกิดขึ้นในแต่ละช่วง ซึ่งเราเรียกว่าวิธีการของการมอดูเลชันทางความกว้างของพัลส์ PWM (Pulse Width Modulation)

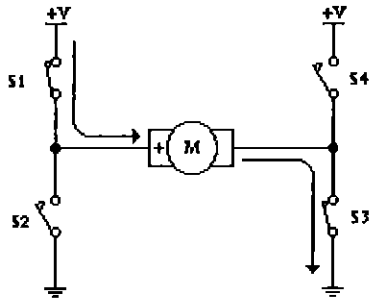
### 2.1.2 การควบคุมทิศทางการหมุนมอเตอร์

หลักการทำงานของมอเตอร์ในตัวหุ่นกู้ภัยนั้นเราตะใช้หลักการทำงานของวงจร H-bridge switching จะประกอบไปด้วยสวิทช์ 4 ตัว คือ S1,S2,S3,S4 ซึ่งในรูป จะใช้ DC Motor เป็นโหลดของวงจร ในสภาวะเริ่มต้น สวิทช์ทุกตัว OFF อยู่ก็จะไม่มีอะไรเกิดขึ้น เนื่องจากไม่มีกระแสไฟฟ้าไหลเข้าสู่มอเตอร์



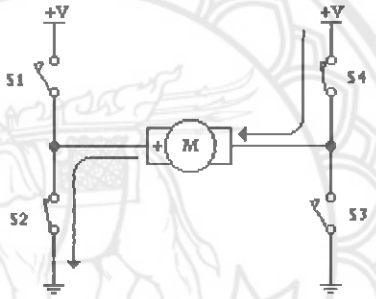

รูปที่ 2.1 ภาพแสดง H-Bridge Switching

เมื่อทำการ ON สวิทช์ S1 และ S3 พร้อมกัน จะเป็นการเชื่อมวงจร ทำให้มีกระแสไฟฟ้าไหลผ่านมอเตอร์จากขั้วบวกของมอเตอร์ไปยังขั้วลบของมอเตอร์ จึงทำให้มอเตอร์สามารถหมุนได้ในทิศทาง Forward (จะหมุนแบบตามเข็มนาฬิกาหรือทวนเข็มนาฬิกานั้นขึ้นอยู่กับลักษณะของการพันขดลวดภายในมอเตอร์




**รูปที่ 2.2** ภาพแสดง H-Bridge Switching เมื่อ S1 และ S3 ON พร้อมกัน

ในทางกลับกัน หากทำการ ON สวิทช์ S2 และ S4 พร้อมกัน จะเป็นการเชื่อมวงจรและทำให้เกิดกระแสไฟฟ้าไหลผ่านมอเตอร์ จากขั้วลบของมอเตอร์ไปยังขั้วบวกของมอเตอร์ จึงทำให้มอเตอร์สามารถหมุนได้และเป็นการหมุนในทิศทาง Backward (กลับทิศทางกับกรณีแรก)




**รูปที่ 2.3** ภาพแสดงH-Bridge Switching เมื่อ S2 และ S4 On พร้อมกัน

## 2.2 วงจรไดรเวอร์มอเตอร์

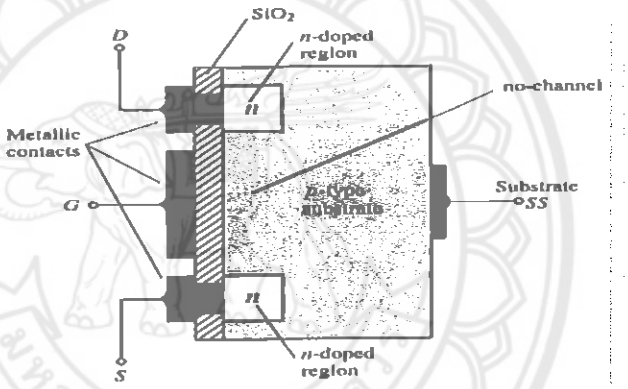
### 2.2.1 วงจรไดรเวอร์มอเตอร์แบบมอสเฟท

ทรานซิสเตอร์แบบมอสเฟท (Metal–Oxide–Semiconductor field-effect transistor Mosfet) เป็นทรานซิสเตอร์ ที่ใช้อิทธิพลสนามไฟฟ้าในการควบคุมสัญญาณไฟฟ้า โดยใช้ออกไซด์ของโลหะในการทำส่วน GATE นิยมใช้ในวงจรดิจิตอล โดยนำไปสร้างลอจิกเกตต่าง ๆ เพราะมีขนาดเล็ก เป็นเฟสที่ประกอบด้วยสารกึ่งตัวนำซึ่งได้รับ การเคลือบผิวบางส่วนด้วยโลหะออกไซด์ข้อเด่นของเฟสชนิดนี้คือ มีค่าความต้านทานอินพุต (หมายถึงค่าความต่านทานที่เกต ) สูงมาก มอสเฟทยังแบ่งเป็น 2 แบบ คือ แบบดีพลีชั่น (Depletion) และแบบเอนฮานซ์เมนต์ (Enhancement) แต่ละประเภทยังแบ่งออกเป็น 2 แบบคือ แบบแชนแนล N และ แบบแชนแนล P มอสเฟทประเภท ดี-พลีชั่นหรือดีมอสเฟท (D-MOSFET) ทั้ง 2 แบบจะทำงานได้ 2 โหมด คือ โหมดดีพลีชั่น (Depletion Mode) และ โหมดเอนฮานซ์เมนต์ (Enhancement Mode) กล่าวคือ ถ้าจ่ายแรงดันลบให้กับดี

มอสเฟทแชนแนล N จะทำงานในโหมดดีพลีชั่น แต่ถ้าจ่ายแรงดันบวกจะทำงานในโหมดเอนฮานซ์เมนต์ ส่วนดีมอสเฟทแชนแนล P ก็จะทำงานคล้ายกันเมื่อ ได้รับแรงดันที่มีขั้วตรงข้ามกับแบบแชนแนล N มอสเฟทประเภทเอนฮานซ์เมนต์หรืออีมอสเฟท (E-MOSFET) มีโครงสร้างบางอย่างคล้ายกับมอสเฟทแบบดีพลีชั่นแต่จะทำงานได้เฉพาะโหมดเอนฮานซ์เมนต์เท่านั้น

**ประเภทของMOSFET**

MOS (Negative MOSFET) เป็นทรานซิสเตอร์ประเภท NPN เมื่อมีความต่างศักย์เป็นบวก (สนามไฟฟ้าแรง) สัญญาณไฟฟ้าจึงจะไหลจาก sorce ไป drain ได้ pMOS (Positive MOSFET) เป็นทรานซิสเตอร์ประเภท PNP เมื่อมีความต่างศักย์ต่ำหรือเป็นลบ (สนามไฟฟ้าอ่อน) สัญญาณไฟฟ้าจึงจะไหลจาก sorce ไป drain ได้




รูปที่ 2.4 โครงสร้างของ MOSFET

nMOS เมื่อปล่อยความต่างศักย์สูง จะเกิดสนามไฟฟ้าในทิศลงอย่างแรง โฮลใน p-type จะถูกผลักลงมาอยู่ด้านล่าง (ตามรูปที่ประกอบข้างบน) ประกอบกับมีอิเล็กตรอนอิสระบางส่วนถูกดูดขึ้นไปด้านบน ส่งผลให้บริเวณด้านบนมีอิเล็กตรอนอิสระมากจนเป็น n-type ได้เรียกว่า channel สัญญาณไฟฟ้าก็จะไหลผ่านช่วง Channel นี้ซึ่งเป็น n-type เหมือนกับ drain และ sorce ได้โดยใช้อิเล็กตรอนอิสระเป็นพาหะ

**pMOS** จะทำงานกลับกับ nMOS โดยเมื่อปล่อยความต่างศักย์ต่ำ(โดยมากมักจะติดลบ) จะเกิดสนามไฟฟ้าในทิศขึ้นอย่างแรง อิเล็กตรอนอิสระใน n-type จะถูกผลักลงมาอยู่ด้านล่าง ประกอบกับมีโฮลบางส่วนถูกดูดขึ้นไปด้านบน ส่งผลให้บริเวณด้านบนมีโฮลมากจนเป็น p-type ได้เรียกว่า channel สัญญาณไฟฟ้าก็จะไหลผ่านช่วง channel นี้ซึ่งเป็น p-type เหมือนกับ drain และ sorce ได้โดยใช้โฮลเป็นพาหะ

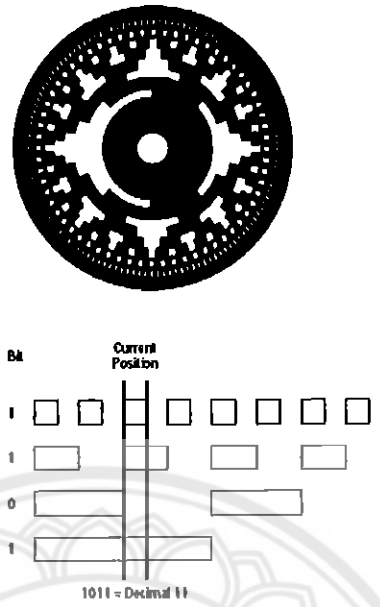
**ตารางที่ 2.1** ลักษณะการหมุนของมอเตอร์เมื่อเปลี่ยนอินพุท

| Inputs | | | Function |
|---|---|---|---|
| En =H | IN1 = L | IN1 = L | Fast Motor Stop |
| | IN1 = L | IN1 = H | Reverse |
| | IN1 = H | IN1 = L | Forward |
| | IN1 = H | IN1 = H | Fast Motor Stop |
| En = L | IN1 = X | IN2 = X | Free Running Motor Stop |

## 2.3 หลักการทำงานของ Encoder

**Optical Encoder** ในระบบควบคุมตำแหน่งที่ซึ่งต้องการการป้อนตำแหน่งที่แม่นยำได้มีการนำอุปกรณ์ป้อนกลับที่ใช้หลักการต่างๆ[4]และ[5] โดยที่อุปกรณ์ที่นิยมใช้ โดยทั่วไปคืออุปกรณ์ที่ใช้หลักการแสง(optical) โดยจะมีแผ่น grating จะเคลื่อนที่ระหว่างตัวกำเนิดแสงและตัวรับแสง เมื่อแสงกระทบกับส่วนของ grating ที่โปร่งแสง ตัวรับแสงก็จะได้รับแสงนั้น เพื่อให้ได้ความละเอียดมากขึ้นก็จะเพิ่ม mask ระหว่าง grating และตัวรับแสงซึ่งจะทำงานคล้ายกับชัตเตอร์ของกล้องถ่ายรูป การออกแบบแหล่งแสงและการ mask จะขึ้นรูปแบบของ Encoder โดยทั่วไปแล้ว Optical encoder มี 2 ประเภทดังนี้

**2.3.1 Absolute encoder** เป็นอุปกรณ์ป้อนกลับตำแหน่งที่ให้สัญญาณออกมาที่แต่ละตำแหน่งของแกน เป็นค่าเฉพาะเจาะจงของตำแหน่งแกนนั้นๆ โดยค่าสัญญาณจะแบ่งเป็นลอจิก 1 และ 0 เมื่อ แสงสามารถผ่าน grating ได้และไม่ได้ค่าลอจิกที่ได้จะแทนลำดับการนับของแต่ละหลักในเลขฐานสอง เพื่อให้เข้าใจยิ่งขึ้นโดยพิจารณาภาพ

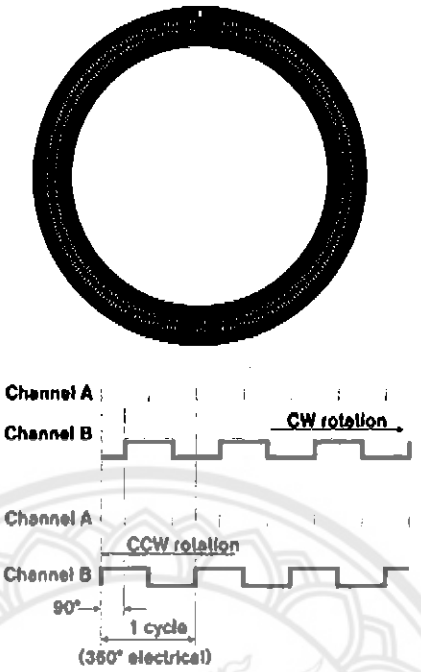



รูปที่ 2.5 จาน disk ของ Absolute Encoder[5]

ความละเอียดของเอนโคเดอร์ชนิดนี้ขึ้นกับจำนวน track เช่น หากเอนโคเดอร์มี 10 track นั่นคือจะมีความละเอียด 2 ยกกำลัง 10 หรือ 1024 ตำแหน่งต่อหนึ่งรอบการหมุน ค่าตำแหน่งที่ได้สามารถ อ่านเป็นค่าตำแหน่งได้เลย รูปแบบของโค๊ดส่วนมากจะเป็น binary code บางครั้งอาจเป็น greycode บางท่านอาจมองเห็นจุดอ่อนของเอนโคเดอร์ชนิดนี้ นั่นคือค่าตำแหน่งที่จำกัดหรือค่ามีค่าสำหรับหนึ่งรอบการหมุนเท่านั้น แต่เราสามารถแก้ปัญหานี้โดยการเพิ่มจำนวนดิส (grating) ซึ่ง ทำให้ได้ระยะตำแหน่งที่เพิ่มขึ้น

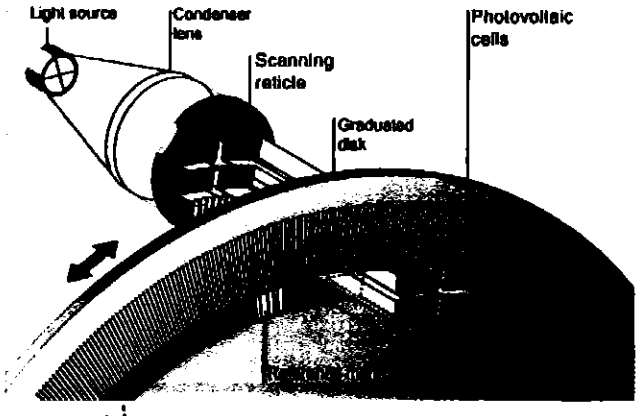
ข้อดีของเอนโคเดอร์ชนิดนี้ได้แก่ ไม่ต้องมีการหาตำแหน่งโฮม (home) เนื่องจากเปิดเครื่องแล้ว ตำแหน่งจะยังคงเป็นตำแหน่งเดิม สัญญาณรบกวนจะไม่มีผลเนื่องจากตำแหน่งที่เปลี่ยนไปจะขึ้น กับการหมุนเท่านั้น แต่เอนโคเดอร์ประเภทนี้มักมีราคาแพง ในโครงการนี้ได้นำเอาหลักการของ Increment Encoder เข้ามาใช้และอ่านค่าออกมาเพื่อตรวจสอบ ตำแหน่งของการเคลื่อนที่และควบคุมความเร็วของมอเตอร์เพื่อนำมาทำการเปรียบเทียบตำแหน่งการเคลื่อนที่กับค่า Input โดยหลักการทำงานของ Encoder มีดังนี้

**2.3.2 Increment Encoder** มีลักษณะเป็นแผ่นกลมมีแกนอยู่ตรงกลาง และ ที่แผ่นกลม จะมีช่องเล็ก ที่แสงสามารถส่องผ่านได้เป็นจำนวนมากซึ่งเรียกช่องนี้ว่าช่อง slit ซึ่งที่ด้านหนึ่งของแผ่นกลมนี้จะมีหลอด LED ซึ่งเป็นตัวส่งแสง infrared ไปยังตัวรับสัญญาณแสง infrared ซึ่งจะอยู่ในด้านตรงกันข้าม

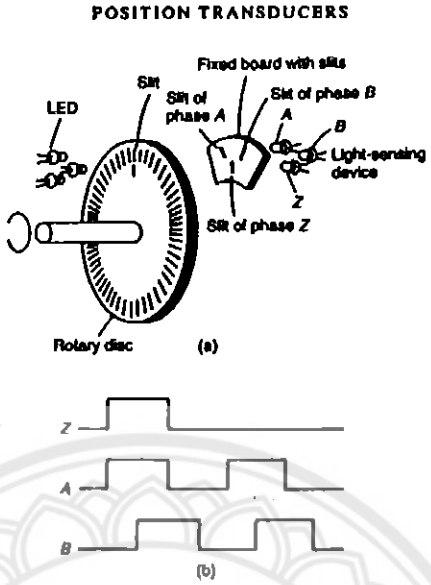



รูปที่ 2.6 จาน disk ของ incremental encoder[5]

โครงสร้างจะประกอบด้วยตัวกำเนิดแสง, ตัวจับแสงซึ่งถูกคั่นกลางด้วยแผ่นจานกลมๆที่มีการทำรูเจาะไว้รอบๆแผ่น (จำนวนรูจะขึ้นอยู่กับความละเอียดของ incremental encoder ) และหน้ากากแยกช่องของสัญญาณพัลส์ A ,B และ Z สัญญาณพัลส์ที่ได้จากเอนโค้ดเดอร์ชนิดนี้จะประกอบด้วย 3 แทรค (tracks) คือ A,BและZ พัลส์ที่เกิดจาก แทรค A และ B จะเกิดการเหลื่อมกันมีความต่างเฟสกัน 90 องศา เพื่อทำหน้าที่รายงานผลของความเร็วและทิศทางการหมุนของมอเตอร์ให้คอนโทรลเลอร์ดังนี้กรณีพัลส์ A เกิดขึ้นก่อน B คอนโทรลเลอร์จะรับรู้ว่ามอเตอร์กำลังหมุนด้วยทิศทางตามเข็มนาฬิกา ส่วนแทรค Z หรือพัลส์อ้างอิง จะเกิดขึ้น1พัลส์ในการหมุน 1 รอบ ทำหน้าใช้อ้างอิงตำแหน่งของโรเตอร์




**รูปที่2.7** รูปแสดงการทำงานของ Encoder[4]

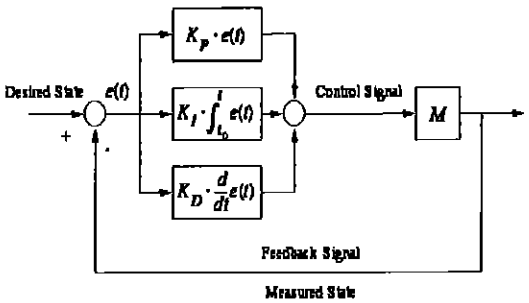



รูปที่2.8 ส่วนประกอบของ Encoder[4]

Incremental Encoder โดยทั่วไปจะไม่นิยมใช้กับระบบเซอร์โวมอเตอร์ที่มีการควบคุมตำแหน่ง เนื่องจากไม่สามารถจำตำแหน่งเดิมได้กรณีที่มีการปิดเครื่องหรือไฟดับ ซึ่งจะต้องทำการหาจุดอ้างอิงใหม่ทุกครั้ง

## 2.4 ระบบการควบคุมป้อนกลับ

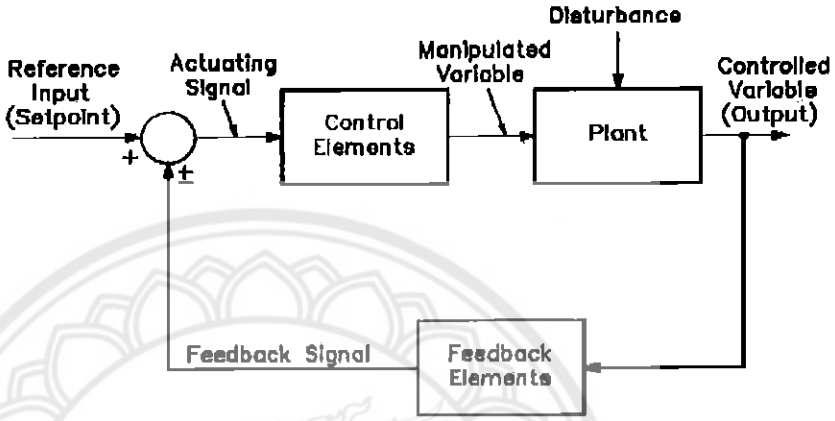
ระบบการควบคุมแบบป้อนกลับ เป็นระบบการควบคุมที่นิยมใช้ในงานอุตสาหกรรมต่างๆ เป็นอย่างมาก เนื่องจากคุณสมบัติที่ว่า ระบบการควบคุมแบบนี้สามารถปรับค่าการควบคุมได้ เมื่ออุปกรณ์ที่ต้องการควบคุมเปลี่ยนไปซึ่งประสิทธิภาพ ในการควบคุมก็ให้ผลดีเป็นที่น่าพอใจ อีกทั้งการควบคุมก็ทำได้ง่ายและไม่ซับซ้อนจนเกินไป โดยปกติแล้วเราจะแบ่งการควบคุมออกเป็น 3 กลุ่ม ดังรูปนี้




**รูปที่ 2.9** ระบบควบคุมแบบ PID

จากรูปจะเห็นว่าเราสามารถแบ่งการควบคุมแบบป้อนกลับออกเป็นสามส่วนดังสมการ

$$m(t) = K_p e(t) + K_I \int e(t)dt + \mathrm{K}_D \frac{de}{dt} \qquad (2.1)$$




รูปที่ 2.10 หลักการทำงานแบบป้อนกลับ

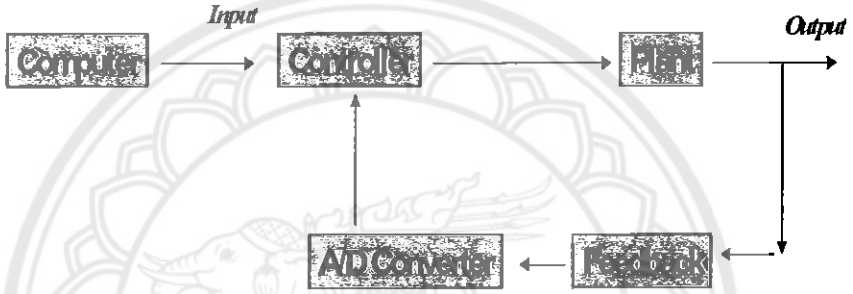
ซึ่งเราสามารถแบ่งการควบคุมออกเป็นส่วนๆดังนี้

**ระบบควบคุมแบบสัดส่วน (Proportional Control)** เป็นการนำสัญญาณความคลาดเคลื่อนระหว่างสัญญาณเข้าและสัญญาณขาออก แล้วนำไปคูณกับค่าคงที่ของการควบคุมแบบสัดส่วน (Proportional Gain) แล้วส่งสัญญาณที่ได้ไปขับอุปกรณ์ซึ่งจะสังเกตได้ว่าหากสัญญาณทั้งสองมีความแตกต่างกันมากๆแล้วระบบจะทำให้เกิดสัญญาณควบคุมมากๆ นั่นจะทำให้อุปกรณ์อยู่ในสภาวะที่เราต้องการได้อย่างรวดเร็ว อย่างไรก็ตามการควบคุมแบบนี้จะทำให้เกิดสภาวะของการสั่นรอบๆจุดของสัญญาณที่เราต้องการ ซึ่งเรียกว่าการเกิด Overshoot โดยหากเราเพิ่มค่าค่าคงที่ควบคุมแบบสัดส่วนมากๆนั้นระบบจะเร็วขึ้นก็จริงแต่อาจเกิดแกว่งของสัญญาณรอบๆจุดที่ต้องการได้

**ระบบควบคุมแบบสะสม (Integral Control)** ในระบบการควบคุมนั้น บางครั้งระบบการควบคุมไม่อาจควบคุมอุปกรณ์ให้ไม่มีความคลาดเคลื่อนของสัญญาณขาเข้า และสัญญาณขาออกได้ จึงมีได้มีการเพิ่มระบบการควบคุมแบบสะสมค่าความคลาดเคลื่อน แล้วนำไปคูณกับค่าคงที่ของการควบคุมแบบสะสม (Integral Control) เพื่อให้เกิดค่าความผิดพลาดน้อยที่สุด

**ระบบควบคุมแบบความแตกต่าง (Differential Control)** ระบบการควบคุมแบบนี้จะหาว่าค่าความเคลื่อนในอดีตกับปัจจุบันมีความแตกต่างกันมากเพียงใด ซึ่งหากมีความแตกต่างมากเมื่อเรานำสัญญาณความแตกต่างไปคูณกับค่าคงที่ของการควบคุมแบบสะสม แล้วก็จะได้สัญญาณที่จะทำให้ระบบนั้นเร็วขึ้นได้

**ระบบควบคุมแบบดิจิตอล** เป็นการควบคุมอุปกรณ์โดยใช้เครื่องมือแบบดิจิตอล เช่นเครื่องคอมพิวเตอร์เป็นตัวควบคุมระบบซึ่งแผนผังการควบคุมแบบดิจิตอลแสดงดังรูป



