การผลิตผสมผสานด้วยคอมพิวเตอร์ : ส่วนระบบลำเลียง

COMPUTER INTEGRATED MANUFACTURING : CONVEYOR SYSTEM SECTION

นายกิตติโชค พัดพรม

นายญาณภัทร ปรีชาสนธิกุล

นายภาณุวัฒน์ นันทะมีชัย

นางสาววรารัตน์ รีเรืองชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

COMPUTER INTEGRATED MANUFACTURING : CONVEYOR SYSTEM SECTION

KITTICHOT PATPROM

YANAPHAT PREECHASONTHIKUL

PANUWAT NANTAMEECHAI

WARARAT REERUANGCHAI

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN CONTROL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

ปริญญานิพนธ์ปีการศึกษา 2553

สาขาวิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การผลิตผสมผสานด้วยคอมพิวเตอร์ : ส่วนระบบลำเลียง

ผู้จัดทำ นายกิตติโชค พัดพรม 50010108

นายญาณภัทร ปรีชาสนธิกุล 50010395

นายภาณุวัฒน์ นันทะมีชัย 50011171

นางสาววรารัตน์ รีเรืองชัย 50011394

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ดร.วันชัย ริ้วรุจา)

.....อาจารย์ที่ปรึกษา

(ดร.วรรณดี เพชรมณีล้ำค่า)

.....อาจารย์ที่ปรึกษา

(อาจารย์เทพจิตร์ เชยโภคา)

การผลิตผสมผสานด้วยคอมพิวเตอร์ : ส่วนระบบลำเลียง

โดย

นายกิตติโชค	พัดพรม	50010108
นายญาณภัทร	ปรีชาสนธิกุล	50010395
นายภาณุวัฒน์	นันทะมีชัย	50011171
นางสาววรารัตน์	รีเรืองงชัย	50011394

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ดร.วันชัย ริ้วรุจา

ดร.วรรณดี เพชรมณีล้ำค่า

อาจารย์เทพจิตร์ เชยโภคา

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอทฤษฎีและการจำลองกระบวนการผลิตที่ส่วนใหญ่ใช้ในอุตสาหกรรมการผลิต โดยมีโครงสร้างของกระบวนการที่ทำการจำลองประกอบด้วย ส่วนที่ทำหน้าที่หยิบจับชิ้นงาน ส่วนที่ทำหน้าที่ขนส่งชิ้นงานไปยังตำแหน่งต่อไป ส่วนที่ทำการกระจายคำสั่งและเก็บข้อมูลจากแต่ละส่วนไว้ มอนิเตอร์ทำหน้าที่แสดงผลการทำงานและทำหน้าที่สั่งการจากห้องควบคุม ซึ่งจุดมุ่งหมายของโครงงานนี้เพื่อทำการศึกษากระบวนการทำงานและการติดต่อสื่อสารกันระหว่างระบบต่างๆ เพื่อให้สามารถทำงานได้อย่างสอดคล้องกันอย่างเป็นระบบ

COMPUTER INTEGRATED MANUFACTURING : CONVEYOR SYSTEM SECTION

By

Kittichot Patprom

Yanaphat Preechasonthikul

Panuwat Nantameechai

Wararat Reeruangchai

Advisors

Assoc.Prof.Dr.Vanchai Riewruja

Dr.Wandee Petchmaneelumka

Mr.Thepjit Cheypoca

ABSTRACT

This project presents the theory and simulation of manufacturing process that are widely used in many manufacturing industries. The simulated model consists of robot arm for holding work piece ,transportation car, the command section and the monitor section that shows the results of work and commands the specimens from the control room. The aim of this project is to study process and communication between different systems in order to develop the system that make the equipments work consistently in a systematic way.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงลงไปได้ด้วยดี เพราะได้รับความช่วยเหลือเป็นอย่างดี จาก รองศาสตราจารย์ดร.วันชัย ริ้วรุจา ดร.วรรณดี เพชรมณีล้ำค่า อาจารย์เทพจิตร์ เชยโภคา ที่กรุณาให้คำปรึกษาแนะนำที่ดีมาโดยตลอดตั้งแต่ต้น รวมทั้งเอื้อเฟื้ออุปกรณ์ที่จำเป็น และความช่วยเหลืออื่นๆที่เป็นประโยชน์ต่อโครงงาน ผู้จัดทำรู้สึกซาบซึ้งและขอกราบขอบพระคุณอย่างสูง

ขอบคุณเพื่อนๆทุกคน ที่ให้กำลังใจ สนับสนุนอุปกรณ์ที่ขาดเหลือ กระตุ้นเตือน รวมทั้งคอยถามไถ่ความคืบหน้าของโครงการอยู่เสมอ

สุดท้ายนี้ผู้จัดทำขอกราบขอบพระคุณบิดา มารดา และครอบครัว ที่คอยเป็นกำลังใจที่ดีตลอดมา รวมถึงสนับสนุนในเรื่องของงบประมาณที่ขาดเหลือ ตลอดจนเป็นแรงบันดาลใจที่ดีที่สุดที่ทำให้โครงงานนี้สำเร็จสมบูรณ์ลงได้

ผู้จัดทำ

นายกิตติโชค พัดพรม

นายญาณภัทร ปรีชาสนธิกุล

นายภาณุวัฒน์ นันทะมีชัย

นางสาววรารัตน์ รีเรืองชัย

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

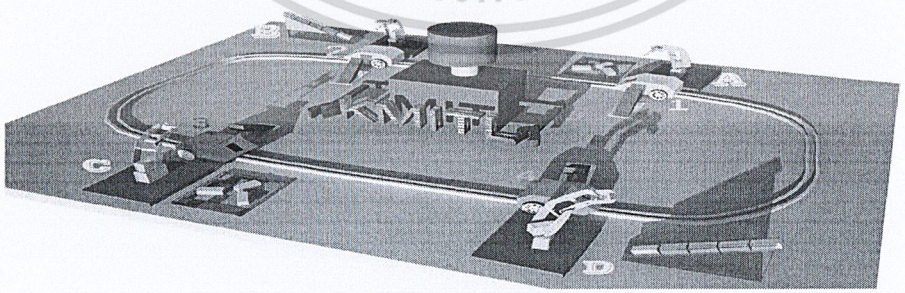
บทที่ 1

บทนำ

1.1 กล่าวนำ

ในปัจจุบันกระบวนการผลิตของโรงงานอุตสาหกรรมส่วนใหญ่ได้มีการนำเอาระบบควบคุมแบบอัตโนมัติเข้ามาใช้งานเพิ่มมากขึ้น โดยมีการนำเอาเทคโนโลยีสมัยใหม่เข้ามาทดแทนแรงงานมนุษย์ที่มีอยู่ เพื่อลดการสูญเสียที่อาจเกิดจากขั้นตอนการผลิต ที่มีความอันตรายต่อบุคคลากร ลดต้นทุนในการผลิต เพิ่มประสิทธิภาพ ความแม่นยำ และความรวดเร็วในกระบวนการผลิต อีกทั้งยังสามารถเก็บรวบรวมข้อมูลและตรวจสอบการทำงานของระบบได้ ทำให้ตระหนักถึงความสำคัญของระบบควบคุมแบบอัตโนมัติ จึงมีความสนใจศึกษาระบบการลำเลียงและแขนกลในโรงงานอุตสาหกรรม และได้ทำการจำลองระบบนี้ขึ้นเพื่อใช้ในการศึกษาทดลองและทำความเข้าใจในกระบวนการ เพื่อพัฒนาขีดความสามารถการในทำงานของระบบควบคุมอัตโนมัติให้ตอบสนองต่อความต้องการของโรงงานในอนาคตได้

การจำลองระบบการผลิตนี้ ประกอบด้วย รถลำเลียงสินค้าซึ่งทำตามคำสั่งที่ส่งผ่านมาทางมอนิเตอร์ โดยรถลำเลียงสินค้าจะวิ่งบนรางไฟฟ้า โดยทำหน้าที่ในการลำเลียงสินค้าจากหน่วยการทำงานหนึ่งไปยังอีกหน่วยการทำงานหนึ่ง และมีแขนกลทำการหยิบจับ ในการนำสินค้าเข้า – ออกระหว่างรถลำเลียงกับหน่วยการทำงาน ซึ่งจะมีหุ่นยนต์แมลง 6 ขา มารับสินค้าจากแขนกลไปส่งต่อให้กับหุ่นยนต์ 2 ขา เพื่อไปเก็บในคลังสินค้าต่อไป



รูปที่ 1.1 ระบบจำลองการผลิตอัตโนมัติ

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. ทำการศึกษาและทดลองการใช้งานอุปกรณ์ต่างๆในระบบควบคุม
2. ทำการศึกษาและจำลองระบบกระบวนการผลิตอัตโนมัติ ซึ่งประกอบด้วย แขนกลทำหน้าที่จับสิ่งของ หุ่นยนต์ลำเลียงสินค้า สถานีที่ทำหน้าที่ติดต่อสื่อระหว่างหุ่นยนต์ลำเลียงสินค้ากับตัวควบคุมหลัก และคอมพิวเตอร์ที่ทำหน้าที่สั่งการและตรวจสอบการทำงาน ซึ่งในโครงงานนี้ใช้กล้องเว็บแคม (Webcam) โดยโครงงานนี้ใช้คอมพิวเตอร์และโปรแกรมวิชวลเบสิก (Visual Basic)
3. ทำการศึกษาการใช้งานไมโครคอนโทรลเลอร์ (Micro Controller) ตระกูลต่างๆ

1.3 ขั้นตอนการศึกษาและการจัดทำโครงงาน

การศึกษาและจำลองระบบกระบวนการผลิตอัตโนมัตินั้น จำเป็นเป็นต้องทำการศึกษาองค์ประกอบของระบบก่อนว่า โดยทั่วไปของระบบๆหนึ่งนั้นประกอบด้วยอะไรบ้าง หลังจากที่ได้ทำการศึกษา ค้นคว้าดูแล้วพบว่า โดยส่วนใหญ่นั้นในกระบวนการผลิตแห่งหนึ่งๆนั้นจะมีการทำงานของหลายๆส่วนเข้าด้วยกัน ซึ่งจะประกอบด้วย ส่วนที่ทำหน้าที่หยิบจับชิ้นงาน ส่วนที่ทำหน้าที่ขนส่งชิ้นงานไปยังตำแหน่งต่อไป ส่วนที่ทำการกระจายคำสั่งและเก็บข้อมูลจากแต่ละส่วนไว้ มอนิเตอร์ทำหน้าที่แสดงผลการทำงานและทำหน้าที่สั่งการจากห้องควบคุม โดยที่แต่ละส่วนสามารถติดต่อสื่อสารกันและทำงานได้อย่างสอดคล้องกัน โดยผ่านการสั่งงานจากมอนิเตอร์และจากนั้นก็ให้ส่วนกระจายคำสั่งและเก็บข้อมูลทำหน้าที่กระจายคำสั่งที่ได้รับมาจากมอนิเตอร์กระจายคำสั่งไปให้แต่ละส่วนที่ทำหน้าที่รับผิดชอบการทำงานนั้นๆต่อไป

จากที่ได้กล่าวมาแล้วข้างต้นโครงงานนี้จึงมีการแบ่งกลุ่มกันไปทำการศึกษาและทำการจำลองการทำงานของแต่ละส่วน โดยกลุ่มของข้าพเจ้าได้ทำการศึกษาและจำลองในส่วนของมอนิเตอร์แสดง กับส่วนเก็บข้อมูลและขนส่งชิ้นงาน

โดยในส่วนของมอนิเตอร์แสดงผลนั้น ให้มีการแสดงผลที่คอมพิวเตอร์ ซึ่งควรที่จะใช้งานได้อย่างสะดวกแต่มีฟังก์ชันการทำงานได้อย่างหลากหลาย เพื่อที่จะให้ผู้ที่ทำการควบคุมระบบอยู่นั้นไม่เกิดความสับสนและง่ายต่อการใช้งาน ใช้โปรแกรมวิชวลเบสิก มีภาษาที่เรียบง่ายพัฒนาเป็นแอพพลิเคชันได้ง่ายและมีเครื่องมือคำสั่งต่างๆ (Tool) มาให้สำเร็จรูป มีพัฒนาเป็นแอพพลิเคชันโดยออกแบบให้เป็นมอนิเตอร์เป็นปุ่มกดเพื่อความงาน เมื่อเรียกใช้งานจึงเพียงแค่คลิกปุ่มคำสั่งที่ต้องการเท่านั้น

ในส่วนเก็บข้อมูลและขนส่งชิ้นงานจะประกอบด้วย รถลำเลียง สถานีที่ทำการติดต่อสื่อสารระหว่างรถลำเลียงกับคอนโทรลเลอร์ที่ทำหน้าที่เก็บข้อมูลและกระจายคำสั่ง คอนโทรลเลอร์ที่ทำหน้าที่เก็บขอมูลและกระจายคำสั่ง

การทำงานของทั้งสามนี้คือเมื่อคอนโทรลเลอร์ได้รับคำสั่งจากมอนิเตอร์แล้วจะทำการตรวจสอบเงื่อนไข ถ้าคำสั่งให้ทำงาน คอนโทรลจะทำการสั่งการไปที่สถานีหรือแขนกลตามเงื่อนไข และเมื่อมอนิเตอร์ต้องการตรวจสอบข้อมูลแล้วคอนโทรลเลอร์จะทำการส่งข้อมูลที่เก็บไว้ไปแสดงผลที่มอนิเตอร์ ส่วนสถานีเมื่อได้รับคำสั่งมาก็จะทำการติดต่อสื่อสารผ่านทางสัญญาณอินฟาเรดไปยังรถลำเลียง แล้วส่งค่าข้อมูลกลับไปยังคอนโทรลเลอร์เพื่อให้เก็บข้อมูลไว้ ในส่วนแขนกลเมื่อได้รับคำสั่งมาก็จะทำการหยิบชิ้นงาน โดยศึกษาการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์ภายนอกรวมถึงวงจรที่ใช้ในการควบคุมซึ่งรายละเอียดจะกล่าวในบทต่อไป

1.4 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวถึงวัตถุประสงค์ หลักการใหม่ ขั้นตอนการศึกษาและการจัดทำโครงการรวมถึงรายละเอียดในแต่ละบท

บทที่ 2 ระบบจำลองกระบวนการอัตโนมัติ กล่าวถึง ระบบรวมของการทำงานทั้งหมด ได้แก่ ระบบลำเลียงสินค้า แขนกล หุ่นยนต์แมลง หุ่นยนต์ 2 ขา

บทที่ 3 ระบบลำเลียงสินค้า กล่าวถึงทฤษฎี หลักการและการออกแบบระบบ

บทที่ 4 ผลการทดลอง เป็นส่วนการทดสอบองค์ประกอบต่างๆในระบบตลอดจนการทดลองระบบจำลองการผลิตอัตโนมัติ

บทที่ 5 สรุปและแนวทางในการพัฒนา จะสรุปผลการดำเนินงาน ปัญหาที่เกิดขึ้น และแนวทางในการปรับปรุงพัฒนาได้

บทที่ 2

ระบบจำลองกระบวนการผลิตอัตโนมัติในโครงงานนี้

การจำลองระบบการผลิตในโครงการนี้ ประกอบด้วย รถลำเลียงสินค้าซึ่งทำตามคำสั่งที่ส่งผ่านมาทางมอนิเตอร์ โดยรถลำเลียงสินค้าจะวิ่งบนรางรถไฟฟ้า โดยทำหน้าที่ในการละเลียงสินค้าจากหน่วยการทำงานหนึ่งไปยังอีกหน่วยการทำงานหนึ่ง และมีแขนกลทำการหยิบจับ ในการนำสินค้าเข้า- ออกระหว่างรถลำเลียงสินค้ากับหน่วยการทำงานซึ่งจะมีหุ่นยนต์แมลง 6 ขามารับสินค้าจากแขนกลเพื่อนำไปส่งต่อให้กับหุ่นยนต์ 2 ขาเพื่อนำไปเก็บในคลังสินค้าต่อไป ซึ่งแต่ละส่วนได้อธิบายในหัวข้อถัดไป

รูปที่ 2.1 ระบบจำลองกระบวนการผลิตอัตโนมัติ

2.1 โครงสร้างของระบบลำเลียงสินค้า

โครงสร้างของระบบลำเลียงสินค้า ประกอบด้วย

. ระบบลำเลียงสินค้า

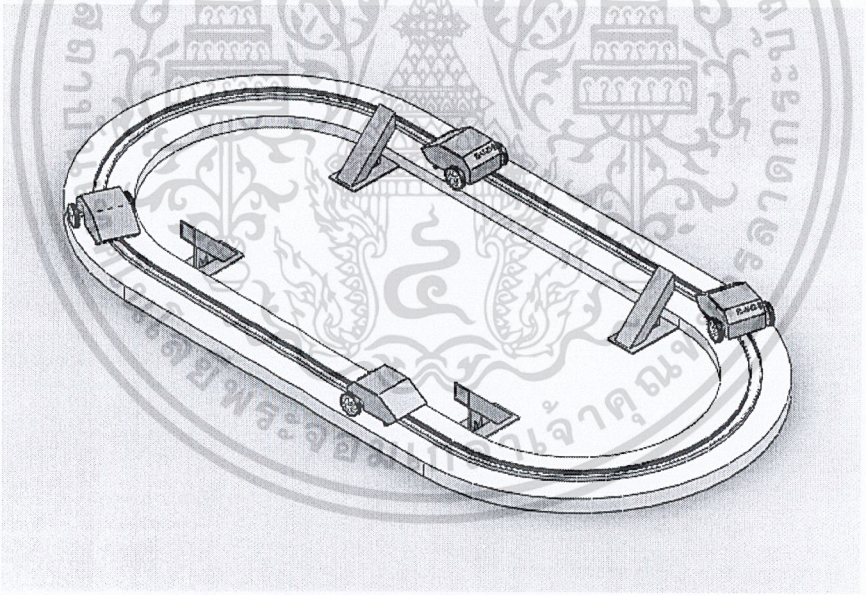
หน้าจอแสดงผล

บอร์ด ET-easy mega 1280

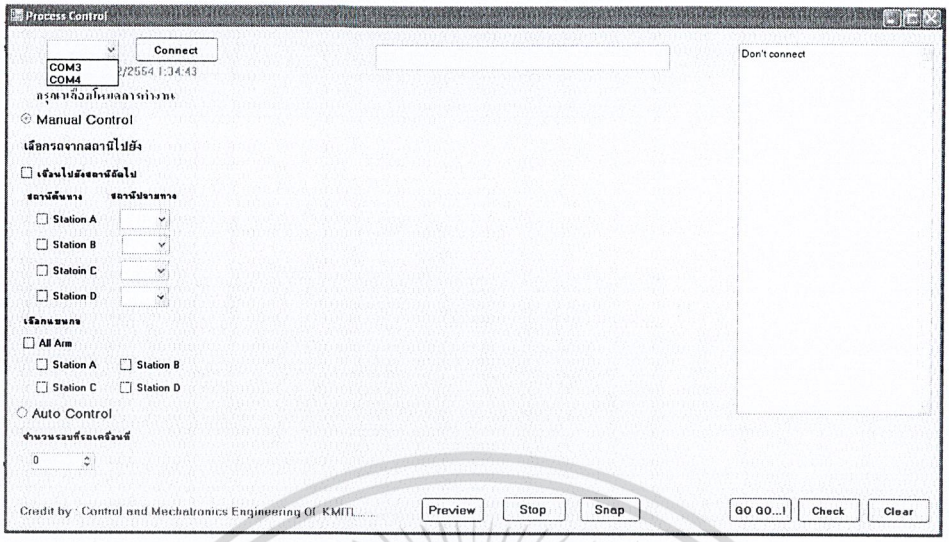
รถลำเลียง

แขนกล

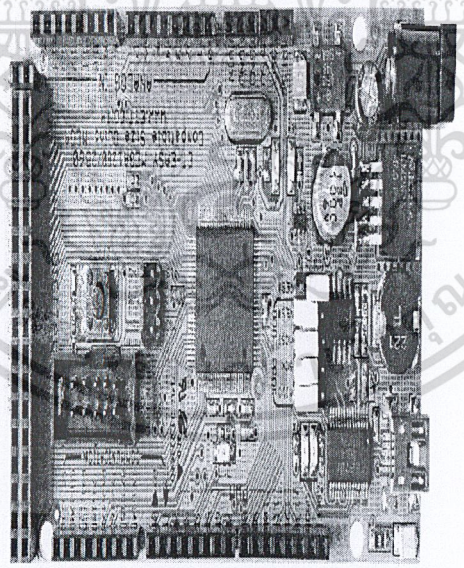
สถานีรับ - ส่งข้อมูล



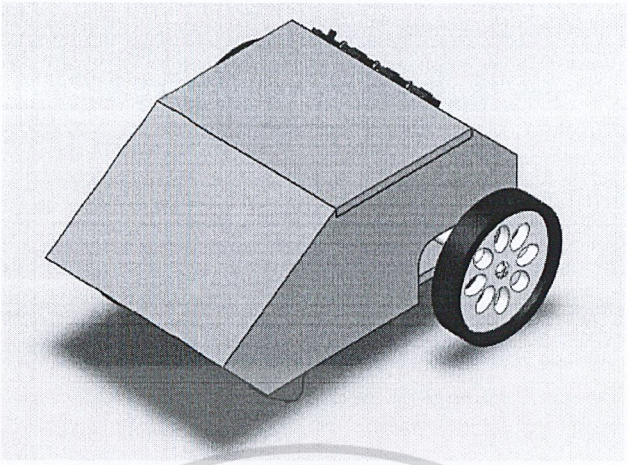
รูปที่ 2.2 ระบบลำเลียงสินค้า



รูปที่ 2.3 หน้าจอแสดงผล

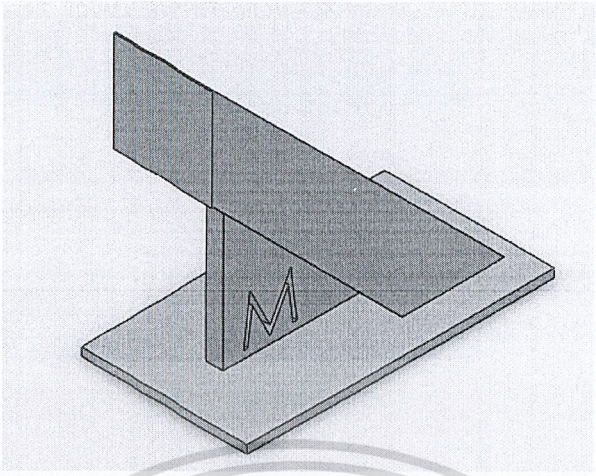


รูปที่ 2.4 บอร์ด ET-easy mega 1280



รูปที่ 2.5 รถลำเลียง

รูปที่ 2.6 แขนกล



รูปที่ 2.7 สถานีรับ-ส่งข้อมูล

2.2 การทำงานของระบบ

การทำงานของระบบแบ่งออกเป็น 2 โหมด คือ

2.2.1 โหมดควบคุมด้วยมือ (Manual Mode)

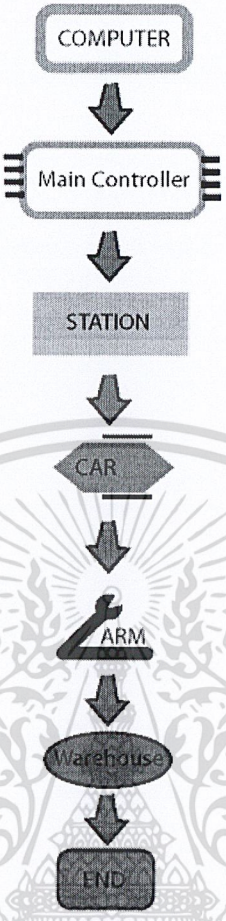
ในการทำงานในโหมดควบคุมด้วยมือ นั้นเราจะเป็นผู้สั่งการทำงานของระบบผ่านโปรแกรมดังภาพด้านบนโดยที่เรามารถกำหนดได้ว่า จะให้รถที่สถานีที่เราต้องการไปจอดในสถานีที่เรากำหนดโดยเราต้องทำการเลือกสถานีต้นทาง และสถานีปลายทาง และทำการสั่งงานโดยปุ่ม “Go Go”

2.2.2 โหมดควบคุมด้วยอัตโนมัติ (Automation Mode)

สำหรับการทำงานใน โหมดควบคุมอัตโนมัติ นั้น จะมีลำดับการทำงานดังนี้

1) รถทั้ง 4 คันจะเข้ามาจอดประจำแต่ละสถานี
2) แขนกลจะทำงานโดยแบ่งการทำงานออกเป็น 2 แบบคือ แขนกลสถานี A กับ C จะหยิบของออกจากหลังรถ แขนกลสถานี B กับ D จะหยิบของมาวางไว้หลังรถ
3) เมื่อแขนกลทำงานเสร็จ รถทั้ง 4 คันจะเคลื่อนที่ไปจอดที่สถานีถัดไป

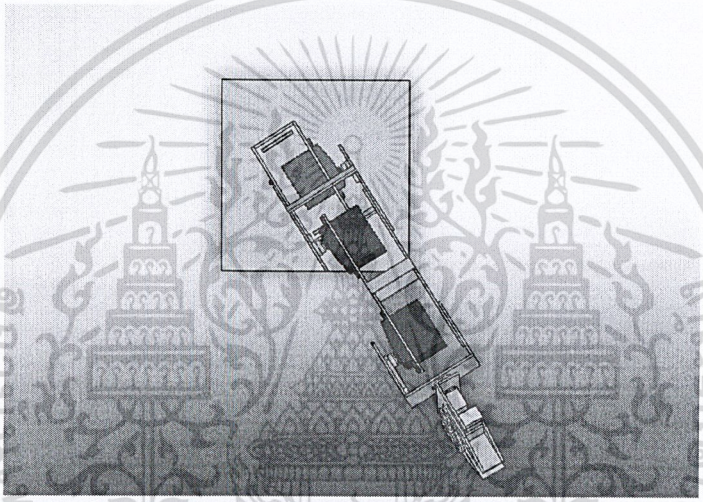
สำหรับการสั่งงานในโหมดอัตโนมัติเราจะต้องกำหนดจำนวนรอบการทำงานของระบบโดยการเลือกที่ปุ่ม “Setting” เมื่อทำการใส่จำนวนรอบเสร็จแล้วให้กดปุ่ม “Run” เพื่อเป็นการสั่งให้ระบบเริ่มทำงาน



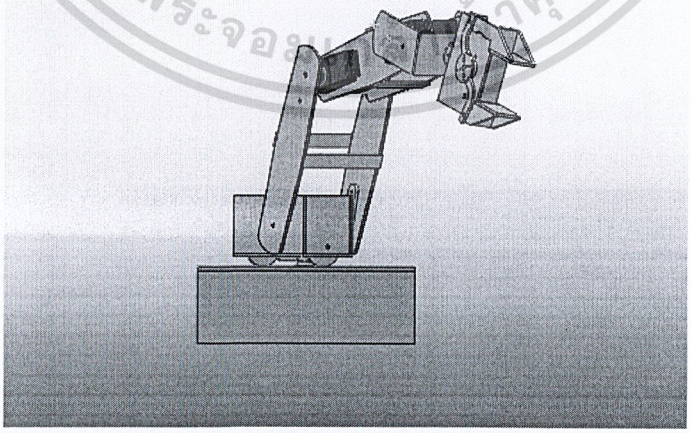
รูปที่ 2.8 การทำงานของระบบ

2.3 แขนกล

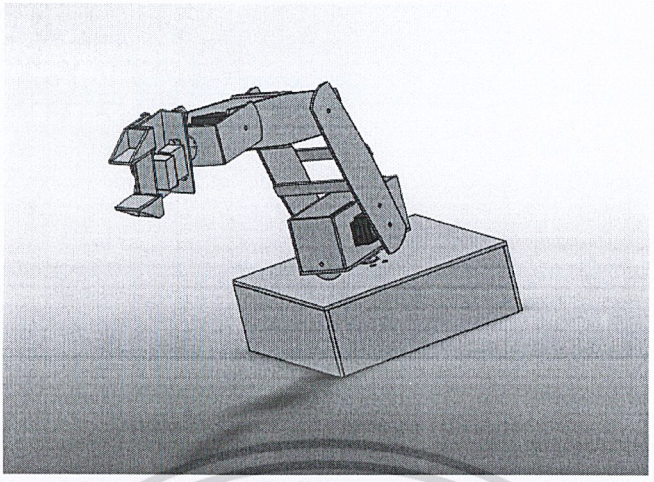
ส่วนประกอบในโครงสร้างของแขนกลนั้นถูกสร้างขึ้นจากแผ่นอคิลิคที่ตัดเป็นชิ้น ตามที่ออกแบบไว้ และในการเคลื่อนที่ของแขนกล ใช้เซอร์โวมอเตอร์ จำนวน 6 ตัว ซึ่งจะมีเซอร์โวมอเตอร์อยู่ในแต่ละข้อของแขนกล ซึ่งจะทำให้การเคลื่อนไหวของแขนกลสามารถเคลื่อนไหวได้คล้ายกับแขนมนุษย์ และสามารถนำไปประยุกต์ใช้ในการหยิบจับสิ่งของได้



รูปที่ 2.9 โครงสร้างแบบ Top ของแขนกล



รูปที่ 2.10 โครงสร้างแบบ Font ของแขนกล

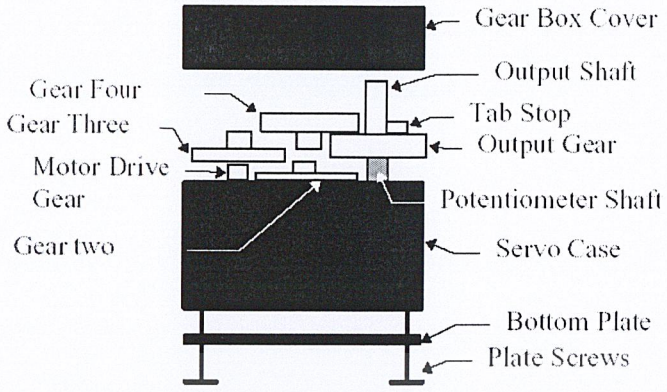


รูปที่ 2.11 โครงสร้างแบบ Isometric ของแขนกล

2.3.1 หลักการทำงานของเซอร์โวมอเตอร์

เซอร์โวมอเตอร์เป็นมอเตอร์ไฟฟ้ากระแสตรง (DC moter) ประกอบด้วยชุดเกียร์ และส่วนควบคุมต่างๆ ไว้ในโมดูลเดียวกัน มีสายต่อใช้งานเพียง 3 เส้นเท่านั้น คือ VCC, GND และ สายสัญญาณควบคุม (Control Line) ซึ่งสามารถควบคุมให้ตัวเซอร์โวมอเตอร์หมุนซ้ายหรือขวาได้ +90 องศา - 90 องศา (180 องศา) โดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณพัลส์วิดมอด (PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 V โดยสามารถสั่งงานในการหมุนให้หมุนไปได้ตามองศาต่างๆที่ต้องการได้ด้วยตัวของเซอร์โวมอเตอร์เองไม่ต้องมีส่วนควบคุมหรือSENSORใดๆกลับมาตรวจสอบอีกทำให้ง่ายและสะดวกในการในการนำไปประยุกต์ใช้งานต่างๆ

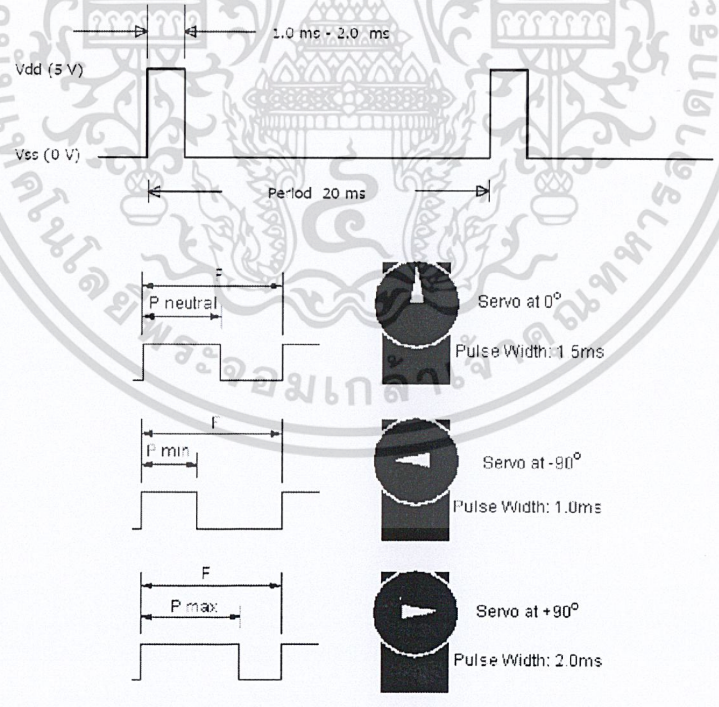
- การควบคุมการทำงานของเซอร์โวมอเตอร์ทำได้โดยการป้อนสัญญาณความกว้างของพัลส์ให้กับตัวเซอร์โวมอเตอร์ซึ่งจะได้ทิศทางการหมุนและตำแหน่งของการหมุน
- สามารถใช้งานกับไฟ DC ได้ 4 - 6 V, หมุนได้ 180 องศา และสามารถปรับแต่งตัวเซอร์โวมอเตอร์ให้สามารถหมุนได้รอบตัวได้
- ขั้วต่อจะเป็นแบบมาตรฐาน : ขั้ว JR TYPE



รูปที่ 2.12 โครงสร้างของเซอร์โวมอเตอร์

หลักการทำงานของเซอร์โวมอเตอร์

การควบคุมการทำงานทำได้โดย การป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้นๆ โดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังรูป



รูปที่ 2.13 จุดอ้างอิงความกว้างของสัญญาณพัลส์

- สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์

- สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม - 90องศา หรือในทิศทางทวนเข็มนาฬิกา

- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม + 90องศา หรือในทิศทางตามเข็มนาฬิกา

ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆ นั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆ โดยอ้างอิงจากจุด ทั้ง 3 จุดที่กล่าวมานี้ ซึ่งสัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้ก็คือ จะอาศัยการเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR) เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุม สัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากัน มอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั่งค่าเวลาความกว้างพัลส์ของ วงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

2.3.2 ชนิดของเซอร์โวมอเตอร์ที่ใช้สำหรับแขนกล

- Servo MG945 Towerpro : Stall torque: 10kg.cm(4.8V),13kg.cm(6V)
  Operating speed: 0.23 sec/60degree(4.8v), 0.2 sec/60degree(6v)
  Operating voltage: 4.8-7.2V
- Servo S3003 : Stall Torque : 3.2kg.cm(4.8V), 4.1kg.cm(6.0V)
  Operating Speed : 0.23sec/60 degrees(4.8V), 0.19sec/60 degrees(6.0V)
  Operating Voltage : 4.8-6.0 V
- Micro Servo SG90 9 g. : Stall Torque : 1.2 kg.cm (4.8V),

Operating Speed : 0.12 sec/ 60 degrees(4.8 V),

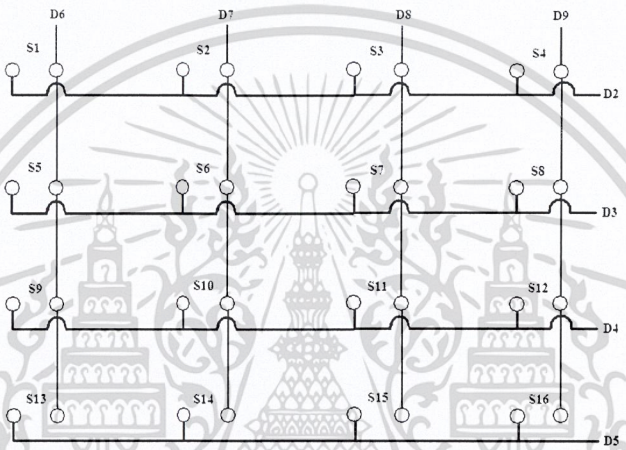
Operating Voltage : 4.0 to 7.2 volts

• Digital Servo : Torgue : Stall Torque : > 1kg.cm (Vcc=5V)

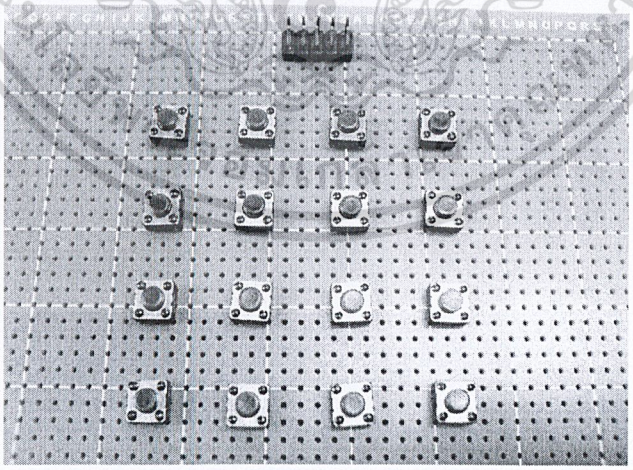
Operating Speed : 0.1 sec/60 degrees

Operating Voltage : DC 5V±1V

2.3.3 วงจรสวิตช์ควบคุม



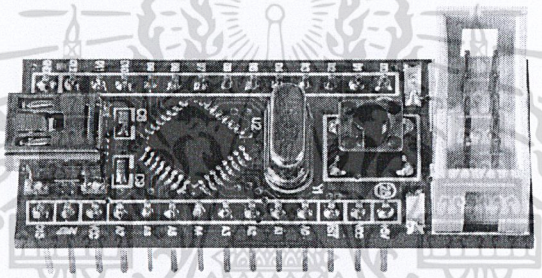
รูปที่ 2.14 วงจรสวิตช์ควบคุม



รูปที่ 2.15 บอร์ดควบคุมการเคลื่อนไหวของหุ่นยนต์

ประยุกต์ใช้วงจรควบคุมมาใช้ในการกำหนดการเคลื่อนที่ของแขนกล โดยใช้ Digital Pin D2-D9 ซึ่งแต่ละ Pin จะส่งสัญญาณ Logic 1 ออกมา เช่น สวิตช์ S1 จะมี Pin D2 กับ D6 เชื่อมกันอยู่ เมื่อกดปุ่ม S1 Logic จาก D2 และ D9 จะเชื่อมถึงกัน ทำให้ Microcontroller รู้ว่ามีการกดปุ่ม S1 ตามที่เขียนโปรแกรมไว้ โดยเซอร์โวมอเตอร์ 1 ตัวจะถูกควบคุมด้วยสวิตช์ 2 ตัว เพื่อควบคุมทิศทางในการหมุนตามเข็มหรือทวนเข็มนาฬิกา ตามที่ต้องการให้เคลื่อนที่ในทิศทางนั้น เมื่อได้ตำแหน่งที่ต้องการแล้วให้กดสวิตช์ที่เป็นการเมมโมรี่ จะสั่งให้จำค่านั้นไว้ ซึ่งในบอร์ดควบคุมนี้สามารถบันทึกค่าได้ทั้งหมด 8 ค่า เมื่อทำการบันทึกค่าทั้งหมดแล้ว ทำการกดสวิตช์ RUN จะเป็นการสั่งให้แขนกลทำงานได้อัตโนมัติตามที่สั่งค่าไว้

2.3.4 บอร์ดควบคุมการทำงานของแขนกล



รูปที่ 2.16 บอร์ด Stamp168

ET-EASY168 STAMP เป็นบอร์ดไมโครคอนโทรเลอร์ในตระกูล AVR8 ขนาดเล็กจิ๋ว โดยมีขนาดของบอร์ดเพียง 2cm x 5cm เท่านั้น ซึ่งขนาดบอร์ด ประมาณเท่ากับตัวถังของไอซี 28 DIP 300 โดยเลือกใช้ไมโครคอนโทรเลอร์ตระกูล AVR8 เบอร์ ATmega168 ของ ATMEL เป็น MCU ประจำบอร์ด โดยเลือกใช้MCU ที่มีรูปร่างตัวถังแบบ 32 TQFP พร้อมวงจรรอบนอกที่จำเป็นอย่าง Oscillator และ Reset รวมไว้ด้วยภายในบอร์ด นอกจากนี้แล้วภายในตัวบอร์ดยังได้รวมเอาไอซี USB Bridge ของ FTDI เบอร์ FT232R เพื่อใช้ติดต่อสื่อสารแบบอนุกรมด้วย RS232 กับคอมพิวเตอร์ PC ผ่านทางพอร์ต USB ได้โดยตรงทำให้บอร์ด ET-EASY168 STAMP เป็นบอร์ดทดลองขนาดเล็กที่เพียบพร้อมไปด้วยวงจรพื้นฐานที่จำเป็นต่อการใช้งานไมโครคอนโทรลเลอร์ตระกูล AVR8 อย่างแท้จริง เพียงแต่เสียบสาย USB จากพอร์ตUSB ของเครื่องคอมพิวเตอร์ PC เข้ากับขั้ว USB ของบอร์ด ET-EASY168 STAMP ก็สามารถทำการเขียนโปรแกรม และ Download Code ให้กับ MCU เพื่อทำการทดลองได้ทันที

คุณสมบัติของบอร์ด

• เลือกใช้ MCU ตระกูล AVR8 เบอร์ ATmega168 ของ ATMEL Run ความถี่ 16.00 MHz

o มีหน่วยความจำ Flash สำหรับเขียนโปรแกรม 16KByte ถ้าใช้การพัฒนาโปรแกรมผ่านระบบ AVRISP หรือ 14Kbyte เมื่อใช้การพัฒนาโปรแกรมผ่านระบบ Boot Loader RS232

o มี SRAM ใช้งานขนาด 1KByte และ EEPROM ใช้งานขนาด 512 Byte

o มี GPIO ใช้งานจำนวน 22 บิต

• Digital GPIO จำนวน 14 บิต

• Analog Input (ADC) ขนาดความละเอียด 10บิต จำนวน 8 ช่อง

• ใช้งานกับแรงดันไฟตรงขนาด +5VDC โดยใช้ได้ทั้งกับแหล่งจ่าย +5VDC/500mA จากพอร์ต USB และจากแหล่งจ่าย +5VDC จากภายนอกได้ด้วย พร้อม LED Power แสดงสถานะของแหล่งจ่าย

• มีวงจร External Reset แบบ RC Reset และ Switch Reset พร้อมภายในบอร์ด

• ขั้วต่อใช้งานวางตัวบน Pin Header ระยะห่าง 2.54mm(100mil) ขนาด 28 Pin (ด้านละ14Pin)ระยะห่าง 600mil(1.5cm) ง่ายต่อการนำไปต่อประยุกต์ใช้งาน และ ขยายวงจร I/O สามารถใช้กับProject Board และ PCB เอนกประสงค์ได้โดยง่าย

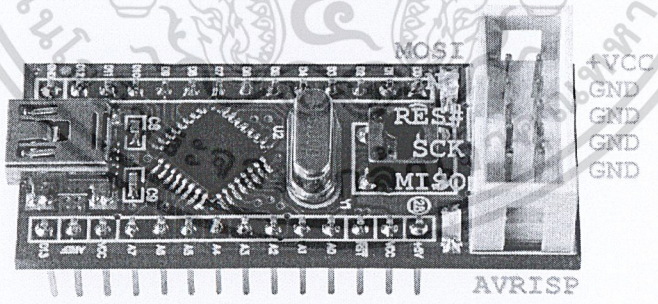
• มีขั้วต่อ USB สำหรับเชื่อมต่อสื่อสารกับคอมพิวเตอร์ PC ผ่าน USB Bridge ของ FTDI ในรูปแบบของการสื่อสารอนุกรม RS232 สำหรับใช้งานสื่อสารและ Download Code ให้กับ MCU ในบอร์ด

• มีขั้ว AVRISP แบบ IDE 10PIN สำหรับใช้ Download โปรแกรมให้กับ MCU ภายในบอร์ดในกรณีไม่ต้องการใช้การพัฒนาโปรแกรมผ่านทาง Boot Loader

• มี LED แสดงสถานะ โดยต่อกับ PB5 ของ AVR (Digital-13 ของ Arduino Project) สำหรับใช้เป็นอุปกรณ์ทดลองการทำงานอย่างง่าย

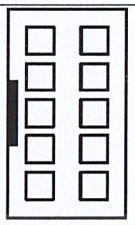
ตารางที่ 2.1 การจัดสรรขาสัญญาณของบอร์ด ET-EASY168 STAMP

AVR	Arduino	Pin	ET-EASY168 STAMP	Pin	Arduino	AVR
PD0	Digital-0	1		28	+5V(+Vin)	+5V(+Vin)
PD1	Digital-1	2		27	+VCC(+5V)	+VCC(+5V)
PD2	Digital-2	3		26	RESET#	RESET(PC6)
PD3	Digital-3	4		25	Analog-0	PC0/ADC0
PD4	Digital-4	5		24	Analog-1	PC1/ADC1
PD5	Digital-5	6		23	Analog-2	PC2/ADC2
PD6	Digital-6	7		22	Analog-3	PC3/ADC3
PD7	Digital-7	8		21	Analog-4	PC4/ADC4
PB0	Digital-8	9		20	Analog-5	PC5/ADC5
PB1	Digital-9	10		19	Analog-6	ADC6
PB2	Digital-10	11		18	Analog-7	ADC7
PB3	Digital-11	12		17	+VCC(+5V)	+VCC(+5V)
PB4	Digital-12	13		16	+AREF	+AREF
GND	GND	14		15	Digital-13	PB5



รูปที่ 2.17 บอร์ด ET-EASY168 STAMP

ตารางที่ 2.2 AVRISP

AVR	Arduino	Pin	AVRISP	Pin	Arduino	AVR
PB3	Digital-11	MOSI		+VCC	+VCC	+VCC
-	-	NC		GND	GND	GND
RES#	RES#	RES#		GND	GND	GND
PB5	Digital-13	SCK		GND	GND	GND
PB4	Digital-12	MISO		GND	GND	GND

หน้าที่ของขาสัญญาณในการใช้งานแบบ “Arduino Project”

• +5V(+Vin) เป็นขาสำหรับใช้เป็นจุดรับแรงดันขนาด +5VDC จากภายนอกเพื่อเป็นแหล่งจ่ายไฟเลี้ยงให้กับบอร์ด

• +VCC(+5V) เป็นขาแหล่งจ่ายไฟจุดเดียวกันกับที่ป้อนให้กับ +VCC ของ MCU ซึ่งจุดนี้จะรับแรงดันมาจาก 2 แหล่ง ด้วยกันคือ ขารับแรงดัน +5V(+Vin) จากขา 28 ของบอร์ด และ จากขา+VUSB(+5V) จากขั้ว USB ของบอร์ด โดยมี Diode ป้องกันการย้อนกลับของแรงดันไว้แล้ว

• +AREF เป็นขาสำหรับรับสัญญาณแรงดันอ้างอิง (Analog Reference) ให้กับวงจร Analog Inputในกรณีต้องการใช้แรงดันอ้างอิงจากภายนอก

• RESET# เป็นขาสัญญาณ RESET ของ CPU ทำงานที่ Logic “0”

• Digital[0..13] เป็นขา I/O แบบ Digital สามารถใช้งานเชื่อมต่อกับสัญญาณ Logic TTL (5V) ต่างๆ

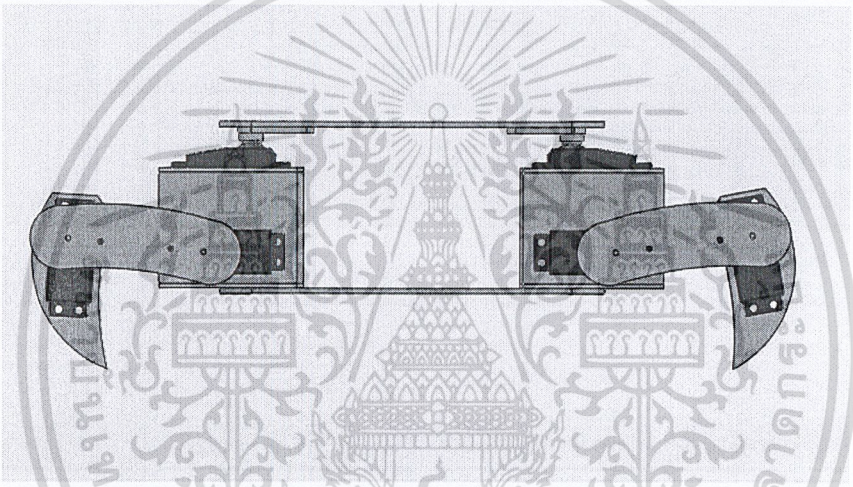
• Analog[0..7] เป็นขา Input แบบ Analog สามารถรับ Input แบบ Analog 0..+5V