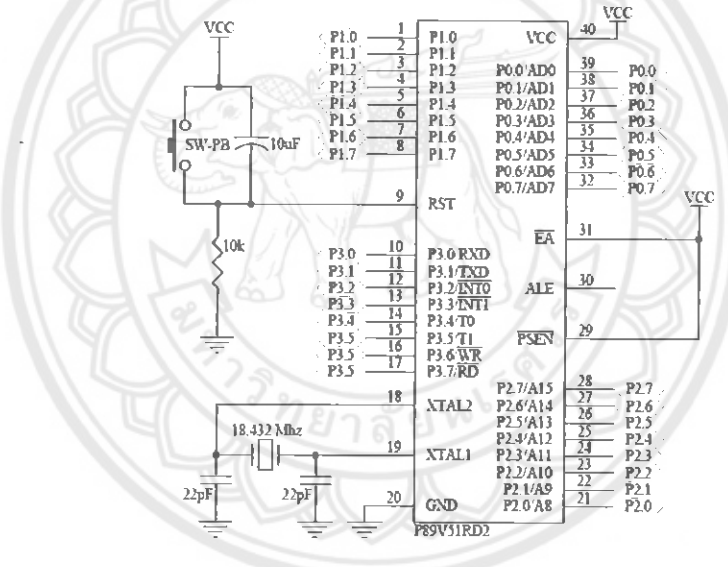

รูปที่ 2.11 ระบบควบคุมแบบดิจิตอล

ในการที่จะสร้างสมการการควบคุมแบบดิจิตอลจากระบบการควบคุมแบบอนาล็อก ทำได้โดยการแปลงสมการแบบอนาล็อกเป็นสมการแบบดิจิตอลซึ่ง ในที่นี้ได้พิจารณาขีดความสามารถของไมโครคอนโทรลเลอร์ซึ่งมีข้อจำกัดต่างๆค่อนข้างมาก ซึ่งไม่สามารถใช้การคำนวณที่ซับซ้อนได้ การควบคุมแบบ PID ในแบบของสัญญาณอนาล็อก

## 2.5 ไมโครคอนโทรลเลอร์ MCS-51 (เบอร์ P89V51RD2)

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เริ่มแรกได้ถูกพัฒนาขึ้นจากบริษัท อินเทล (Intel Corporation) และได้มีการนำไปใช้งานกันอย่างแพร่หลายตั้งแต่ปี 1980 ในช่วงเวลาที่ผ่านมาได้มีบริษัทผู้ผลิตหลายบริษัท เช่น Dallas, Philips, Atmel ได้รับลิขสิทธิ์ในการผลิต และจำหน่าย จากบริษัท อินเทล และบริษัทต่าง ๆ ก็ได้พัฒนาความสามารถของไมโครคอนโทรลเลอร์ MCS-51 รุ่นใหม่ ๆ ให้มีความสามารถ และมีความเร็วเพิ่มขึ้น แต่ยังคงโครงสร้างพื้นฐานที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล 8051 ซึ่งมีรายละเอียดดังนี้

1. เป็นไมโครคอนโทรลเลอร์ที่มีหน่วยประมวลผลกลางแบบ 8 บิต
2. มีคำสั่งคำนวณทางคณิตศาสตร์ และตรรกศาสตร์ (Boolean processor)
3. มีแอดเดรสบัสขนาด 16 บิตทำให้สามารถอ้างตำแหน่งหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้ 64 กิโลไบต์
4. มีหน่วยความจำ (RAM) ภายในขนาด 256 ไบต์ (8052/8032)
5. มีพอร์ตอนุกรมทำงานแบบฟลูดูเพล็กซ์ (Full Duplex) 1 พอร์ต
6. มีพอร์ตอินพุต/เอาต์พุตแบบขนานจำนวน 32 บิต
7. มีไทม์เมอร์ 2 ตัว (8051/8031) หรือ 3 ตัว (8052/8032)
8. มีวงจรควบคุมการเกิดอินเตอร์รัพท์ 5 ประเภท (8051/8031) หรือ 6 ประเภท (8052/8032) มีวงจรออสซิลเลเตอร์ภายในตัว




**รูปที่ 2.12** วงจรไมโครคอนโทรลเลอร์ P89V51RD2

ไมโครคอนโทรลเลอร์ MCS-51 มีวงจรออสซิลเลเตอร์อยู่ภายใน ดังนั้นในการใช้งานจึงสามารถต่อคริสตอล และตัวเก็บประจุเข้ากับคริสตอลได้โดยตรง โดยความถี่ของคริสตอลที่ต่อเข้ากับไมโครคอนโทรลเลอร์จะเป็นตัวระบุความเร็วในการทำงานโดยตรง ในไมโครคอนโทรลเลอร์ MCS-51 ปกติ 1 แมชชีนไซเคิล (Machine Cycle) จะใช้สัญญาณนาฬิกาจำนวน 12 ลูก และในการทำงานแต่ละคำสั่งไมโครคอนโทรลเลอร์จะใช้เวลาในการทำงาน 1 - 4 แมชชีนไซเคิล ขึ้นอยู่กับความซับซ้อนของคำสั่งนั้น

ในปัจจุบันผู้ผลิตได้พัฒนาให้ไมโครคอนโทรลเลอร์สามารถทำงานได้เร็วขึ้น โดยเพิ่มความสามารถในการรองรับคริสตอลความถี่ที่สูงขึ้น รวมไปถึงการปรับปรุงการทำงานภายใน ให้ไมโครคอนโทรลเลอร์ใช้จำนวนสัญญาณนาฬิกาในการสร้างแมชชีนไซเคิลน้อยลง โดยในบางรุ่น 1 แมชชีนไซเคิลใช้สัญญาณนาฬิกาเพียงแค่ 1 ลูกเท่านั้น

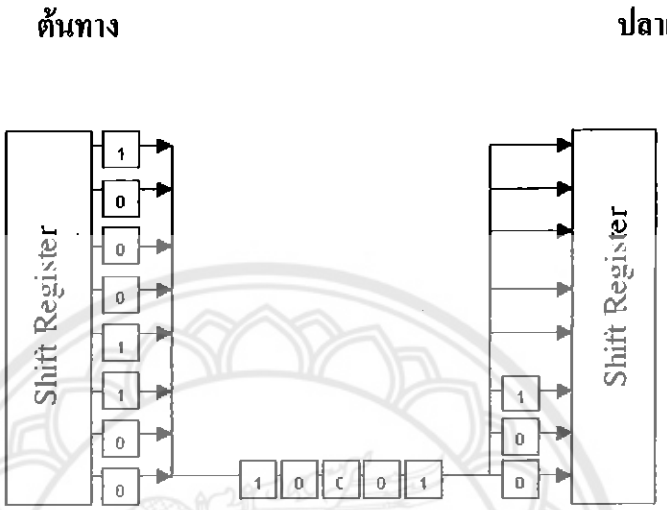
สำหรับไมโครคอนโทรลเลอร์ MCS-51 ที่เราจะมาลองเล่นกันนั้นเป็นรุ่น P89V51RD2 ของบริษัท Philips ที่เลือกรุ่นนี้เนื่องจากเป็นรุ่นที่สามารถรองรับการดาวน์โหลดโปรแกรมแบบ ISP (In System Programming) ผ่านพอร์ตอนุกรมได้โดยตรง ไม่ต้องอาศัยอุปกรณ์ หรือวงจรเพิ่มเติมในการดาวน์โหลดโปรแกรม จึงทำให้สามารถใช้งานได้อย่างสะดวก รวมถึงราคาของ P89V51RD2 ที่ไม่แพง เมื่อเทียบกับความสามารถ และประสิทธิภาพของมัน P89V51RD2 สามารถทำงานในโหมด X2 ซึ่ง จะทำให้สามารถทำงานได้เร็วกว่า MCS-51 พื้นฐาน 2 เท่า (1 แมชชีนไซเคิล ใช้สัญญาณนาฬิกา 6 ลูก) เมื่อใช้คริสตอลความถี่ที่เท่ากัน ในการทำงานในโหมด X2 นี้ P89V51RD2 สามารถใช้คริสตอลความถี่สูงสุด 20MHz ส่วนในการทำงานในโหมด X1 สามารถใช้คริสตอลความถี่สูงสุด 40 MHz ภายใน P89V51RD2 มีหน่วยความจำโปรแกรมแบบแฟลชขนาด 64 กิโลไบต์ นอกจากนั้นยังมี หน่วยความจำข้อมูลภายนอกเพิ่มเติมขนาด 1 กิโลไบต์ อยู่ภายในตัวชิพด้วย

ส่วนสำคัญของ port ในไมโครคอนโทรลเลอร์ที่ใช้ในการศึกษาและทำโครงงานนี้คือ พอร์ท Timer ต่างๆ T0,T1,T2 โดยจะใช้งาน port Timer เป็นตัวนับหรือเคาเตอร์จะใช้นับสัญญาณพัลส์ที่เข้ามาจากภายนอกที่เข้ามายังไมโครคอนโทรลเลอร์ ทางขา T0 หรือ T1 โดยค่าในรีจิสเตอร์จะเพิ่มขึ้นถ้าสัญญาณที่เข้ามามีการเปลี่ยนแปลงจากลอจิก " 1 " เป็นลอจิก " 0 "

## 2.6 การสื่อสารแบบอนุกรม

การส่งข้อมูลแบบขนานนั้น สายส่งข้อมูลแบบขนานที่ใช้สำหรับเชื่อมต่อระหว่างพอร์ต I/O กับอุปกรณ์ภายนอกนอกจากจะมีความยาวได้เพียง 1 หรือ 2 เมตรเท่านั้น ทั้งนี้เนื่องจากค่าคาพา-ซิแดนซ์ในสายจะจำกัดระยะทางในการส่งข้อมูล แต่ถ้าเราต้องการให้สามารถส่งข้อมูลได้ในระยะทางไกลขึ้น เราก็ต้องนำวงจรขับพิเศษมาใช้ การส่งข้อมูลแบบขนานจะต้องส่งสัญญาณจำนวน 1 เส้นสำหรับข้อมูลในแต่ละบิต ซึ่งทำให้การโอนย้ายข้อมูลแบบขนาน 1 ไบต์มีราคาสูงกว่าการโอนย้ายข้อมูลแบบอนุกรมถึง 8 เท่า เช่นเดียวกันด้วยเหตุผลด้านราคาและความไม่สะดวกที่พบในการโอนย้ายข้อมูลแบบขนาน จึงได้มีอุปกรณ์หลายชนิดที่ใช้ในการสื่อสารแบบอนุกรม ทั้งๆที่ไมโครโปรเซสเซอร์ในเครื่องไมโครคอมพิวเตอร์จะยังคงใช้การโอนย้ายข้อมูลแบบขนาน เมื่อเราศึกษาการโอนย้ายข้อมูลแบบอนุกรม เราก็จะต้องเรียนรู้โพรโตคอลที่เกี่ยวกับการส่งข้อมูลแบบอนุกรม ซึ่งแบ่งออกได้เป็น 3 อย่าง อย่างแรกคือ วิธีการแปลงข้อมูลแบบขนานเป็นแบบอนุกรมและการแปลงข้อมูลแบบอนุกรมเป็นข้อมูลแบบขนาน อย่างที่สองคือ ชนิดของวงจรและรูปแบบ

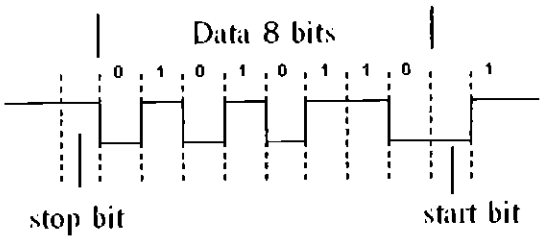
สัญญาณที่ใช้ในการส่งข้อมูลในระยะไกล อย่างที่สามคือรูปแบบของข้อมูลที่ส่งไปและการควบคุมการโอนย้ายข้อมูล



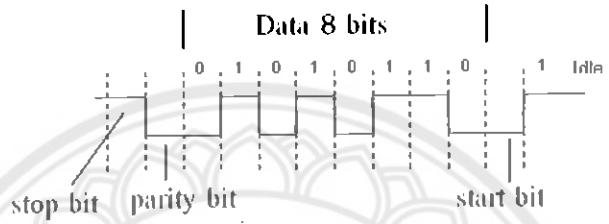

รูปที่ 2.13 รูปแบบการส่งข้อมูลแบบอนุกรม

**2.6.1 การเชื่อมต่อแบบอนุกรม UART**

การแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม โดยเริ่มแรกข้อมูลแบบขนานจะถูกนำไปเก็บไว้ในรีจีสเตอร์ที่เลื่อนค่าได้ (Shift register) จากนั้นเราจะใช้สัญญาณนาฬิกาในการเลื่อนค่าในรีจีสเตอร์ออกมาทีละบิต (โดยจะเลื่อนค่าไปทางขวามือ) โดยบิตแรกที่ถูกเลื่อนออกมาคือ บิต LSB ของข้อมูลและบิตที่สองที่ถูกเลื่อนออกมาก็คือ บิตที่อยู่ถัดไปจากบิต LSB และบิตต่อๆไป สำหรับบิตสุดท้ายที่ถูกเลื่อนออกมาก็คือ บิต MSB ของข้อมูล เมื่อเรานำบิตที่ 8 ของข้อมูลมาใช้ในการตรวจสอบความผิดพลาดในการสื่อสารข้อมูลซึ่งเราเรียกบิตนี้ว่า บิตพาริตี้ (Parity bit) UART ส่วนใหญ่สามารถสร้างและทำการตรวจสอบข้อมูลนั้นว่าเป็นพาริตี้คู่หรือเป็นพาริตี้คี่ได้ ในการสร้างพาริตี้คู่ UART จะทำการเซตหรือเคลียร์ค่าในบิตพาริตี้เพื่อให้ข้อมูลทั้ง 8 บิตนั้นมีเลข 1 จำนวนคู่ตัว และ ในการสร้างพาริตี้คี่ UART จะทำการเซตหรือเคลียร์ค่าในพาริตี้เพื่อให้ข้อมูลทั้ง 8 บิตนั้น มีเลข 1 จำนวนคี่ตัว การส่งข้อมูลของ UART จะเป็นแบบอะซิงโครนัส ซึ่งก็หมายความว่าเวลาระหว่างคำอัตราการส่งข้อมูลของ UART จะไม่ขึ้นกับจังหวะการทำงานของไมโคร โดยไมโครคอนโทรลเลอร์และ UART จะมีวงจรสร้างสัญญาณนาฬิกาของมันเอง แต่ถ้าเราพบว่าไมโครคอนโทรลเลอร์และ UART ทำงานร่วมกันอย่างเข้าจังหวะแต่การทำเช่นนี้ก็เพื่อเป็นการลดส่วนของวงจรฮาร์ดแวร์ที่ใช้สร้างสัญญาณนาฬิกา



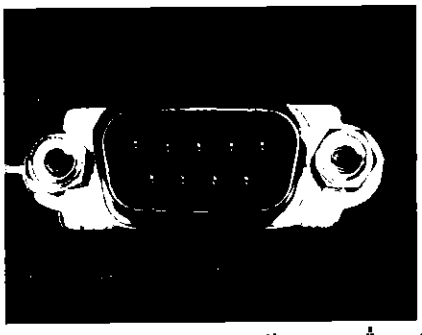

**รูปที่ 2.14** การสื่อสารแบบอะซิงโคนัสที่ไม่มีพาริตี้บิต



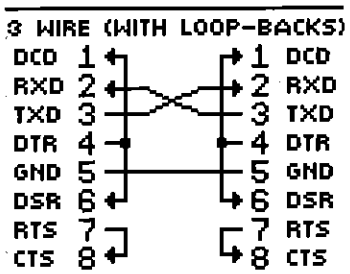

**รูปที่ 2.15** การสื่อสารแบบอะซิงโคนัสที่มีพาริตี้บิต

การเชื่อมต่อระหว่างพอร์ตอนุกรม โดยทั่วไปเราจะพบเส้นส่งสัญญาณอนุกรมแบบมาตรฐาน EIA RS-232 มากที่สุดซึ่งเราจะเรียกว่า RS-232 สายส่งสัญญาณ RS-232 นี้ได้ถูกนำไปใช้ในหน่วยแสดงผล เครื่องพิมพ์โมเด็ม และอุปกรณ์อื่น ๆ ซึ่งจะมีความยาวของสายไม่เกิน 50 ฟุต

มาตรฐาน RS-232 ได้กำหนดให้ค่าสัญญาณไฟฟ้าที่มีระดับแรงดันไฟฟ้าเท่ากับ 3 โวลต์หรือสูงกว่า ที่มีค่าทางตรรกะเป็น 1 และกำหนดค่าสัญญาณไฟฟ้าที่มีระดับแรงดันเท่ากับ -3 โวลต์หรือต่ำกว่า มีค่าทางตรรกะเป็น 0 วงจรไอซีที่สร้างสัญญาณเหล่านี้ต้องการแหล่งจ่ายไฟขนาด +12 โวลต์ RS-232 จะใช้สาย 1 เส้นสำหรับส่งข้อมูลและใช้สายอีก 1 เส้นสำหรับข้อมูล โดยสัญญาณในแต่ละสายนี้จะถูกอ้างอิงกับกราวนด์ (ขาเบอร์ 7) มาตรฐาน RS-232 นี้ยังได้กำหนดสัญญาณตอบรับเพื่อใช้ในการควบคุมการรับ/ส่งข้อมูลด้วย



**รูปที่ 2.16** Serial Port (Com Port) ใช้ในการเชื่อมต่อการส่งสัญญาณ




**รูปที่ 2.17** การต่อสายสัญญาณตามมาตรฐาน RS-232

## 2.7 ระบบการสื่อสารผ่านเครือข่ายไร้สาย Wireless LAN

การใช้งานระบบเครือข่ายแบบไร้สายถือว่าเป็นสิ่งที่มีความสะดวก สบายในยุคนี้ และในปัจจุบันนี้มีการเจริญเติบโตการติดต่อสื่อสารแบบไร้สายกันอย่างแพร่หลายและขยายวงกว้างไปมาก โดยที่นับตั้งแต่เกิดมาตรฐาน IEEE 802.11 เกิดขึ้นมาก็มีการพัฒนาและปรับปรุงเครือข่ายการติดต่อสื่อสารแบบไร้สายมากยิ่งขึ้นจนให้ผู้ใช้ได้รับความสะดวกสบายเพิ่มมากขึ้น จึงสามารถตอบสนองผู้ใช้ได้อย่างสมบูรณ์ทั้งในด้านความสะดวกสบาย ความปลอดภัย เป็นต้น เราจึงคำนึงถึงข้อดีจากที่ได้ลองใช้งานแล้วคิดว่าน่าจะยังมาประยุกต์ใช้กับการทำงานในรูปแบบอื่นๆได้

### 2.7.1 มาตรฐานเครือข่ายไร้สาย IEEE 802.11

เป็นเครือข่ายแบบไร้สายภายใต้มาตรฐาน IEEE 802.11 ที่ได้รับการตีพิมพ์โดยสถาบัน IEE (The Institute of Electronics and Electrical Engineers) ซึ่งมีข้อกำหนดระบุไว้ว่า ผลิตภัณฑ์เครือข่ายไร้สายในส่วนของ PHY Layer นั้นมีความสามารถในการรับส่งข้อมูลที่ความเร็ว 1, 2, 5.5, 11 และ 54 เมกะบิตต่อวินาที โดยมีสื่อนำสัญญาณ 3 ประเภทให้เลือกใช้งานอันได้แก่ คลื่นวิทยุย่านความถี่ 2.4 กิกะเฮิรตซ์, 2.5 กิกะเฮิรตซ์และคลื่นอินฟราเรด ส่วน.ในระดับชั้น MAC Layer นั้นได้กำหนดกลไกของการทำงานแบบ CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) ซึ่งมีความคล้ายคลึงกับ CSMA/CD (Collision Detection) ของมาตรฐาน IEEE 802.3 Ethernet ซึ่งนิยมใช้งานบนระบบเครือข่ายแลนใช้สาย โดยมีกลไกในการเข้ารหัสข้อมูลก่อนแพร่กระจายสัญญาณไปบนอากาศ พร้อมกับมีการตรวจสอบผู้ใช้งานอีกด้วย

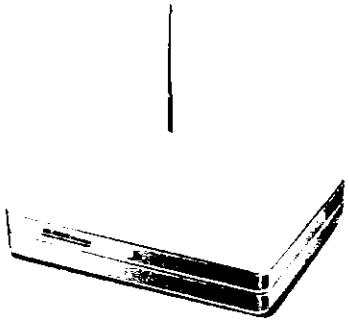
มาตรฐาน IEEE 802.11 ในยุคเริ่มแรกนั้นให้ประสิทธิภาพการทำงานที่ค่อนข้างต่ำ ทั้งไม่มีการรับรองคุณภาพของการให้บริการที่เรียกว่า QoS (Quality of Service) ซึ่งมีความสำคัญในสภาพแวดล้อมที่มีแอพพลิเคชันหลากหลายประเภทให้ใช้งาน นอกจากนั้นกลไกในเรื่องการรักษาความปลอดภัยที่นำมาใช้ก็ยังมีช่องโหว่จำนวนมาก IEEE จึงได้จัดตั้งคณะทำงานขึ้นมาหลายชุดด้วยกัน เพื่อทำการพัฒนาและปรับปรุงมาตรฐานให้มีศักยภาพเพิ่มสูงขึ้น

**ตารางที่ 2.2** Wireless Networking Standards

| | UWB | Bluetooth | Wi-Fi | Wi-Fi | Wi-Fi | WiMAX | WiMAX | Edge | CDMA2000 / 1 x EV-DO | WCDMA / UMTS |
|---|---|---|---|---|---|---|---|---|---|---|
| Standard | 802.15.3a | 802.15.1 | 802.11a | 802.11b | 802.11g | 802.16d | 802.16e | 2.5G | 3G | 3G |
| Usage | WPAN | WPAN | WLAN | WLAN | WLAN | WMAN Fixed | WMAN Portable | WWAN | WWAN | WWAN |
| Throughput | 110-480Mbps | Up to 720Kbps | Up to 54Mbps | Up to 11Mbps | Up to 54Mbps | Up to 75Mbps (20MHz BW) | Up to 30Mbps (10MHz BW) | Up to 384Kbps | Up to 2.4 Mbps (typical 300-600Kbps) | Up to 2Mbps (Up to 10Mbps with HSDPA technology) |
| Range | Up to 30 feet | Up to 30 feet | Up to 300 feet | Up to 300 feet | Up to 300 feet | Typical 4-6 miles | Typical 1-3 miles | Typical 1-5 miles | Typical 1-5 miles | Typical 1-5 miles |
| Frequency | 7.5GHz | 2.4GHz | 5GHz | 2.4GHz | 2.4GHz | Sub 11GHz | 2-6GHz | 1900MHz | 400, 800, 900, 1700, 1800, 1900, 2100MHz | 1800, 1900, 2100MHz |

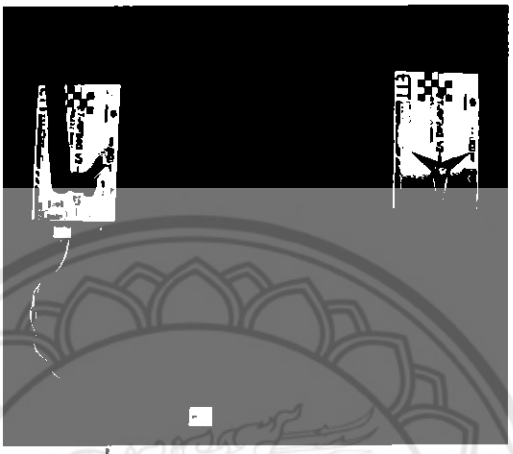
### 2.7.2 มาตรฐานที่ใช้ในการติดต่อสื่อสาร IEEE 802.11g

คณะทำงานชุด IEEE 802.11g ได้ใช้นำเทคโนโลยี OFDM มาประยุกต์ใช้ในช่องสัญญาณวิทยุความถี่ 2.4 GHz ซึ่งอุปกรณ์ IEEE 802.11g WLAN มีความสามารถในการรับส่งข้อมูลด้วยความเร็วสูงสุดที่ 54 Mbps ส่วนรัศมีสัญญาณของอุปกรณ์ IEEE 802.11g WLAN จะอยู่ระหว่างรัศมีสัญญาณของอุปกรณ์ IEEE 802.11a และ IEEE 802.11b เนื่องจากความถี่ 2.4 GHz เป็นย่านความถี่สาธารณะสากล อีกทั้งอุปกรณ์ IEEE 802.11g WLAN สามารถทำงานร่วมกับอุปกรณ์ IEEE 802.11b WLAN ได้ (backward-compatible) ดังนั้นจึงมีแนวโน้มสูงว่าอุปกรณ์ IEEE 802.11g WLAN จะได้รับความนิยมอย่างแพร่หลายหากมีราคาไม่แพงจนเกินไปและน่าจะมาแทนที่ IEEE 802.11b ในที่สุด ตามแผนการแล้วมาตรฐาน IEEE 802.11g จะได้รับการตีพิมพ์ประมาณช่วงกลางปี พ.ศ. 2546



**รูปที่ 2.18** ตัวอุปกรณ์ Wireless Router

โดยโครงงานนี้จะใช้เป็นตัวกลางในการติดต่อสื่อสารระหว่างคนบังคับหุ่นยนต์โดยผ่านทาง RS232 Wireless โดยที่เราจะทำการทั้งส่งอักขระเพื่อทำการควบคุมหุ่นยนต์ให้เป็นไปตามที่เราต้องการและ RS232 Wireless ยังเป็นตัวส่งข้อมูลค่าที่ได้จากตัวนับพัลส์ของสัญญาณของ Encoder มายังเครื่องผู้บังคับอีกด้วย เราจึงเลือกมา RS232 Wireless มาประยุกต์ใช้ในโครงงานนี้