2.4 หุ่นยนต์แมลง

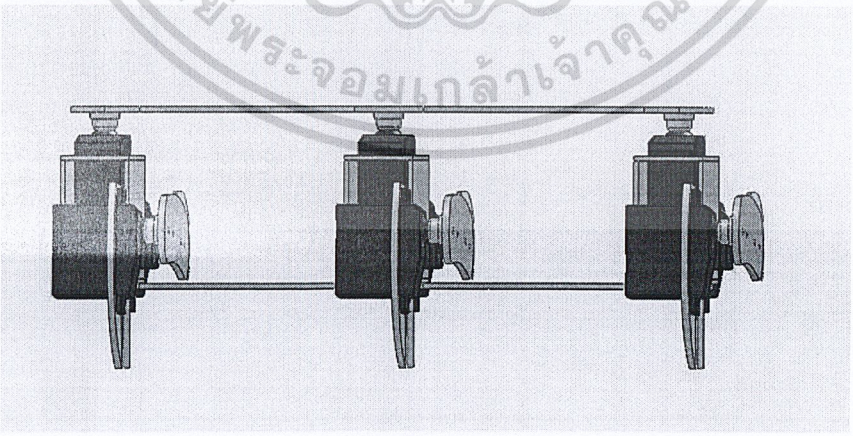
2.4.1 ทฤษฎีและหลักการทำงานของหุ่นยนต์แมลง

หุ่นยนต์แมลงประกอบด้วย 2 ส่วนคือ ฮาร์ดแวร์และส่วนซอฟต์แวร์

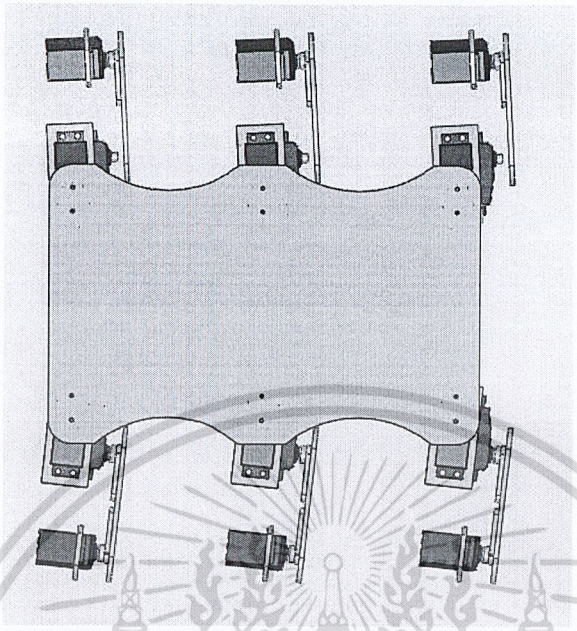
- ส่วนฮาร์ดแวร์ ประกอบด้วยเซอร์โวมอเตอร์, โครงสร้างหุ่นยนต์แมลง , บอร์ดควบคุม ET-Easy168 STAMP , วงจรที่ใช้ในการควบคุมหุ่นยนต์แมลง
- ส่วนซอฟต์แวร์ ประกอบด้วย โปรแกรมArduino



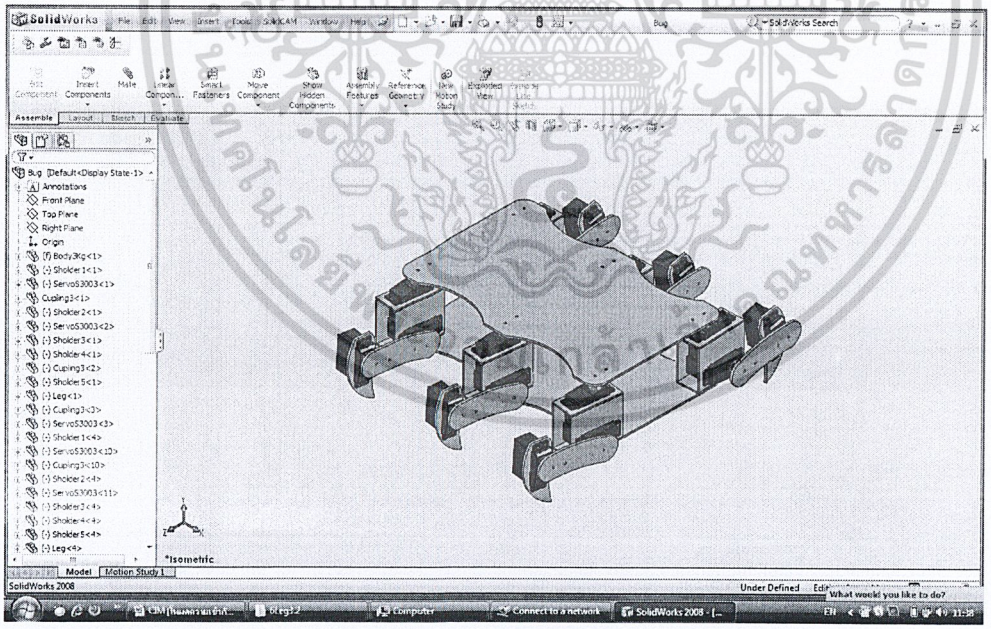
รูปที่ 2.18 โครงสร้างหุ่นยนต์แมลง ด้านหน้า



รูปที่ 2.19 โครงสร้างหุ่นยนต์แมลง ด้านข้าง



รูปที่ 2.20 โครงสร้างหุ่นยนต์แมลง ด้านบน



รูปที่ 2.21 โครงสร้างหุ่นยนต์แมลง มุม Isometric

2.4.2 ข้อมูลของเซอร์โวมอเตอร์ของหุ่นยนต์แมลง

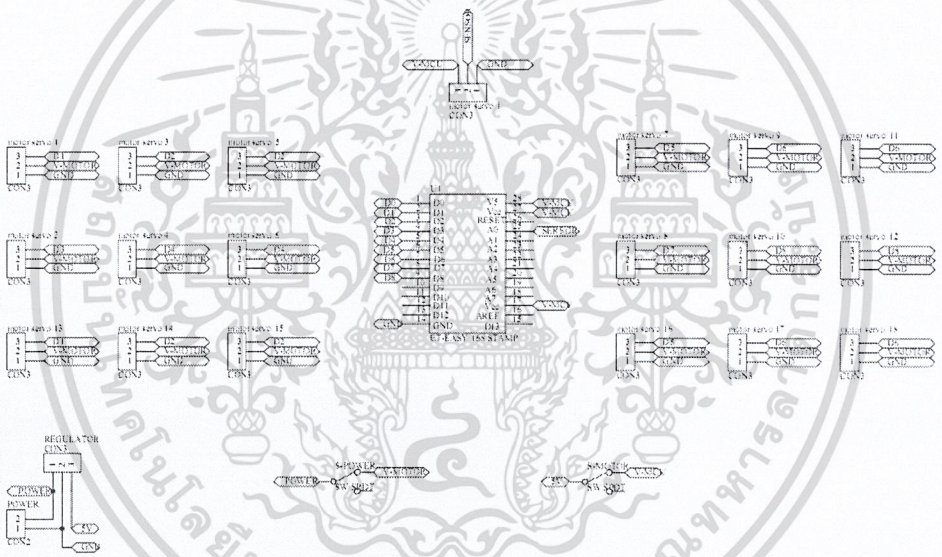
Speed : 0.23sec/60degrees at 4.8V , 0.23sec/60degrees at 6V

Torque : 10kg/cm(4.8V) , 12kg/cm(6V)

Size : 40.7 mm. X 19.7 mm. X 42.9 mm.

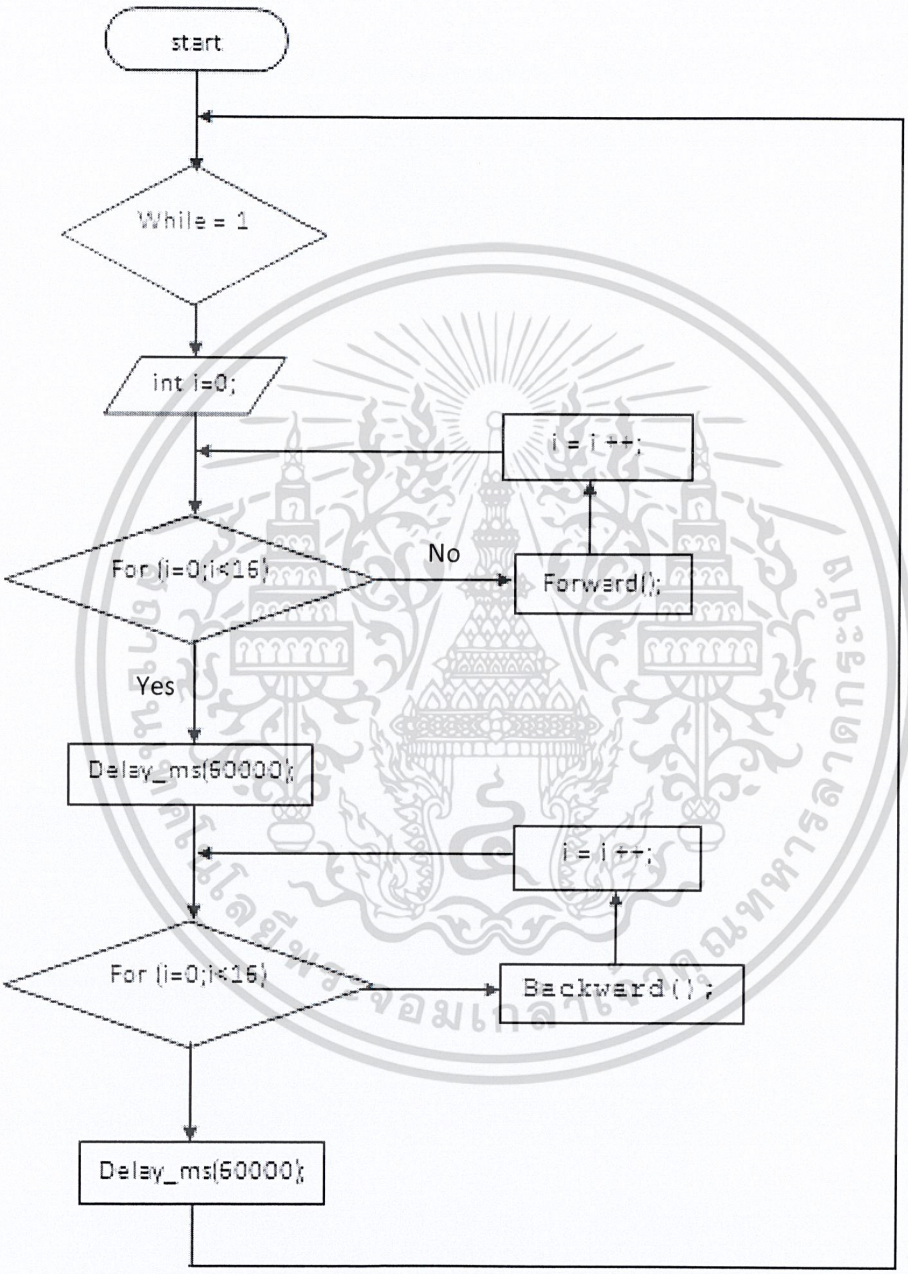
Weight : 55g

2.4.3 วงจรที่ใช้ในการควบคุมหุ่นยนต์แมลง



รูปที่ 2.22 วงจรที่ใช้ในการควบคุมหุ่นยนต์แมลง

2.4.4 ผังงานของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์แมลง



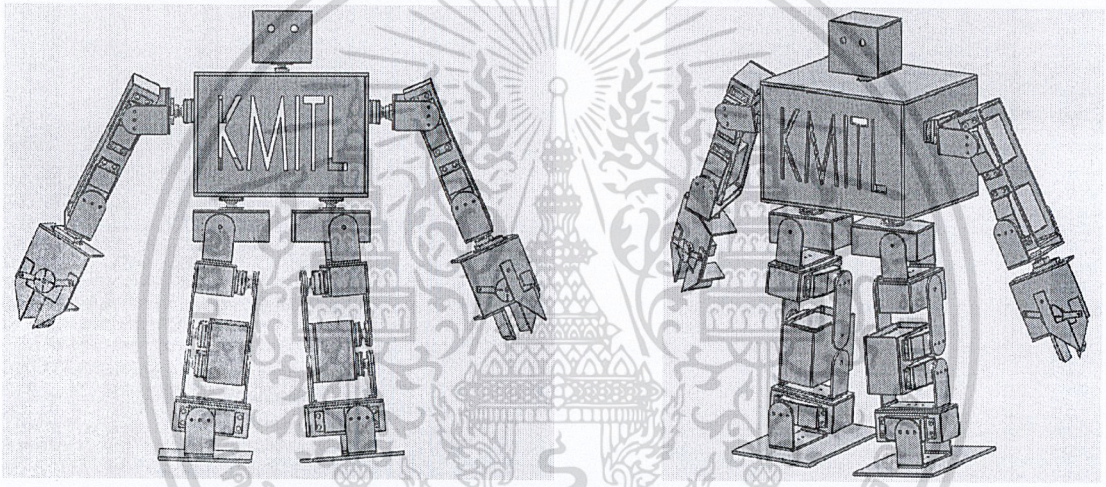
รูปที่ 2.23 ผังงานของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์แมลง

2.5 หุ่นยนต์ 2 ขา

2.5.1 ทฤษฎีและหลักการทำงานของหุ่นยนต์ 2 ขา

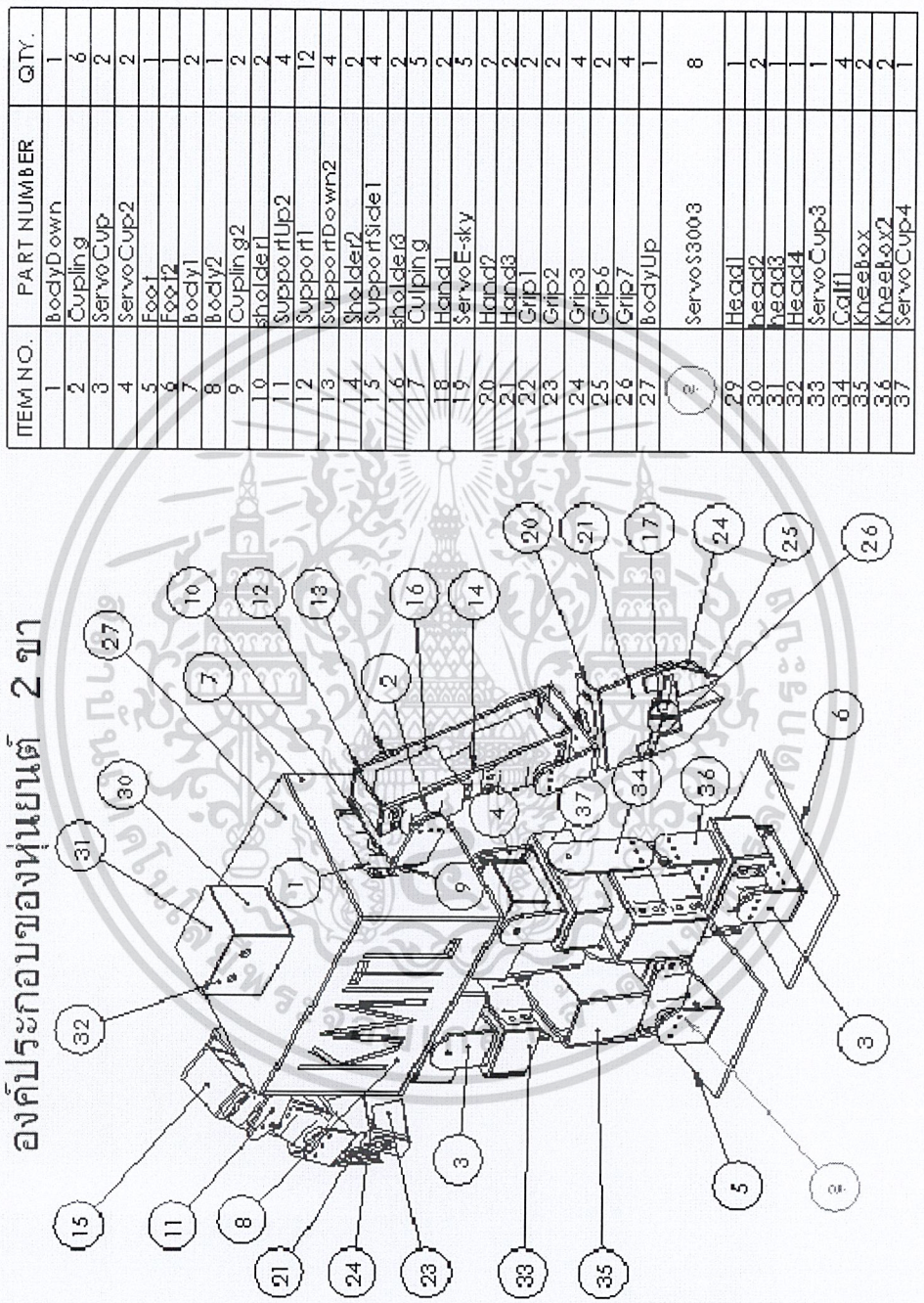
หุ่นยนต์ 2 ขา ประกอบด้วย 2 ส่วนคือ ฮาร์ดแวร์และส่วนซอฟต์แวร์

- ส่วนฮาร์ดแวร์ ประกอบด้วยเซอร์โวมอเตอร์, โครงสร้างหุ่นยนต์ 2 ขา , บอร์ดควบคุม ET-Easy168 STAMP , วงจรที่ใช้ในการควบคุมหุ่นยนต์ 2 ขา
- ส่วนซอฟต์แวร์ ประกอบด้วย โปรแกรมArduino



รูปที่ 2.24 โครงสร้างหุ่นยนต์ 2 ขา

2.5.2 รูปแบบโครงสร้างหุ่นยนต์ 2 ขา และการออกแบบ

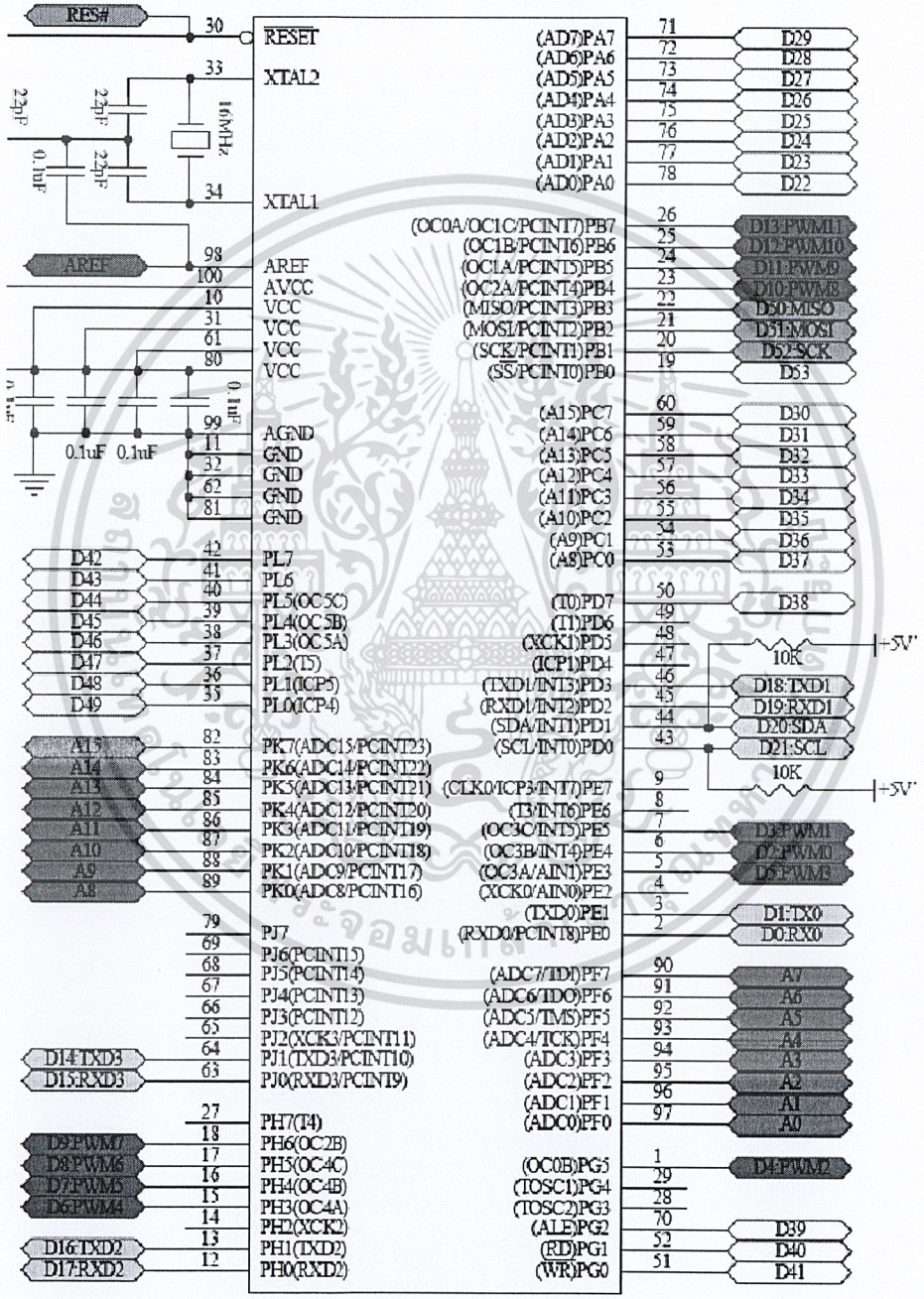


ITEM NO.	PART NUMBER	QTY.
1	BodyDown	1
2	Cupling	6
3	ServoCup	2
4	ServoCup2	2
5	Foot	1
6	Foot2	1
7	Body1	2
8	Body2	1
9	Cupling2	2
10	sholder1	2
11	SupportUp2	4
12	Support1	12
13	SupportDown2	4
14	Sholder2	2
15	SupportSide1	4
16	sholder3	2
17	Culping	5
18	Hand1	2
19	ServoE-sky	5
20	Hand2	2
21	Hand3	2
22	Grip1	2
23	Grip2	2
24	Grip3	4
25	Grip6	2
26	Grip7	4
27	BodyUp	1
28	ServoS3003	8
29	Head1	1
30	head2	2
31	Head3	1
32	Head4	1
33	ServoCup3	1
34	Calf1	4
35	Kneebox	2
36	KneeBox2	2
37	ServoCup4	1

รูปที่ 2.25 โครงสร้างหุ่นยนต์ 2 ขา และการออกแบบ

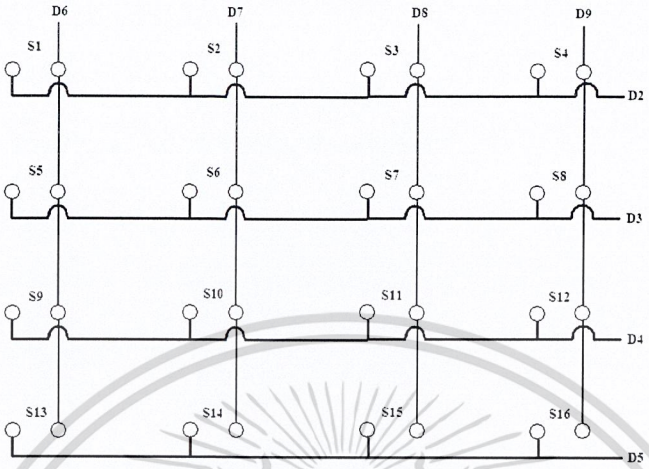
2.5.3 วงจรของหุ่นยนต์ 2 ขา

ในการควบคุมหุ่นยนต์ 2 ขา ได้ใช้ Microcontroller AVR MEGA 1280 ในการควบคุมการทำงาน



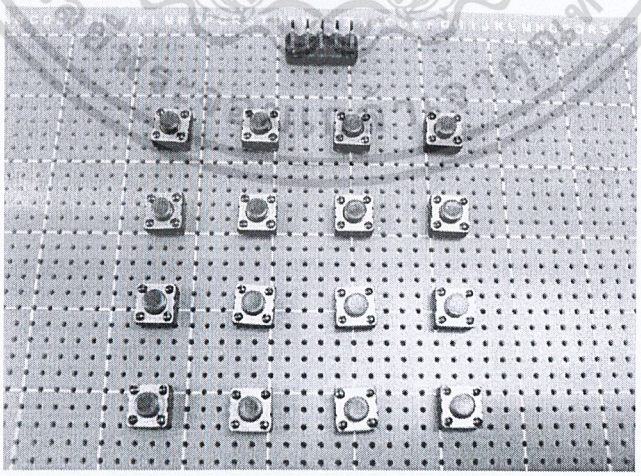
รูปที่ 2.26 Microcontroller AVR MEGA 1280

2.5.4 วงจรสวิตช์ควบคุมหุ่นยนต์ 2 ขา



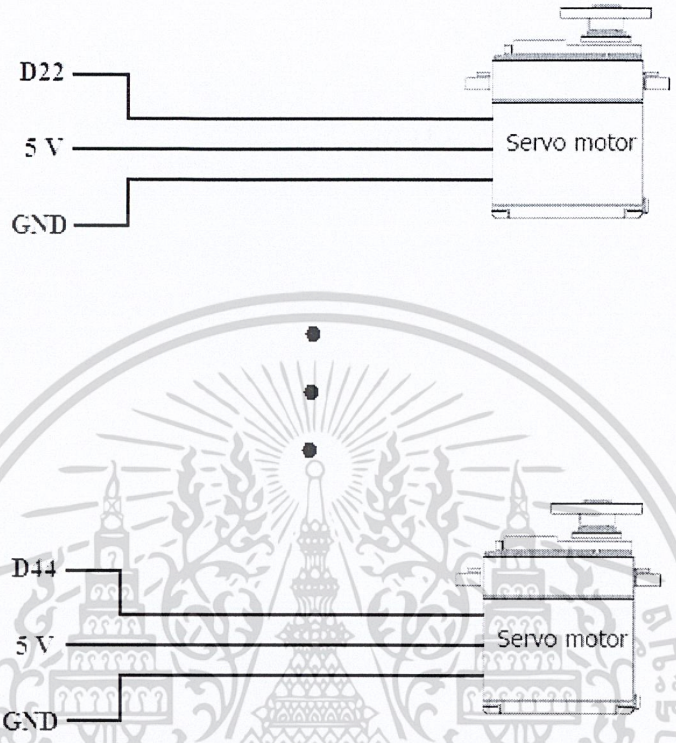
รูปที่ 2.27 วงจรสวิตช์ควบคุม

ในการควบคุมหุ่นยนต์ 2 ขา ในเบื้องต้น ในการจัดท่าหุ่นยนต์ ได้ทำการประยุกต์ใช้วงจรควบคุมการจัดท่าของหุ่นยนต์แขนกลมาใช้ในการจัดท่าการเดินของหุ่นยนต์ 2 ขา โดยใช้ Digital Pin D2-D9 ซึ่งแต่ละ Pin จะส่งสัญญาณ Logic 1 ออกมา เช่น สวิตช์ S1 จะมี Pin D2 กับ D6 เชื่อมกันอยู่ เมื่อกดปุ่ม S1 Logic จาก D2 และ D9 จะเชื่อมถึงกัน ทำให้ Microcontroller รู้ว่ามีการกดปุ่ม S1 ตามที่เขียนโปรแกรมไว



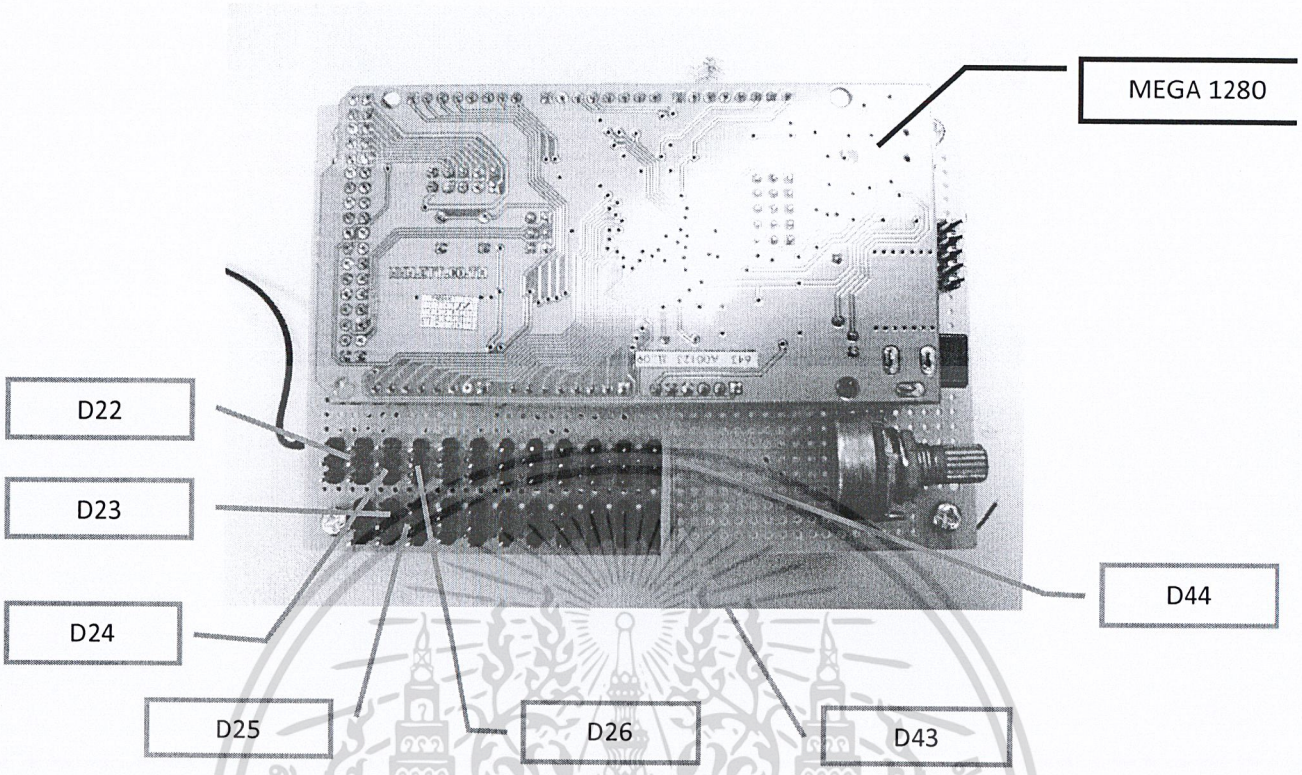
รูปที่ 2.28 บอร์ดสวิตช์ควบคุมการเคลื่อนไหวของหุ่นยนต์ 2 ขา

2.5.5 การเชื่อมต่อเซอร์โวมอเตอร์กับ pin ของ AVR MEGA 1280



รูปที่ 2.29 การเชื่อมต่อเซอร์โวมอเตอร์ กับ pinของ AVR MEGA 1280

หุ่นยนต์ 2 ขา ประกอบด้วยเซอร์โวมอเตอร์ จำนวน 23 ตัว ควบคุมเซอร์โวมอเตอร์โดยเชื่อมต่อสายสัญญาณกับ Pin D22 – D44 ที่กำหนดให้ปล่อยค่า Duty Cycle Pulse ตามที่เขียนโปรแกรมไว้



รูปที่ 2.30 pin D22-D24

บทที่ 3

ระบบลำเลียง

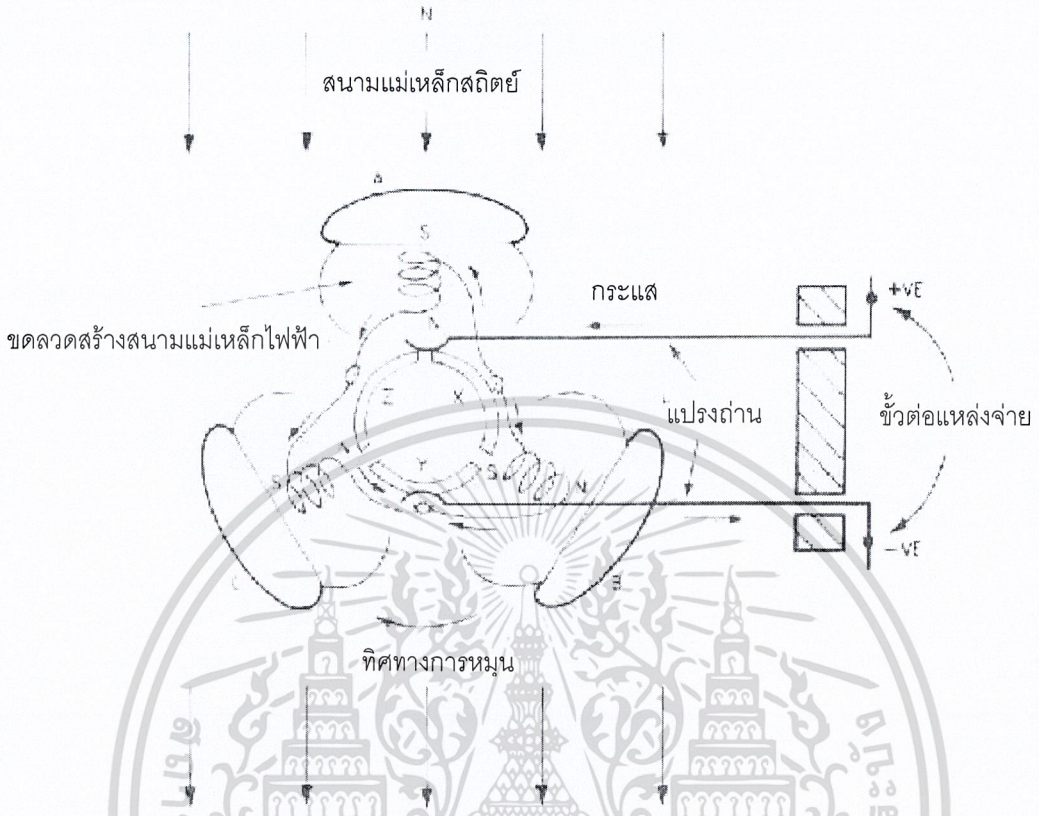
3.1 ทฤษฎีและความรู้ที่เกี่ยวข้อง

จากที่ได้กล่าวในบทที่ 1 แล้วว่า ก่อนที่จะมีการออกแบบและจำลองระบบกระบวนการผลิตอัตโนมัตินั้น จำเป็นต้องศึกษาองค์ประกอบต่างๆที่จำเป็นของระบบที่จะทำการจำลองให้เข้าใจเสียก่อน พบว่าในกระบวนการผลิตที่จำลองนั้นในแต่ละส่วนนั้นมีส่วนที่สำคัญหลายส่วนดังนั้นในบทนี้จะศึกษาและอธิบายถึงองค์ประกอบต่างๆ ที่จะนำไปใช้งานจริงในระบบที่ทำการจำลอง ซึ่งประกอบด้วยการประมวลผลภาพดิจิตอล มอเตอร์ไฟฟ้ากระแสตรง วงจรที่ใช้ในการรับส่งสัญญาณอินฟาเรด รวมถึงรายละเอียดของพอร์ตขนาน และพอร์ตอนุกรม และตัวควบคุมแบบต่างๆ

3.1.1 มอเตอร์กระแสตรง

3.1.1.1 หลักการทำงานของมอเตอร์กระแสตรง

รูปที่ 3.1 โครงสร้างทั่วไปของมอเตอร์กระแสตรง

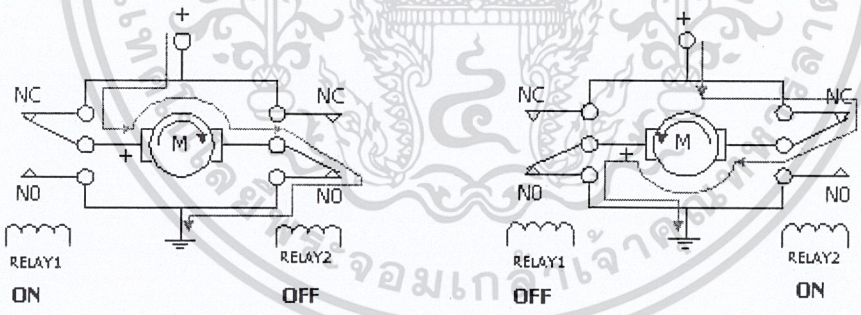


รูปที่ 3.2 โครงสร้างทั่วไปของมอเตอร์กระแสตรง

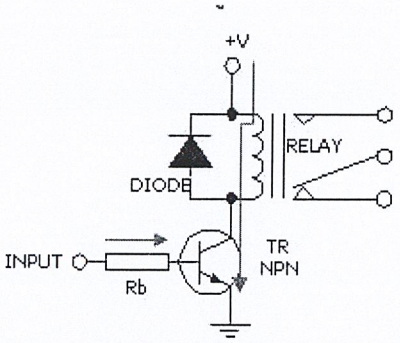
จากในรูปทางเดินของฟลั๊กซ์แม่เหล็ก และสนามแม่เหล็กจะเกิดจากแท่งแม่เหล็กเฟอร์ไรต์ 2 ชิ้นที่ขึ้นรูปเป็นแบบโค้งยึดติดกับตัวถังได้พอดี เพื่อที่จะให้เส้นแรงแม่เหล็กวิ่งเข้าสู่ใจกลางของมอเตอร์ได้ ดังนั้นความเข้มของแม่เหล็กจะขึ้นอยู่กับขนาดความหนาของแม่เหล็กด้วย ซึ่งส่งผลให้ฟลั๊กซ์แม่เหล็กวิ่งไปบนตัวถังโลหะ กระแสไฟฟ้าในขดลวดที่พันกับทุ่นโรเตอร์ก็จะทำให้เกิดสนามแม่เหล็กไฟฟ้าและต้านกับสนามแม่เหล็กถาวร จึงเกิดเป็นแรงบิดเพื่อที่จะหมุนทุ่นโรเตอร์ ให้ไปในทิศทางเดียวกันกับทิศทางของสนามแม่เหล็กที่มีแรงมากกว่า กระแสก็จะไหลผ่านไปยังทุ่นโรเตอร์ โดยผ่านแปรงถ่าน ซึ่งจะสัมผัสกับแหวนตัวนำในทุ่นโรเตอร์ และแหวนคอมมิวเตเตอร์ ซึ่งจะถูกแบ่งออกเป็น 3 เซกเมนต์ เพื่อที่จะทำหน้าที่นำกระแสเข้าขดลวดนั่นเอง

3.1.1.2 การขับและกลับทิศทางของมอเตอร์กระแสตรง

ในการใช้ไอซีไมโครคอนโทรเลอร์เป็นตัวควบคุมการหมุน และทิศทางของมอเตอร์กระแสตรงนั้น เราจะต้องมีส่วนของวงจร ที่เรียกว่าวงจรขับมอเตอร์ (Driver) ในส่วนของวงจรกลับทิศทางของมอเตอร์นั้น สามารถที่จะใช้รีเลย์ต่อวงจร สวิตซ์เพื่อกลับทิศทางของขั้วไฟกระแสตรง หรืออาจใช้อุปกรณ์สารกึ่งตัวนำที่เป็นวงจรขับกำลังเช่น ทรานซิสเตอร์ มอสเฟต แล้วแต่วิธีที่เราจะเลือกใช้งาน จากรูปเป็นการใช้รีเลย์ควบคุมการเปลี่ยนทิศทางการหมุนของมอเตอร์ โดยการควบคุมการปิด - เปิดที่รีเลย์ 2 ตัว ซึ่งจะทำหน้าที่กลับทิศทางของขั้วไฟที่ป้อนให้กับมอเตอร์ โดยการสลับการทำงานของรีเลย์ เช่นให้รีเลย์ตัวที่ 1 ทำงาน (ON) และรีเลย์ตัวที่ 2 หยุดทำงาน (OFF) จะทำให้มอเตอร์หมุนไปทางซ้าย และในทำนองเดียวกันถ้าหากรีเลย์ตัวที่ 1 หยุดทำงาน (OFF) และรีเลย์ตัวที่ 2 ทำงาน (ON) ก็จะทำให้มอเตอร์หมุนไปทางขวา

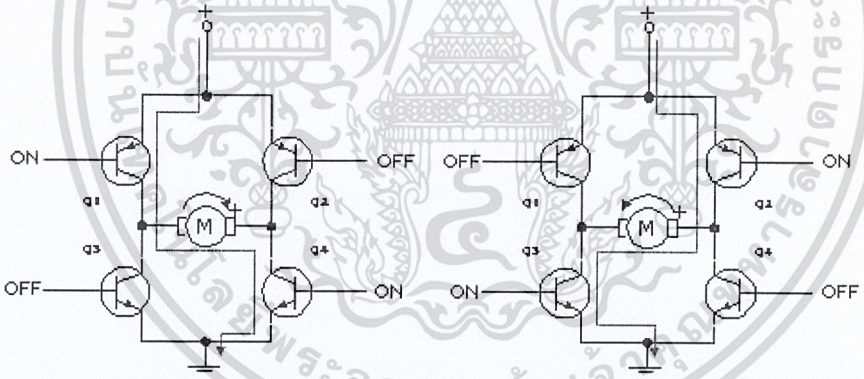


รูปที่ 3.3 การกลับทิศทางของมอเตอร์กระแสตรงโดยใช้รีเลย์



รูปที่ 3.4 การใช้ทรานซิสเตอร์เพื่อขับรีเลย์ให้ทำงาน

จากรูปที่ 3.4 เป็นวงจรขับรีเลย์โดยใช้ทรานซิสเตอร์ทำหน้าที่ขยายกระแส ด้วยเหตุผลเพราะไม่สามารถจะใช้ขาเอาต์พุตของไมโครคอนโทรลเลอร์ป้อนกระแสไฟที่ขดลวดของรีเลย์โดยตรงได้ เนื่องจากว่ากระแสที่จ่ายออกมาจากขา เอาต์พุตของไมโครคอนโทรลเลอร์มีค่าน้อยเกินไป ดังนั้นเราจึงต้องมีส่วนของวงจร ทรานซิสเตอร์เพื่อที่จะทำการขยายกระแสให้เพียงพอในการป้อนให้กับขดลวดของรีเลย์ ส่วนไดโอดนำมาต่อไว้สำหรับป้องกันแรงดันย้อนกลับที่เกิดจากการเหนี่ยวนำของสนาม-แม่เหล็กในขณะเกิดการยุบตัวซึ่งอาจจะทำให้ทรานซิสเตอร์เสียหายได้

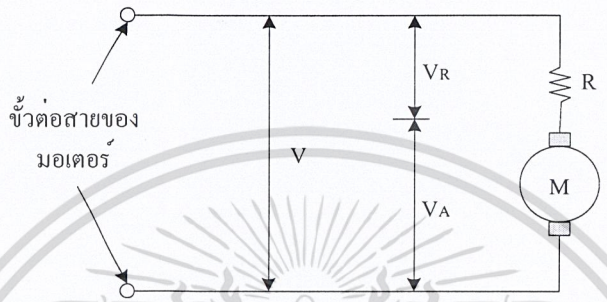


รูปที่ 3.5 การใช้ทรานซิสเตอร์เป็นวงจรขับและกำหนดทิศทางของมอเตอร์กระแสตรง

จากรูปที่ 3.5 เป็นวงจรลิเนียร์บริดจ์แอมป์ ซึ่งจะประกอบไปด้วยทรานซิสเตอร์กำลัง 4 ตัวที่ทำหน้าที่ขับ และควบคุมทิศทางการหมุนของมอเตอร์ ถ้าหากกำหนดให้ทรานซิสเตอร์ Q1 และ Q4 อยู่ในสภาวะทำงาน (Active) กระแสไฟฟ้าจะไหลผ่านทรานซิสเตอร์จากซ้ายไปขวา โดยผ่านมอเตอร์ กระแสตรงทำให้มอเตอร์หมุนไปทางขวา ในทำนองเดียวกันถ้าหากเราทำให้ทรานซิสเตอร์ Q2 และ Q3 อยู่ในสภาวะทำงานกระแสไฟฟ้าก็จะไหลจากทางขวาไปทางซ้ายซึ่งจะส่งผลให้มอเตอร์กลับทิศทางการหมุนจากทางขวาไปทางซ้าย

3.1.1.3 คุณสมบัติของมอเตอร์กระแสตรง

ในการอธิบายคุณสมบัติของมอเตอร์กระแสตรงให้ละเอียดนั้นต้องพิจารณาแรงดันที่ป้อนและความต้านทานของโรเตอร์ด้วย วงจรภายในของมอเตอร์เขียนได้ดังรูปที่ 3.6



รูปที่ 3.6 วงจรภายในของมอเตอร์กระแสตรง

โดยสมมติให้ทุ่นโรเตอร์ไม่มีความต้านทานอยู่เลย อนุกรมกับความต้านทานซึ่งในที่นี้ก็คือความต้านทานของขดลวดนั่นเอง แรงดันที่ขั้วต่อสายของมอเตอร์ก็คือผลบวกระหว่างแรงดันที่ทุ่นโรเตอร์ (V_A) และ แรงดันตกคร่อมความต้านทานขดลวด (V_R)

แรงดัน V_A ถูกเรียกว่า แรงเคลื่อนเหนี่ยวนำป้อนกลับ (Back EMF) ซึ่งเกิดขึ้นในโรเตอร์ขณะที่หมุน แรงดันที่เกิดขึ้นนี้เป็นไปตากกฎของการเหนี่ยวนำแม่เหล็กไฟฟ้าจากการเคลื่อนที่ของตัวนำในสนามแม่เหล็กสัมพันธ์กับแรงเคลื่อนเหนี่ยวนำแม่เหล็กและความเร็วในการเคลื่อนที่ของตัวนำแรงดันที่เกิดขึ้นจะมีขั้วตรงกันข้ามกับแรงดันที่ป้อนให้กับมอเตอร์และแปรผันตรงกับความเร็วในการหมุน ผลบวกของแรงดันที่ทุ่นโรเตอร์ (V_A) และแรงดันตกคร่อมขดลวด (V_R) ต้องเท่ากับแรงดันที่ป้อนให้กับมอเตอร์ (V)

$$V = V_A + V_R \qquad (V) \tag{3.1}$$

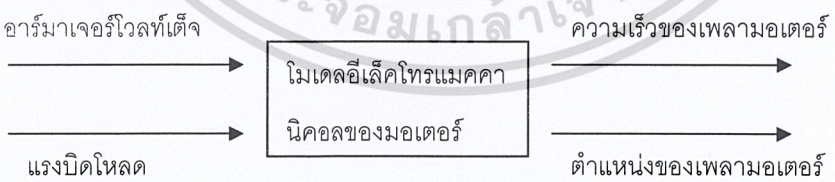
เมื่อพิจารณาตั้งแต่มอเตอร์หยุดนิ่ง ความเร็วมีค่าเป็นศูนย์ ดังนั้น $V_A = 0$, $V_R = V$ กระแสที่ไหลในมอเตอร์หาได้จาก

$$I=V_R/R \tag{3.2}$$

เมื่อมอเตอร์เริ่มหมุนจะมีความเร็ว และ V_A เพิ่มขึ้นเป็นเส้นตรงตามความเร็ว V_R ซึ่งมีค่าเท่ากับความแตกต่างระหว่าง V_A และ V จะเริ่มลดลงกระแส I ก็จะเริ่มลดลงเช่นกันขณะที่มอเตอร์ยังมีความเร่งอยู่ ความเร็วจะเพิ่มขึ้น แรงบิดจะลดลงจนกว่าจะถึงจุดซึ่งแรงบิดจองมอเตอร์รับภาระโหลดได้สมดุลพอดี ขณะที่มอเตอร์ไม่มีโหลดและหมุนอย่างอิสระจะมีเพียงค่าความฝืดของแบริ่งและแรงต้านอากาศทำให้ V_A เกือบเท่ากับค่า V

3.1.1.4 โมเดลคณิตศาสตร์ของดีซีมอเตอร์

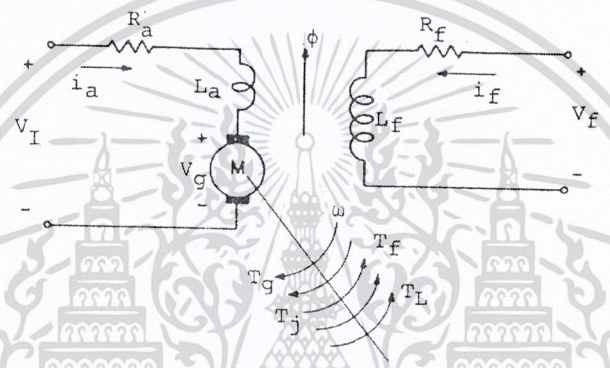
ดีซีมอเตอร์ที่ใช้ร่วมกับดีซีแอมปลิไฟล์ทั้งในระบบการบังคับตำแหน่งและการบังคับความเร็วมักจะได้รับการประยุกต์ใช้เป็นส่วนประกอบสร้างกำลังงานในระบบการนำร่องและระบบบังคับต่างๆ และเนื่องจากวิทยาการเกี่ยวกับสารแม่เหล็ก และการขยายด้วยโซลิดสเตททำให้ดีซีมอเตอร์แบบแม่เหล็กถาวรได้รับความนิยมใช้เป็นส่วนประกอบการขับเคลื่อนในระบบการบังคับแบบปิดลูฟต่างๆ มากขึ้น การออกแบบและการชดเชยระบบดังกล่าวได้อย่างเหมาะสมจะต้องใช้โมเดลทางคณิตศาสตร์ของส่วนประกอบทั้งหมดในระบบ