รูปที่ 2.19 อุปกรณ์ RS232 Wireless

## 2.8 ระบบGUI

หมายถึงส่วนที่มีการเชื่อมต่อประสานกราฟิกกับผู้ใช้งาน(Graphical User Interface,) กล่าวคือเป็นวิธีการใช้งานคอมพิวเตอร์ผ่านทางสัญลักษณ์หรือสภาพนอกเหนือจากตัวอักษร ส่วนใหญ่จะมีองค์ประกอบเป็นแบบรูปภาพซะเป็นส่วนใหญ่ ลองลงมาอาจจะเป็นสัญลักษณ์หรือตัวอักษรก็ได้[3]

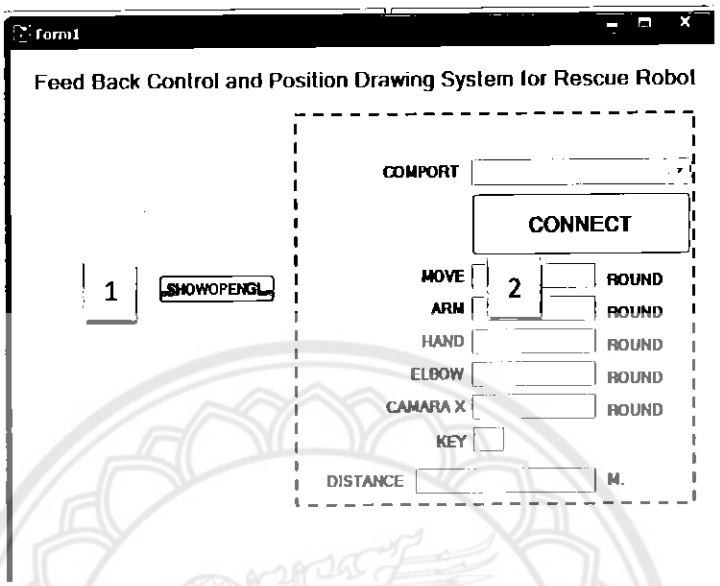
GUI ได้มีการพัฒนาขึ้นมาโดยใช้งานระบบกราฟิกแทนที่ระบบอักษร โดยบางคนจะเรียกระบบนี้ว่า PARC User Interface หรือ PUI โดยมีบริษัทแอปเปิลคอมพิวเตอร์ได้นำมาใช้กับเครื่องแมคอินทอชและวินโดวส์และล่าสุดก็ยังมีในระบบปฏิบัติการยูนิกซ์อีกด้วย รูปร่างหน้าตาของ GUI ตามด้านล่างนี้

### 2.8.1 การออกแบบ GUI

ในส่วนของการเชื่อมต่อระหว่างผู้ใช้กับส่วนแสดงผลวิธีการออกแบบ GUI โปรแกรมที่ใช้จะเป็นโปรแกรม Microsoft Visual Studio 2008 ซึ่งใช้ภาษา Visual Basic ในการพัฒนาโปรแกรม โดยการออกแบบ GUI จะแบ่งเป็น 2 ส่วน คือ

2.8.1.1 ส่วนที่ 1 ภาพแสดงลักษณะการทำงานของหุ่นยนต์โดยแสดงผลเป็นภาพในส่วนของการเคลื่อนไหวของหุ่นยนต์ในลักษณะต่างๆ

2.8.2.2 ส่วนที่ 2 เป็นการเชื่อมต่อกับผู้ใช้โดยตรงผ่านทางการสื่อสารอนุกรมและเป็นส่วนแสดงผลของการประมวลผลไมโครคอนโทรลเลอร์




รูปที่ 2.20 แสดงส่วนของ GUI ของผู้บังคับหุ่นยนต์

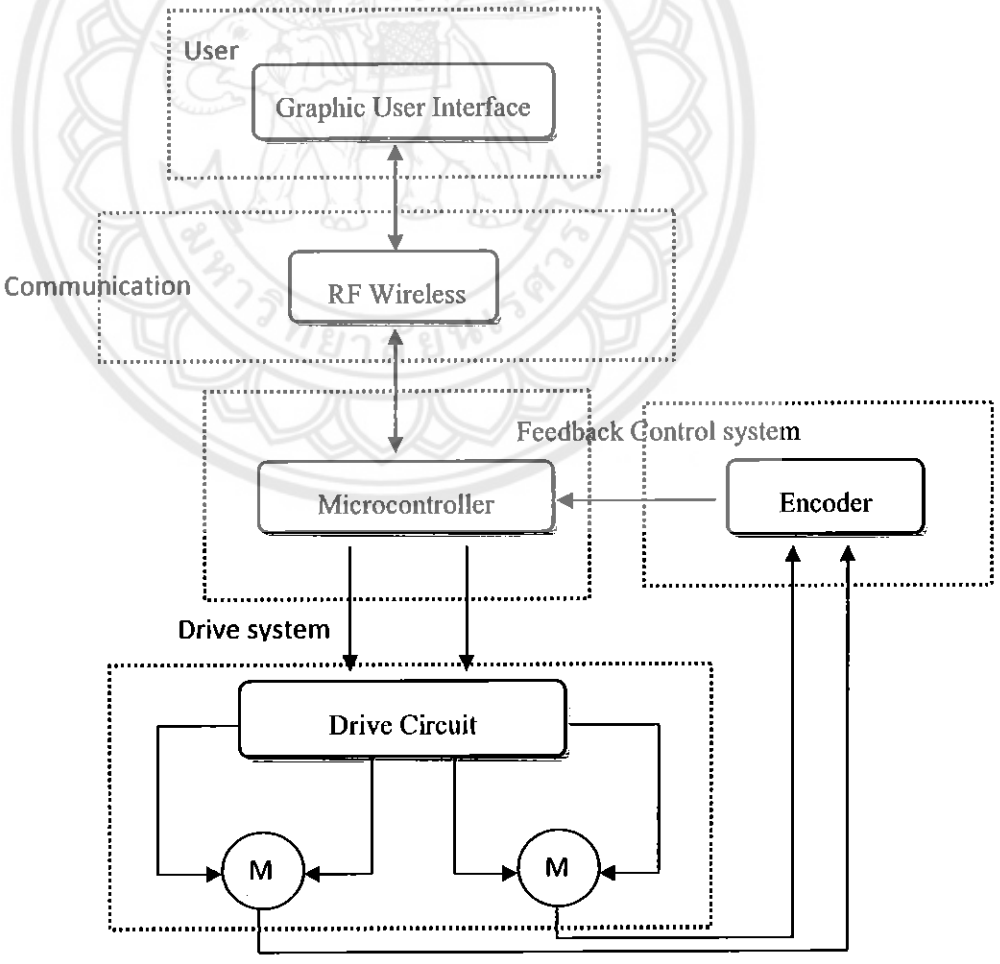
จากหลักการและทฤษฎีที่เกี่ยวข้องในการดำเนินโครงงานได้กล่าวมาข้างต้นนั้น สามารถศึกษาเพิ่มเติมในส่วนที่เราต้องการจะเน้นได้ตามความต้องการของผู้ดำเนินโครงงานได้เลยแต่ระบบของการทำงานนั้นยังคงต้องมีองค์ประกอบในการดำเนินโครงงานดังนี้ ในส่วนของควบคุมมอเตอร์นั้นต้องใช้การเขียนโปรแกรมของไมโครคอนโทรลเลอร์เพื่อไปสั่งการให้มอเตอร์เกิดการขับเคลื่อนจึงทำให้เอ็นโค้ดเดอร์นั้นหมุนจนครบรอบแล้วนำค่าไปประมวลผลในไมโครคอนโทรลเลอร์จากนั้นไมโครคอนโทรลเลอร์ก็จะนำค่าที่ได้ส่งไปให้ในส่วนของการแสดงผลออกทางหน้าจอด้วยเช่นกัน

# บทที่ 3
# วิธีการดำเนินโครงงาน

โครงงาน ระบบควบคุมป้อนกลับและวาดตำแหน่งเชิงกลสำหรับหุ่นยนต์ จะเป็นการควบคุมเกี่ยวกับการเคลื่อนที่ของหุ่นยนต์ได้บางส่วนและการจะสามารถระบุตำแหน่งของการเคลื่อนที่ของหุ่นยนต์ในข้อต่อของมอเตอร์ในส่วนต่างๆได้ เพื่อทำให้คนบังคับหุ่นยนต์นำข้อมูลรูปจำลองของตัวหุ่นยนต์นั้นมาวิเคราะห์ข้อมูลความเป็นไปได้ที่จะทำให้หุ่นยนต์นั้นเคลื่อนที่ผ่านอุปสรรคที่อยู่ข้างหน้านั้นเป็นผลสำเร็จ

## 3.1 หลักการทำงาน

ในการทำงานของการจำลองภาพหุ่นยนต์ในตำแหน่งของข้อต่อในส่วนต่างๆของหุ่นยนต์เบื้องต้นนั้น จะต้องอาศัยหลักการควบคุมในส่วนของมอเตอร์ หลักการการป้อนค่ากลับของEncoderและหลักการของการแสดงผลในรูปแบบ3D



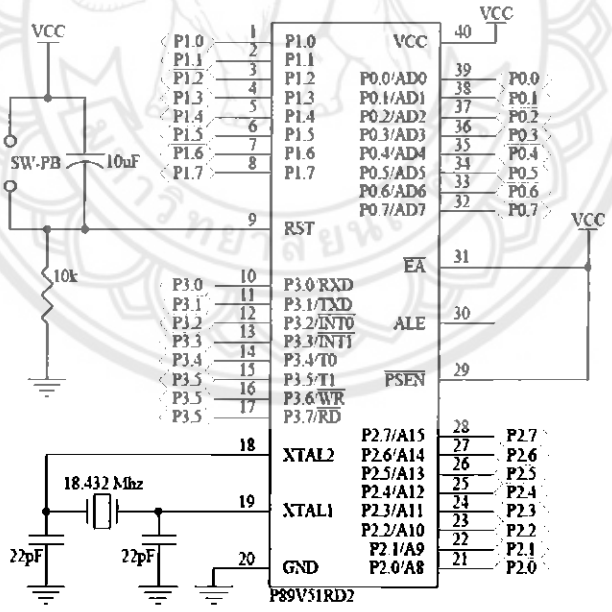

**รูปที่ 3.1** หลักการทำงานแบบภาพรวม

จากรูปที่ 3.1 มีการทำงานหลายๆ ส่วนประกอบกันโดยจะมีไมโครคอนโทรลเลอร์เป็นศูนย์กลางในการทำงานต่างๆ คือ การควบคุมมอเตอร์ การรับค่าจากEncoder และการติดต่อกับผู้ควบคุมทาง Graphic User Interface โดยผ่านการติดต่อสื่อสารกันแบบไร้สาย ( Wireless LAN ) โดยทางไมโครคอนโทรลเลอร์จะเป็นคนติดต่อผ่านเครือข่ายไร้สายไปยังผู้ควบคุม ดังนั้นไมโครคอนโทรลเลอร์จึงเป็นส่วนสำคัญจึงต้องออกแบบตั้งแต่ต้น

## 3.2 การออกแบบชุดควบคุมมอเตอร์

### 3.2.1 การออกแบบชุดไมโครคอนโทรลเลอร์

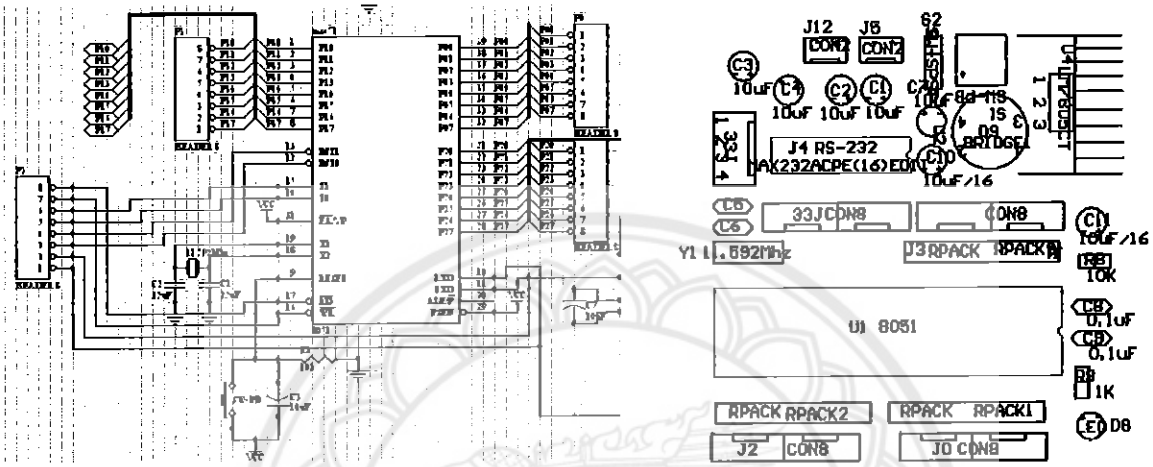
บอร์ดควบคุมไมโครคอนโทรลเลอร์จะมีหน้าที่ทำการรับคำสั่งจากผู้ควบคุมผ่านทางคอมพิวเตอร์เพื่อไปสั่งการให้ตัวมอเตอร์ที่อยู่ในหุ่นยนต์นั้นเคลื่อนที่ได้ตามโปรแกรมที่ผู้ควบคุมสั่ง โดยมอเตอร์ที่เราต้องการควบคุมทั้งหมดนั้นมีด้วยกัน 4 ตัว ซึ่งชุดควบคุมของมอเตอร์นั้น 1 ชุดควบคุมสามารถควบคุมมอเตอร์ได้ 2 ตัว จึงทำให้ต้องมีชุดควบคุมมอเตอร์ 2 ชุด โดยที่ชุดควบคุมมอเตอร์ของแต่ละชุดจะถูกสั่งการจากพอร์ทของไมโครคอนโทรลเลอร์ 1 พอร์ท ควบคุม



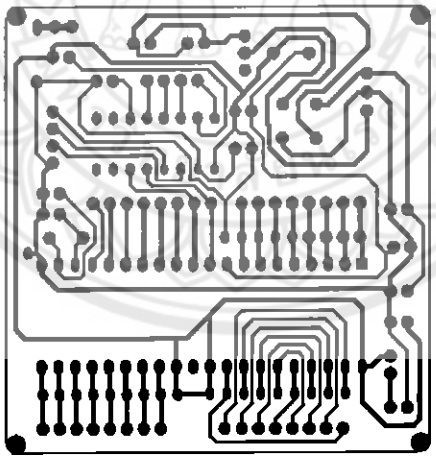

**รูปที่ 3.2** วงจรไมโครคอนโทรลเลอร์ P89V51RD2

จากรูป 3.2 เป็นไมโครคอนโทรลเลอร์ของ Philip ตระกูล MSC-51 P89V51RD2 โดยจะใช้พอร์ทในการควบคุมมอเตอร์โดยใช้พอร์ท 0 ในการควบคุม ซึ่งเป็นพอร์ทเอาต์พุตของไมโครคอนโทรลเลอร์โดยทำการต่อกับชุดควบคุมมอเตอร์

การออกแบบบอร์ดไมโครคอนโทรลเลอร์นี่จะใช้โปรแกรม Protel 99 SE ในการออกแบบรายวงจร โดยเริ่มจากการวาดวงจรเป็นแบบ Schematic จากนั้นเราทำการแปลงวงจรที่เราได้มาเป็นไฟล์ PCB เพื่อทำให้เป็นรายวงจรการเชื่อมต่อจาก Schematic จากราย PCB เราก็ทำการ Convert ไฟล์เป็นรายวงจรสำหรับกัดปริ้นได้

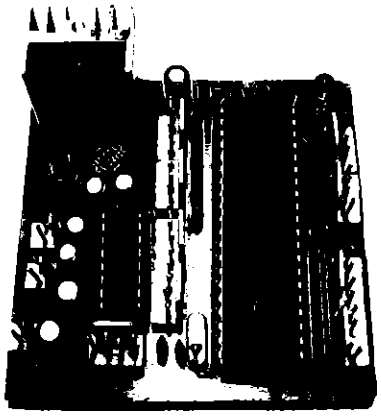



**รูปที่ 3.3** การออกแบบวงไมโครคอนโทรลเลอร์เป็นแบบ Schematic และการวางอุปกรณ์ในบอร์ด



**รูปที่ 3.4** รายวงจรไมโครคอนโทรลเลอร์เป็นแบบที่ Convert จากไฟล์ PCB

จากรูป 3.4 เราก็จะนำมาปริ้นลงแผ่นใสแล้วรีดรายวงจรที่ปริ้นนั้นลงแผ่นทองแดงเพื่อจะได้ติดรายทองแดงจากนั้นนำรายที่ติดหมึกดำนั้นไปแช่ในกรดกัดปริ้นจนทองแดงบนแผ่นนั้นๆหายไปหมายยกเว้นทองคองตรงหมึกสีดำยังอยู่ จากนั้นนำไปล้างน้ำแล้วทำการเคลือบบอร์ดด้วยน้ำยาเคลือบอีกทีเพื่อไม่ให้รายทองแดงนั้นหลุดลอกออกจากตัวบอร์ดแล้วทำการเจาะรูอุปกรณ์ต่างๆที่จะใส่ให้หมด

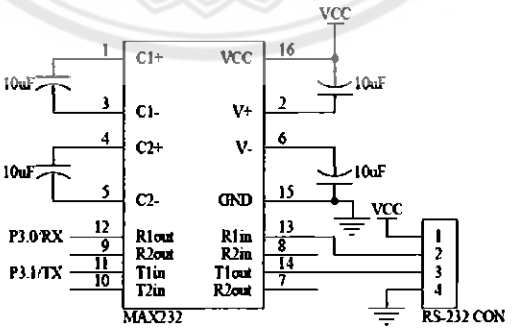


**รูปที่ 3.5** บอร์ดไมโครคอนโทรลเลอร์สมบูรณ์

ไมโครคอนโทรลเลอร์ จะเป็นตัวควบคุมการทำงานทั้งหมดของหุ่นยนต์ โดยที่ไมโครคอนโทรลเลอร์ จะทำงานก็ต่อเมื่อได้รับคำสั่งและนำมาประมวลผลเพื่อที่จะสั่งงานให้ชุดควบคุมมอเตอร์นั้นขับเคลื่อนหุ่นยนต์

### 3.2.2 การติดต่อพอร์ทอนุกรม

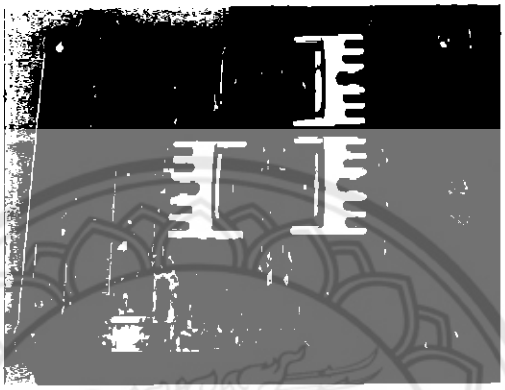
จากบอร์ดไมโครคอนโทรลเลอร์ที่มีสัญญาณในระดับ TTL จึงทำให้การติดต่อสื่อสารกับคอมพิวเตอร์นั้นไม่สามารถทำได้ เพราะในคอมพิวเตอร์ใช้ระบบการสื่อสารแบบอนุกรม หรือ RS-232 (COM Port) มา จึงทำให้บอร์ดไมโครคอนโทรลเลอร์นั้นจะต้องมีไอซีเพื่อแปลงระดับสัญญาณซึ่งไอซีตัวนี้จะทำหน้าที่รับสัญญาณ RS-232 มาแล้วทำการแปลงเป็น TTL และในทางกับกันก็รับสัญญาณ TTL แล้วทำการแปลงเป็น RS-232 ได้เหมือนกันโดยไอซีที่กล่าวมานี่คือ MAX-232



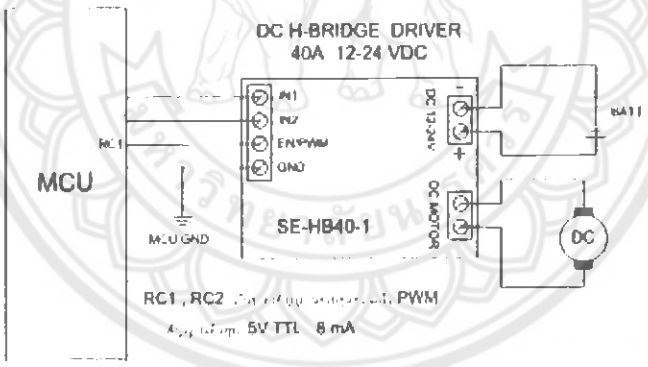

**รูปที่ 3.6** วงจรติดต่อผ่านพอร์ทอนุกรม RS-232

### 3.2.3 ชุดควบคุมมอเตอร์แบบ H-Bridge

มอเตอร์และการควบคุมมอเตอร์นั้น มอเตอร์จะสามารถเปลี่ยนแปลงทิศทางในการหมุนของ DC มอเตอร์ แบบแม่เหล็กถาวรได้นั้น เกิดจากการที่เราสลับขั้วของแหล่งจ่ายไฟฟ้าที่ต่อเข้ากับมอเตอร์และในทางปฏิบัติจะใช้เป็นวงจรอิเล็กทรอนิกส์ที่เรียนว่า H-Bridge เป็นตัวจัดการทำงาน ส่วนของโครงงานนี้ได้ศึกษา DUAL FULL-BRIDGE DRIVER รุ่น SE-HB40-1 ดังรูปที่(3.7)



รูปที่ 3.7 บอร์ดควบคุมมอเตอร์ H-Bridge ขนาด 40A




**รูปที่ 3.8** ภาพแสดงการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับบอร์ด SE-HB40-1

คุณสมบัติของ Module ความคุมทิศทางการหมุนของมอเตอร์ขับเคลื่อน แบบ H-Bridge รุ่น SE-HB40-1

**Output**

- Motor DC Supply 12-24 V 40A (Max.)
- Full-Complementary Power MOSFET Driver

  With ultra-fast reverse recovery protection diodes

**Input**

- Full Opto-isolated input interface signals
- 5V 8 mA TTL – Level

**Drive Mode:** independently with

- ON – OFF Control
- Direction Control
- Speed Control (PWM Drives)

**PWM Frequency :** 400 Hz - 1000 Hz ( 800 Hz Recommend )

**เหตุผลที่เลือกใช้ H-Bridge รุ่น SE-HB40-1**

หากมีการขับเกินกำลังจนมอสเฟทพัง ให้ตรวจสอบโดยการใช้มิเตอร์วัดการลัดวงจรของมอสเฟททั้ง 4 ตัว หากตัวใดเสียขั้วมอสเฟทจะลัดวงจร ให้เปลี่ยนมอสเฟทได้เลย วงจรส่วนอื่นปลอดภัย ไม่จำเป็นต้องตรวจสอบ

- การเปลี่ยนมอสเฟท เบอร์ IRF4905 ไม่มีจำหน่าย ให้ใช้เบอร์ IRF9540 แทน
- เบอร์ IRFZ44 มีจำหน่าย หรือให้ใช้เบอร์ HUF75639P3
- IRF3710PBF จะทำให้ทนแรงดันได้ดีกว่า

จากการทดลองไอซีความคุมมอเตอร์สำเร็จรูป เบอร์ L298D ร่วมกับ Relay 10 A ไม่เหมาะสมกับการสวิตซ์ซึ่งจากการบังคับแบบไม่ต่อเนื่อง เช่น เมื่อกดบังคับให้หุ่นยนต์เดินหน้าและหยุด สลับกันอย่างรวดเร็วจะทำให้หน้าสัมผัสของ Relay เกิดอุณหภูมิสูงทำให้ส่วนหน้าสัมผัสโลหะติดกันไม่แยกจากกันทำให้ควบคุมหุ่นยนต์ไม่ได้จึงเลือกใช้โมดูล H-Bridge รุ่น SE-HB40-1 แทน

จากแผ่นริ้นที่หาซื้อตามท้องตลาดเพื่อนำมาออกแบบวงจรเอง จะทำให้เกิดปัญหาเมื่อวงจรเกิดความร้อนและความถี่สูงระหว่าง ลายทองแดงทำให้ประสิทธิภาพของวงจรที่ออกแบบเองเกิดปัญหาความผิดพลาดของการส่งสัญญาณบ่อยครั้งจากการทดลอง และที่สำคัญคือลายทองแดง บางมากจะต้องใช้สายไฟเดินบริเวณที่มีกระแสไฟฟ้าไหลผ่านมาก ๆ สามารถทนกระแสได้ถึง 40 A ซึ่งแตกต่างจากวงจรที่ออกแบบขึ้นเอง ที่ทนกระแสได้น้อยกว่า