รูปที่ 3.7 อินพุตและเอาต์พุตของโมเดลทางคณิตศาสตร์ของมอเตอร์

3.1.1.5 โมเดลอิเล็กโทรแมคคานิคอล

ส่วนสำคัญของดีซีมอเตอร์แบบฟิลด์แยกกระตุ้นมีโมเดลดังแสดงในรูป



รูปที่ 3.8 โมเดลของดีซีมอเตอร์แบบฟิลด์แยกกระตุ้น

R_a: ความต้านทานของอาร์เมเจอร์

L_a: อินดัคแต้นซ์ของอาร์เมเจอร์

V_g: โวลท์เต็จกำเนิดในอาร์เมเจอร์(โวลท์เต็จย้อนกลับ)

R_f: ความต้านทานของฟิลด์

L_f: อินดัคแต้นซ์ของฟิลด์

ϕ : ช่องว่างอากาศของเส้นแรงสนามแม่เหล็ก

ω : ความเร็วของเพลาอาร์เมเจอร์

T_g: แรงบิดที่พัฒนาขึ้นในมอเตอร์

T_f: แรงบิดเสียดทานของมอเตอร์

T_J: แรงเฉื่อยของมอเตอร์

T_L: แรงบิดโหลดบนเพลาของมอเตอร์

ขั้นแรกเราจะหาสมการพื้นฐานโมเดลของดีซีมอเตอร์ได้จากลูฟของอาร์เมเจอร์

$$V_i(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + V_g(t) \tag{3.3}$$

เทอมโวลท์เต็จ $V_g(t)$ ในสมการ (3.3) คือโวลท์เต็จย้อนกลับของมอเตอร์ซึ่งเกิดขึ้นเมื่อเส้นลวดตัวนำของอาร์มาเจอร์หมุนตัดเส้นแรงแม่เหล็กซึ่งเกิดขึ้นในกระแสของฟิลด์ (i_f) ตามกฎของฟาราเดย์ลูฟของเส้นลวดตัวนำหมุนในฟิลด์แม่เหล็กคงที่จะมีการเหนี่ยวนำโวลท์เต็จขึ้นในขดลวดนั้น

$$V_i(t) = \frac{d\lambda(t)}{dt} \tag{3.4}$$

เมื่อ $\lambda(t)$ คือเส้นแรงแม่เหล็กที่ลิงเคจ(linkages) ไปยังขดลวดและ t คือเวลาในการหมุนของค็อมนิวเทเตอร์ของมอเตอร์ การควบคุมวงจรของแต่ละส่วนของตัวนำในโรเตอร์จะเกิดโวลท์เต็จขึ้นในส่วนของตัวนำนั้นตามสมการ (3.4) เมื่อ $\frac{d\lambda(t)}{dt}$ จะเป็นสัดส่วนต่อเส้นแรงแม่เหล็กในช่องว่างอากาศและความเร็วเชิงมุม $\omega(t)$ เราจะได้ว่า

$$V_g(t) = K\phi(t)\omega(t) \tag{3.5}$$

สมมติให้กระแสของฟิลด์มีค่าคงที่และไม่คิดถึงส่วนการเปลี่ยนแปลงในเส้นแรงฟิลด์เนื่องจากอาร์เมเจอร์รีแอกชั่นเส้นแรงฟิลด์ก็จะมีค่าคงที่ดังน้นสมการ (3.5) ก็จะเป็น

$$V_g(t) = K_e\omega(t) \tag{3.6}$$

เมื่อเราสมมติให้เส้นแรงของฟิลมีค่าคงที่ แรงบิดของแม่เหล็กไฟฟ้าซึ่งเกิดขึ้นแก่โรเตอร์ของมอเตอร์จะเป็นสัดส่วนกับกระแสของอาร์มาเจอร์

$$T_g(t) = K_t\, i_a(t) \tag{3.7}$$

เมื่อ K_t คือ ค่าคงที่ของแรงบิดของมอเตอร์

กำลังงานเชิงกลที่เกิดขึ้นในโรเตอร์คือผลคูณของแรงบิดที่เกิดขึ้นและความเร็วเชิงมุม

$$P_g(t) = T_g(t)\, \omega(t) \tag{3.8}$$

กำลังงานเชิงกลที่เกิดขึ้นในโรเตอร์ทั้งหมดนี้จะจ่ายไปยังโหลดที่ต่ออยู่กับเพลาของมอเตอร์แต่กำลังงานนี้บางส่วนจะสูญเสียไปในมอเตอร์ การสูญเสียจากแรงเสียดทาน หมายถึงความหน่วงเนื่องจากลมที่มีต่อโรเตอร์ แรงเสียดทานตัวรองรับโรเตอร์ กระแสที่ไหลวนในเหล็กของโรเตอร์และฮีสเทรีซิส โดยแรงบิดต่างๆแสดงดังนี้

รูปที่ 3.9 แรงบิดต่างๆที่เกิดขึ้นต่อโหลดของมอเตอร์

$T_g(t)$: แรงบิดของมอเตอร์

$T_f(t)$: แรงบิดที่ต้องชนะการสูญเสียเนื่องจากการเสียดทาน

$T_J(t)$: แรงบิดเพื่อใช้เพิ่มอัตราเร่งแก่ความเฉื่อยของโหลด

$T_L(t)$: แรงบิดโหลด

ในช่วงเวลาใดๆก็ตาม แรงบิดของมอเตอร์จะต้องเท่ากับและมีทิศทางตรงข้ามกับผลรวมของแรงบิด $T_f(t)$ $T_J(t)$ และ $T_L(t)$ ดังนั้น

$$T_g(t) = T_f(t) + T_L(t) + J\frac{d\omega(t)}{dt} \tag{3.9}$$

เมื่อ J คือผลรวมของโมเม้นท์แรงเฉื่อยของโรเตอร์และโหลดที่ต่ออยู่ที่เพาของมอเตอร์

ผลรวมของแรงบิดเสียดทานที่ประกอบกันขึ้นที่เพลาของมอเตอร์ซึ่งเป็นลิเนียร์ฟังก์ชันกับความเร็วเชิงมุมของโรเตอร์เรียกว่า ส่วนประกอบของวิสคอสฟริคชั่นและมักจะอยู่ในเทอมที่แยกออกจากฟริคชั่นอื่นๆ ซึ่งแสดงได้ด้วยสมการต่อไปนี้

$$T_g(t) = T_f(t) + T_L(t) + J\frac{d\omega(t)}{dt} + B\omega(t) \tag{3.10}$$

เมื่อ B คือสัมประสิทธิ์ของวิสคอสฟริคชั่นของมอเตอร์และโหลดที่ต่ออยู่กับเพลาของมอเตอร์ $T_f(t)$ คือผลรวมของฟริคชั่นของโหลดและมอเตอร์ทั้งหมด มีแรงต้านของลมและการสูญเสียกำลังในเหล็กของเพลามอเตอร์ยกเว้นวิสคอสฟริคชั่น

สมการ (3.3) (3.6) (3.7) และ (3.9) เป็นชุดสมการพื้นฐานของดีซีมอเตอร์โมเดลและสมการเหล่านี้เราสามารถจะหาทรานสเฟอร์ฟังก์ชั่นของดีซีมอเตอร์ได้ โดยใส่ลาปลาสทรานสฟอร์มทั้งสองข้างของชุดสมการพื้นฐานได้เป็น

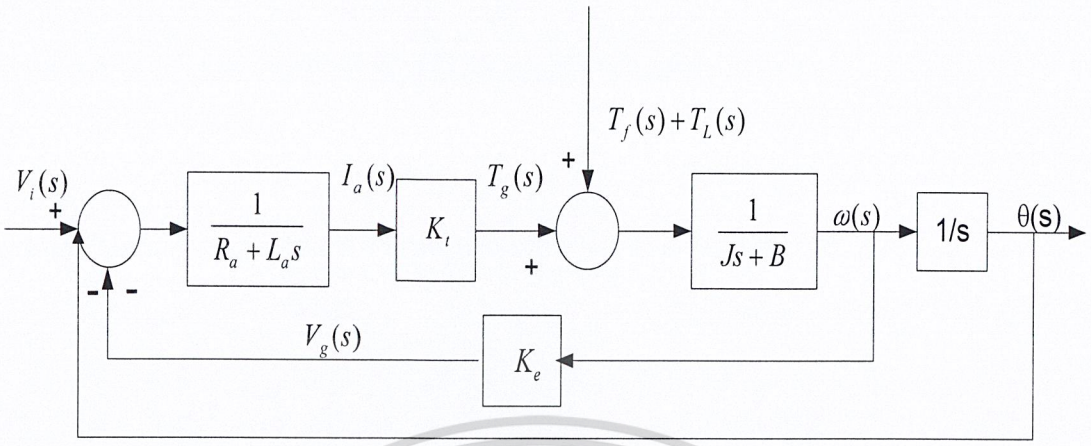
$$V_i(s) - V_g(s) = (R_a + sL_a)\, I_a(s) \tag{3.11}$$

$$V_g(s) = K_e\omega(s) \tag{3.12}$$

$$T_g(s) = K_t\, I_a(s) \tag{3.13}$$

$$T_g(s) - T_f(s) - T_L(s) = (B + sJ)\,\omega(s) \tag{3.14}$$

สามารถเขียนเป็นบล็อกไดอะแกรมที่แสดงสมการพื้นฐานเหล่านี้ได้ดังนี้



รูปที่ 3.10 บล็อกไดอะแกรมของดีซีมอเตอร์โมเดล

3.1.1.6 ข้อสังเกต

สมมติว่าโวลท์เต็จที่ป้อนให้กับวงจรอาร์มาเจอร์ของมอเตอร์มีค่าคงที่ดังนั้นมอเตอร์จะหมุนด้วยความเร็วคงที่คือทำงานอยู่ที่สภาวะสงบนิ่งด้วยโหลดที่คงที่ กำลังงานเชิงกลที่เกิดขึ้นโดยโรเตอร์จะหาได้จากสมการ (3.8) จะได้

$$P_g = T_g \omega = K_t I_a \omega \tag{3.15}$$

เมื่อทุกเทอมในสมการสุดท้ายมีค่าคงที่เนื่องจากมอเตอร์ทำงานอยู่ที่สภาวะสงบนิ่งกำลังไฟฟ้าที่ถูกดูดกลืนโดยอาร์เมเจอร์ต้องเท่ากับ

$$P = V_g I_a = K_e \omega I_a \tag{3.16}$$

ดังนั้นเราจะได้ว่ากำลังงานเชิงกลที่เกิดขึ้นต้องเท่ากับกำลังงานไฟฟ้าที่ถูกดูดกลืนในโรเดอร์คือ สรุปได้ว่า $K_e = K_t$

บล็อกไดอะแกรมของรูปที่ 3.10 แสดงถึงระบบที่มีสองอินพุท และมีเอาต์พุทเป็นความเร็วเชิงมุม $\omega(s)$ และการเคลื่อนที่แบบเชิงมุม จากรูปที่ 3.10 ความเร็วเอาท์พุทของระบบเขียนได้เป็น

$$\omega(s) = G_1(s)V_i(s) + G_2(s)[T_f(s) + T_L(s)] \tag{3.17}$$

เมื่อ

$$G_1(s) = \left.\frac{\omega(s)}{V_i(s)}\right|_{T_f(s)+T_L(s)=0} \tag{3.18}$$

$$G_2(s) = \left.\frac{\omega(s)}{T_f(s)+T_L(s)}\right|_{V_i(s)=0} \tag{3.19}$$

$G_1(s)$ คือทรานสเฟอร์ฟังก์ชั่นระหว่างโวลท์เต็จและความเร็ว

$$G_1(s) = \frac{\omega(s)}{V_i(s)} = \frac{K_t}{(L_a s + R_a)(Js + B) + K_t K_e}$$

$$\mathbf{q} = \frac{K_m}{\alpha S^2 + \beta s + 1} \tag{3.20}$$

เมื่อ

$$K_m = \frac{K_t}{R_a B + K_t K_e}$$

$$\alpha = \frac{L_a J}{R_a B + K_t K_e}$$

$$\beta = \frac{R_a J + L_a B}{R_a B + K_t K_e}$$

สมการ (3.20) เป็นโวลท์เต็จทรานสเฟอร์ฟังก์ชั่นของดีซีมอเตอร์ในเมื่อสมมติว่า T_f และ T_L มีค่าเป็นศูนย์ สมการ (3.20) สามารถเขียนใหม่ได้เป็น

$$G_1(s) = \frac{K_t}{R_a B(1+\tau_e s)(1+\tau_m s) + K_t K_e} \tag{3.21}$$

เมื่อ $\tau_e = \frac{L_a}{R_a}$ = ไทม์คอนสแต้นท์ทางไฟฟ้า

$\tau_m = \frac{J}{B}$ = ไทม์คอนสแต้นท์ทางเชิงกล

ถ้าอินดัคแต้นท์ของอาร์มาเจอร์มีค่าน้อย ไทม์คอนสแต้นท์ทางไฟฟ้าสามารถตัดทิ้งได้และได้สมการเป็น

$$G_V(s) = \frac{\omega(s)}{V_i(s)} = \frac{K_t}{R_a(Js+B) + K_t K_e} \tag{3.22}$$

$$= \frac{K_m}{\tau s + 1} \tag{3.23}$$

เมื่อ $\tau = \frac{R_a J}{R_a B + K_t K_e}$

ในสมการ (3.23) ค่าคงที่ K_m อาจเรียกได้ว่าเป็นค่าคงที่ของมอเตอร์

ทรานสเฟอร์ฟังก์ชั่นแรงบิดโหลด $G_2(s)$ หาได้เป็น

$$G_2(s) = \frac{\omega(s)}{T_f(s) + T_L(s)} = \frac{-\frac{1}{Js+B}}{1 + \frac{K_t K_e}{(Js+B)(L_a s + R_a)}}$$

$$= \frac{-\frac{R_a}{K_t} K_m \left[\frac{L_a}{R_a} s + 1\right]}{\alpha s^2 + \beta s + 1} \tag{3.24}$$

ซึ่งถ้าอินดัคแต้นท์ของอาร์มาเจอร์ไม่นำมาคิด จะทำให้ได้สมการ ดังนี้

$$G_L(s) = \frac{\omega(s)}{T_f(s) + T_L(s)} = \frac{-\frac{R_a}{K_t} K_m}{\tau s + 1} \tag{3.25}$$

ซึ่งจากสมการที่ (3.16) เมื่อให้ค่าของ ว่า T_f และ T_L มีค่าเป็นศูนย์จะทำให้ค่าทรานสเฟอร์ฟังก์ชั่นมีค่าดังนี้

$$\omega(s) = G_v(s)V_i(s) = \frac{K_m}{\tau s + 1} V_i(s) \tag{3.26}$$

โมเดลคณิตศาสตร์ในการควบคุมตำแหน่งของมอเตอร์ จาก transfer function โมเดลคณิตศาสตร์ของดีซีมอเตอร์ ซึ่งจะได้ model ของการควบคุมความเร็วของมอเตอร์ในรูปของสมการอันดับหนึ่งเป็นดังนี้ คือ

$$\omega(s) = G_v(s)V_i(s) = \frac{K_m}{\tau s + 1} V_i(s) \tag{3.27}$$

จากสมการข้างต้นดังกล่าวนั้นจะเห็นถึงความสัมพันธ์ระหว่างความเร็วเชิงมุม (output) และ ค่าแรงดันที่ป้อน (input)

และในการควบคุมตำแหน่งจะมีการผ่านตัว Integrator (1/s) ทำให้ได้เอาท์พุตคือ Θ(s) ซึ่งจากรูปที่ 3.10. และ transfer function ที่ได้จากหัวข้อข้างต้นนั้นเมื่อทำการผ่าน Integrator (1/s) เข้าไปจะทำให้ได้ค่าของ output เป็นมุมในการเคลื่อนที่ของมอเตอร์ ซึ่งเขียนในรูปของสมการได้ดังนี้

$$\theta(s) = \frac{\omega(s)}{s} = \left[\frac{1}{s}\right]\left[\frac{K_m V_i(s)}{\tau s + 1}\right] \tag{3.28}$$

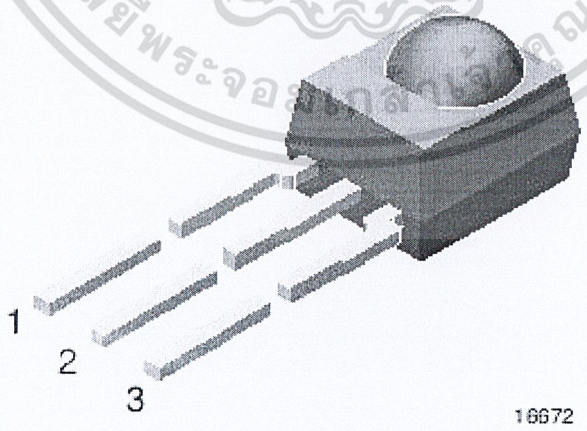
ส่วนในการหาโมเดลมอเตอร์ที่ใช้ในการทดลองนั้น เรากำหนดให้โมเดลของมอเตอร์เป็นโมเดลมอเตอร์อันดับหนึ่ง (First Order) ซึ่งเราได้ทำการทดลองเพื่อหาโมเดลที่เป็นลักษณะอันดับหนึ่ง

3.1.2 สัญญาณอินฟาเรด

รังสีอินฟราเรด (อังกฤษ: Infrared (IR)) มีชื่อเรียกอีกชื่อว่า รังสีใต้แดง หรือรังสีความร้อน เป็นคลื่นแม่เหล็กไฟฟ้าที่มีความยาวคลื่นอยู่ระหว่างคลื่นวิทยุและ แสงมีความถี่ในช่วง 1011 – 1014 เฮิร์ตซ์ มีความถี่ในช่วงเดียวกับไมโครเวฟ มีความยาวคลื่นอยู่ระหว่างแสงสีแดงกับคลื่นวิทยุสสารทุกชนิดที่มีอุณหภูมิ อยู่ระหว่าง -200 องศาเซลเซียสถึง 4,000 องศาเซลเซียส จะปล่อยรังสีอินฟาเรดออกมา คุณสมบัติเฉพาะตัวของรังสีอินฟราเรด เช่น ไม่เบี่ยงเบนในสนามแม่เหล็กไฟฟ้า ที่แตกต่างกันก็คือคุณสมบัติที่ขึ้นอยู่กับความถี่ คือยิ่งความถี่สูงมากขึ้น พลังงานก็สูงขึ้นด้วย ดังนั้นในการใช้ประโยชน์ ใช้ในการควบคุมเครื่องใช้ระบบไกล (remote control) สร้างกล้องอินฟราเรดที่สามารถมองเห็นวัตถุในความมืดได้ เช่น อเมริกาสามารถใช้กล้องอินฟราเรดมองเห็นเวียตกงได้ตั้งแต่สมัยสงครามเวียตนาม และสัตว์หลายชนิดมีนัยน์ตารับรู้รังสีชนิดนี้ได้ ทำให้มองเห็นหรือล่าเหยื่อได้ในเวลากลางคืน

3.1.2.1 สัญญาณอินฟราเรดที่เลือกใช้ TSOP34836

เป็นอุปกรณ์ขนาดเล็กที่ใช้ในการรับสัญญาณอินฟราเรดโดยข้อมูลจะถูกส่งผ่านหลอดอินฟราเรด ซึ่งการส่งข้อมูลผ่านอินฟราเรดเราต้องทำการเลือกความถี่ช่วงสัญญาณ เพื่อที่จะทำการเลือกตัวรับได้ว่าจะให้รับสัญญาณอินฟราเรดในช่วงใด ซึ่ง TSOP34836 จะสามารถรับสัญญาณอินฟราเรดได้ในช่วง 38KHz โดยลักษณะของ TSOP34836 เป็นดังรูปที่ 3.10



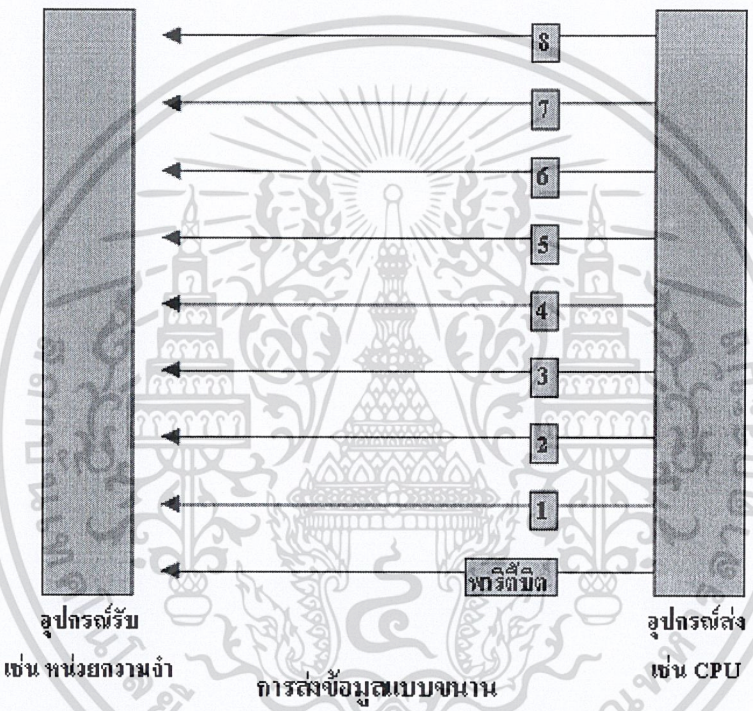
รูปที่ 3.11 ลักษณะของ TSOP34836

3.1.3 การส่งสัญญาณข้อมูล

การส่งสัญญาณข้อมูลแบ่งออกได้เป็น 2 ประเภทคือ

1. การส่งแบบขนาน
2. การส่งแบบอนุกรม

3.1.3.1 การส่งแบบขนาน (Parallel Transmission)



รูปที่ 3.12 การส่งข้อมูลแบบขนาน

การส่งแบบขนานนั้นจะทำการส่งข้อมูลทีละหลาย ๆ บิต เช่น ส่ง 10011110 ทั้ง 8 บิต ออกไปพร้อมกันโดยผ่านสายส่งข้อมูลที่มี 8 เส้น ส่วนการส่งข้อมูลแบบอนุกรม ข้อมูลจะถูกส่งออก ไปทีละบิตต่อเนื่องกันไป เช่นถ้าข้อมูลคือ 10011110 เลข 0 ทางขวามือสุดเป็นบิตที่ 1 เรียงลำดับไปจนครบ 8 บิต โดยการส่งนั้นจะใช้สายส่งเส้นเดียวเท่านั้น ดังภาพ แสดงการส่งข้อมูลแบบขนานและแบบอนุกรม ตัวอย่างการใช้งานที่เห็นชัดของการส่งข้อมูลแบบขนาน เช่น การต่อเครื่องพิมพ์เข้ากับ

เครื่องคอมพิวเตอร์ ซึ่งปกติจะใช้สายยาว 5 เมตร ถึง 10 เมตรเท่านั้นและตัวอย่างการส่งข้อมูลแบบอนุกรม เช่นการต่อเทอร์มินัลเข้ากับคอมพิวเตอร์แม่ที่อยู่ห่างกันสัก 100 เมตร ซึ่งทำให้ประหยัดสาย