โดยการออกแบบโปรแกรมการควบคุมจะเป็นไปตามตารางที่ 3.1 โดยโปรแกรมจะรับอินพุตจากส่วนควบคุมระดับบน ถ้าส่วนควบคุมระดับบนส่งข้อมูลมาสั่งงานส่วนควบคุมระดับล่าง (MCU) ก็จะส่งแรงดันเอาต์พุต ประมาณ 5 V ตามจำนวนบิตที่วงจรความคุมทิศทางมอเตอร์ H-Bridge รุ่น SE-HB40-1 ดังนี้

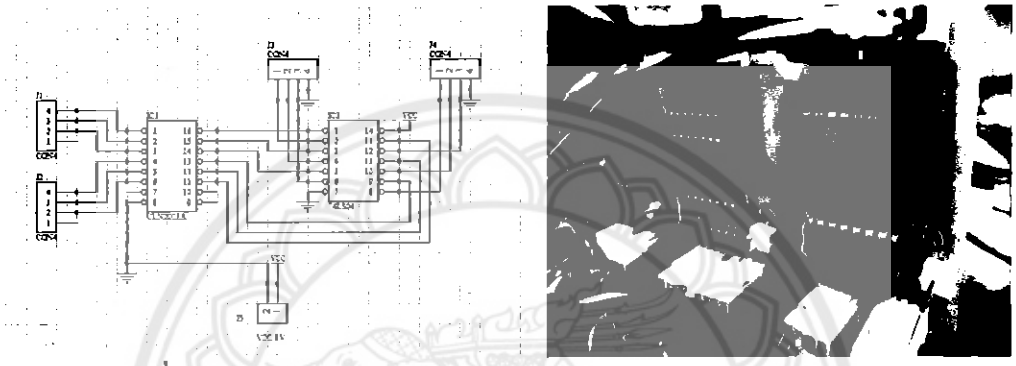
**ตารางที่ 3.1 การเคลื่อนที่ของหุ่นยนต์**

| สถานะ | Wheel Left | | | | Wheel Right | | | |
|---|---|---|---|---|---|---|---|---|
| | IN1 | IN2 | EN | GND | IN1 | IN2 | EN | GND |
| Forward Robot | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| Backward Robot | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| Turn Left Robot | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Turn Right Robot | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| Turn Short Left Robot | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| Turn Short Right Robot | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

จากตารางที่ 3.1 จะทำให้ทราบได้ว่าถ้าต้องการให้หุ่นยนต์เคลื่อนที่ไปในทิศทางใด ก็ต้องเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ ให้ส่งแรงดันออกที่พอร์ตของไมโครคอนโทรลเลอร์ ซึ่งในที่นี้การควบคุมจะส่งเอาต์พุตออกที่พอร์ต 2 ของบอร์ดไมโครคอนโทรลเลอร์ตามบิตแต่ละบิตของพอร์ตไมโครคอนโทรลเลอร์ ที่ช่อง สถานะ IN1 และ IN2 คือ ขาควบคุมทิศทางการหมุนของมอเตอร์ และ EN คือ ขา Enable มีหน้าที่ในการเปิดการทำงานของวงจรควบคุมหารหมุนของมอเตอร์ และยังเป็นขาที่ใช้ปรับความเร็วของมอเตอร์ในการใช้วิธีการ Pulse Wide Modulation ตามที่ได้กล่าวแล้วในบทที่ 2 และขา GND คือ ขากราวด์ที่ต้องต่อร่วมกับบอร์ดไมโครคอนโทรเลอร์ด้วย เพื่อให้กระแสไหลได้ครบวงจรหรือมองเป็นกราวด์ร่วนนั้นเอง เมื่อวงจรควบคุมทิศทางการหมุนของมอเตอร์ ได้รับการส่งเอาท์พุตจากไมโครคอนโทรเลอร์เรียบร้อย มอเตอร์จะหมุนโดยมีจุดต่อมอเตอร์ และรับแรงดันจากภายนอกเพื่อให้มอเตอร์ได้รับกระแส โดยแหล่งจ่ายแรงดันจากภาย นอกนี้จะเลือกใช้แบตเตอรี่รถจักรยานยนต์ที่มีแรงดัน 12 VDC 3A/hr ซึ่งมีราคาถูกและสามารถประจุไฟเข้าได้อย่างสะดวก แต่มีน้ำหนักมากถ้าเลือกเป็นแบตเตอรี่แบบใช้น้ำกลั่น และในที่นี้เลือกแบบแบตเตอรี่แห้ง และการเลือกแรงดันที่จ่ายให้มอเตอร์นั้น พิจารณาโดยมอเตอร์เป็นมอเตอร์กระแสตรงที่ทนแรงดันสูงสุดไม่เกิน 12 V

เมื่อออกแบบรหัสควบคุมเพื่อเตรียมการเขียนโปรแกรมแล้วนั้น การออกแบบวงจรเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับชุดควบคุมทิศทางการหมุนของมอเตอร์นั้น กระแสที่ออกจากพอร์ตของไมโครคอนโทรลเลอร์ ไม่เพียงพอต่อการควบคุมชุดควบคุมทิศทางการหมุนของมอเตอร์เนื่องจาก ชุดควบคุมทิศทางการหมุนของมอเตอร์ต้องการกระแสเข้าที่ขา IN1, IN2, EN ที่กระแส สูงกว่า 8 mA แต่ไมโครคอนโทรลเลอร์จ่ายกระแสออกพอร์ตได้เพียง 1 mA. เท่านั้น จึงต้องออกแบบวงจรขยายกระแสให้กับบอร์ดควบคุมทิศทางการหมุนของมอเตอร์ โดยใช้อุปกรณ์ดังต่อไปนี้ คือ IC ULN2003A และ IC Not Gate

7404 โดย IC ULN2003A โดยไอซีตัวนี้จะทำหน้าที่ขยายกระแสให้ ให้กับบอร์ดควบคุมทิศทางการหมุนของมอเตอร์และต่อกับ IC Not Gate 7404 เพื่อกลับบิตและส่งเข้าให้กับบอร์ดควบคุมทิศทางการหมุนของมอเตอร์ การทำงานของวงจรขยายกระแสมีลักษณะเหมือนกับวงจรบัฟเฟอร์ บางวงจรหรือ IC สำเร็จรูปบางตัวเช่น IC 74LS245 ทำหน้าที่ขยายกระแสจากพอร์ตของไมโครคอนโทรลเลอร์ หรืออาจจะใช้วิธีการ ต่อตัวต้านทานอนุกรมกับแหลงจ่ายแรงดัน 5 Volt เพื่อทำให้เกิดกระแสที่แต่ละบิตสูงขึ้นแต่กระแสก็ยังไม่เพียงพอ ตัวอย่างการต่อวงจร โดยใช้ IC ULN2003A และ IC Not Gate 7404 ดังรูปที่ 3.8



รูปที่ 3.9 วงจรขยายกระแสด้วย IC ULN2003A และ IC Not Gate 7404

เมื่อประกอบอุปกรณ์ดังแสดงในรูปที่ 3.10 เรียบร้อย ก็จะสามารถนำไปใช้งานในควบคุมการหมุนของมอเตอร์ได้โดยกระแสที่ได้จากวงจร Buffer ที่ออกแบบขึ้นให้กระแสสูงถึง 500 mA จากการแสดงคุณสมบัติจากเอกสารแสดงคุณสมบัติ

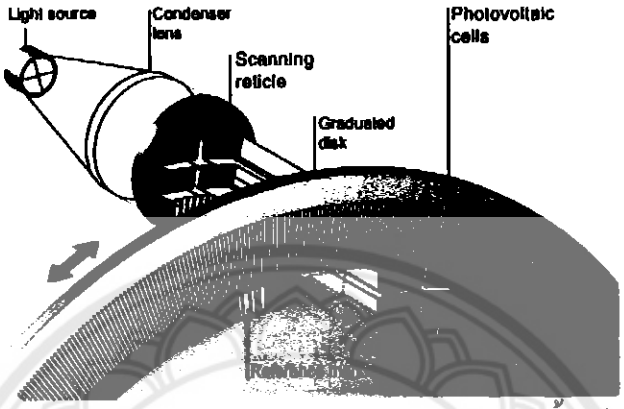
## 3.3 การออกแบบชุดทดลองการทำงาน Encoder

ชุดการทดลอง Encoder นั้นจะต้องอุปกรณ์ประกอบไปด้วยดังนี้

1. ชุดไมโครคอนโทรลเลอร์
2. ชุดควบคุมการทำงานของมอเตอร์แบบ H-Bridge
3. Encoder

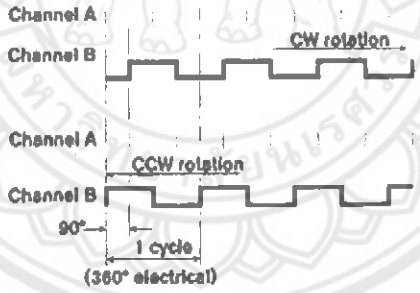
การออกแบบชุดทดลองการทำงานของ Encoder นี้เป็นการทดลองเพื่อศึกษาการทำงานของ Encoder โดยจะเริ่มศึกษาจากสัญญาณพัลส์ของ Encoder ก่อนโดยสัญญาณพัลส์นั้นจะเกิดจากการที่ Encoderทำงานโดยการหมุนเมื่อเราทำการควบคุมมอเตอร์นั้นเอง สัญญาณพัลส์เกิดจากโครงสร้างจะประกอบด้วยตัวกำเนิดแสง, ตัวจับแสงซึ่งถูกคั่นกลางด้วยแผ่นจานกลมๆที่มีการทำรูเจาะไว้รอบๆแผ่น (จำนวนรูจะขึ้นอยู่กับความละเอียดของ incremental encoder ) และหน้ากากแยกช่องของสัญญาณพัลส์ A ,B และ Z สัญญาณพัลส์ที่ได้จากเอนโค้ดเดอร์ชนิดนี้จะประกอบด้วย 3 แทรค (tracks) คือ A,BและZ พัลส์ที่เกิดจาก แทรค A และ B จะเกิดการเหลื่อมกันมีความต่างเฟสกัน 90 องศา เพื่อทำหน้าที่รายงานผลของ

ความเร็วและทิศทางการหมุนของมอเตอร์ให้คอนโทรลเลอร์ดังนี้กรณีพัลส์ A เกิดขึ้นก่อน B คอนโทรลเลอร์จะรับรู้ว่ามอเตอร์กำลังหมุนด้วยทิศทางตามเข็มนาฬิกา ส่วนแทรค Z หรือพัลส์อ้างอิง จะเกิดขึ้น1พัลส์ในการหมุน 1 รอบ ทำหน้าใช้อ้างอิงตำแหน่งของโรเตอร์

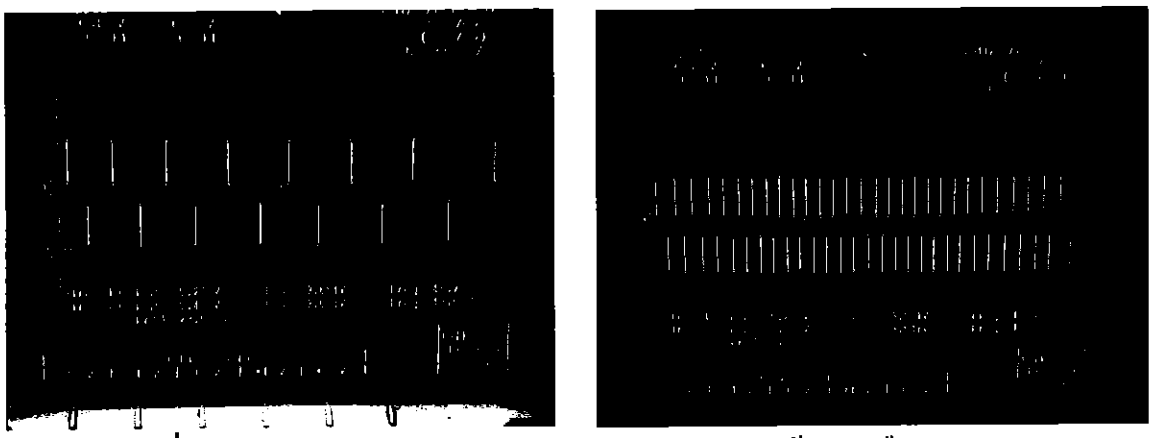



รูปที่ 3.10 ส่วนประกอบของ Encoder และการทำงานเบื้องต้น[4]

จากการวิเคราะห์สัญญาณ output ของ Encoder พบว่าคลื่นของสัญญาณจะเป็นไปตามรูปที่ 3.11




รูปที่ 3.11 รูปของสัญญาณ output ที่เกิดจาก Encoder



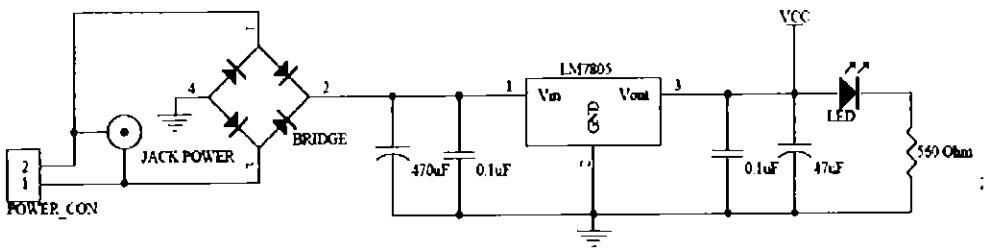
รูปที่ 3.12 รูปของสัญญาณด้านซ้ายมือหมุนแบบ CCW และด้านขวามือหมุนแบบ CW

อธิบายรูปที่ 3.12 เป็นการทดลองโดยการวัดคลื่นสัญญาณที่ออกมาจากแทรค A และ B ของ Encoderโดยสัญญาณคลื่นลูกล่างนั้นเกิดจากแทรค A และคลื่นลูกบนนั้นเกิดจากแทรค B โดยนำมาเปรียบเทียบให้เห็นว่ามีความต่างเฟสของลูกคลื่นนั้นเป็น 90 องศา โดยนำจะนำสัญญาณที่ได้นั้นมาเชื่อมต่อกับขา T0 กับ T1 ของไมโครคอนโทรลเลอร์ที่ พอร์ท P1.5 และ P1.6 เพื่อนำสัญญาณนั้นมาเขียนโปรแกรมเพื่อกระทำกับสัญญาณที่เราได้ โดยชุดทดลองของเรานั้นจะนำ Encoder มาติดเข้ากับแกนหมุนของมอเตอร์แล้วมีชุดควบคุมการหมุนของมอเตอร์แบบ H-Bridge โดยการทดลองนี้จะใช้ ไอซี TA7279 เป็นตัวควบคุม โดย ไอซี TA7279 จะมีการทำงานโดยการสั่งงานของไมโครคอนโทรลเลอร์การทำงานจะต้องสั่งงานผ่านขาIN1 และ ขา IN2 จากนั้นไอซี TA7279 จะทำการหมุนมอเตอร์ โดยค่าที่สั่งเข้ามาทางขา IN1 และ IN2 นั้นต้องเป็นสัญญาณระดับ TTL โดยมี 0V. กับ 5V. โดยต้องทำการสั่งงานเป็นตรงข้ามกันระหว่างขา IN1 กับ IN2 โดยมีค่าตารางความจริงดังนี้

ตารางที่ 3.2 ตารางความจริงการทำงานของ ไอซี TA7279

| IN1 | IN2 | OUT1 | OUT2 | Status |
|---|---|---|---|---|
| 1 | 1 | L | L | BRAKE |
| 0 | 1 | L | H | FWD/REV |
| 1 | 0 | H | L | REV/FWD |
| 0 | 0 | High Impedance | | STOP |

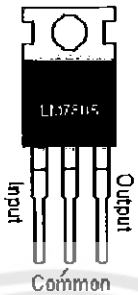
แล้วยังชุดการทดลองการทำงานของ Encoder นี้ยังมีบอร์ดแปลงไฟ 12v. เป็น 5v. ประกอบด้วยเนื่องจากไฟเลี้ยงให้บอร์ดไมโครคอนโทรลเลอร์และบอร์ดควบคุมมอเตอร์นั้นใช้ไฟเลี้ยง 12v. แต่ไฟเลี้ยงของ Encoder นั้นเป็นไฟ 5v.จึงต้องทำบอร์ดเพื่อแปลงไฟจาก 12v. เป็น 5v. ด้วยโดยมีลักษณะของวงจร ไอซี และรูปร่างดังรูปภาพที่ 3.13




**รูปที่ 3.13** รูปวงจรบอร์ดแปลงไฟ 12v. เป็น 5v.

โดยบอร์ดแปลงไฟจะมีอุปกรณ์หลักคือ ไอซี LM7805 เป็น Dual power supply Regulator คือไอซีที่ใช้ในการควบคุมไฟเลี้ยงให้กับวงจรอิเล็กทรอนิกส์โดยจะดูได้จากรหัส 2 ตัวท้ายว่าเราจะต้องการแปลง

ไฟ DC ให้มีแรงดันในระดับใด เช่นที่ใช้คือรหัส 7805 จะทำให้แรงดันไฟฟ้าที่ขาออกจาก Regulator นั้นออกมามีค่า 5 โวลต์ แต่ถ้ารหัสของไอซีแปลงไฟนี้เป็น 7808 จะทำให้แรงดันไฟฟ้าที่ขาออกมาจะมีค่าเป็น 8 โวลต์ นั้นเอง




รูปที่ 3.14 ไอซี LM7805 และบอร์ดแปลงไฟ 12 โวลต์

เมื่อเรานำบอร์ดไมโครคอนโทรลเลอร์มาเชื่อมต่อกับบอร์ดควบคุมมอเตอร์และบอร์ดควบคุมมอเตอร์ทำการหมุนมอเตอร์แล้วทำให้ Encoder เกิดสัญญาณพัลส์ส่งกลับมายังไมโครคอนโทรลเลอร์เพื่อประมวลผลโดยมีบอร์ดจ่ายไฟ 12 โวลต์ และมี แรงดันจากการแปลงไฟ 5 โวลต์ที่เป็นแหล่งจ่ายให้บอร์ดต่างๆด้วย โดยจะแสดงรูปของการเชื่อมต่อบอร์ดต่างๆในรูปที่ 3.15

**รูปที่ 3.15** ชุดทดลองการทำงานของ Encoder

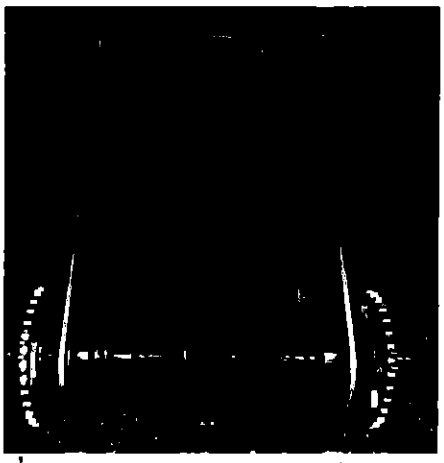
## 3.4 โครงสร้างหุ่นยนต์กู้ภัย

เนื่องจากโครงการนี้เป็นโครงการหุ่นยนต์กู้ภัย วัสดุที่เลือกใช้จึงต้องการคุณสมบัติที่มีความคงทนและแข็งแรง ส่วนประกอบหลักของหุ่นยนต์คือตัวฐานของหุ่นยนต์ที่จะประกอบไปด้วยมอเตอร์ชุดขับเคลื่อนล้อ 2 ข้างด้วยกัน, แล้วมีโซ่ที่ติดยางใช้ในการเคลื่อนที่ของหุ่นยนต์ โครงสร้างของหุ่นยนต์ส่วนใหญ่จะเป็นเหล็กและอลูมิเนียม เนื่องจากต้องการให้มีความแข็งแรง

### 3.4.1 โครงสร้างฐานล่างของหุ่นยนต์

โครงสร้างของของหุ่นยนต์ในส่วนของการขับเคลื่อนในส่วนข้างล่าง ส่วนใหญ่วัสดุที่ใช้จะทำจากอลูมิเนียมโดยที่มีความหนาของแต่ละชิ้นอลูมิเนียมมีขนาด 5 มิลลิเมตร โดยจะใช้ด้วยกันทั้งหมด 4 ชิ้น ประกอบไปด้วย โครงอลูมิเนียมที่ยึดกับมอเตอร์ขับเคลื่อนทั้ง 2 ตัว จำนวน 2 แผ่น และอลูมิเนียมที่ใช้ในการทำแขนของหุ่นยนต์ทั้ง 2 ข้างอีก 2 แผ่น

รูปที่ 3.16 โครงสร้างของหุ่นยนต์



รูปที่ 3.17 โครงสร้างของหุ่นยนต์(2)

จากนั้นนำแผ่นอลูมิเนียมมายึดระหว่างแผ่นอลูมิเนียมที่ยึดกับมอเตอร์ทั้ง 2 ข่างเอาไว้โดยทำเป็นแบบปูพื้นฐานข้างล่างให้กับหุ่นยนต์โดยแผ่นอลูมิเนียมที่ใช้ปูพื้นและแผ่นอลูมิเนียมที่เป็นตัวยึดชิ้นงานทั้ง 2 ข้างนั้นเราจะใช้อลูมิเนียมขนาด 3 มิลลิเมตร

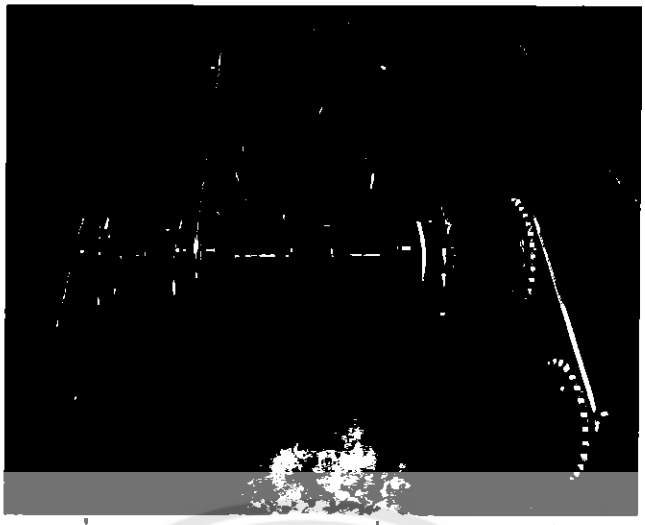
### 3.4.2 โครงสร้างชุดขับเคลื่อนของหุ่นยนต์

มอเตอร์ที่ใช้ในการเคลื่อนที่ของหุ่นยนต์นั้นส่วนใหญ่จะเป็นมอเตอร์กระแสตรงทั้งหมด โดยมีมอเตอร์หลักที่ใช้ในการเคลื่อนที่ของหุ่นยนต์นั้นใช้ มอเตอร์กระจกไฟฟ้า

รูปที่ 3.18 มอเตอร์กระจกไฟฟ้า

การติดตั้งมอเตอร์กระจกไฟฟ้านั้นจะทำการยึดกับแผ่นอลูมิเนียมแผ่นที่มีความหนา 5 มิลลิเมตร โดยทำการยึดในแนวนอนโดยให้มอเตอร์กระจกนั้นขนานไปกับแผ่นอลูมิเนียมแล้วใช้เจาะรูที่แผ่นอลูมิเนียมสำหรับยึดแล้วใส่น็อตยาวยึดระหว่างมอเตอร์กับอลูมิเนียม จากนั้นทำการติดเฟืองเข้ากับมอเตอร์กระจกโดยเฟืองที่ใช้นั้นเป็นเฟืองโซ่จักรยาน เพื่อใช้โซ่จักรยานเป็นตัวขับเคลื่อนเหมือนกัน

ชุดเพลาขับเคลื่อนล้อจะมีเพลาขนาดเส้นผ่านศูนย์กลาง 26 มิลลิเมตร เป็นเพลาที่ใช้ขับเคลื่อนหลัก โดยบนเพลานั้นจะทำการประกอบกับชุดลูกปืนและ ชุดขับเคลื่อน สเตอร์ของโซ่จักรยาน และสเตอร์ของโซ่เบอร์ 40 โดยโซ่เบอร์ 40 ใช้ในการขับเคลื่อนล้อด้านหน้าและด้านหลัง เพลาขับเคลื่อนหลักนั้นจะมีการแบ่งครึ่งโดยแทนใส่มอเตอร์สำหรับหมุนแขน เพื่อที่จะใช้เพลาของ 2 ข้างนั้นอิสระต่อกันเพื่อใช้สำหรับเวลาหุ่นยนต์เลี้ยวซ้ายหรือเลี้ยวขวา เพลาในส่วนที่ 2 คือเพลาในที่ยัดเข้าไปในเพลาขับเคลื่อนอีกทีโดยเพลาในนั้นมีไว้สำหรับใช้ในกายกแขนของหุ่นยนต์ขึ้นลงไปมาโดยมีมอเตอร์และชุดในการหมุนเพลานั้นอยู่ตรงกลางของหุ่นยนต์เพื่อทำหน้าที่ในการหมุนเพลาในให้แขนข้างหน้ายกขึ้นมานั้นเอง ดูจากรูป 3.19