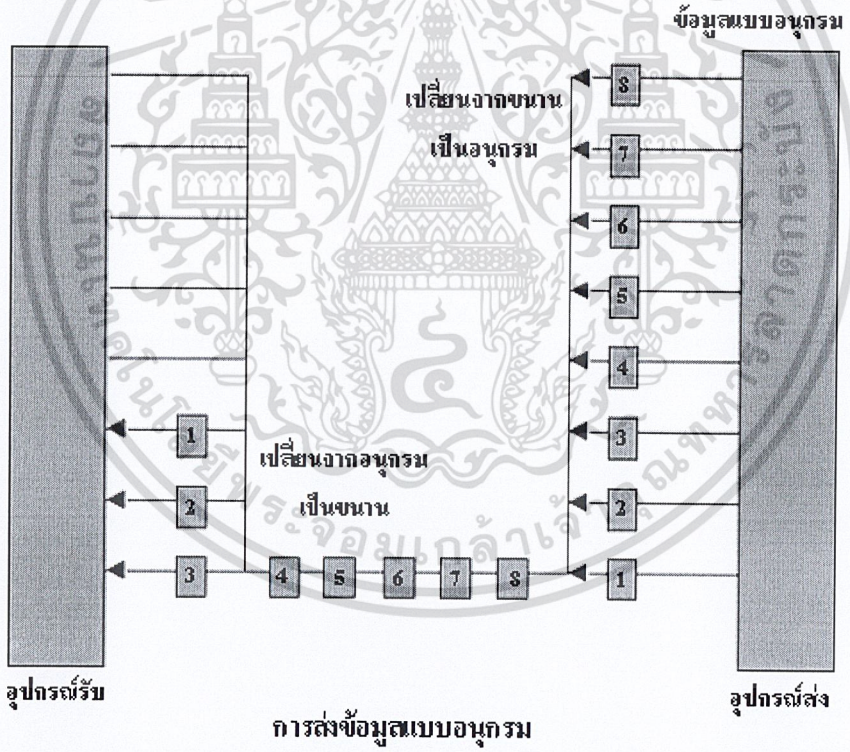
ข้อดี คือ สามารถส่งข้อมูลได้รวดเร็ว เพราะส่งครั้งละ 8 บิท

ข้อเสีย คือ ใช่ส่งแต่เฉพาะใกล้ ๆ เท่านั้น ราคาแพง

3.1.3.2 การส่งแบบอนุกรม (Serial Transmission)

การส่งข้อมูลแบบอนุกรมแบ่งออกได้เป็น 2 ประเภทได้แก่

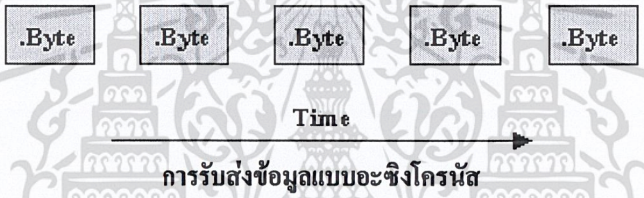
1. การส่งข้อมูลแบบอะซิงโครนัส (asynchronous data transmission)
2. การส่งข้อมูลแบบซิงโครนัส (synchronous data transmission)



รูปที่ 3.13 การส่งข้อมูลแบบอนุกรม

การส่งข้อมูลแบบอะซิงโครนัส (asynchronous data transmission)

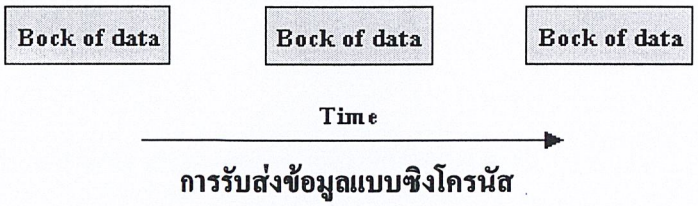
มักจะใช้กับเทอร์มินัลธรรมดา (dumb terminal) ไว้สำหรับรับข้อมูลจากคอมพิวเตอร์แม่และแสดงผลที่จอ โดยไม่สามารถเปลี่ยนแปลงข้อมูลได้ การส่งข้อมูลแบบนี้มักจะมีอัตราในการรับส่งข้อมูลที่แน่นอนมีหน่วยเป็นบิตต่อวินาที (bit per second) เมื่ออุปกรณ์อะซิงโครนัสจะส่งข้อมูล 1 ไบต์ ก็จะส่งบิตเริ่มต้น (start bit) ก่อน ซึ่งมักจะเป็น "0" และตามด้วยข้อมูลทั้ง 8 บิตใน 1 ไบต์ แล้วจึงจะส่งบิตหยุด (stop bit) ซึ่งมักจะเป็น "1" บิตทั้งหมดนี้ จะรวมกันเป็น 10 บิต ในการส่งข้อมูลเรียงตามลำดับดังนี้ 1 บิตเริ่มต้น 7 บิตข้อมูล (data bit) 1 บิต ภาวะเสมอมูล และ 1 บิตหยุด กระบวนการเหล่านี้จะห่างกัน 1 วินาที ที่จะส่งข้อมูลชุดต่อไป ซึ่งก็หมายถึงว่าเมื่อคอมพิวเตอร์แม่ได้รับบิตเริ่มต้น ก็คาดหวังว่าจะได้รับอีก 9 บิตภายในเวลา 1 วินาที



รูปที่ 3.14 การรับส่งข้อมูลแบบอะซิงโครนัส

ในระบบนี้จะเกี่ยวข้องกับเวลาว่าเมื่อไรบิตต่อไปจะมาถึง ถ้าไม่ตรงตามที่กำหนดไว้ การส่งข้อมูลก็จะล้มเหลว ระบบนี้เหมาะในการส่งอักขระจากเทอร์มินัลมายังคอมพิวเตอร์แม่ทันที เคาะแป้นพิมพ์ของเทอร์มินัลก็จะรู้ทันทีว่าจะต้องส่งไบต์ใดโดยเติมบิตเริ่มต้นและบิตหยุดที่หัวและท้ายของข้อมูลไบต์นั้น ตามลำดับให้ครบ 10 บิตที่จะส่ง ในการส่งข้อมูลอัตราการส่งข้อมูลอาจจะเป็น 110, 300, 1,200, 2,400, 4,800, 9,600, 19,200 บิตต่อวินาที โดยที่ทางด้านส่งและด้านรับจะต้องมีการตั้งค่าความเร็วให้เท่ากัน

การส่งข้อมูลแบบซิงโครนัส(synchronous data transmission)



รูปที่ 3.15 การรับส่งข้อมูลแบบซิงโครนัส

จะไม่ใช้บิตเริ่มต้นและบิตหยุด จะไม่มีการการหยุดชั่วขณะระหว่างอักขระ จะใช้วิธีให้จังหวะเวลาทั้งสองทางที่ติดต่อกัน มีอยู่สองวิธีที่ปฏิบัติคือ ใช้อักขระซิงก์ (sinc character) หรือใช้สัญญาณนาฬิกา (clock singal) การใช้อักขระซิงก์ไว้หน้าบล็อก (block) ของอักขระที่ใหญ่ โดยการใส่อักขระซิงก์ไว้หน้าบล็อกของข้อมูลอักขระซิงก์นี้เป็นบิตจำนวนหนึ่งที่ทางอุปกรณ์เครื่องรับสามารถใช้ ในการกำหนดอัตราเร็วของข้อมูลให้ตรงกับทางอุปกรณ์เครื่องส่ง การใช้สัญญาณนาฬิกาของด้านส่ง และสัญญาณนาฬิกาของด้านรับจะใช้คนละสายหรือคนละช่องสัญญาณในการส่งข่าวสารเกี่ยวกับเวลาของข้อมูลที่จะส่ง โดยทั่วไปการส่งข้อมูลแบบซิงโครนัสจะทำงานภายใต้การควบคุมของโปรโตคอลในระบบนั้น ๆ และนิยมใช้กับเทอร์มินัลฉลาดและเทอร์มินัลอัจฉริยะ การส่งข้อมูลจะนำข้อมูล 1 ไบท์ มาส่งออกไปตามสายไฟฟ้าเรียงกันไปจนครบ 8 บิท ซึ่งเท่ากับ 1 ตัวอักษร

ข้อดี คือ สามารถส่งได้ระยะทางไกลมากกว่า Parallel Tranmission

ข้อเสีย คือ ความเร็วในการรับส่งข้อมูลมีจำกัด ต้องคำนึงถึงรายละเอียดในการรับส่งข้อมูล

3.2 หลักการทำงานของระบบลำเลียง

ในส่วนของระบบลำเลียง จะประกอบไปด้วยหลายส่วนทำงานประกอบเข้าด้วยกัน โดยในแต่ละส่วนนั้นจะทำหน้าที่แตกต่างกันออกไป แล้วจะมีการติดต่อสื่อสารกันเพื่อให้สามารถทำการประสานกันได้อย่างดี ซึ่งจะแบ่งการควบคุมออกเป็นส่วนๆดังนี้

ส่วนที่ทำการติดต่อสื่อสารกับรถ

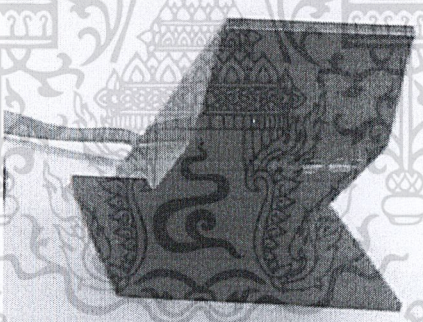
ส่วนที่ทำการควบคุมการส่งสัญญาณ

ส่วนของการแสดงผลและสั่งการระบบ

3.2.1 ส่วนที่ทำการติดต่อสื่อสารกับรถ

ส่วนที่จะทำการติดต่อสื่อสารกับรถนั้นจะประกอบไปด้วยวงจร 2 วงจร คือ

3.2.1.1วงจรสถานีรับ – ส่งสัญญาณ



รูปที่ 3.16 สถานีรับ - ส่งสัญญาณ

รายละเอียดของสถานี

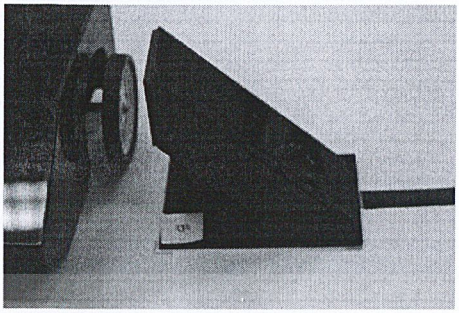
ไมโครคอนโทรลเลอร์ PIC16F690

ภาษาที่เขียน ภาษาซี

โปรแกรมที่ใช้เขียนคำสั่ง PIC C Compiler (CCS compiler)

PICkit 2 V 2.51

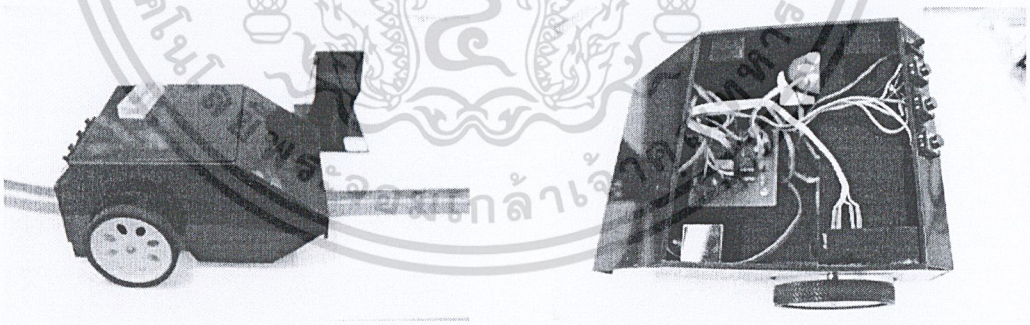
หลักการทำงาน



รูปที่ 3.17 ส่วนรับส่ง-สัญญาณ

วงจรสถานีส่วนรับ – ส่งสัญญาณ ติดต่อสื่อสารกับส่วนที่ทำการควบคุมการส่งสัญญาณ และติดต่อสื่อสารกับรถขนส่งสินค้า โดยนั้นจะทำหน้าที่ตรวจจับว่าที่สถานีนั้นมีอยู่หรือไม่ ด้วยการรับสัญญาณอินฟาเรด โดยส่วนที่ทำการควบคุมการส่งสัญญาณจะส่งสัญญาณเข้ามาที่สถานีเพื่อให้ทำการตรวจสอบรถขนส่งสินค้าที่จอดอยู่ที่สถานี หากสถานีไม่มีสัญญาณอินฟาเรดเข้ามาจะแจ้งไปยังส่วนที่ทำการควบคุมการส่งสัญญาณว่า สถานีนั้นว่าง ไม่มีรถจอด แต่หากมีการรับสัญญาณที่ส่งมาจากรถขนส่งสินค้า ทางสถานีจะแจ้งกลับไปว่า ที่สถานีนั้นมีรถจอดอยู่ ซึ่งเป็นรถหมายเลขอะไร และหน้าที่อีกอย่างของสถานี คือสั่งให้รถทำการเคลื่อนที่ไปยังสถานีที่เราต้องการ

3.2.1.2 วงจรของส่วนส่งสินค้า



รูปที่ 3.18 รถขนส่งสินค้า

รายละเอียดของสถานี

ไมโครคอนโทรลเลอร์ : PIC16F690

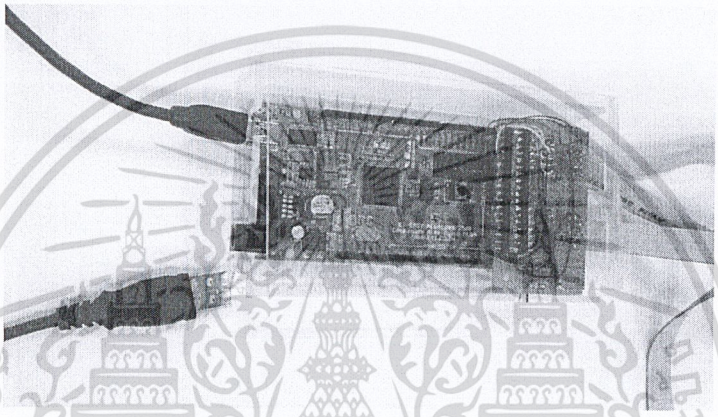
ภาษาที่เขียน : ภาษาซี

โปรแกรมที่ใช้เขียนคำสั่ง : PIC C Compiler (CCS compiler) PICkit 2 V 2.51

หลักการทำงาน

ที่ตัวรถของส่งสินค้าจะมีวงจรทำหน้าที่รับ – ส่งสัญญาณอินฟาเรด กับตัวสถานี โดยเมื่อรถขนส่งสินค้าเข้ามาที่สถานีครั้งแรก จะส่งสัญญาณไปยังสถานี เพื่อบอกกับสถานีว่ารถที่เข้าจอดที่สถานีหมายเลขอะไร และเมื่อสถานีส่งสัญญาณเข้าที่รถ และไมโครคอนโทลเลอร์ในรถจะทำการตัดสินใจเพื่อให้รถทำการเคลื่อนที่ไปยังสถานีปลายทางตามสัญญาณที่ได้รับมา

3.2.2 ส่วนที่ทำการควบคุมการส่งสัญญาณ



รูปที่ 3.19 บอร์ด ET-easy mega1280

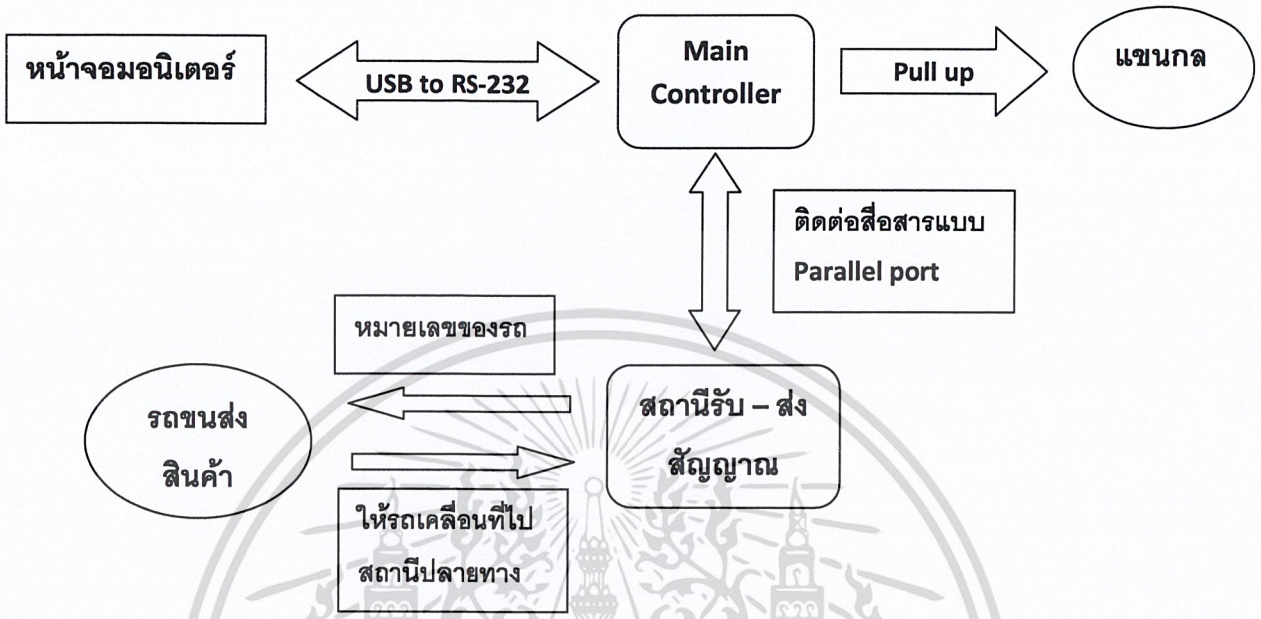
รายละเอียดของสถานี

ไมโครคอนโทรลเลอร์ บอร์ด ET-easy mega1280

ภาษาที่เขียน ภาษาซี

โปรแกรมที่ใช้เขียนคำสั่ง Ardiuno v.18

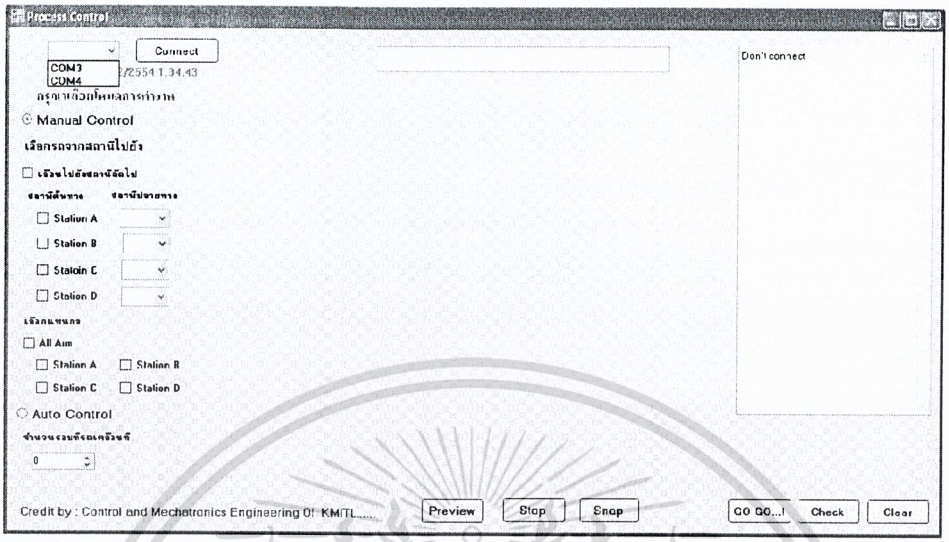
หลักการทำงานของบอร์ด ET-easy mega 1280



รูปที่ 3.20 การติดต่อสื่อสารของบอร์ด ET-easy mega 1280

บอร์ด ET- easy mega 1280 จะทำหน้าที่เป็นคอนโทลเลอร์เลอร์หลักที่จะทำการเชื่อมต่อส่วนต่างๆ เข้าด้วยกัน และเป็นตัวตัดสินใจว่าจะส่งสัญญาณไปส่งส่วนไหน โดยเมื่อมอนิเตอร์ต้องการจะตรวจสอบหมายเลขรถที่จอดแต่ละสถานีจะส่งสัญญาณมาทาง USB to Serial Port เพื่อแปลงสัญญาณจาก RS-232 เข้า USB Port จากนั้นบอร์ดจะติดต่อสื่อสารเข้าไปที่สถานี โดยส่งสัญญาณผ่านทาง Parallel Port เพื่อให้สถานีส่งหมายเลขรถที่จอดอยู่กลับมา แล้วET-easy mega 1280 จะทำการส่งไปแสดงผลที่มอนิเตอร์ และเมื่อมีคำสั่งมาจากผู้ที่ทำการควบคุมระบบว่าต้องการให้รถเคลื่อนที่ไปยังสถานีปลายทางไหน ก็จะรับคำสั่งมาจากมอนิเตอร์แล้วส่งสัญญาณไปสั่งที่สถานีเพื่อให้สถานีส่งสัญญาณไปสั่งที่ตัวรถอีกทีหนึ่งให้รถเคลื่อนที่ไปสถานีปลายทางตามที่ได้รับสัญญาณสั่งมา นอกจากนั้นยังทำหน้าที่เป็นตัวติดต่อสื่อสารกับไมโครคอนโทรลเลอร์ที่ควบคุมแขนกลด้วย เพื่อให้แขนกลทำงาน เมื่อรถเข้ามาจอดที่สถานี

3.2.3 ส่วนของการแสดงผลและสั่งงานระบบ



รูปที่ 3.21 หน้าจอแสดงผลและสั่งงานของระบบ

รายละเอียด

ซอฟต์แวร์ที่ใช้พัฒนาโปรแกรม Microsoft visual basic 6

หลักการทำงาน

เมื่อเรียกโปรแกรมขึ้นมาแล้ว โปรแกรมจะให้เราทำการlog in comportเพื่อให้เราเชื่อมต่อcomport ที่จะใช้ในการสื่อสารกับบอร์ด ET – easy mega 1280 เมื่อเราเชื่อมต่อกับบอร์ดแล้ว จะปรากฏหน้าจอมอนิเตอร์ดังรูป โดยมอนิเตอร์จะมีระบบสั่งงาน 2 แบบ

1. แบบควบคุมด้วยมือ (Manual Mode)

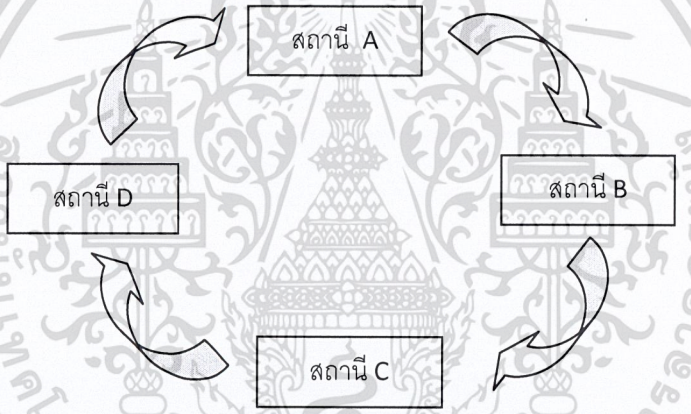
จะเป็นการสั่งว่าเราต้องการที่จะให้รถที่จอดที่สถานีต้นทางไปจอดยังสถานีปลายทางไหน โดยจะไม่มีการจอดทีสถานีที่เป็นทางผ่าน เช่น เมื่อต้องการสั่งให้รถที่จอดที่สถานี A เคลื่อนที่ไปยังสถานี C รถก็จะทำการเคลื่อนที่ไปยังสถานี C เมื่อเคลื่อนที่ถึงสถานี B รถจะไม่จอดที่สถานี B แต่จะเคลื่อนที่ผ่านไป จนกระทั่งถึงสถานี C แล้วจึงจอด เป็นต้น

นอกจากนั้นการทำงานของแขนกล ก็จะทำงานเมื่อเราสั่งจากทางมอนิเตอร์ จะไม่ทำงานเองเมื่อรถเข้าไปจอดที่สถานี

2. แบบอัตโนมัติ (Automatic Mode)

สำหรับการสั่งงานในAuto Mode เราจะต้องกำหนดจำนวนรอบการทำงานของระบบโดยการเลือกที่ปุ่ม “Setting” เมื่อทำการใส่จำนวนรอบเสร็จแล้วให้กดปุ่ม “Run” เพื่อเป็นการสั่งให้ระบบเริ่มทำงาน แล้วรถจะทำการเคลื่อนที่ไปยังสถานีต่อไปแล้วหยุดเพื่อรอให้แขนกลทำงานจนเสร็จจากนั้นจะเคลื่อนที่ไปยังสถานีต่อไปแล้วทำงานอย่างเดิม จนกระทั่งวนกลับมาที่สถานีเดิมจึงจะนับเป็นหนึ่งรอบ มันจะส่งสัญญาณให้บอร์ดคอนโทรลเลอร์จนกระทั่งทำงานครบตามจำนวนรอบที่สั่งไว้

โดยลำดับการเคลื่อนที่ของรถขนส่งสินค้าเป็นดังนี้



รูปที่ 3.22 ผังการทำงาน

บทที่ 4

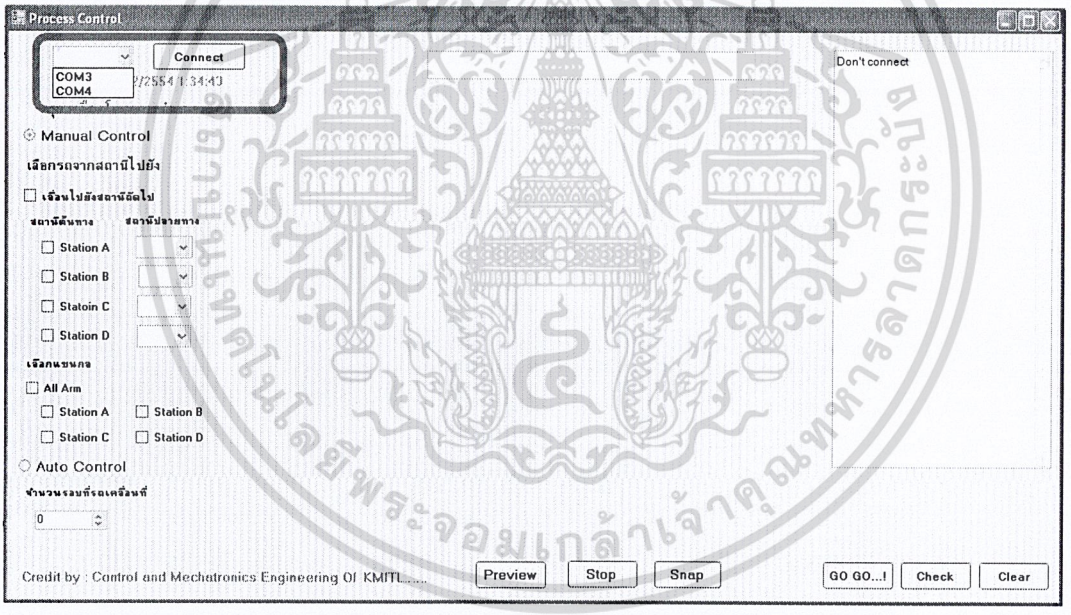
ผลการทดลอง

ในบทนี้ จะกล่าวถึงการทดลองและผลการทดลอง ของโปรแกรม และการทำงานของระบบลำเลียง เมื่อมีการสั่งการจากโปรแกรม โดยมีรายละเอียดของการทดลองดังนี้

4.1 การทดลองโปรแกรม และผลที่ได้จากการทำงาน

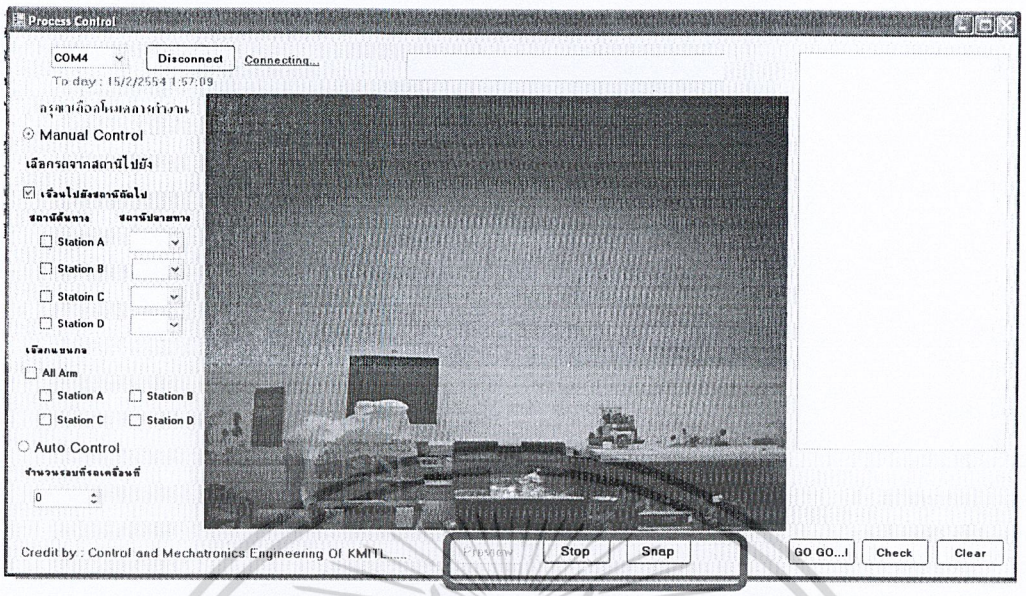
ในส่วนนี้ เป็นการศึกษาเฉพาะส่วนโปรแกรม Visual basic 2008 และ ผลที่ได้จากการสั่งจากโปรแกรม ซึ่งผลการทดลองมีดังนี้

หลังจากทำการรันโปรแกรมจาก Visual basic 2008



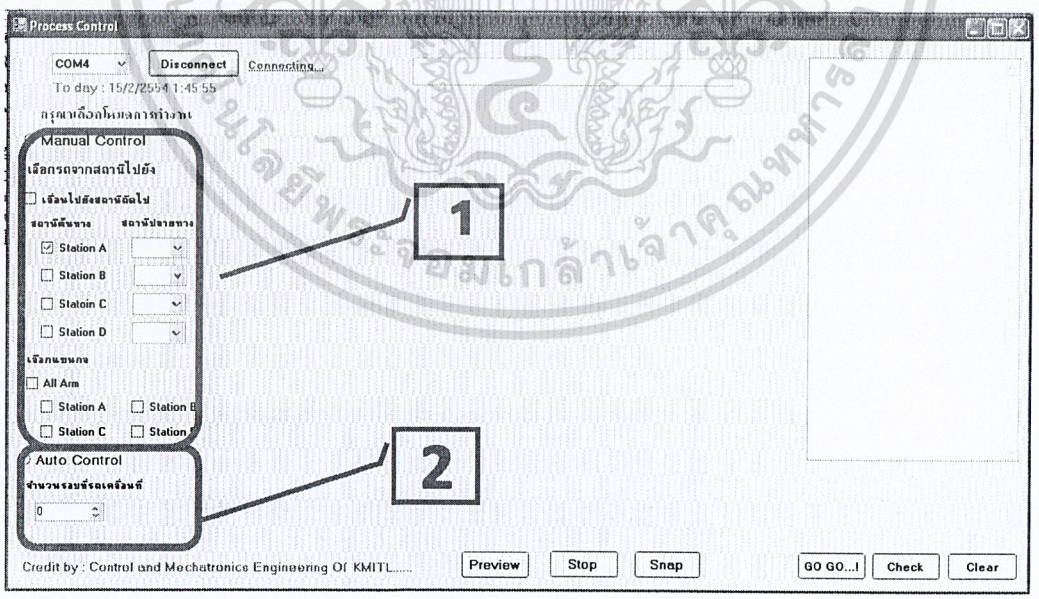
รูปที่ 4.1 การเลือก port เพื่อ connect

ทำการเลือก com port และกดปุ่ม connect เพื่อทำการเชื่อมต่อระบบ



รูปที่ 4.2 การดูการทำงานจริงผ่านกล้อง webcam

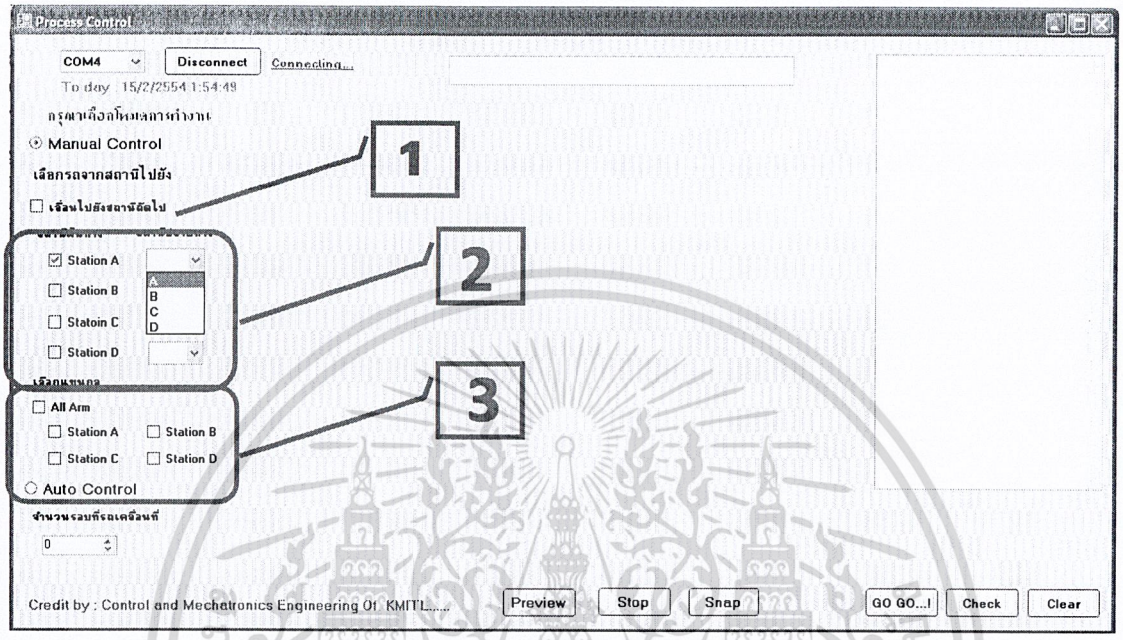
เราสามารถดูการทำงานจริงของระบบโดยผ่านทางกล้อง webcam โดยกดปุ่ม Preview และ เราสามารถทำการ save รูปภาพในขณะนั้นได้โดยการกดปุ่ม Snap และทำการปิดกล้องได้โดยการกดปุ่ม Stop



รูปที่ 4.3 การเลือกโหมดการทำงาน

เมื่อเราเข้ามายังตัวโปรแกรมแล้วเราสามารถเลือกการทำงานของระบบลำเลียงได้ 2 โหมด คือ

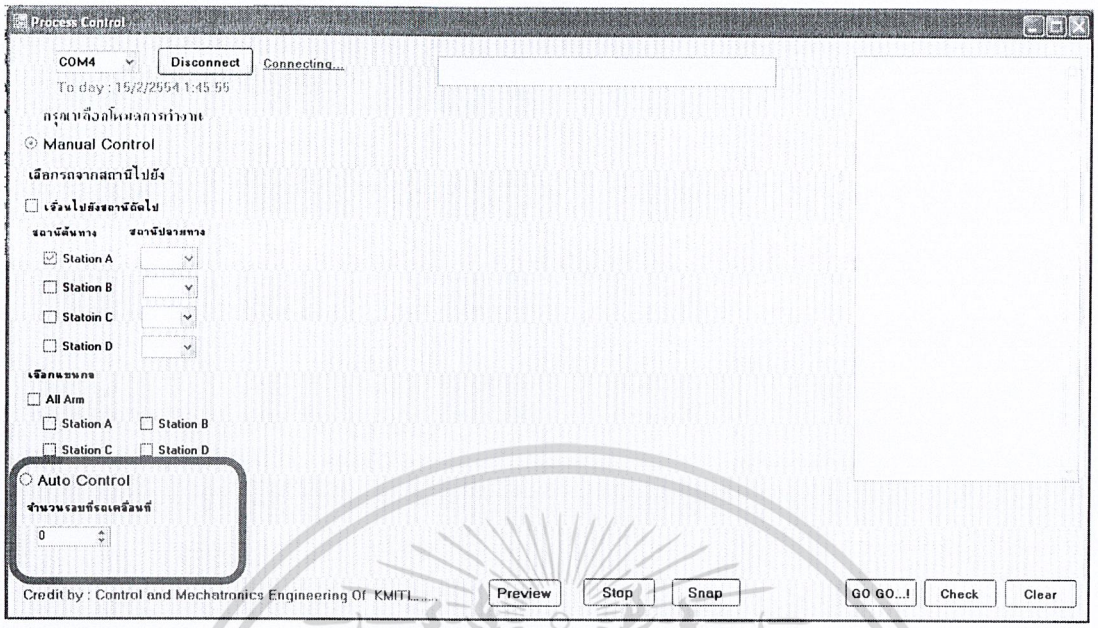
1. โหมดควบคุมด้วยมือ และ 2. โหมดควบคุมอัตโนมัติ



รูปที่ 4.4 การเลือกโหมดควบคุมด้วยมือ

ในโหมดควบคุมด้วยมือนั้น เราอาจแบบเป็น 3 ส่วนหลักๆ คือ

1. เป็นการสั่งการให้รถเคลื่อนที่ไปยังสถานีถัดไป โดยไม่มีการวิ่งข้ามสถานี
2. เป็นการสั่งการให้รถเคลื่อนที่ไปยังสถานีที่ต้องการได้
3. เป็นกานสั่งการให้แขนกลในสถานีต่างให้ทำงาน



รูปที่ 4.5 การเลือกโหมดอัตโนมัติ

และในโหมดควบคุมอัตโนมัตินั้น เราสามารถกำหนดจำนวนรอบการทำงานได้ว่าต้องการให้ระบบของเราทำงานได้กี่รอบ โดยเมื่อรถเคลื่อนที่มาถึงยังสถานี แขนกลจะทำการหยิบจับสิ่งของเองโดยอัตโนมัติ ไม่จำเป็นต้องรอคำสั่ง

รูปที่ 4.6 การตรวจสอบสถานะการทำงาน

และเราสามารถตรวจสอบสถานะการทำงาน ทั้งระบบได้ด้วยการกดปุ่ม Check ไม่ว่าจะเป็นการตรวจสอบว่าสถานีไหนมีรถคันใดจอดอยู่บ้าง และเราสามารถทำการลบข้อมูลได้โดยการกดปุ่ม Clear

บทที่ 5

สรุปและแนวทางในการพัฒนา

5.1 สรุปผลการทดลอง

โครงงานนี้คือโครงงานระบบลำเลียงสินค้าที่มีความสำคัญอย่างมากกับกระบวนการผลิตในอุสาหกรรมเป็นการจำลองการลำเลียงสินค้าด้วยรถขนส่งและแขนกลแบบอัตโนมัติ โดยมีการสั่งงานจากคอมพิวเตอร์ มีการวางระบบโดยนำทฤษฎีและความรู้ทางด้านอิเล็กทรอนิกส์มาประยุกต์ใช้ในการสร้างวงจรควบคุม ในส่วนของชิ้นงานได้นำอุปกรณ์ต่างๆมาใช้ในการควบคุม เช่น เซอร์โวมอเตอร์ ไมโครคอนโทลเลอร์ เป็นต้น มาทำงานรวมกับวงจรอิเล็กทรอนิกส์และโปรแกรมเพื่อควบคุมการทำงานของระบบให้มีประสิทธิภาพ

จากการทดลองระบบลำเลียงสินค้าเมื่อเราทำการเขียนโปรแกรมและทำการสั่งให้ระบบทำงานจากคอมพิวเตอร์ ระบบจะทำงานตามที่เรากำหนด คือ รถจะวิ่งออกจากสถานีใดแล้วไปจอดที่สถานีใด ที่เหลืออีกสามคันจะวิ่งไปจอดถัดไปอีกหนึ่งสถานีอย่างเช่น ถ้าเราต้องการให้รถคันที่หนึ่ง วิ่งไปยังสถานีที่สี่ ดังนั้น รถคันที่สองจะวิ่งไปจอดที่สถานีที่หนึ่ง รถคันที่สามจะวิ่งไปจอดที่สถานีที่สอง และรถคันที่สี่จะวิ่งไปจอดสถานีที่สาม จะเป็นอย่างนี้ไปเรื่อยๆ ไม่ว่าเราจะทำการเลือกรถคันที่เท่าไร สถานีต้นทางหรือปลายทางใด รถคันที่เหลือจะวิ่งไปจอดตามสถานีตามที่กล่าวดังข้างต้น

5.2 ปัญหาที่พบและแนวทางแก้ไข

1. เนื่องจากม้วนทองแดงที่อยู่ใต้ท้องรถไม่สัมผัสกับราง ทำให้รถหยุดวิ่งในบางจังหวะ ทำให้ต้อง reset ระบบเพื่อเริ่มทำงานใหม่ จึงแก้ไขด้วยการเปลี่ยนม้วนทองแดงใต้ท้องรถใหม่
2. พื้นรางมีสิ่งสกปรกอยู่ทำให้ sensor ใต้ท้องรถตรวจจับว่าเป็นสีดำทำให้รถจอดก่อนสถานีจึงต้องแก้ไขด้วยการติดแทบสีขาวใต้ราง

5.3 ข้อเสนอแนะและแนวทางในการค้นคว้าและพัฒนา

โครงงานระบบลำเลียงสินค้ามีความสำคัญอย่างมากกับกระบวนการผลิตในอุตสาหกรรมเพราะการใช้ระบบลำเลียงสินค้า มีการนำเอาเทคโนโลยีสมัยใหม่เข้ามาทดแทนแรงงานมนุษย์ที่มีอยู่ เพื่อลดการสูญเสียที่อาจเกิดจากขั้นตอนการผลิต ที่มีความอันตรายต่อบุคคลากร ลดต้นทุนในการผลิต เพิ่มประสิทธิภาพ ความแม่นยำ ความรวดเร็วในกระบวนการผลิต อีกทั้งยังสามารถเก็บรวบรวมข้อมูลและตรวจสอบการทำงานของระบบได้ ดังนั้นควรพัฒนาเซนเซอร์ให้สามารถทำงานในสถานที่จริง ตำแหน่งที่ใช้งานจริง พัฒนาส่วนโปรแกรมให้รองรับกับเหตุการณ์ในสถานที่จริง เพื่อที่จะได้ทำทำงานอย่างมีประสิทธิภาพที่สุด

ภาคผนวก

ภาคผนวก ก

โปรแกรมควบคุมส่วนต่างๆ

โปรแกรมการควบคุมรถลำเลียง

```
/* Include ------------------------------------------------------------*/
#include <16F690.h>
#device *=16
#device ICD=TRUE
#device adc=8
#FUSES NOWDT            //No Watch Dog Timer
#FUSES HS               //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPROTECT        //Code not protected from reading
#FUSES NOBROWNOUT       //No brownout reset
#FUSES MCLR             //Master Clear pin enabled
#FUSES NOCPD            //No EE protection
#FUSES NOPUT            //No Power Up Timer
#FUSES IESO             //Internal External Switch Over mode enabled
#FUSES FCMEN            //Fail-safe clock monitor enabled
#define  KEYHIT_DELAY  500    // in milliseconds
#use delay(clock=20000000)
#use rs232(baud=2400,parity=N,xmit=PIN_B7,rcv=PIN_B5,bits=8, stream=AVR,timeout=KEYHIT_DELAY)
/* Define -------------------------------------------------------------*/
#define RobotNumber '2'         // RobotNumber 1 to 6
#define Analog0 4               // AnalogChannel 4
#define MotorDrive0 PIN_C3      // MotorDrivePin
#define MotorDrive1 PIN_C4      // MotorDrivePin
#define MPWMPin PIN_C6          // Manual PWM Pin_C2
#define ResMPWM 100             // Maximum Resolution PWM
#define IrFreq 528              // Ir Receive Freq 38 kHz.
#define NcPin0 PIN_A0           // Empty Pin0
#define NcPin1 PIN_B4           // Empty Pin1
#define NcPin2 PIN_C6           // Empty Pin2
#define NcPin3 PIN_B6           // Empty Pin3
```

```
#define NcPin4 PIN_C7                // Empty Pin4
#define Button0 !Input(NcPin4)       // Input0 use to memo
#define Button1 !Input(NcPin2)       // Input1 use to run, skip
#define Led0   NcPin0                // Output0 use to run status
#define Led1   NcPin1                // Output1 use to memo status
#define Led2   NcPin3                // Output2 use to sense
#define MaxSpeed 100                 // Maximum Motor Speed.
#define MinSpeed 80                  // Minimum Motor Speed.
/* Private Variable ------------------------------------------------------*/
boolean toggle0 = 1;                 // External Interrupt Toggle.
boolean ANL0;                        // Analog0[AN4] Logic
unsigned char ADC_Value;             // Analog to Digital Value
unsigned char Setpoint;              // ADC Setpoint Value.
unsigned char Range;                 // Error Range.
char Loopi;                          // Counter Loop.
unsigned char Duty;                  // Set Duty Cycle Of Manual PWM.
unsigned char Mode;                  // Select MotorMode.
unsigned char skipStation = 1;       // Value of skiping station
/* Private Function ------------------------------------------------------*/
void MotorForward(char);
void MotorStop(void);
void MotorBackward(char);
int stacker(void);
unsigned char receive(void);
/* All Interrupt Service Routine -----------------------------------------*/
/**
 * @ brief RTCC Interrupt Service Routine.
 *       Read Sensor.
 */
#int_RTCC
void  RTCC_isr(void)
{
  /* PWM Motor Control */
  if(Duty >= Loopi){
    if(Mode == 1)
    {
      Output_High(MotorDrive0);
      Output_Low(MotorDrive1);
    }
```

```
    else if(Mode == 2)
    {
      Output_High(MotorDrive0);
      Output_High(MotorDrive1);
    }
    else if(Mode == 3)
    {
      Output_Low(MotorDrive0);
      Output_High(MotorDrive1);
    }
  }
  else if((Loopi >Duty) && (Loopi <= ResMPWM)){
    Output_Low(MotorDrive0);
    Output_Low(MotorDrive1);
  }
  else if(Loopi > ResMPWM){
    Loopi = 0;
  }
  Loopi++;
}
/**
 * @brief External Interrupt Service Routine.
 */
#int_EXT
void  EXT_isr(void)
{
  if(toggle0){
    SET_PWM1_DUTY(IrFreq/2);
    EXT_INT_EDGE(L_TO_H);
  }
  else if(!toggle0){
    SET_PWM1_DUTY(0);
    EXT_INT_EDGE(H_TO_L);
  }
  toggle0 = !toggle0;
}


/**
 * @brief TIMER1 Interrupt Service Routine.
```

```
 */
#int_TIMER1
void  TIMER1_isr(void)
{
  /* ADC Reading Control. ************************************************/
  SET_ADC_CHANNEL(Analog0);
  ADC_Value =  READ_ADC();

  if(ADC_Value < Setpoint + 3*Range)
  {
    ANL0 = 1;              // ANL0 = 1 When found Blackline.
    Output_High(Led2);
  }
  else
  {
    ANL0 = 0;              // ANL0 = 0 When not fount Blackline.
    Output_Low(Led2);
  }
}
//@brief Main Func.
int main()
{
  /* Initial Module ******************************************************/
  Setup_Adc_ports(sAN4|VSS_VDD);
  Setup_Adc(ADC_CLOCK_DIV_32);
  Setup_Spi(SPI_SS_DISABLED);
  Setup_Timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  Setup_Timer_1(T1_INTERNAL|T1_DIV_BY_1);
  Setup_Timer_2(T2_DIV_BY_1,131,1);
  Setup_Ccp1(CCP_PWM);
  Set_Pwm1_Duty(0);
  Setup_Comparator(NC_NC_NC_NC);
  Enable_Interrupts(INT_RTCC);
  Enable_Interrupts(INT_EXT);
  EXT_INT_EDGE(H_TO_L);
  Enable_Interrupts(INT_TIMER1);
  Enable_Interrupts(GLOBAL);
  /* Begin Process *******************************************/
  Output_High(led0);            // turn on running status led
```

```
  Output_Low(led1);              // turn off memo status led
  Output_Low(led2);              // turn off sense status led
  MotorStop();
  delay_ms(10);
  Setpoint = Read_Eeprom(1);          // read memory
  delay_ms(10);
  Range = 5 + (float)Setpoint/15.00;     // adjust sensor
    /* Memory Value Sensor ************************************/
  while(TRUE){
    if(Button1){                 // push to memo
      Write_Eeprom(1,ADC_Value);
      delay_ms(10);
      Setpoint = Read_Eeprom(1);
      delay_ms(10);
      Range = 5 + (float)Setpoint/15.00;
      Output_High(Led1);             // turn on memo status led
    }
    else if(Button0){              // push to run
      Output_Low(Led0);              // turn off running status led
      delay_ms(500);
      break;
    }
    else{
      Output_Low(Led1);              // non-memo status led
    }
  }
  Output_High(Led0);               // turn on running status led
    /* Robot Running. ***************************************/
  MotorForward(MaxSpeed);
  delay_ms(200);                 // Random error case.

 /* Robot Start Runing At Setpoint Line. ********************/
  if(ANL0){
    MotorForward(100);
    while(ANL0);
  }
    //* @brief Infinity Loop.
  while(TRUE){
      /* Check Line 1 And Forward. *********/
```

```
do{
  MotorForward(MaxSpeed);
}while(!ANL0);
while(ANL0);
  /* Slowly Then Check Line 2. *********/
//do{
//  MotorForward(MinSpeed);
//}while(!ANL0);
if(--skipStation == 0){          // Skip Check and Reduce
  /* Stop Motor. ***************************************/
  Output_Low(Led0);              // turn off running status led
  MotorStop();
  Disable_Interrupts(INT_RTCC);
  delay_ms(50);
  Disable_Interrupts(INT_TIMER1);
  delay_ms(50);
       // delay
  delay_ms(200);
  // send RobotNumber
  printf("%c",0x01);
  delay_ms(25);

  printf("%c",0x01);
  delay_ms(25);
     printf("[");                // to station protocol
  delay_ms(25);
     printf("%c",RobotNumber);
  delay_ms(25);
     delay_ms(25);
  Enable_Interrupts(INT_RTCC);
  delay_ms(25);
  Enable_Interrupts(INT_TIMER1);

  // receive skipStation
  skipStation = receive();
     Output_High(Led0);           // turn on running status led
}
 /* Running Again ***************************************/
do{
```

```
      Output_Low(Led0);              // turn on running status led
        MotorForward(MaxSpeed);
      }while(ANL0);
    }
    return 0;
}
  //* @brief Forward Motor.
void MotorForward(char i){
    Mode = 1;
    Duty = i;
}
/**
  * @brief Stop Motor.
  */
void MotorStop(){
    Mode = 2;
    Duty = 100;
}
/**
  * @brief Backward Motor.
  */
void MotorBackward(char i){
    Mode = 3;
    Duty = i;
    }
unsigned char receive(void){
    unsigned char value;
    unsigned char value2 = 0;
    unsigned char keep = '1';
    do
    {
        while(!kbhit());
        do
        {
          value=getc();
          if (value == '<'){
             value2 = value;
          }
        }while (RS232_ERRORS);  //RS232_ERRORS will be 0 when there is a timeout
```

```
    }while(value2 !='<');
    value2 = 0;
    while(!kbhit());
    do
    {
      value=getc();
      if((value == '1')||(value == '2')||(value == '3')||(value == '4')){
        keep = value;
      }
    }while (RS232_ERRORS);      //RS232_ERRORS will be 0 when there is a timeout
    keep = keep - 48;
    return keep;
}
```