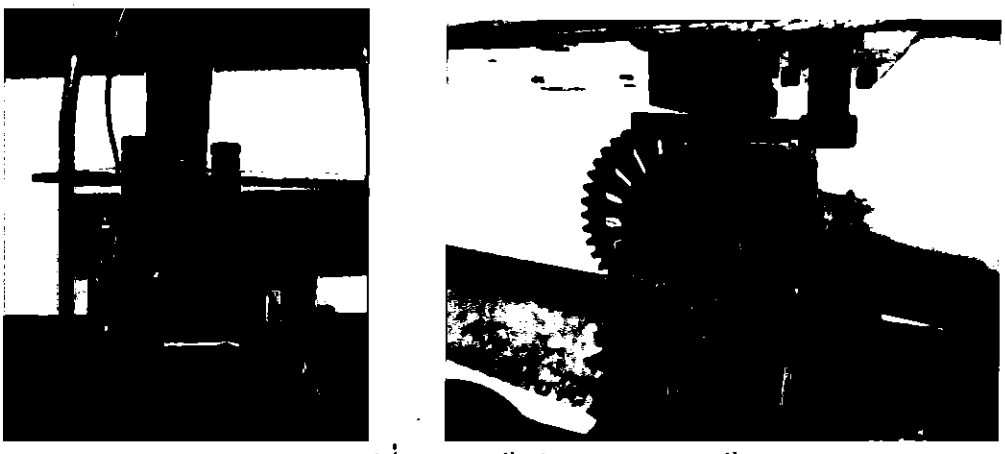
รูปที่ 3.19 โครงสร้างชุดขับเคลื่อนของหุ่นยนต์กู้ภัย

เมื่อเราทำการประกอบโครงสร้างของชุดขับเคลื่อนของหุ่นยนต์เข้ากับชุดขับเคลื่อนของแขนหน้าเข้าด้วยกันแล้วแล้วก็นำโซ่ที่ติดยางที่ใช้ในการยึดเกาะกับพื้นผิวขรุขระ เอามาติดตั้งเอาสเตอร์ที่เราเตรียมไว้บนแกนการควบเคลื่อน

**รูปที่ 3.20 ชุดขับเคลื่อนของหุ่นยนต์กู้ภัย**

**3.4.3 โครงสร้างข้อต่อตามส่วนต่างๆของหุ่นยนต์**

หุ่นยนต์กู้ภัยใน 1 ตัวนั้นจะมีข้อต่อเป็นส่วนประกอบของหุ่นยนต์นั้นมีหลายส่วน เช่น ส่วนแขนล่างของหุ่นยนต์ ส่วนของแขนบนของหุ่นยนต์ ส่วนของข้อศอกของหุ่นยนต์ เป็นต้น โดยข้อต่อพวกนี้เราจะใช้มอเตอร์ในการสั่งงานด้วยกันทั้งหมดโดยจะแยกการออกแบบดังนี้



รูปที่ 3.21 ชุดข้อต่อของฐานรองกล้อง

โดยฐานรองข้อต่อกล้องออกแบบให้สามารถหมุนกล้องได้ให้ทั้งในแนวแกนตั้ง (ขึ้น - ลง) และในแนวแกนนอน (ซ้าย – ขวา)

รูปที่ 3.22 ชุดข้อต่อในส่วนของข้อศอก และชุดข้อต่อในส่วนของแขนท่อนบน

ชุดข้อต่อของข้อศอกของหุ่นยนต์และชุดข้อต่อของแขนบนของหุ่นยนต์ นั้นจะออกแบบเพื่อให้เคลื่อนที่ในแนวแกนตั้ง ( ขึ้น - ลง )

รูปที่ 3.23 ชุดข้อต่อในส่วนของแขนล่าง

รูปที่ 3.24 แสดงรูปหุ่นยนต์กู้ภัย

### 3.5 การติดตั้ง Encoder

ในการทำงานของ Encoder นั้นเกิดจากการหมุนแกนของตัว Encoder ไปในทิ้งทางซ้าย ขวาแล้ว สัญญาณข้อมูลที่ออกมาจะออกมาตามสายสัญญาณ A กับ B โดยที่ สัญญาณจะออกมาในรูปของระดับสัญญาณ H กับ L โดยข้อมูลที่ส่งออกมาจากสาย A และ B นั้นจะมีค่าของสัญญาณเหลื่อมกันอยู่ 90 องศา แล้วในโครงงานนี้จะมี Encoder ด้วยกัน 3 ตัว โดยจะจำลองการทำงานในส่วนของข้อต่อในส่วนต่างๆ จำนวน 2 ข้อต่อ และนำสัญญาณที่ออกมาจาก Encoder นั้นมาคำนวณเพื่อหาความสัมพันธ์ของระยะทางอีก 1 ตัวด้วยกัน การติด Encoder ตามข้อต่อในส่วนต่างๆนั้นจะแสดงให้เห็นในรูปที่ 3.24 , 3.25 , 3.26

รูปที่ 3.25 แสดงการติดตั้ง Encoder ในส่วนของล้อขับเคลื่อน

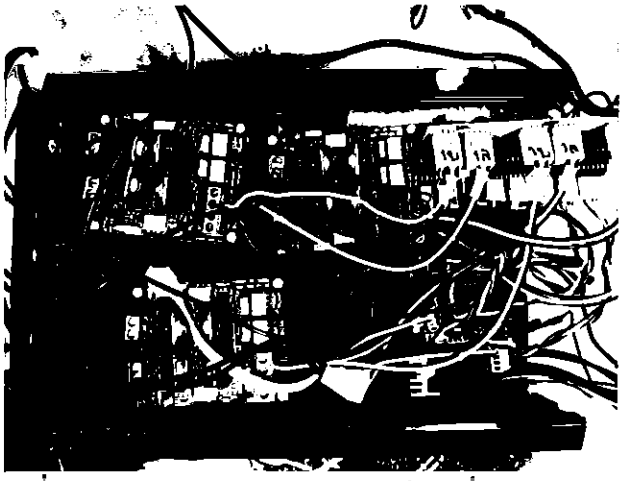
**รูปที่ 3.26** แสดงการติดตั้ง Encoder ในส่วนของล้อข้อต่อแขนล่าง

**รูปที่ 3.27** แสดงการติดตั้ง Encoder ในส่วนของล้อข้อต่อแขนบน

## 3.6 การติดตั้งชุดควบคุมและติดตั้งอุปกรณ์ในการติดต่อสื่อสาร

### 3.6.1 การติดตั้งชุดควบคุม

จากขั้นตอนการดำเนินงานที่ 3.2 เราได้ทำการสร้างวงจรและทดสอบเรียบร้อยแล้วจะต้องทำการติดตั้งชุดควบคุมเข้ากับกล้องอเนกประสงค์ โดยทำการเดินสายไฟ ติดตั้งสวิตซ์เปิดไฟให้เรียบร้อยทุกแหล่งจากไฟและต้องทำการกันกระแทกด้วยโฟมหั่นบางๆเพื่อลองบอร์ดวงจรกระแทกกับกล้อง



รูปที่ 3.28 แสดงการประกอบอุปกรณ์การเชื่อมต่อสายไฟ

จากนั้นทำการติดตั้งกล่องควบคุมเข้ากับตัวหุ่นยนต์

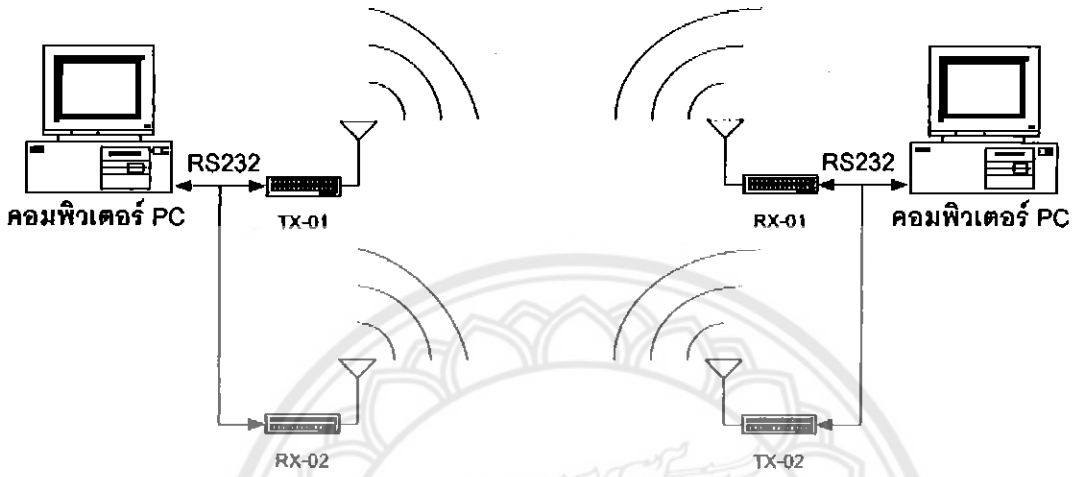
รูปที่ 3.29 แสดงการประกอบกล่องควบคุมเข้ากับหุ่นยนต์

**3.6.2 การติดตั้งอุปกรณ์สื่อสาร**

การใช้ RS-232 Wireless นั้นเพื่อต้องการรับส่งข้อมูลแบบไร้สาย โดยการใช้นั้นจะต้องมี 2 ตัว คือ ทางฝั่งของหุ่นยนต์ 1 ตัว และฝั่งผู้ควบคุมอีก 1 ตัว ซึ่งก่อนใช้งานนั้นต้องทำการตั้งค่าให้ทั้ง 2 ตัวทำงานร่วมกันได้ คือต้องตั้งสัญญาณรับส่งของทั้ง 2 ตัว

RS232 Wireless นั้นเป็นการสื่อสารที่ใช้อุปกรณ์ RS232 to RF-wireless ในสัญญาณความถี่ 2.4GHz เป็นชุดแปลงสัญญาณระหว่าง RS232 และ RF-wireless โดยในโหมดการทำงานของการส่งข้อมูล (Transmitter) จะทำหน้าที่รอรับข้อมูลจากพอร์ตสื่อสารอนุกรม RS232 จากขา RX แล้วแปลงเป็นสัญญาณความถี่ (GFSK) ส่งออกไปในอากาศ และในทางกลับกันในโหมดการทำงานแบบรับ (Receiver) ก็จะทำหน้าที่คอยตรวจจับข้อมูลที่อยู่ในรูปของสัญญาณความถี่จากด้าน RF เพื่อแปลงกลับเป็นข้อมูลแบบ RS232 ส่งออกไปทางขา TX โดยตัวอุปกรณ์จะต้องมีการกำหนดค่าต่าง ๆ ที่เหมือนกันที่ทำให้สามารถรับส่งข้อมูล

กันได้ โดยต้องมีการกำหนดอัตราความเร็วในการรับส่งส่งข้อมูลของตัวเครื่อง กำลังในในการรับส่งข้อมูล และโหมดการทำงานของช่องรับส่งข้อมูลให้ถูกต้อง โดยการรับส่งข้อมูลจะเป็นแบบ Full Duplex คือ สามารถที่จะรับส่งในเวลาเดียวกันได้พร้อม ๆ กัน โดยรูปจะแสดงดังนี้



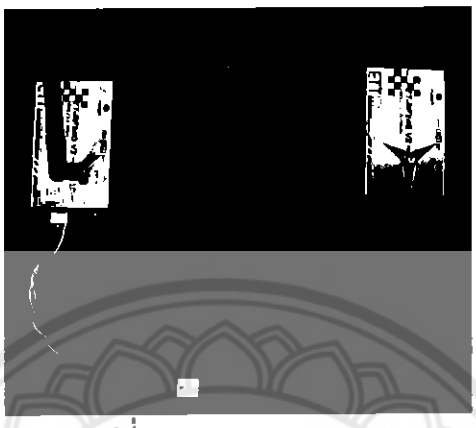

รูปที่ 3.30 การรับส่งข้อมูลแบบ Full Duplex

โดยจะกำหนดโหมดการทำงานเป็น RF Receive Only และ RF Transmit Only ฝั่งละ 1 ชุด ดังนี้

ตารางที่ 3.3 การกำหนดค่าของอุปกรณ์ RF แบบ Full Duplex

| ค่า Configuration | ET-RF24G V2.0 ฝั่งต้นทาง ตัวที่1 | ET-RF24G V2.0 ฝั่งปลายทาง ตัวที่2 |
|---|---|---|
| User RS232 Baud Rate | 9200 Bps | 9200 Bps |
| RF Data Rate | 250 Kbps | 250 Kbps |
| RF Operation Mode | Auto Direction | Auto Direction |
| RF Power Gain | +0dBm | +0dBm |
| RXD ID Code | 01 | 02 |
| TXD ID Code | 02 | 01 |
| RF Frequency Channel | 0 | 0 |

โดยการกำหนดค่าต่าง ๆ นี้จะทำให้อุปกรณ์สามารถที่จะรับส่งข้อมูลในเวลาเดียวกันได้ เราสามารถที่จะเปลี่ยนแปลงค่าต่าง ๆ ได้โดยสังเกตที่ RXD ID Code และ TXD ID Code จะมีค่าต่อข้ามกันเสมือนแบ่งเป็นช่องในการรับส่งข้อมูลให้ถูกต้อง



รูปที่ 3.31 RS-232 Wireless

เมื่อนำอุปกรณ์ทุกชิ้นมาประกอบกันเป็นหุ่นยนต์กู้ภัย 1 ตัวรูปร่างหน้าตาของหุ่นยนต์ก็จะเป็นแบบรูปที่ 3.31

รูปที่ 3.32 แสดงรูปหุ่นยนต์กู้ภัย

## 3.7 การออกแบบและการเขียนโปรแกรมไมโครคอนโทรลเลอร์

การเขียนโปรแกรมที่ตัวของไมโครคอนโทรลเลอร์นี้สามารถแบ่งได้ 2 รูปแบบดังต่อไปนี้

### 3.7.1 การเขียนโปรแกรมควบคุมการทำงานของมอเตอร์

การเขียนโปรแกรมควบคุมการทำงานของมอเตอร์นี้เราจะทำการเพื่อไปสั่งงานให้มอเตอร์ทำงาน โดยการเขียนโปรแกรมควบคุมการทำงานของมอเตอร์นั้น เราจะเขียนผ่านพอร์ท 0 , พอร์ท 1 , พอร์ท 2 ของไมโครคอนโทรลเลอร์โดยมีการสั่งงานไปยังพอร์ทที่เรากำหนดค่า โดยค่าที่กำหนดนั้นจะมีด้วยกัน 3 ค่า คือ ค่า IN1 , IN2 , EN โดยที่ IN1 คือว่า อินพุต1 IN2 คือค่า อินพุต2 และEN คือค่า สั่งให้ทำงานบอร์ดไดร์มอเตอร์นั้นทำงานหรือไม่ทำงานก็ได้ โดยถ้าเราเขียนโปรแกรมด้วยการส่งค่า 0 ไปให้กับบอร์ดไดร์มอเตอร์นั้นจะทำให้มอเตอร์ไม่ทำงานหรือหยุดการทำงานทันที แต่ถ้าส่ง 1 ไปให้กับบอร์ดไดร์มอเตอร์จะทำให้มอเตอร์นั้นจะทำงานตามค่าของ IN1 กับ IN2โดยการเขียนโปรแกรมนี้ทำให้บอร์ดไมโครคอนโทรลเลอร์มีการเชื่อมต่อกับบอร์ด Buffer เพื่อเพิ่มกระแสให้กับสัญญาณที่ส่งเข้ามาแล้วบอร์ด Buffer นั้นจะส่งเอาต์พุตไปเข้าที่บอร์ดไดร์มอเตอร์แบบ H-Bridge อีกทีนึงเพื่อทำการไดร์มอเตอร์ตามค่าอินพุต1 และอินพุต2 สั่งการควบคุมมา

### 3.7.2 การเขียนโปรแกรมเชื่อมต่อกับ Encoder

การเขียนโปรแกรมเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับ Encoder นั้นจะเป็นรูปแบบของการที่ไมโครคอนโทรลเลอร์รับค่าสัญญาณจาก Encoder แล้วนำค่าสัญญาณนั้นมาหาเป็นระยะทางที่หุ่นยนต์เคลื่อนที่กับนำค่าของสัญญาณนั้นมาเปลี่ยน แปลงรูปจำลองการทำงานของหุ่นยนต์ผ่านทางจอแสดงผล (GUI) โดยการเชื่อมต่อระหว่าง Encoder กับไมโครคอนโทรลเลอร์นั้นเราจะทำได้จากการนำสายสัญญาณเอาต์พุตของ Encoder ทั้ง 2 เส้นคือ เส้น A กับเส้น B นั้นทำการเชื่อมต่อกับพอร์ท 3.5 และพอร์ท 3.6 ซึ่งทั้ง 2 พอร์ท ในตระกูลไมโครคอนโทรลเลอร์ของ MCS-51 จะเป็นขา Timer0 และขา Timer1 โดยเราจะทำการเขียนโปรแกรมในรูปแบบการนับ(Counter)ค่าของลูกคลื่นของสัญญาณที่ขา A กับ B ซึ่งจะให้ไมโครคอนโทรลเลอร์แสดงผลออกมาเป็นจำนวนรอบของ Encoder ที่หมุนตามข้อต่อต่าง และนำยังมีส่วนการรับค่าของสัญญาณพัลส์มาคำนวณเพื่อนำมาใช้วัดระยะทางของหุ่นยนต์แล้วส่งกลับไปที่หน้าจอแสดงผลของผู้ควบคุม โดยผ่านทางRS232 wireless

## 3.8 การออกแบบและการเขียนโปรแกรมหน้าจอการแสดงผล GUI (*Graphical user interface)*

หลักการทำงานของ GUI (*Graphical user interface)* ในการควบคุมการทำงานของหุ่นยนต์เข้าไปในสถานที่ที่ไม่สามารถมองเห็นได้จะต้องมีตัวชี้ว่าหุ่นยนต์เราอยู่ในลักษณะใด เพื่อให้การบังคับหุ่นยนต์ได้ง่ายขึ้น ซึ่งประกอบด้วย 2 ส่วนคือ ภาพแสดงลักษณะการทำงานของหุ่นยนต์ และเชื่อมต่อกับผู้ใช้โดยตรงผ่านทางการสื่อสารอนุกรมและเป็นส่วนแสดงผลของการประมวลผลไมโครคอนโทรลเลอร์

ส่วนของการออกแบบ GUI ในส่วนของ ภาพแสดงลักษณะการทำงานของหุ่นยนต์จะใช้ ไลบลารี่ OpenGL ในการสร้างภาพเสมือนการเคลื่อนไหวของหุ่นยนต์ในลักษณะต่างๆ

### 3.8.1 การเขียนโปรแกรมแสดงผลหน้าจอผู้ใช้

ในการการออกแบบ GUI ในส่วนของการเชื่อมต่อระหว่างผู้ใช้กับส่วนแสดงผล วิธีการออกแบบ GUI โปรแกรมที่ใช้จะเป็นโปรแกรม Microsoft Visual Studio 2008 ซึ่งใช้ภาษา Visual Basic ในการพัฒนาโปรแกรม โดยการออกแบบ GUI จะแบ่งเป็น 2 ส่วน คือ

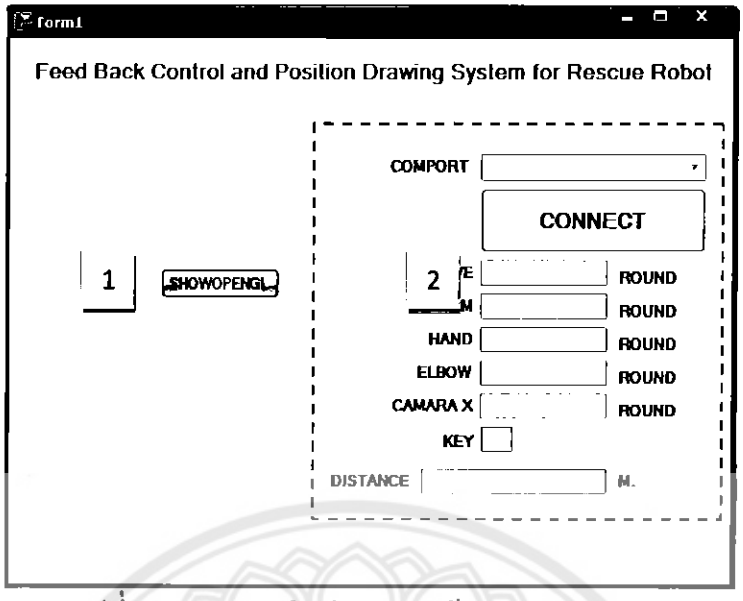
ส่วนที่ 1 ภาพแสดงลักษณะการทำงานของหุ่นยนต์โดยแสดงผลเป็นภาพในส่วนของการเคลื่อนไหวของหุ่นยนต์ในลักษณะต่างๆ

ส่วนที่ 2 เป็นการเชื่อมต่อกับผู้ใช้โดยตรงผ่านทางการสื่อสารอนุกรมและเป็นส่วนแสดงผลของการประมวลผลไมโครคอนโทรลเลอร์มีรายละเอียดดังนี้

- COMPORT ใช้เลือกช่องทางในการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ผ่านทางการสื่อสารแบบอนุกรมโดยมีปุ่มCONNECTไว้เพื่อให้เกิดการสื่อสารระหว่าไมโครคอนโทรลเลอร์กับหน้าจอแสดงผล (GUI)

- MOVE BOX , ARM BOX , HAND BOX , ELBOW BOX , CAMARAX BOX ในส่วนนี้เป็นการแสดงจำนวนรอบการหมุนของ Encoder ที่รับค่ามาจากไมโครคอนโทรลเลอร์ส่งมาทาง RS232 wireless

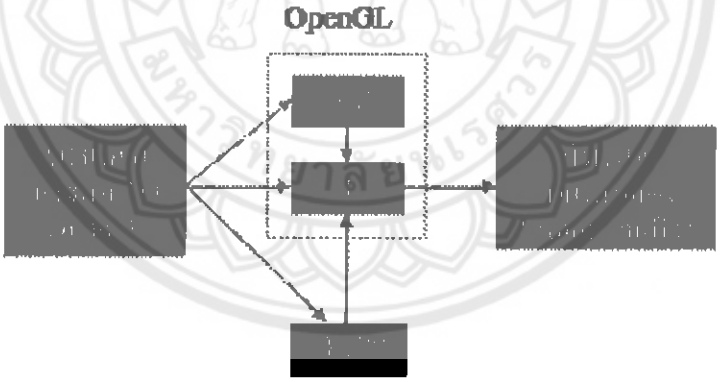
- DISTANCE BOX เป็นส่วนของการแสดงผลการวัดระยะทางของหุ่นยนต์โดยตัวเลขที่ออกมานั้นคือค่าระยะทางของหุ่นยนต์ที่วิ่งบนพื้นซึ่งเกิดจากการคำนวณตัวเลขของไมโครคอนโทรลเลอร์มีหน่วยวัดระยะทางเป็นเมตร




รูปที่ 3.33 แสดงรูปในส่วนของหน้าจอแสดงผล(GUI)

**3.8.2 การเขียนโปรแกรมแสดงผลภาพโดย OpenGL (Open source of Graphic Library)**

องค์ประกอบโดยรวมของ OpenGL จะประกอบด้วยไลบรารีกลุ่มใหญ่ๆอยู่ 3 กลุ่มสำหรับการเรียกใช้งานอันได้แก่ GL GLU และ GLUT




**หมายถึงคล่องคับทางเรียกใช้งานกลุ่มฟังก์ชันของคล่องที่ชี้**

**รูปที่ 3.34** แสดงโครงสร้างของOpenGL (Open source of Graphic Library)

จากรูปโครงสร้างของการใช้งาน OpenGL ในรูปที่ 1 ผู้ใช้งานจะเป็นผู้เขียนโปรแกรมประยุกต์ซึ่งโดยปกติคือภาษา CหรือC++ภายในโปรแกรมประยุกต์ที่สร้างขึ้นจะต้องรวมเอาไลบลารี่ของOpenGL เข้าร่วมไปด้วยเพื่อที่จะเรียกใช้งานได้ทางกราฟิกส์ การรวมเอาฟังก์ชันต่างๆของ OpenGL เพื่อใช้งานจะกระทำได้โดยเริ่มจากการรวมไฟล์ส่วนหัวคือ "GL/glut.h" เข้ากับโปรแกรมที่สร้างขึ้น โปรแกรมที่ได้รวมไฟล์ส่วนหัว glut.h นี้ จะสามารถเรียกใช้ฟังก์ชันต่างๆของ OpenGL ได้ ฟังก์ชันต่างๆที่เป็นของ OpenGL จะขึ้นต้น

ด้วยคำว่า "gl" ฟังก์ชันสำหรับ glu จะขึ้นด้วย "glu" และเช่นเดียวกันสำหรับ glut จากรูปของโครงสร้างแสดงให้เห็นว่าฟังก์ชันของ gl จะเป็นแกนหลักและอาจถูกเรียกใช้โดย glu หรือ glut

เป้าหมายสำหรับ OpenGL ในแบบปกติคือการเรนเดอร์วัตถุเพื่อให้เกิดภาพไปปรากฏยังบัพเฟอร์จอแสงผลเราเรียกบัพเฟอร์ในลักษณะนี้ว่า บัพเฟอร์จอแสดงผลที่จัดหาโดยวินโดวส์อย่างไรก็ตาม OpenGL ในปัจจุบันที่รองรับการทำงานระดับฮาร์แวร์ที่มากขึ้นซึ่งเราเรียกว่า ส่วนขยายของ OpenGL บัพเฟอร์จอแสดงผลที่มีอยู่เป็นบัพเฟอร์ที่เชื่อมตรงกับสถาปัตยกรรมตามแนวท่อของ OpenGL โดยตรง ทำให้การเรนเดอร์ออกได้โดยตรงไปสู่เฟรมบัพเฟอร์ซึ่งเราเรียกลักษณะนี้ว่าการเรนเดอร์แบบเชื่อมตรงในอีกแบบหนึ่งโปรแกรมประยุกต์หรือโปรแกรมเมอร์จะเป็นผู้สร้างเฟรมบัพเฟอร์ขึ้นมาใช้งานเองซึ่งเรียกว่าเฟรมบัพเฟอร์ที่จัดหาโดยโปรแกรมประยุกต์บัพเฟอร์ดังกล่าวไม่จำเป็นจะต้องออกจอแสดงผลโดยตรงเสมอการใช้งานเป็นไปในลักษณะของการปิดการเชื่อมตรงไปยังเฟรมบัพเฟอร์ปกติ การเรนเดอร์จะเก็บเป็นรูปภาพไว้ เพื่อที่จะนำเอาไปใช้ในการแสดงผลในคราวต่อไป

โมเดลต่างๆของ OpenGL (Open source of Graphic Library) ที่เหมาะสมกับการใช้งานบนวินโดวส์มีดังนี้

| | |
|---|---|
| GL | Graphic Library[3] |
| GLU | Graphic Library Utility[3] |
| GLUT | Graphic Library Utility Toolkit[1] |
| WGL | Graphic Library for Windows (WigGLe) |
| ARB | Architecture Review Board |

สำหรับ ARB จะเป็นฟังก์ชันในส่วนต่อขยายของ OpenGL ฟังก์ชันในส่วนนี้จะเกิดจากการตกลงกันระหว่างผู้สร้างกราฟิกส์ซึ่งผู้ผลิตเหล่านั้นจะมีส่วนในการกำหนดการทำงานของฟังก์ชันเพื่อให้ฟังก์ชันของ ARB เหมาะสมหรือเข้าได้กับฮาร์ดแวร์ สำหรับที่มาของฟังก์ชันที่ยอมรับโดยARB จะเริ่มจากฟังก์ชันในลักษณะของส่วนขยายก่อนและเมื่อได้รับการยอมรับ
บนอินเทอร์เน็ต มีผู้พัฒนาโปรแกรมอยู่เป็นจำนวนมาก ผู้พัฒนาดังกล่าวได้พัฒนารหัสโปรแกรมที่รองรับฟังก์ชันของ OpenGLและมีการเปิดให้ใช้งานในลักษณะของแหล่งโปรแกรมแบบเปิดส่วนใหญ่มักรองรับงานของOpenGL โดยการเพิ่มคุณลักษณะของส่วนติดต่อเชื่อมประสานกราฟิกส์กับผู้ใช้ให้กับ OpenGL เพื่อให้ OpenGL มีความน่าสนใจมากขึ้น

**3.8.2.1 การติดตั้ง OpenGL (Open source of Graphic Library) ใช้งานกับ VC++**

OpenGL จะเป็นการใช้งานในส่วนไลบลารี่ "gl" และ "glu" ซึ่งผู้ใช้จะต้องมี "opengl32.lib" อยู่ด้วย กับการแปลงโปรแกรมหลังจากนั้นโปรแกรมได้ถูกพัฒนาเป็นGLUTขึ้นมา ซึ่งจะรวมเอา ทั้ง "gl" และ "glu" เข้ากับ "glut" และการเรียกใช้งานไลบลารี่ทั้งสองถูกปรับให้สามารถเรียกผ่าน "glut" ได้โดยตรง ดังนั้นผู้ใช้จึงเพียงรวมไลบลารี่"glut"เข้ากับการแปลงไฟล์ก็จะสามารถใช้งานOpenGLได้เสมือนใช้งาน"gl" และ "glu" ร่วมอยู่โดยไม่จำเป็นต้องรวมเอา gl และ glu เข้าเป็นไฟล์ส่วนหัวโดยตรง ในการสร้างงานกราฟิกส์โดยการใช้ OpenGL ผู้ที่ต้องการใช้งานจะต้องเพิ่มไลบลารี่เข้ากับระบบของภาษา C ที่มีใช้อยู่เพื่อให้ใช้งานได้ สำหรับขั้นตอนการติดตั้งนั้น VC++ แต่ละรุ่นจะมีรายละเอียดของการติดตั้งที่ต่างกันเล็กน้อย ดังนั้นจึงขอยกตัวอย่างการติดตั้ง OpenGL บน VC++ หรือ VC .net framework

ไฟล์ที่ใช้งาน

glut32.dll, glut.h, glut32.lib

ไฟล์ดังกล่าวหาได้จาก OpenGL.org

สำหรับ .net framework 2005, 2008

เนื่องจากเริ่มใช้งาน OpenGL เบื้องต้นจาก VC ซึ่ง glut32.dll, glut.h และ glut32.lib จะได้จากการแปลงจาก VC++ สำหรับผู้ใช้ที่ใช้ VC++.net จำเป็นที่จะต้องทำการแปลงไฟล์ใหม่อีกครั้ง สำหรับตัวโปรแกรมต้นฉบับผู้ใช้สามารถหาได้จาก OpenGL.org หัวข้อ glut

**3.8.2.2 ขั้นตอนการติดตั้ง**

1. สำเนา glut32.dll ไปไว้ที่โฟลเดอร์ system32 ตัวอย่างเช่น c:\windows\system32\ โดยนำไปวางไว้ยังโฟลเดอร์นั้น

2. สำเนา glut32.lib ไปไว้ยังโฟลเดอร์ "lib" ของโปรแกรม VC ตัวอย่างเช่น

C:\Program Files\Microsoft Visual Studio\VC98\Lib // สำหรับ VC

C:\Program Files\Microsoft Visual Studio 9.0\VC\Lib // สำหรับ VC.net framework

3. สร้างโฟลเดอร์ "GL" ไว้ใต้โฟลเดอร์

C:\Program Files\Microsoft Visual Studio\VC98\Include // สำหรับ VC

C:\Program Files\Microsoft Visual Studio 9.0\VC\Include

- นำ glut.h ไว้ภายใต้โฟลเดอร์ GL ดังกล่าว

**3.8.2.2 โครงสร้างโปรแกรมกราฟิกส์โดยใช้ OpenGL (Open source of Graphic Library)**

ในการเขียนโปรแกรมกราฟิกส์จะมีลักษณะที่คล้ายๆกันสำหรับ OpenGL ในที่นี้จะใช้วิธีการเขียนโปรแกรมด้วยภาษา CหรือC++ รูปแบบของโปรแกรมที่ง่ายต่อการเขียนใช้งานจะเป็นในลักษณะของ

โปรแกรมแบบคอนโซลหรือโปรแกรมที่ยังอิงอยู่กับระบบปฏิบัติการของดอสโดยโปรแกรมจะแบ่งออกเป็นชนิดของฟังก์ชันต่างๆดังนี้

ฟังก์ชันที่ผู้เขียนสร้างขึ้นเพื่อให้ทำงานตามที่ผู้เขียนกำหนด บางฟังก์ชันที่สร้างขึ้นในลักษณะนี้จะเกี่ยวข้องกับการเขียนกราฟิกส์ ฟังก์ชันการทำงานที่สำคัญมีได้ดังนี้

main – เป็นฟังก์ชันหลักซึ่งเขียนโดยผู้เขียนโปรแกรมมักใช้กำหนดค่าเริ่มต้นต่างๆ และเป็นที่อยู่ของฟังก์ชัน และท้ายที่สุดจะต้องตามด้วยฟังก์ชัน glutMainloop(); เพื่อที่จะให้โปรแกรมกราฟิกส์มีการตรวจสอบการทำงานของฟังก์ชันเรียกกลับ

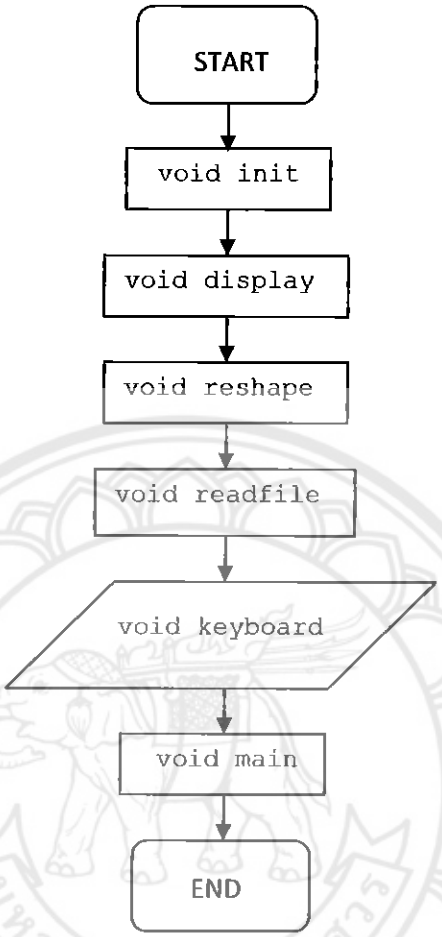
display – เป็นฟังก์ชันที่ใช้สำหรับการนิยามวัตถุ การแปลงวัตถุและมุมมองของกราฟิกส์ สำหรับการใช้ฟังก์ชัน เป็นฟังก์ชันเรียกกลับในฟังก์ชัน main โดยเรียกใช้ฟังก์ชันglutDisplayFunc(); เช่น glutDisplayFunc(display);

reshape – ฟังก์ชันที่เกี่ยวข้องกับการปรับปรุงหรือวินโดวส์มีการเปลี่ยนแปลงขนาดเนื่องจากผู้ใช้ขยายหรือหดหน้าต่างของ OpenGL และการฉายภาพจากสามมิติเป็นสองมิติ การเรียกใช้ฟังก์ชันด้วยฟังก์ชัน glutReshapeFunc(); ตัวอย่างเช่น glutReshapeFunc(reshape);

นอกจากนี้ยังมีฟังก์ชันเรียกกลับสำหรับการโต้ตอบกับผู้ใช้ เช่น glutMouseFunc() , glutKeyboardFunc() , glutIdleFunc() เป็นต้น ซึ่งสามารถหารดูรายละเอียดได้ที่คู่มือของ OpenGL

ฟังก์ชันเรียกกลับ (callback function) เป็นฟังก์ชันที่ใช้สำหรับเรียกกลับเป็นฟังก์ชัน ของ GLUT ซึ่งจะลงท้ายด้วยคำว่า "Func" เช่น glutReshapeFunc, glutDisplayFunc โดยที่พารามิเตอร์ของมันจะเป็นฟังก์ชันการทำงานที่สอดคล้องกับการทำงานของมันที่ควรเป็น ฟังก์ชันเรียกกลับนี้ใช้สำหรับตรวจจับเหตุการณ์ที่กำหนดใน OpenGL

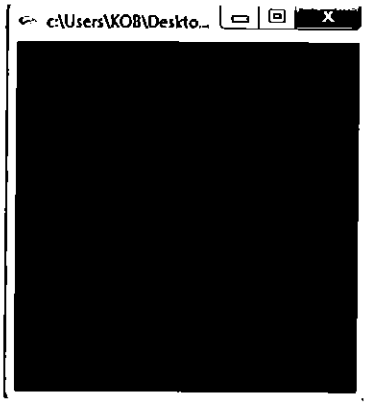
ฟังก์ชันกำหนดการวนรอบ ฟังก์ชัน glutMainLoop() สำหรับกราฟิกส์จะเป็นการทำงานที่จะต้องทำการเขียนซ้ำอยู่ตลอดเวลาเพื่อที่จะให้วัตถุปรากฏที่หน้าจอ ดังนั้นฟังก์ชัน glutMainLoop() จะทำหน้าที่ในการสร้างรอบวนดังกล่าว เสมือนกับว่าฟังก์ชันต่างๆถูกกำหนดอยู่ภายใต้รอบวนของ while(1) ซึ่งมันจะทำงานภายในรอบวนตลอดจนกว่าจะปิดโปรแกรม




**รูปที่ 3.35** แสดง Flow chart การทำงานของโปรแกรมสร้างภาพจำลองของหุ่นยนต์

จากรูปที่ 3.35 แสดง Flow chart การประมวลผลเพื่อให้ออกมาเป็นภาพการแสดงผลสถานะของหุ่นยนต์โดยการรับค่าจาก Text.file เข้ามาแล้วนำค่าที่ได้ไปผ่านกระบวนประมวลผลเพื่อให้เกิดการเคลื่อนไหวของรูปภาพสถานะของหุ่นยนต์ซึ่งจะอธิบายแต่ละขั้นตอนการทำงานตั้งแต่เริ่มต้นโดยจำลองการทำงานโดยการกดปุ่มเพื่อให้ภาพเกิดการเคลื่อนไหวในรูปแบบเสมือนการเคลื่อนไหวของหุ่นยนต์จริงก่อนที่จะรับค่าจาก Text.file

x.x.x. void init เป็นการเปิดการใช้3Dซึ่งใช้เก็บความลึกในแนวแกนZที่แต่ละพิกเซลของหน้าจอเพื่อเปรียบเทียบป้องกันการวาดวัตถุซ้อนทับกันผิดลำดับละเป็นการตั้งค่าสีพื้น ดังรูปที่ 3.36




**รูปที่3.36** แสดงการทำงานของ void init ที่ถูกตั้งให้พื้นหลังเป็นสีดำ

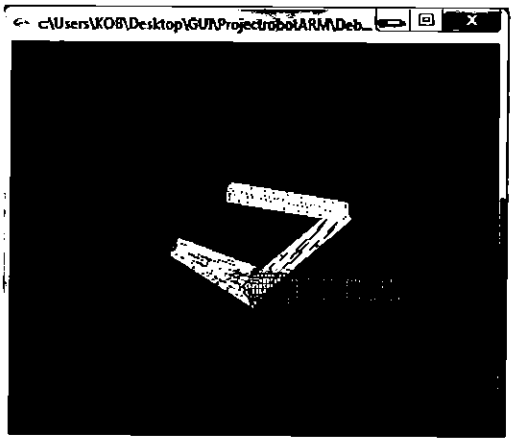
x.x.x.void display กระบวนการทำงานในการสร้างรูปออกมา ซึ่งประกอบด้วย สี ความยาวของรูป ลักษณะของรูป

x.x.x.void reshape เป็นฟังก์ชั่นกำหนดขนาดการแสดงผลออกทางจอภาพจะถูกเรียกใช้เมื่อหน้าต่างของ OpenGL ถูกปรับหรือถูกเปิดเป็นครั้งแรก

x.x.x.void readfile เป็นฟังก์ชั่นการอ่านค่าจาก Text.file ที่ได้จาก Hyper terminal ที่ถูกเก็บข้อมูลในรูปแบบ Text.file และนำค่าที่ได้ดังกล่าวไปกระทำให้รูปแสดงสถานะของหุ่นยนต์เกิดการเคลื่อนไหว

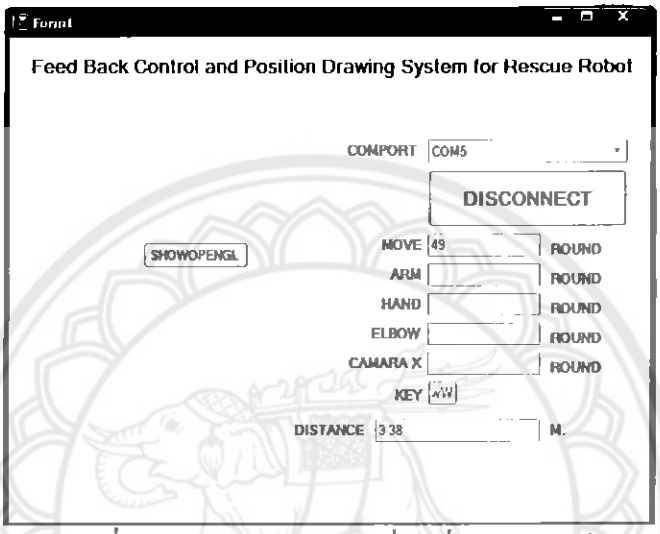
x.x.x. void keyboard เป็นฟังก์ชั่นทดสอบว่ารูปสถานะหุ่นยนต์ที่เกิดจากกระบวนการ void display สามารถเคลื่อนที่ได้หรือไม่ ซึ่งฟังก์ชันนี้ผู้ใช้จะนำไปประยุกต์เพื่อรับค่าจาก Text.file แล้วทำให้รูปสถานะหุ่นยนต์เกิดการเคลื่อนไหว

x.x.x.void main เป็นฟังก์ชันหลักในการประมวลโดยนำค่าต่างๆจากฟังก์ชั่นต่างที่ผู้ใช้กำหนดมาเพื่อที่จะแสดงผลในที่นี้จะแสดงผล ดังรูปที่ 3.37

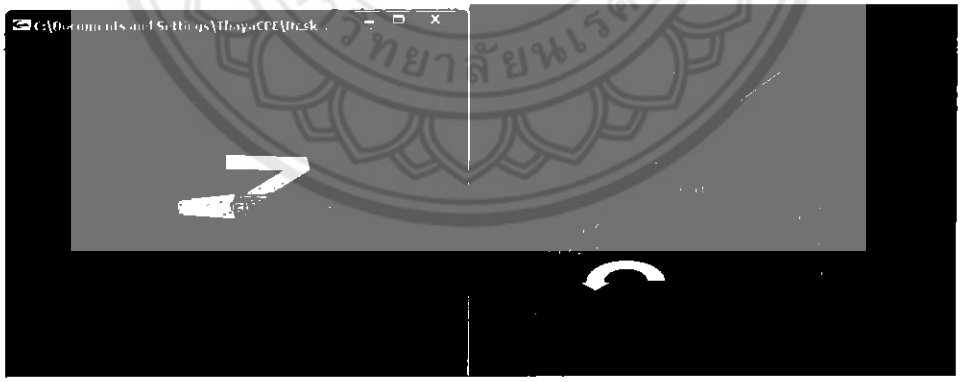



**รูปที่3.37** แสดงภาพจำลองของหุ่นยนต์กู้ภัย

จากการดำเนินงานตั้งแต่ต้นจนมาถึงขั้นตอนในการสร้างรูปจำลองของหุ่นยนต์เพื่อแสดงตำแหน่งเชิงกลนั้นก็จะมีวิธีการตามที่กล่าวมาข้างต้นแต่จะมีมาเพิ่มก็คือในส่วนของการวัดระยะทางของหุ่นยนต์ที่เคลื่อนที่จากจุดเริ่มต้นจนถึงจุดที่หุ่นยนต์อยู่ณ.ขณะนั้นด้วยการแสดงผลค่าระยะทางที่ได้ในช่องว่างที่ชื่อ Distance บนหน้าจอของผู้บังคับด้วยซึ่งสามารถดูได้จาก รูปที่3.38 และถ้าหุ่นยนต์ถูกบังคับรูปจำลองของหุ่นยนต์ก็จะแสดงออกทางหน้าจอแสดงผลก็สามารถที่จะเปลี่ยนตามผู้ควบคุมได้แล้วเช่นกันดูได้จากรูปที่3.39




รูปที่3.38 ค่าระยะทางการเคลื่อนที่ของหุ่นยนต์



(ก) (ข)

**รูปที่3.39** เป็นการเปรียบเทียบระหว่างรูปจำลองกับรูปหุ่นยนต์จริง

(ก) เป็นรูปจำลองหุ่นยนต์ของแขนล่างที่หมุน 180 องศา

(ข) เป็นรูปหุ่นยนต์ของแขนล่างที่หมุน 180 องศา

สังเกตการเปลี่ยนแปลงได้โดยรายละเอียดทั้งหมดในการทดสอบจะแสดงในดูในบทที่4 ต่อ

# บทที่ 4

# ผลการทดลอง

ในบทที่ 4 นี้จะกล่าวถึงการทดสอบการทำงานของโปรแกรมและทดสอบกับหุ่นยนต์กู้ภัยที่พัฒนาขึ้นมาว่าสามารถทำงานได้ตามจุดประสงค์ที่ได้ตั้งไว้อย่างไร สามารถสั่งการทำงานควบคุมการทำงานของมอเตอร์ตามข้อต่อต่างๆของหุ่นยนต์ได้และสามารถรับค่าจาก Encoder เพื่อนำค่าที่ได้มาประมวลทางรูปภาพของหุ่นยนต์เพื่อบอกสถานะและตำแหน่งตามข้อต่อต่างๆของหุ่นยนต์ได้ จากนั้นหาข้อผิดพลาดของโปรแกรมในส่วนต่างๆมาวิเคราะห์ถึงปัญหาที่เกิดขึ้น และหาแนวทางในการปรับปรุงแก้ไขข้อผิดพลาดที่เกิดขึ้น และแนวทางในการพัฒนาต่อไป

## 4.1 ขั้นตอนในการทดสอบหุ่นยนต์

ขั้นตอนการทดลองของหุ่นยนต์นั้นจะแบ่งเป็น 2 ประเภทคือ ทดลองโปรแกรมและทดลองหุ่นยนต์

### 4.1.1 การทดลองโปรแกรม

การทดลองโปรแกรมนั้นจะแบ่งออกเป็น 2 ส่วนด้วยกัน

4.1.1.1 การเขียนโปรแกรมในไมโครคอนโทรลเลอร์

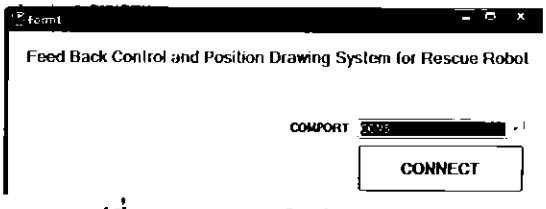
4.1.1.2 การเขียนโปรแกรมควบคุมและการติดต่อกับไมโครคอนโทรลเลอร์

### 4.1.2 การทดลองตัวหุ่นยนต์

ในการทดลองกับตัวของหุ่นยนต์นั้นจะเป็นการทดลองในส่วนของการควบคุมมอเตอร์ตามข้อต่อต่างๆของหุ่นยนต์กู้ภัย และในส่วนของการเคลื่อนที่ของหุ่นยนต์เช่น เดินหน้า ถอยหลัง เลี้ยวซ้าย เลี้ยวขวา ในพื้นราบเพื่อเป็นการทดสอบเกี่ยวกับโปรแกรมที่พัฒนาขึ้นมาเพื่อหาระยะทางการเคลื่อนที่ของหุ่นยนต์ด้วย

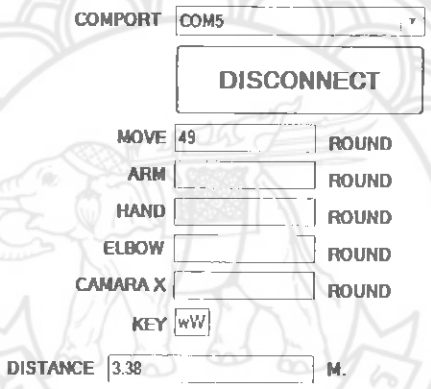
## 4.2 การทดสอบโปรแกรมที่พัฒนา

โปรแกรมการติดต่อและควบคุมหุ่นยนต์ เมื่อทำการเปิดโปรแกรมหน้าต่าง GUI ขึ้นมาแล้วจะเริ่มจากการเลือกใส่ค่าของCom Port ที่ใช้ในการติดต่อสื่อสารจากไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ เพื่อทำการควบคุมหุ่นยนต์จากการสั่งงานทางคอมพิวเตอร์ได้ โดยการเลือกค่า Com Port นั้นจะดูจาก Serial Port ของทางคอมพิวเตอร์ แล้วทำการกดเลือกค่า Com Port ที่เราต้องการตามด้วย ปุ่ม CONNECT ที่อยู่ข้างล่างช่อง Com Port เพื่อทำการเชื่อมต่อการสื่อสาร ตัวอย่างรูปที่ 4.1



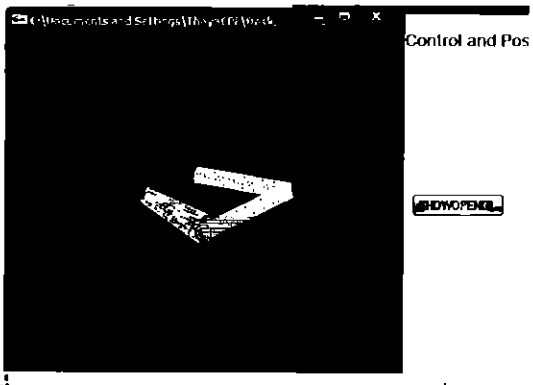

**รูปที่ 4.1** แสดงการติดต่อ Serial Port

ในส่วนต่อไปจะเป็นส่วนที่แสดงค่าตัวเลขการนับรอบซึ่งจะใช้ Encoder การหมุนรอบ และจะใช้ Encoder ในการวัดระยะทางแสดงออกเป็นหน่วยเมตร โดยจะทำการแสดงในช่อง DISTANCE

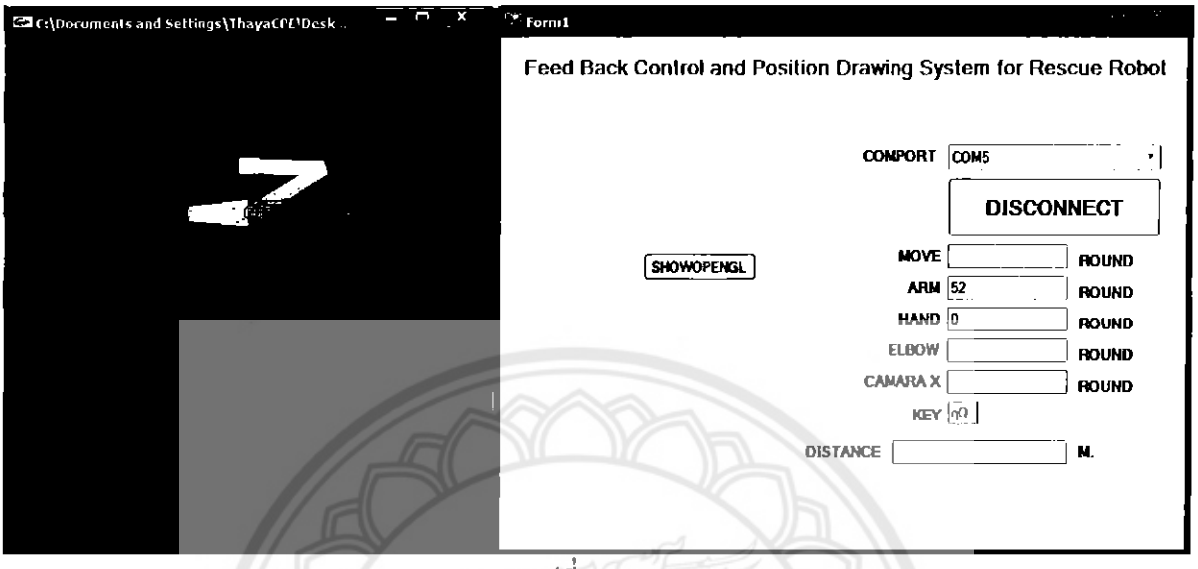



รูปที่ 4.2 แสดงการจำนวนรอบการหมุนของ Encoder และการวัดระยะทาง

ในส่วนการแสดงผลภาพจำลองของหุ่นยนต์นั้นจะทำจากการเรียกไฟล์ .EXE ของการทำภาพจำลอง นำมา link ร่วมกับหน้าจอควบคุม โดยจะทำการเรียกจากปุ่มกดมาใช้ โดยให้กดที่ปุ่ม Open GL




**รูปที่ 4.3** แสดงจากนำภาพจำลองของหุ่นยนต์มาแสดงผล




รูปที่ 4.4 (ก)

รูปที่ 4.4 (ข)

**รูปที่ 4.4** การทดสอบโปรแกรมผ่านหน้าจอแสดงผล

(ก) การแสดงผลโปรแกรมควบคุมและจำลองภาพหุ่นยนต์กู้ภัย

(ข) แสดงภาพหุ่นยนต์ที่ทำการทดลอง

## 4.3 การทดสอบตัวหุ่นยนต์

### 4.3.1 เดินหน้า – ถอยหลัง

การทดลองการเดินหน้าและถอยหลังของหุ่นยนต์ทำได้ตามที่เขียนโปรแกรมไว้

### 4.3.2 การเลี้ยวของหุ่นยนต์

การทดลองการเลี้ยวของหุ่นยนต์นั้นจะมี 2 แบบ คือ เลี้ยวแบบมุมกว้าง และเลี้ยวมุมแคบ

4.3.2.1 การเลี้ยวแบบมุมกว้าง คือการเลียวโดยใช้ล้อข้างเดียวส่วนอีกล้อจะหยุดนิ่ง เช่นจะเลี้ยวขวา ก็ให้ล้อขวาหมุนไปข้างหน้า ส่วนล้อซ้ายนั้นจะหยุดนิ่ง จะเป็นการเลี้ยวแบบมุมกว้าง

4.3.2.2 การเลี้ยวแบบมุมแคบ คือการเลียวโดยใช้ล้อทั้ง 2 ข้างหมุนไปในทางทิศตรงข้ามกัน เช่น เลี้ยวขวา ก็ให้ล้อหมุนไปข้างหน้า ส่วนล้อซ้ายจะหมุนไปข้างหลัง จะเป็นการเลี้ยวแบบมุมแคบ การเลียวแบบมุมแคนจะเลี้ยวได้เร็วกว่ามุมกว้าง

### 4.3.3 การเคลื่อนที่แขนล้อของหุ่นยนต์

หุ่นยนต์สามารถใช้แขนล้อยกตัวเองได้ และใช้แขนล้อช่วยพยุงตัวในตอนที่หุ่นยนต์เคลื่อนที่ในสภาพผิวที่ขรุขระได้เป็นอย่างดี แต่แขนของหุ่นยนต์นั้นทำงานช้ามาก เป็นเพราะอัตราการทดของเฟืองโซ่ เพราะจะได้แรงบิตที่สูงแต่จะแรกกับความเร็วที่เสียไป

### 4.3.4 การเคลื่อนที่แขนของหุ่นยนต์

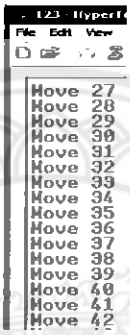
หุ่นยนต์สมมารถยกแขนได้ตามต้องการ แต่จะมีปัญหาตรงเมื่อหุ่นเคลื่อนที่จะเกิดการสั้นของแขน ทำให้แขนไม่นิ่ง จึงต้องทำการแก้ไขโดยการใส่ตัวซับแรงกระแทกเข้าไป ซึ่งดัดแปลงมาจากตัวกันสะบัดของรถจักรยานยนต์แล้วนำมาใส่เข้ากับหุ่น ทำให้แขนของหุนนั่นนิ่งมากขึ้น

## 4.4 การทดสอบ Encoder

โปรแกรมที่พัฒนาในส่วนของ Encoder นั้นหลักๆด้วยกันมี 2 อย่าง คือการรับค่าการนับรอบของ Encoder ที่จะทำโดยการนับค่าพัลส์สัญญาณของเอาต์พุตที่เกินมาจากการหมุนครบรอบของ Encoder โดยจะแสดงเป็นรอบของการหมุน และอีกส่วนเป็นการนำค่าจากการนับรอบนั้นมาคำนวณเพื่อทำการวัดระยะทางการเคลื่อนที่ของหุ่นยนต์

**4.4.1 การทดสอบการนับรอบของ Encoder**

โปรแกรมที่เราพัฒนานั้นได้มีการรับค่าสัญญาณเอาต์พุตเข้ามาของ Encoder ที่พอร์ท Timer0 และ Timer1 จากนั้นสัญญาณนั้นจะมีการส่งมาเป็นสัญญาณ ดิจิตอล ในระดับ TTL คือมี 0 กับ 1 เท่านั้น โดยโปรแกรมของเราจะเริ่มทำการเริ่มนับที่ขอบขาขึ้นครั้งแรกจากนั้นจะทำการนับที่ขอบขาลงของสัญญาณว่ามีการเปลี่ยนแปลงเป็นจำนวน 288 ค่าเมื่อไหร่ก็จะทำการคืนค่ากลับเป็น 1 รอบของการหมุนนั้นเอง




รูปที่ 4.5 จำนวนรอบที่แสดงออกมาจากการนับ แสดงผ่าน Hyperterminal

**4.4.2 การทดสอบการวัดระยะทาง**

Encoder จะใช้เป็นการตรวจสอบระยะทางและความเร็วของล้อทั้ง 2 ข้างขณะที่หุ่นยนต์เคลื่อนที่ ความละเอียดของ Encoder จะเท่ากับ 288 ในโปรแกรมจะนับทุกๆครั้งเมื่อ Encoder นับได้ 16 ดังนั้น 1 รอบของ Encoder ตัวไมโครคอนโทลเลอร์จะนับได้ $\frac{288}{16} = 18$ ครั้ง

การทดลองจะทำการทดลองในระยะต่างๆ เพื่อตรวจสอบดูว่า Encoder จะบอกระยะทางได้เท่ากันทุกครั้งหรือไม่ ผลการทดลองเป็นไปตามตารางที่ 4.1

**ตารางที่ 4.1 ผลการทดลองการวัดระยะทางโดยใช้ Encoder**

| ระยะทางจริง (cm.) | จำนวนพัลส์ของ Encoder | ระยะทาง / จำนวนพัลส์ |
|---|---|---|
| 50 | 7 | 7.14 |
| 100 | 15 | 6.67 |
| 150 | 23 | 6.52 |
| 200 | 30 | 6.67 |
| 250 | 37 | 7.14 |
| 300 | 42 | 7.00 |

| ระยะทางจริง (cm.) | จำนวนพัลส์ของ Encoder | ระยะทาง / จำนวนพัลส์ |
|---|---|---|
| 350 | 50 | 6.90 |
| 400 | 58 | 6.82 |
| 450 | 66 | 6.85 |
| 500 | 73 | 6.89 |
| 600 | 87 | 6.90 |
| 700 | 101 | 6.93 |
| 800 | 115 | 6.96 |
| 900 | 132 | 6.82 |
| 1000 | 147 | 6.80 |
| เฉลี่ย | | 6.87 |

จากตารางที่ 4.1 จากการหาค่าเฉลี่ยของ ระยะทาง / จำนวนพัลส์ ซึ่งจะได้ 6.87 ซึ่งแสดงว่า ใน 1 พัลส์ของ Encoder นั้นจะได้ระยะทางเท่ากับ 6.87 เซนติเมตร ซึ่งจะนำตัวเลขนี้ไปใส่ในโปรแกรมเพื่อที่จะสามารถรู้ได้ว่าหุ่นยนต์วิ่งไปได้ระยะทางเท่าไหร่

**4.4.3 การทดสอบภาพจำลองของหุ่นยนต์กู้ภัย**

การทดสอบการแสดงภาพจำลองหุ่นยนต์กู้ภัยนั้น จะต้องอาศัยการทำงานของโปรแกรมจากข้อ 4.4.1 โดยจะทำการนับรอบการหมุนของ Encoder เข้ามาจากนั้นนำค่าที่ได้จากการนับนั้นมาเปรียบเทียบกับการเปลี่ยนรูปร่างของหุ่นยนต์โดยที่เริ่มจากการทดสอบการหมุนภาพในส่วนของแขนหุ่นยนต์ตำแหน่งที่ 0 องศาไปจนถึง 180 องศา เพื่อที่เราจำทำการตั้งเงื่อนไขในการเปลี่ยนแปลงของภาพได้ โดยจะแสดงการทดลองในการนับค่าทั้งหมด 10 รอบ แล้วนำมาหาค่าเฉลี่ย ว่าต้องใช้การหมุนของ Encoder เท่าไหร่ในตารางที่ 4.2

**ตารางที่ 4.2 ผลการทดลองหาจำนวนรอบการหมุนของแขน**

| ครั้งที่ | จำนวนรอบ Encoder |
|---|---|
| 1 | 51 |
| 2 | 52 |
| 3 | 52 |

| ครั้งที่ | จำนวนรอบ Encoder |
|---|---|
| 4 | 53 |
| 5 | 51 |
| 6 | 51 |
| 7 | 52 |
| 8 | 51 |
| 9 | 52 |
| 10 | 51 |

จากตางรางที่ 4.2 เราจะนำค่าจากากรทดลองมารวมกันแล้วหารด้วย 10 เพื่อหาค่าเฉลี่ย ค่าเฉลี่ยที่เกิดจากการคำนวณได้เท่ากับ 51.6 จะประมาณโดยการปัดขึ้นเป็น 52 รอบ ต่อการเปลี่ยนรูปของแขนทำมุม 180 องศา

รูปที่ 4.6 ภาพการทำงานของแขนหุ่นยนต์จาก 0 องศา ถึง 180 องศา จากแกนX

ต่อไปเป็นการหาค่าเฉลี่ยของการนับรอบที่มีต่อการเปลี่ยนแปลงภาพจำลองในตำแหน่งของแขนบน โดยแขนบน นี้เราจะทำการหาค่าจำนวนรอบของ Encoder ที่ทำไปในมุมที่เริ่ม 20 องศา สิ้นสุดที่ 90 องศาของรูปโดยจะทำการทดลองเป็นจำนวน 10 ครั้งแล้วนำมาหาค่าเฉลี่ยโดยจะแสดงในตารางที่ 4.3

**ตารางที่ 4.3** ผลการทดลองหาจำนวนรอบการหมุนของแขน

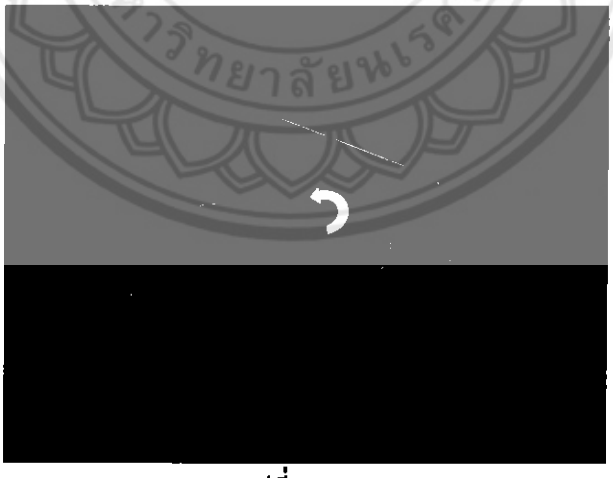
| ครั้งที่ | จำนวนรอบ Encoder |
|---|---|
| 1 | 72 |
| 2 | 73 |
| 3 | 72 |
| 4 | 70 |
| 5 | 72 |
| 6 | 71 |
| 7 | 72 |
| 8 | 71 |
| 9 | 72 |
| 10 | 72 |

จากตางรางที่ 4.3 เราจะนำค่าจากากรทดลองมารวมกันแล้วหารด้วย 10 เพื่อหาค่าเฉลี่ย ค่าเฉลี่ยที่เกิดจากการคำนวณได้เท่ากับ 71.7 จะประมาณโดยการปัดขึ้นเป็น 72 รอบ ต่อการเปลี่ยนรูปของแขนบนทำมุม 90 องศา

**รูปที่ 4.7** ภาพการทำงานของแขนบนหุ่นยนต์จาก 10 องศา ถึง 90 องศา ทำมุมกับแกนX

การทดลองสุดท้ายคือการทดลองการจำลองภาพของหุ่นยนต์กู้ภัยโดยการแสดงผลผ่านทางหน้าจอของผู้ควบคุมโดยการทดลองในส่วนนี้เราจะต้องเพิ่มโค้ดในส่วนของการติดต่อระหว่างโปรแกรมสร้างภาพจำลองของหุ่นยนต์กู้ภัยเข้ากับส่วนแสดงผลหน้าจอของผู้ควบคุมโดยจะเพิ่มโค้ดในส่วนของการรับไฟล์ .exe จากโปรแกรมจำลองภาพของหุ่นยนต์และเพิ่มโค้ดโปรแกรมการเขียนค่าตัวเลขในช่องต่างๆลงไฟล์ .txt เพื่อที่จำนำค่าต่างๆนั้นมาคำนวณในโปรแกรมภาพจำลองเพื่อทำให้ภาพจำลองหุ่นยนต์กู้ภัยนั้นเปลี่ยนค่าตามรอบการนับของ Encoder ได้ โดยเราจะทำการทดลองรูปจำลองด้วยกัน 3 แบบ โดยจะแสดงในรูปที่


รูปที่ 4.8 (ก)

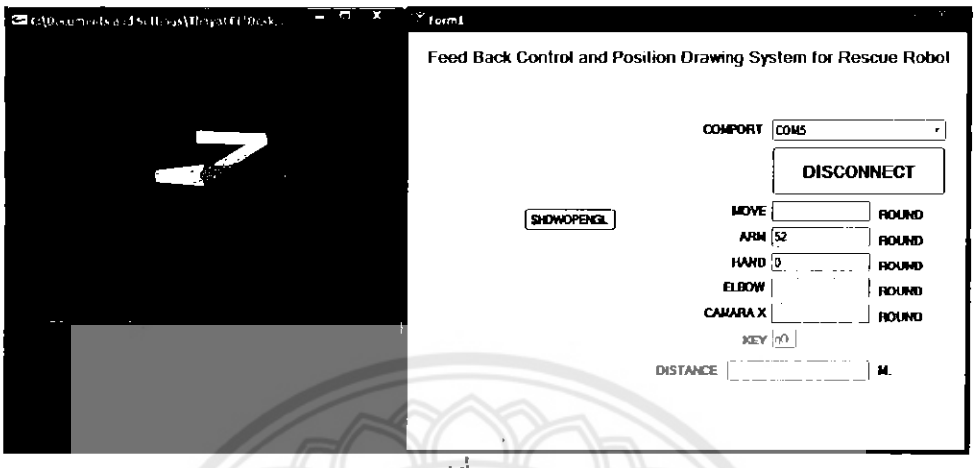


รูปที่ 4.8 (ข)

**รูปที่ 4.8 รูปเริ่มต้นก่อนมีการควบคุม**

**(ก) รูปภาพจำลองหุ่นยนต์ก่อนการทำงาน**

**(ข) รูปหุ่นยนต์ก่อนการทำงาน**



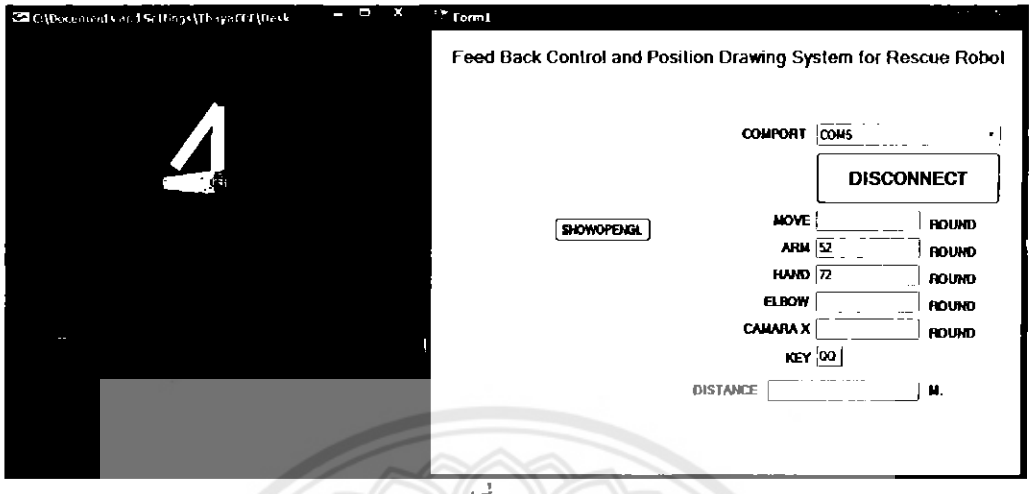

รูปที่ 4.9 (ก)

รูปที่ 4.9 (ข)

**รูปที่ 4.9** แสดงการเปลี่ยนแปลงของแขนจากการนับรอบ

(ก) รูปภาพจำลองหุ่นยนต์เมื่อมีการบังคับแขน

(ข) รูปหุ่นยนต์เมื่อมีการบังคับแขน




รูปที่ 4.10 (ก)

รูปที่ 4.10 (ข)

รูปที่ 4.10 แสดงการเปลี่ยนแปลงของแขนบนจากการนับรอบ

(ก) รูปภาพจำลองหุ่นยนต์เมื่อมีการบังคับแขนบน

(ข) รูปหุ่นยนต์เมื่อมีการบังคับแขนบน

จากการทดสอบการดำเนินงานการพัฒนาโปรแกรมที่ได้ทำมานั้นผลที่ได้รับจากการทดลองส่วนใหญ่แล้วเป็นไปตามที่เราได้คาดหมายไว้ว่าจะได้ผลลัพธ์ที่มีค่าตรงกับที่เราคำนวณส่วนใหญ่ผลลัพธ์ที่เกิดจากความคาดเคลื่อนนั้นจะมีสาเหตุที่เป็นไปได้หลายสาเหตุด้วยกันไม่ว่าจะเป็นการเสียดสีกันของล้อ การที่กำลังแบตเตอรี่อ่อน และอาจจะเป็นค่าการคาดเคลื่อนหรือองศาของมุมเริ่มนั้นไม่ตรงกับที่เราคำนวณไว้ซึ่งจะบอกโดยระเอียดต่อไปในบทที่ 5

# บทที่ 5
# บทสรุปและข้อเสนอแนะ

## 5.1 สรุปผลการทดลอง

1. สามารถตรวจสอบระยะทางของล้อทั้งสองข้างได้มีหน่วยเป็นเมตร ค่าความละเอียด 0.068 เมตร

2. สามารถควบคุมมอเตอร์ด้วยคอมพิวเตอร์ผ่านชุดไมโครคอนโทรลเลอร์ MCS51 แบบไร้สายได้ โดยระยะทางระหว่างผู้ควบคุมกับตัวหุ่นยนต์นั้นจะมีระยะไกลได้ไม่เกิน 180 เมตร ในพื้นที่โล่ง

3. สามารถเปลี่ยนแปลงภาพหุ่นยนต์กู้ภัยจากการรับค่าเป็นแบบจำนวนรอบการหมุนของ Encoder ได้

## 5.2 ปัญหาที่พบ

1. อุปกรณ์ที่ทำชุดขับเคลื่อนช่วงล่างของหุ่นยนต์ยังไม่ค่อยได้มาตรฐานเท่าที่ควร

2. การออกแบบข้อต่อในส่วนต่างๆของหุ่นยนต์กู้ภัยนั้นยังขาดพวกวัสดุที่นำมาทำและยังขาดความรู้เรื่องการออกแบบต่างๆ

3. ปัญหาเรื่องการติดตั้ง Encoder นั้นเนื่องจาก Encoder ที่ซื้อมานั้นมีขนาดใหญ่และติดตั้งยาก

4. แบตเตอรี่ในส่วนของแขนท่อนบนหมดเร็วเนื่องจากแขนท่อนบนรับน้ำหนักเยอะเกินไป

5. น้ำหนักของแขนท่อนบนมีน้ำหนักมากเกินไป

6. สัญญาณของ Encoder เกินการชนกันจึงทำให้ไมโครคอนโทรลเลอร์ประมวลผลไม่ได้

7. พอร์ท Timer ของไมโครคอนโทรลเลอร์มีน้อยเกินไป

8. Timer ของโปรแกรมในส่วนแสดงผลมีค่าที่ไม่แน่นอน

## 5.3 แนวทางการแก้ปัญหาและข้อเสนอแนะ

1. สำคัญที่สุดคือการออกแบบหุ่นยนต์และอุปกรณ์ที่ใช้ทำหุ่นยนต์ เราควรเลือกวัสดุอุปกรณ์ที่นำมาทำหุ่นยนต์นั้นด้วยความรอบคอบและคำนึงถึงผลกระทบที่จะตามมาภายหลังจากการประกอบหรือติดตั้งอุปกรณ์นั้นๆลงไปด้วย

2. ปัญหาการติดตั้ง Encoder เราควรเปลี่ยนมาใช้มอเตอร์แบบที่มี Encoder ติดอยู่ท้ายมอเตอร์เลยโดยตรงเพื่อความสะดวกในการใช้และการติดตั้ง

3. ควรเลือกไมโครคอนโทรลเลอร์ในการทำงานให้เหมาะสมกับงานที่ทำว่ามีพอร์ท ที่ต้องการใช้นั้นครบหรือเพียงพอต่องานที่ทำหรือไม่

4. การติดต่อสื่อสารควรมีการหาอุปกรณ์ที่ใช้ในการติดต่อสื่อสารระหว่างผู้ใช้กับหุ่นยนต์ในมีความไกลเพิ่มขึ้น โดยอาจจะใช้ wireless ในการติดต่อสื่อสารเพื่อความไกลของสัญญาณมากขึ้น

## 5.4 แนวทางในการพัฒนาเพิ่มเติม

### 5.4.1 แนวทางการพัฒนาในส่วนของตัวหุ่นยนต์

1. พัฒนาเกี่ยวกับชุดขับเคลื่อนและมอเตอร์ที่ใช้ในการขับเคลื่อนของหุ่นยนต์กู้ภัย เช่นมอเตอร์ที่มี Encoder ติดท้ายมอเตอร์ให้เลย

2. ติดตั้ง Encoder ให้กับมอเตอร์ทุกจุด เพื่อที่จะสามารถบอกลักษณะของหุ่นยนต์ได้อย่างชัดเจน ซึ่งจะทำให้ตอนบังคับรู้ได้ว่าหุ่นยนต์อยู่ในลักษณะใดและทั้งแกน x และ y ของการหมุน

### 5.4.2 แนวทางการพัฒนาในส่วนของการควบคุมและการเชื่อมต่อ

1. การพัฒนาระบบควบคุม Microcontroller เป็นประเภทอื่นหรือตระกูลอื่นที่มีความเร็วมากขึ้น เช่น ARM7, AVR, PIC

2. เปลี่ยนอุปกรณ์การสื่อสารแบบไร้สาย เพื่อให้ควบคุมได้ไกลมากยิ่งขึ้น เช่น การใช้เทคโนโลยี 3G (Third Generation) เข้ามาควบคุมหุ่นยนต์ซึ่งจะบังคับได้ไกลมาก สามารถบังคับที่ไหนก็ได้ที่มีสัญญาณโทรศัพท์

### 5.4.3 แนวทางในการพัฒนาในส่วนของซอฟแวร์

1. การพัฒนาจากการใช้คอมพิวเตอร์ควบคุม เป็นการใช้โทรศัพท์มือถือหรืออุปกรณ์อื่นที่มีขนาดพกพาควบคุมหุ่นยนต์แบบไร้สาย

2. การใช้ระบบ AI เข้ามาช่วยวิเคราะห์ เช่นวิเคราะห์ว่าเป็นสิ่งมีชีวิตหรือไม่ และใช้ AI เข้ามาช่วยในการควบคุมหุ่นยนต์เช่น ข้างหลังเป็นทางชันหรือมีสิ่งกีดขวางอยู่ไม่สามารถไปได้ เป็นต้น

3. การแสดงแผนที่ ตำแหน่งและเส้นทางที่หุ่นยนต์เคลื่อนที่พบผู้ประสบภัย

4. การพัฒนาเรื่องการแสดงภาพจากกล้อง IP โดยผ่าน IP server แล้วส่งสัญญาณแบบ wireless เพื่อนำภาพจากกล้องมาทำเป็นภาพแบบ พาโนรามา โดยใช้กล้อง 2 ตัวมาทำ เพื่อความกว้างในการมองเห็นเพื่อขึ้น

5. การพัฒนาโดยติด Sensor IR กับหุ่นยนต์เพื่อตรวจสอบส่งกรีดข้างทั้งรอบด้านของหุ่นยนต์โดยมีการคืนค่ากลับเป็นระยะทางจากหุ่นยนต์ถึงวัตถุเพื่อส่งผลดีต่อการบังคับ

# เอกสารอ้างอิง


[1] รศ.ธีรวัฒน์ ประกอบผล: **การประยุกต์ใช้งานไมโครคอนโทรลเลอร์**. พิมพ์ครั้งที่ 8. กรุงเทพ ฯ: บริษัท ดวงกมลสมัย จำกัด. พ.ศ. 2547

[2] อุดม จีนประดับ. **ไมโครคอนโทลเลอร์ MCS-51**. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร: ศูนย์ตำรา สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ 2541.

[3] อ.วิทวัส วิทยชำนาญกุล . **การสร้างภาพสามมิติโดยใช้ OpenGL**. การใช้ OpenGL เบื้องต้น. สืบค้นเมื่อวันที่ 10 ธันวาคม 2553, http://202.183.233.73/SMCsdown/2549-2/Selected%20Topic%20Advance%20Computer%20Graphic/Data/Book/OpenGL.pdf

[4] กฤติกร สุขศิริพงศ์วาสี. ระบบเซอร์โวและการบำรุงรักษาเครื่องจักรซีเอ็นซี. อุปกรณ์ที่เกี่ยวกับ Encoder. สืบค้นเมื่อวันที่ 9 ธันวาคม 2553, http://www.bpcd.net/machine/ CNC_training /CNC_SDI.pdf

[5] eestudy มหาวิทยาลัยขอนแก่น. **Motion Control Hardware**. Encoder control. สืบค้นเมื่อวันที่ 12 ธันวาคม 2553, http://eestud.kku.ac.th/~u4316562/hardware.html#

[6] กิตินันท์ พลสวัสดิ์. เริ่มต้น VisualBasic 2008 ฉบับโปรแกรมเมอร์. พิมพ์ครั้งที่ 1. นนทบุรี: ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์ จำกัด. พ.ศ.2552

[7] บัญชา ปะสีละเตสัง. พัฒนาแอปพลิเคชันด้วย VisualBasic2008. พิมพ์ครั้งที่ 1. กรุงเทพ: ซีเอ็ดยูเคชั่น บมจ. พ.ศ.2552

# ภาคผนวก

## โปรแกรมส่วนควบคุมระดับล่าง(Microcontroller)

```
// Project Feed Back Control and Position Drawing System for Rescue
Robot
// Thaya Thermlarp 49371033
// C Keil
// 11/03/53
// Naresuan University

#include <reg52.h>
#include <stdio.h>
#include <intrins.h>
//------------------- Function ----------------------
int cmdRS;
sbit C1 = P3^4 ;
sbit C0     = P3^5 ;
void dusec (unsigned int count);
void start232 (void);
void startinterupt(void);
void serial_ISR(void);
void serial_2(void);
void serial_ISR_Job(void);
int roundE10,roundE20,roundE21,roundE30,roundE31;
double distance ;
//-----------------------------------------
void start232 (void)
{
      SCON = 0x52;       // set RS232 parameter
    TMOD = 0x55;
      RCAP2H =0xFF;      // Set Baud rate to 9600 bps
      RCAP2L = 0xDC;     // Set Baud rate to 9600 bps and 0xDC =
Crystal = 11.059200 MHz
      T2CON = 0x34;      // Set Timer2 as Baudrate Generate

      //-------------------------
      TH0 = TH1 = 0xff;
      TL0 = TL1 = 0x88;

      TR1 = 1;
      TR0 = 1;
}
void startinterupt(void)              //Founction Interrupt Enable
register
{
      EA = 1;                                          //add Interrupt
Enable
      ES = 1;

      ET0 = 1;
      ET1 = 1;
      //Interrupt priority register
      PT1 = 1;      //Timer 1
      PT0 = 1;      //Timer 0
      PS = 0;             //serial port
      //*************************

}
/*************** Encoder ***************/
int count0 = 0,count1 = 0;
```

```
int pls,T_2;
//************************** COUNTER ENCODER TIMER 1
*****************************
void Timer1_Handler(void) interrupt 3 //Right
{
      TF1 = 0;

       if(cmdRS == 'g')         // MOVE ARM UP
      {
            count0--;
            roundE21 = count0 ;
            printf("A %D \n\r",roundE21) ;
      }
      else if(cmdRS == 'h')         //     MOVE ARM Down
      {
            count0++;
            roundE21 = count0 ;
            printf("A %D \n\r",roundE21) ;
      }
       else if(cmdRS == 'z')       //MOVE Hand Up
      {
            count0++;
            roundE31 = count0 ;
            printf("H %D \n\r",roundE31) ;
      }
      else if(cmdRS == 'x')          //MOVE Hand Down
      {
            count0--;
            roundE31 = count0 ;
            printf("H %D \n\r",roundE31) ;
      }

      TH1 = 0xff;
      TL1 = 0x88;
      TR1 = 1;

}
//************************ COUNTER ENCODER TIMER 0
*******************************
void Timer0_Handler(void) interrupt 1 //Left
{

      TF0 = 0;
      if(cmdRS == 'w')          // MOVE Up
      {
            count1++;
            roundE10 = count1 ;
            printf("M %D \n\r",roundE10) ;
            distance = (roundE10 * 6.9)/ 100;
            printf("L %3.2fM\n\r",distance);
      }
      else if(cmdRS == 's')          // MOVE Down
      {
            count1--;
            roundE10 = count1 ;
            printf("M %D \n\r",roundE10) ;
            distance = (roundE10 * 6.9)/ 100;
            printf("L %3.2fM\n\r",distance);
      }
      else if(cmdRS == 'g')          // MOVE ARM UP
      {
```

```
                count1--;
                roundE20 = count1;
                printf("A %D \n\r",roundE20) ;
        }
        else if(cmdRS == 'h')    // MOVE ARM DOWN
        {
                count1++;
                roundE20 = count1;
                printf("A %D \n\r",roundE20) ;
        }
        else if(cmdRS == 'z')    // MOVE HAND UP
        {
                count1++;
                roundE30 = count1;
                printf("H %D \n\r",roundE30) ;
        }
        else if(cmdRS == 'x')     // MOVE HAND DOWN
        {
                count1--;
                roundE30 = count1;
                printf("H %D \n\r",roundE30) ;
        }

        TH0 = 0xff;
        TL0 = 0x88;
        TR0 = 1;
}
//--------- Delay Control -----------
void dusec (unsigned int count) {  // mSec Delay
    unsigned int i;                // Keil CA51 (x2)
    while (count)
      {
                serial_2();
                if(cmdRS == '0')
                {
                        break;
                }
                //else{
                i = 100;
                        while (i>0)
                        i--;
                        count--;
                //}
    }
}
void serial_ISR(void) interrupt 4
{
      if(RI)
      {
                cmdRS = SBUF;
                //---------------------------------------
                serial_ISR_Job();
                dusec(100000);
                //---------------------------------------
                RI = 0;
      }
}
//-----------------------------------------
void serial_2(void)
{
      if(RI)
```

```
        {
                cmdRS = SBUF;
                //----------------------------------------
                serial_ISR_Job();
                //----------------------------------------
                RI = 0;
        }
}
void serial_ISR_Job(void)
{
        if(cmdRS == 'w')          //Forword
        {
        P2 = 0x63;
        }
        else if(cmdRS == 's')     //Backword
        {
        P2 = 0x36;
        }
        else if(cmdRS == 'a')     //Turn Right
        {
        P2 = 0x66;
        }
        else if(cmdRS == 'd')     //Turn Right
        {
        P2 = 0x33;
        }
////////////////////////////////////////////
        else if(cmdRS == 'z')     //Hand Up
        {
        P1 = 0x60;
        }
        else if(cmdRS == 'x')     //Hand down
        {
        P1 = 0x30;
        }
        else if(cmdRS == 'c')     //Elbow Up
        {
        P1 = 0x03;
        }
        else if(cmdRS == 'v')     //Elbow Down
        {
        P1 = 0x06;
        }
////////////////////////////////
        else if(cmdRS == 'g')     //ARM Up
        {
        P0 = 0x60;
        }
        else if(cmdRS == 'h')     //ARM Down
        {
        P0 = 0x30;
        }
        else if(cmdRS == 'j')     //Camera Up
        {
        P0 = 0x06;
        }
        else if(cmdRS == 'k')     //Camera Down
        {
        P0 = 0x03;
        }
////////////////////////////////
```

```
        else if(cmdRS == 'p')    //Reset Counter
        {
        count0 = 0;    // Reset Counter0
              count1 = 0;    // Reset Counter1
        }
        else
        {
              P2 = 0x00;
              P1 = 0x00;
        P0 = 0x00;
        }
 }
/*************** Main Function *****************/
void main(void)
{
        TH0 = 0xff;
        TL0 = 0x88;

        TH1 = 0xff;
        TL1 = 0x88;

        start232();
        startinterupt();
        printf ("Microcontroller OK \n\r");
        C1 = 1;
        C0 = 1;
        count0 = 0;

        while(1)
        {
        }
}
```

**โปรแกรมหน้าจอแสดงผลพัฒนาโดยภาษา visual basic**

```
Imports System.IO
'Option Explicit On

Public Class Form1
    Inherits System.Windows.Forms.Form
    Private mIniFile As w2IniFile
    Private mPort As New IO.Ports.SerialPort
    Friend WithEvents lstData As System.Windows.Forms.ListBox
    Dim data As String
    Dim send As String
    Dim a() As String
    Dim b As String
    Dim myProcID As Long


    Private Declare Function OpenProcess Lib "Kernel32" (ByVal _
    dwDesiredAccess As Long, ByVal bInheritHandle As Long, _
    ByVal dwProcessId As Long) As Long

    Declare Sub Sleep Lib "kernel32" Alias "Sleep" (ByVal
dwMilliseconds As Long)

    Private Declare Function TerminateProcess Lib "Kernel32" _
    (ByVal hProcess As Long, ByVal uExitCode As Long) As Long

    Private Declare Function CloseHandle Lib "Kernel32" _
    (ByVal hObject As Long)
```

```
    Private Const PROCESS_TERMINATE As Long = &H1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        Dim s As String

        'ini file
        mIniFile = New
w2IniFile(IO.Path.Combine(System.AppDomain.CurrentDomain.BaseDirector
y(), "wterm.ini"))

        'set port properties
        PORTCOM.DropDownStyle = ComboBoxStyle.DropDownList
        For Each s In My.Computer.Ports.SerialPortNames
            PORTCOM.Items.Add(s)
        Next
        s = mIniFile.ReadParameter("Port", "Port", "")
        PORTCOM.SelectedIndex = PORTCOM.Items.IndexOf(s)
        Timer1.Interval = 18
        Timer2.Interval = 5

    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        'Timer1.Enabled = False

        data = SerialPort1.ReadExisting
        a = data.Split(" ")
        If a(0) = "M" Then
            MOVEBOX.Text = a(1)
        ElseIf a(0) = "A" Then
            ARMBOX.Text = a(1).ToString
            Dim f1 As StreamWriter = File.CreateText("D:\TestA.txt")
            f1.Close()
            Dim sw As StreamWriter = File.AppendText("D:\TestA.txt")
            'Sleep(2)
            Dim Strr As String
            Strr = a(1)
            sw.Write(Strr)
            sw.Close()
        ElseIf a(0) = "H" Then
            HANDBOX.Text = a(1)
            Dim f2 As StreamWriter = File.CreateText("D:\TestH.txt")
            f2.Close()
            Dim sw As StreamWriter = File.AppendText("D:\TestH.txt")
            Dim Strr As String
            Strr = a(1)
            sw.Write(Strr)
            sw.Close()
        ElseIf a(0) = "E" Then
            ELBOWBOX.Text = a(1)
        ElseIf a(0) = "C" Then
            CAMARAXBOX.Text = a(1)
        ElseIf a(0) = "L" Then
            DISTANCEBOX.Text = a(1)
        End If
        SerialPort1.Write(TbxKeyCode.Text)
```

```
        '  myProcID = Shell("C:\Documents and
Settings\ThayaCPE\Desktop\GUIROBOT\ProjectrobotARM\Debug\Projectrobot
ARM.exe", AppWinStyle.Hide)

        '  Kill all notepad process
        '  Dim pProcess() As Process =
System.Diagnostics.Process.GetProcessesByName("ProjectrobotARM")

        '  For Each p As Process In pProcess
        '  p.Kill()
        '  Next
        Timer1.Enabled = True
    End Sub

    Private Sub CONNECT_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles CONNECT.Click
        If CONNECT.Text = "CONNECT" Then
            With SerialPort1
                .PortName = PORTCOM.Text
                .BaudRate = 9600
                .Parity = IO.Ports.Parity.None
                .DataBits = 8
                .StopBits = IO.Ports.StopBits.One
            End With
            SerialPort1.Open()
            Timer1.Start()
            Timer2.Start()
            CONNECT.Text = "DISCONNECT"
        Else
            SerialPort1.Close()
            CONNECT.Text = "CONNECT"
            Timer1.Stop()
            Timer2.Stop()
        End If
    End Sub
    Private Sub Form1_Load() Handles Me.Load
        Me.KeyPreview = True
    End Sub
    Private Sub Form1_KeyDown(ByVal sender As Object, ByVal e As
KeyEventArgs) Handles Me.KeyDown, HANDBOX.KeyDown, ARMBOX.KeyDown
        TbxKeyCode.Text = e.KeyCode.ToString()

    End Sub

    Private Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer2.Tick
        If TbxKeyCode.Text = "g" Or TbxKeyCode.Text = "G" Then
            SerialPort1.Write("g")
        ElseIf TbxKeyCode.Text = "q" Or TbxKeyCode.Text = "Q" Then
            SerialPort1.Write("q")
        ElseIf TbxKeyCode.Text = "h" Or TbxKeyCode.Text = "H" Then
            SerialPort1.Write("h")
        ElseIf TbxKeyCode.Text = "w" Or TbxKeyCode.Text = "W" Then
            SerialPort1.Write("w")
        ElseIf TbxKeyCode.Text = "s" Or TbxKeyCode.Text = "S" Then
            SerialPort1.Write("s")
        ElseIf TbxKeyCode.Text = "a" Or TbxKeyCode.Text = "A" Then
            SerialPort1.Write("a")
        ElseIf TbxKeyCode.Text = "d" Or TbxKeyCode.Text = "D" Then
            SerialPort1.Write("d")
        ElseIf TbxKeyCode.Text = "z" Or TbxKeyCode.Text = "Z" Then
```

```
            SerialPort1.Write("z")
        ElseIf TbxKeyCode.Text = "x" Or TbxKeyCode.Text = "X" Then
            SerialPort1.Write("x")
        ElseIf TbxKeyCode.Text = "c" Or TbxKeyCode.Text = "C" Then
            SerialPort1.Write("c")
        ElseIf TbxKeyCode.Text = "v" Or TbxKeyCode.Text = "V" Then
            SerialPort1.Write("v")
        ElseIf TbxKeyCode.Text = "j" Or TbxKeyCode.Text = "J" Then
            SerialPort1.Write("j")
        ElseIf TbxKeyCode.Text = "k" Or TbxKeyCode.Text = "K" Then
            SerialPort1.Write("k")
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        myProcID = Shell("C:\Documents and
Settings\ThayaCPE\Desktop\TestARM2\testArm2\Debug\testArm2.exe",
AppWinStyle.NormalFocus)
    End Sub
End Class
```

**ส่วนสร้างภาพจำลองหุ่นยนต์กู้ภัย ใช้ Library OpenGL**

```
#include <windows.h>
#include <GL/glut.h>
#include "stdio.h"
#include "stdlib.h"
#include <conio.h>
#include <iostream>
#include <fstream>

#define SIZEW 50.0 // กำหนดขอบเขตของวัตถุ ในฟังชันก์  void reshape
GLUquadricObj *theObj; // ประกาศตัวแปร quadric ชื่อ theobj
int i,arm,hand,shoulder1,elbow,arm2;

void init(void)
{
   glEnable(GL_DEPTH_TEST); // เปิด Depth Buffer ใช้เก็บความลึกz ที่แต่ละพิกเซลของหน้าจอเพื่อ
เปรียบเทียบป้องกันการวาดวัตถุซ้อนทับกันผิดลำดับ (Depth คือ เปิดหรือปิดการเขียนไปยังบัฟเฟอร์ความลึก)
   glClearColor(0.0, 0.0, 0.0, 0.0); // ตั้งค่าสีพื้นหลัง
   theObj = gluNewQuadric();// สร้าง Quadric ให้ theObj
}
void display(void)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //เคลียร์ค่าในบัฟเฟอร์
สี ซึ่งเก็บรูปที่แสดงออกทางหน้าจอให้เป็นสีพื้นหลังตามที่กำหนด และเคลียร์ค่าใน Depth Buffer
    glPushMatrix();//เป็นคำสั่งที่ใช้วาดรูปสองหรือสามมิติ     คำสั่งนี้ไว้ใช้สำหรับคัดลอกแมตริกทรานฟอร์ม หรือเก็บไว้ใน
Stack
      gluQuadricDrawStyle(theObj,GLU_LINE); // กำหนดลักษณะการวาดรูปทรงของ Obj
    glRotatef(75.0, 0.0, 1.0, 0.0); //มุมมอง
      ////////////////////// Shoulder1 ////////////////////////////
      glRotatef(shoulder1, 1.0, 0.0, 0.0); //ระนาบหมุน ฟ้า
    glPushMatrix();
    glTranslatef(5.0, 0.0, 0.0); //ตำแหน่ง
      ////////////////////// ARM ////////////////////////////////
    glRotatef(arm, 1.0, 0.0, 0.0); // การหมุนระนาบหมุน เหลือง
```

```
    glColor3f(1.0, 1.0, 0.0);//สีเหลือง
      gluCylinder(theObj, 4.0, 2.0, 20.0, 20, 50); //ความใหญ่ยาว
    glPopMatrix(); //สำหรับดึงเมตริกที่เก็บไว้ออกมาจาก stack
    ////////////////////// HAND ////////////////////////////////
      glColor3f(0.0, 1.0, 1.0);// สีเขียวน้ำเงิน
    gluCylinder(theObj, 4.0, 4.0, 30.0, 20, 40); //ความใหญ่ยาว
      glRotatef(hand, 1.0, 0.0, 0.0); //ระนาบหมุน ฟ้า
    glPushMatrix();
      ////////
    glTranslatef(0.0, 0.0, 25.0); //ตำแหน่ง
    glRotatef(220.0, 1.0, 0.0, 0.0); //ระนาบหมุน เหลือง
    glColor3f(1.0, 1.0, 0.0);//สีเหลือง
      glRotatef(arm2, 1.0, 0.0, 0.0); //ระนาบหมุน เหลือง
    glColor3f(1.0, 1.0, 1.0);//สีขาว

    ////////////////////// เขียนขนาดของรูป //////////////////////
      gluCylinder(theObj, 2.0, 2.0, 25.0, 20, 70); //ความใหญ่ยาว
    glPopMatrix();
    glColor3f(1.0, 1.0, 1.0);//สีขาว
    gluCylinder(theObj, 2.0, 2.0, 25.0, 20, 70);//ความใหญ่ยาว
      glPopMatrix();
      glColor3f(1.0, 0.0, 1.0);//สีม่วง
    gluCylinder(theObj, 2.0, 2.0, 25.0, 20, 70);//ความใหญ่ยาว
      ////////////////////////////////////////////////////////
    glPopMatrix();
    glutSwapBuffers(); // คำสั่งแสดงสิ่งที่วาดออกทางหน้าจอ
}
 void reshape(int w, int h)
{
      glViewport(0, 0, w, h); //เป็นการจัดพิกัดของ Opengl ให้เท่ากับหน้าต่างของ Windows
      glMatrixMode(GL_PROJECTION); //เปลี่ยนโหมดของเมตริกซ์จาก GL_MODELVIEW เป็น
GL_PROJECTION << มีผลกับการที่วัตถุถูกฉายลงบนระนาบ 2 มิติใดบ้างมีผลกระทบกับวัตถุ บรรทัดนี้บอกถึงการเปลี่ยนโหมดเมตริกซ์
      glLoadIdentity(); // แทนเมตริกซ์ที่มีอยู่ด้วยเมตริกเอกลักษณ์ เพราะเมตริกซ์การแปลงตัวใหม่จะถูกคูณกับเมตริกซ์
ที่มีอยู่ หากไม่เคลียร์จะเกิดข้อผิดพลาดขึ้น
      if(w<=h && w>0)
      glOrtho(-SIZEW,SIZEW,(GLdouble)-h/w*SIZEW, //
      (GLdouble)h/w*SIZEW,0.0,2*SIZEW); //
      else if(h>0)
      glOrtho((GLdouble)-w/h*SIZEW,(GLdouble)w/h*SIZEW, // เป็นการกำหนดว่าถ้า
รูปมีพิกัดเกินขอบเขตจะถูกตัดออกไม่นำมาแสดง
      -SIZEW,SIZEW,0.0,2*SIZEW); //
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glTranslatef(0.0,0.0,-SIZEW);
}
void readfile(/*unsigned char myint, int myint1, int myint2*/)
{
      {
      int myint;   ////////////////arm
      int myint1;  ////////////////hand
      int i;
      char c[100];
      bool sel;
      sel=true;
      //while(1)
      {
```

```
        {
          ////////////////////////ARM////////////////////////
          FILE *fp;
        printf("\n");
          fp=fopen("D:ARM.txt","r");

                        for( i=0;i<5;i++)
                    {

                        {
                              c[i] = fgetc(fp);


                        }
                    }
                  c[i] = fgetc(fp) ;
                  myint = atoi(c) ;
                  arm = myint*-1.74  ;
          }
        }

    ////////////////////////Hand////////////////////////////
        {
          FILE *fp;
        printf("\n");
          fp=fopen("D:HAND.txt","r");

                        for( i=0;i<5;i++)
                    {

                        {
                              c[i] = fgetc(fp);


                        }
                    }
                  c[i] = fgetc(fp) ;
                  myint1 = atoi(c) ;
                  hand = (myint1*-0.86) - 20  ;
                    if (sel=true)
                    {
                        fgetc(fp);
                        sel=false;
                    }
                    fclose(fp);
                    printf("myint %i \n",myint);
                    printf("myint %i \n",myint1);
                    //getch();

                }
      }
    }

void main (int argc, char** argv)
{
    glutInit(&argc, argv); ////////////////////////////////////////
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutInitWindowSize(700, 700);
    glutCreateWindow(argv[0]);            /////// เป็นการตั้งพารามิเตอร์ของ
OpenGL
    init(); ////////////////////////////////////////////////////
```

```
	glutReshapeFunc(reshape); ////////////////////////////////////
	//glutKeyboardFunc(keyboard);
	glutDisplayFunc(readfile);
	glutIdleFunc(display);                    /////// เป็นการประกาศฟังก์ชั่นที่ทำหน้าที่
ต่างๆ
	glutMainLoop();//////////////////////////////////////////////////
รอคำสั่งจากผู้ใช้
}
```

# ประวัติผู้เขียนโครงงาน


ชื่อ นายธญา เติมลาภ

ภูมิลำเนา 575/23 ถ.สนามบิน ต.ในเมือง อ.เมือง
จ.พิษณุโลก 65000


ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนพิษณุโลกพิทยาคม จังหวัดพิษณุโลก
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร


E-mail: sincerity_blue@hotmail.com



ชื่อ นายภูเบศ ประสาทชัย

ภูมิลำเนา 409/46 ถ.สนามบิน ต.ในเมือง อ.เมือง
จ.พิษณุโลก 65000


ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนพิษณุโลกพิทยาคม จังหวัดพิษณุโลก
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร


E-mail: desolation_Kob@hotmail.com