โปรแกรมควบคุมรถลำเลียงสินค้า

```
/* Include ------------------------------------------------------------*/
#include <16F690.h>

#device *=16
#device ICD=TRUE
#device adc=8

#FUSES NOWDT            //No Watch Dog Timer
#FUSES HS               //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPROTECT        //Code not protected from reading
#FUSES NOBROWNOUT       //No brownout reset
#FUSES MCLR             //Master Clear pin enabled
#FUSES NOCPD            //No EE protection
#FUSES NOPUT            //No Power Up Timer
#FUSES IESO             //Internal External Switch Over mode enabled
#FUSES FCMEN            //Fail-safe clock monitor enabled

#define  KEYHIT_DELAY   500    // in milliseconds
#use delay(clock=20000000)
#use rs232(baud=2400,parity=N,xmit=PIN_B7,rcv=PIN_B5,bits=8, stream=AVR,timeout=KEYHIT_DELAY)

/* Define -------------------------------------------------------------*/
#define RobotNumber '4'         // RobotNumber 1 to 6

#define Analog0 4               // AnalogChannel 4
#define MotorDrive0 PIN_C3      // MotorDrivePin
#define MotorDrive1 PIN_C4      // MotorDrivePin

#define MPWMPin PIN_C6          // Manual PWM Pin_C2
#define ResMPWM 100             // Maximum Resolution PWM

#define IrFreq 528              // Ir Receive Freq 38 kHz.

#define NcPin0 PIN_A0           // Empty Pin0
#define NcPin1 PIN_B4           // Empty Pin1
#define NcPin2 PIN_C6           // Empty Pin2
#define NcPin3 PIN_B6           // Empty Pin3
```

```
#define NcPin4 PIN_C7              // Empty Pin4

#define Button0 !Input(NcPin4)      // Input0 use to memo
#define Button1 !Input(NcPin2)      // Input1 use to run, skip
#define Led0   NcPin0              // Output0 use to run status
#define Led1   NcPin1              // Output1 use to memo status
#define Led2   NcPin3              // Output2 use to sense

#define MaxSpeed 100               // Maximum Motor Speed.
#define MinSpeed 80                // Minimum Motor Speed.

/* Private Variable ---------------------------------------------------*/
boolean toggle0 = 1;               // External Interrupt Toggle.
boolean ANL0;                      // Analog0[AN4] Logic
unsigned char ADC_Value;           // Analog to Digital Value
unsigned char Setpoint;            // ADC Setpoint Value.
unsigned char Range;               // Error Range.
char Loopi;                        // Counter Loop.
unsigned char Duty;                // Set Duty Cycle Of Manual PWM.
unsigned char Mode;                // Select MotorMode.
unsigned char skipStation = 1;     // Value of skiping station

/* Private Function ---------------------------------------------------*/
void MotorForward(char);
void MotorStop(void);
void MotorBackward(char);
int stacker(void);
unsigned char receive(void);

/* All Interrupt Service Routine --------------------------------------*/
/**
 * @ brief RTCC Interrupt Service Routine.
 *       Read Sensor.
 */
#int_RTCC
void  RTCC_isr(void)
{
  /* PWM Motor Control */
  if(Duty >= Loopi){
```

```
    if(Mode == 1)
    {
      Output_High(MotorDrive0);
      Output_Low(MotorDrive1);
    }
    else if(Mode == 2)
    {
      Output_High(MotorDrive0);
      Output_High(MotorDrive1);
    }
    else if(Mode == 3)
    {
      Output_Low(MotorDrive0);
      Output_High(MotorDrive1);
    }
  }
  else if((Loopi >Duty) && (Loopi <= ResMPWM)){
    Output_Low(MotorDrive0);
    Output_Low(MotorDrive1);
  }
  else if(Loopi > ResMPWM){
    Loopi = 0;
  }
  Loopi++;
}

/**
 * @brief External Interrupt Service Routine.
 */
#int_EXT
void  EXT_isr(void)
{
  if(toggle0){
    SET_PWM1_DUTY(IrFreq/2);
    EXT_INT_EDGE(L_TO_H);
  }
  else if(!toggle0){
    SET_PWM1_DUTY(0);
    EXT_INT_EDGE(H_TO_L);
```

```
  }
  toggle0 = !toggle0;
}

/**
 * @brief TIMER1 Interrupt Service Routine.
 */
#int_TIMER1
void  TIMER1_isr(void)
{
  /* ADC Reading Control. ************************************************/
  SET_ADC_CHANNEL(Analog0);
  ADC_Value =  READ_ADC();

  if(ADC_Value < Setpoint + 3*Range)
  {
    ANL0 = 1;              // ANL0 = 1 When found Blackline.
    Output_High(Led2);
  }
  else
  {
    ANL0 = 0;              // ANL0 = 0 When not fount Blackline.
    Output_Low(Led2);
  }
}

/**
 * @brief Main Func.
 */
int main()
{
  /* Initial Module ******************************************************/
  Setup_Adc_ports(sAN4|VSS_VDD);
  Setup_Adc(ADC_CLOCK_DIV_32);
  Setup_Spi(SPI_SS_DISABLED);
  Setup_Timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  Setup_Timer_1(T1_INTERNAL|T1_DIV_BY_1);
  Setup_Timer_2(T2_DIV_BY_1,131,1);
  Setup_Ccp1(CCP_PWM);
```

```
  Set_Pwm1_Duty(0);
  Setup_Comparator(NC_NC_NC_NC);
  Enable_Interrupts(INT_RTCC);
  Enable_Interrupts(INT_EXT);
  EXT_INT_EDGE(H_TO_L);
  Enable_Interrupts(INT_TIMER1);
  Enable_Interrupts(GLOBAL);

////////////////////////////////////////////////////////////////////////////////
  /* Begin Process ***************************************/
  Output_High(led0);              // turn on running status led
  Output_Low(led1);               // turn off memo status led
  Output_Low(led2);               // turn off sense status led
  MotorStop();
  delay_ms(10);
  Setpoint = Read_Eeprom(1);          // read memory
  delay_ms(10);
  Range = 5 + (float)Setpoint/15.00;     // adjust sensor

  /* Memory Value Sensor ***********************************/
  while(TRUE){
    if(Button1){                  // push to memo
      Write_Eeprom(1,ADC_Value);
      delay_ms(10);
      Setpoint = Read_Eeprom(1);
      delay_ms(10);
      Range = 5 + (float)Setpoint/15.00;
      Output_High(Led1);              // turn on memo status led
    }
    else if(Button0){               // push to run
      Output_Low(Led0);             // turn off running status led
      delay_ms(500);
      break;
    }
    else{
      Output_Low(Led1);             // non-memo status led
    }
  }
  Output_High(Led0);              // turn on running status led
```

```
/* Robot Running. ****************************************/
MotorForward(MaxSpeed);
delay_ms(200);                  // Random error case.

/* Robot Start Runing At Setpoint Line. ********************/
if(ANL0){
  MotorForward(100);
  while(ANL0);
}

/**
 * @brief Infinity Loop.
 */
while(TRUE){

  /* Check Line 1 And Forward. *********/
  do{
    MotorForward(MaxSpeed);
  }while(!ANL0);
  while(ANL0);

  /* Slowly Then Check Line 2. *********/
  //do{
  //  MotorForward(MinSpeed);
  //}while(!ANL0);

  if(--skipStation == 0){          // Skip Check and Reduce
    /* Stop Motor. ****************************************/
    Output_Low(Led0);              // turn off running status led
    MotorStop();

    Disable_Interrupts(INT_RTCC);
    delay_ms(50);
    Disable_Interrupts(INT_TIMER1);
    delay_ms(50);

    // delay
    delay_ms(200);
```

```
        // send RobotNumber
        printf("%c",0x01);
        delay_ms(25);

        printf("%c",0x01);
        delay_ms(25);

        printf("[");              // to station protocol
        delay_ms(25);

        printf("%c",RobotNumber);
        delay_ms(25);

        delay_ms(25);
        Enable_Interrupts(INT_RTCC);
        delay_ms(25);
        Enable_Interrupts(INT_TIMER1);

        // receive skipStation
        skipStation = receive();

        Output_High(Led0);          // turn on running status led
      }

      /* Running Again ****************************************/
      do{
       Output_Low(Led0);          // turn on running status led
        MotorForward(MaxSpeed);
      }while(ANL0);
    }
    return 0;
}

/**
 * @brief Forward Motor.
 */
void MotorForward(char i){
  Mode = 1;
  Duty = i;
```

```
}

/**
 * @brief Stop Motor.
 */
void MotorStop(){
  Mode = 2;
  Duty = 100;
}

/**
 * @brief Backward Motor.
 */
void MotorBackward(char i){
  Mode = 3;
  Duty = i;
}

unsigned char receive(void){
  unsigned char value;
  unsigned char value2 = 0;
  unsigned char keep = '1';
  do
  {
      while(!kbhit());
      do
      {
        value=getc();
        if (value == '<'){
           value2 = value;
        }
      }while (RS232_ERRORS);  //RS232_ERRORS will be 0 when there is a timeout
  }while(value2 !='<');
  value2 = 0;
  while(!kbhit());
  do
  {
    value=getc();
    if((value == '1')||(value == '2')||(value == '3')||(value == '4')){
```

```
        keep = value;
      }
    }while (RS232_ERRORS);      //RS232_ERRORS will be 0 when there is a timeout
    keep = keep - 48;
    return keep;
}
```

โปรแกรมควบคุมMain Controller

```
/* Include ----------------------------------------*/
#include <avr/io.h>
#include <avr/interrupt.h>
#include <compat/deprecated.h>
unsigned char keeping;
unsigned char flag;
unsigned char stackBu[7];
uint8_t readingBu[4];
// Parallel start up pin
uint8_t A = 22;
uint8_t B = 29;
uint8_t C = 36;
uint8_t D = 43;
/* Function ---------------------------------------*/
/**
 * @brief Write Parallel
 */
uint8_t reading (uint8_t pinNum){
   uint8_t readBu[3];
   readBu[0] = digitalRead(pinNum);
   readBu[1] = digitalRead(pinNum+1);
   readBu[2] = digitalRead(pinNum+2);
   return (readBu[0]*1 + readBu[1]*2 + readBu[2]*4);
}
/**
 * @brief Write Parallel
 */
void writing (uint8_t pinNum,uint8_t wr){
 uint8_t wrBu[3];
 switch(wr){
   case 0: wrBu[0] = 0; wrBu[1] = 0; wrBu[2] = 0;
        break;
   case 1: wrBu[0] = 1; wrBu[1] = 0; wrBu[2] = 0;
        break;
   case 2: wrBu[0] = 0; wrBu[1] = 1; wrBu[2] = 0;
        break;
```

```
    case 3: wrBu[0] = 1; wrBu[1] = 1; wrBu[2] = 0;
        break;
    case 4: wrBu[0] = 0; wrBu[1] = 0; wrBu[2] = 1;
        break;
    case 5: wrBu[0] = 1; wrBu[1] = 0; wrBu[2] = 1;
        break;
    case 6: wrBu[0] = 0; wrBu[1] = 1; wrBu[2] = 1;
        break;
    case 7: wrBu[0] = 1; wrBu[1] = 1; wrBu[2] = 1;
        break;
   }
   for(uint8_t i = 0; i<=2 ; i++){
    if(wrBu[i] == 1){
     digitalWrite(pinNum+i, HIGH);
    }
    else{
     digitalWrite(pinNum+i, LOW);
    }
   }
}
/**
 * @brief Parallel Control
 */
uint8_t parallelCtrl(uint8_t pin,uint8_t data){
 uint8_t readValue;
 writing(pin,0);       //close slave read
 digitalWrite(pin+6,LOW);
 delay(20);
 readValue = reading(pin+3);
 writing(pin,data);
 delay(20);            //delay for ready
 digitalWrite(pin+6,HIGH);
 delay(20);
 writing(pin,0);       //close slave read
 return readValue;
}
/**
 * @brief Setup Arduino
 */
```

```
void setup()  {
 Serial.begin(9600);

 // Station A port setup (D22-D28)
 pinMode(22,OUTPUT);    //MOSI
 pinMode(23,OUTPUT);    //MOSI
 pinMode(24,OUTPUT);    //MOSI
 pinMode(25,INPUT);     //MISO
 pinMode(26,INPUT);     //MISO
 pinMode(27,INPUT);     //MISO
 pinMode(28,OUTPUT);    //SCK
 digitalWrite(28,HIGH); //Clock High
 writing(22,0);
 delay(50);
 // Station B port setup (D29-D35)
 pinMode(29,OUTPUT);    //MOSI
 pinMode(30,OUTPUT);    //MOSI
 pinMode(31,OUTPUT);    //MOSI
 pinMode(32,INPUT);     //MISO
 pinMode(33,INPUT);     //MISO
 pinMode(34,INPUT);     //MISO
 pinMode(35,OUTPUT);    //SCK
 digitalWrite(35,HIGH); //Clock High
 writing(29,0);
 delay(50);
 // Station C port setup (D36-D42)
 pinMode(36,OUTPUT);    //MOSI
 pinMode(37,OUTPUT);    //MOSI
 pinMode(38,OUTPUT);    //MOSI
 pinMode(39,INPUT);     //MISO
 pinMode(40,INPUT);     //MISO
 pinMode(41,INPUT);     //MISO
 pinMode(42,OUTPUT);    //SCK
 digitalWrite(42,HIGH); //Clock High
 writing(36,0);
 delay(50);
 // Station D port setup (D43-D49)
 pinMode(43,OUTPUT);    //MOSI
 pinMode(44,OUTPUT);    //MOSI
```

```
  pinMode(45,OUTPUT);    //MOSI
  pinMode(46,INPUT);     //MISO
  pinMode(47,INPUT);     //MISO
  pinMode(48,INPUT);     //MISO
  pinMode(49,OUTPUT);    //SCK
  digitalWrite(49,HIGH); //Clock High
  writing(43,0);
  delay(1000);

  // Go! Go! Go!
  Serial.println("command");
}
/**
 * @brief Super Loop
 */
void loop(){
  // Keyboard input
    unsigned char iBu = 0;
  do{
    while(Serial.available() <= 0);
    stackBu[iBu] = Serial.read();
    Serial.print(stackBu[iBu],BYTE);
  }while((stackBu[iBu] != ' ')&&(iBu++<=7));
  // Filter data : just pass '1' to '4'
  if((stackBu[1] >= 54)&&(stackBu[0] <= 60)){
    stackBu[1] = '1';
  }
  Serial.println("");
  // Selection Mode
  // Check All Parallel Data
  if((stackBu[0]=='c')&&(stackBu[1]=='h')&&(stackBu[2]=='e')&&(stackBu[3]=='c')&&(stackBu[4]=='k')){
     Serial.println("Check All Data");
     Serial.print(" A=");
     readingBu[0] = parallelCtrl(A,0);
     Serial.print(readingBu[0], DEC);
     Serial.print(", B=");
     readingBu[1] = parallelCtrl(B,0);
     Serial.print(readingBu[1], DEC);
     Serial.print(", C=");
```

```
    readingBu[2] = parallelCtrl(C,0);
    Serial.print(readingBu[2], DEC);
    Serial.print(", D=");
    readingBu[3] = parallelCtrl(D,0);
    Serial.print(readingBu[3], DEC);
    Serial.print("\n");
  }
  // Send to A
  else if((stackBu[0] == 'A')&&((stackBu[1]>=65)&&(stackBu[1]<=68))){
   if(stackBu[1] == 'A'){
    keeping = '4';
   }
   else if(stackBu[1] == 'B'){
    keeping = '1';
   }
   else if(stackBu[1] == 'C'){
    keeping = '2';
   }
   else if(stackBu[1] == 'D'){
    keeping = '3';
   }
   flag = parallelCtrl(A,keeping-48);
   if(flag == 0){
    Serial.print("Parking Empty");
   }
   else{
    Serial.print("Robot ");
    Serial.print(flag, DEC);
    Serial.print(" go to station ");
    Serial.print(stackBu[1]);
   }
   Serial.print("\n");
  }
  // Send to B
  else if((stackBu[0] == 'B')&&((stackBu[1]>=65)&&(stackBu[1]<=68))){
   if(stackBu[1] == 'A'){
    keeping = '3';
   }
   else if(stackBu[1] == 'B'){
```

```
    keeping = '4';
   }
   else if(stackBu[1] == 'C'){
    keeping = '1';
   }
   else if(stackBu[1] == 'D'){
    keeping = '2';
   }
   flag = parallelCtrl(B,keeping-48);
   if(flag == 0){
    Serial.print("Parking Empty");
   }
   else{
    Serial.print("Robot ");
    Serial.print(flag, DEC);
    Serial.print(" go to station ");
    Serial.print(stackBu[1]);
   }
   Serial.print("\n");
  }
  // Send to C
  else if((stackBu[0] == 'C')&&((stackBu[1]>=65)&&(stackBu[1]<=68))){
   if(stackBu[1] == 'A'){
    keeping = '2';
   }
   else if(stackBu[1] == 'B'){
    keeping = '3';
   }
   else if(stackBu[1] == 'C'){
    keeping = '4';
   }
   else if(stackBu[1] == 'D'){
    keeping = '1';
   }
   flag = parallelCtrl(C,keeping-48);
   if(flag == 0){
    Serial.print("Parking Empty");
   }
   else{
```

```
      Serial.print("Robot ");
      Serial.print(flag, DEC);
      Serial.print(" go to station ");
      Serial.print(stackBu[1]);
     }
     Serial.print("\n");
    }
    // Send to D
    else if((stackBu[0] == 'D')&&((stackBu[1]>=65)&&(stackBu[1]<=68))){
     if(stackBu[1] == 'A'){
      keeping = '1';
     }
     else if(stackBu[1] == 'B'){
      keeping = '2';
     }
     else if(stackBu[1] == 'C'){
      keeping = '3';
     }
     else if(stackBu[1] == 'D'){
      keeping = '4';
     }
     flag = parallelCtrl(D,keeping-48);
     if(flag == 0){
      Serial.print("Parking Empty");
     }
     else{
      Serial.print("Robot ");
      Serial.print(flag, DEC);
      Serial.print(" go to station ");
      Serial.print(stackBu[1]);
     }
     Serial.print("\n");
    }
    // For Wrong Instruction
    else{
     Serial.println("Wrong Instruction");
    }
}
```

โปรแกรมควบคุมสถานีรับ-ส่งสัญญาณ

```
/* Includes ------------------------------------------------------*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<compat/deprecated.h>
#define F_CPU 8000000UL
#include<util/delay.h>

#define FOSC 8000000// Clock Speed
#define BAUD 2400
#define MYUBRR FOSC/16/BAUD-1

/* Define ------------------------------------------------------*/
#define F_IR 38000              // Frequency of IR Receiver Module
#define TOP_PWM F_CPU/2/F_IR   // Maximum Value of Duty Cycle
#define data0 (PINC&0x01)
#define data1 (PINC&0x02)>>1
#define data2 (PINC&0b00000100)>>2

/* Private Variable ------------------------------------------------------*/
unsigned char toggle = 1;       // Exti toggle
volatile unsigned char stTemp;          // stack temorary.
unsigned char dataTemp;          // data temporary.
volatile unsigned char dataStacked = '0';     // Data is stacked from serial
unsigned char dataBuffered;        // Data is bufferd from spi to serial
unsigned char keep = 0;
volatile char stFlag = 0;

/* Private Function ------------------------------------------------------*/
/**
  * @brief Delay Millisecond.
  */
void delay_ms(unsigned int i){
   for(;i>0;i--)
      _delay_ms(1);
}
```

```
/**
 * @brief Initialize PORT.
 */
void Init_PORT(){
  sbi(DDRB,1);   // PWM
  //sbi(PORTB,1);
  //cbi(DDRD,2);   // INT0
  //cbi(DDRD,0);   // RXD
  //sbi(DDRD,1);   // TXD
}
/**
 * @brief Initialize UART.
 */
static void Init_USART(unsigned int baud)
{
  // Set baud rate *.bps.
  UBRRH =(unsigned char)(baud>>8);
  UBRRL =(unsigned char)baud;

  // Enable receiver , tramsmitter , Enable Receiver Interrupt.
  UCSRB = (1<<RXEN) | (1<<TXEN) | (1<<RXCIE);

  // Set frame format: 8data, NoneParity, 1stop bit.
  UCSRC = (1<<URSEL)|(0<<USBS)| (1<<UCSZ1) | (1<<UCSZ0);
}
/**
 * @brief Receive RXEN
 */
unsigned char receive(void){
  while(!(UCSRA&0x80));
  return UDR;
}
/**
 * @brief Transmit TXEN.
 */
static void transmit(unsigned char c){
  while(!(UCSRA&0x20));
  UDR = c;
  UCSRA |= 0x20;
```

```
}
/**
 * @brief Initialize Interrupt.
 */
void Init_Interrupt(){
  MCUCR = (1<<ISC01)|(0<<ISC00);   // INT0 Detect falling edge
  GICR  = (1<<INT0);          // INT Enable

  /* Set SREG for Global Interrupt */
  sei();
}
/**
 * @brief Initialize PWM.
 */
void Init_PWM(){
  TCCR1A = 0xF2;
  TCCR1B = 0x11;
  ICR1 = TOP_PWM;
  OCR1A = TOP_PWM;
  OCR1B = TOP_PWM/2;
}
/**
 * @brief Initialize parallel
 */
void parallelSetup(void){
  // PC3 , PC4, PC5 -> MISO
  // PC0 , PC1, PC2 -> MOSI
  // PB2 -> SCK
  // MOSI
  cbi(DDRC,0);
  cbi(DDRC,1);
  cbi(DDRC,2);
  // SCK
  cbi(DDRB,2);
  // MISO
  sbi(DDRC,3);
  sbi(DDRC,4);
  sbi(DDRC,5);
  // clear
```

```
    cbi(PORTC,3);
    cbi(PORTC,4);
    cbi(PORTC,5);
}
uint8_t parallelRead(void){
    uint8_t readBu[3];
     uint8_t readValue;
     readBu[0] = data0;
     readBu[1] = data1;
     readBu[2] = data2;
     readValue = (readBu[0]*1 + readBu[1]*2 + readBu[2]*4);
   return readValue;
}
void parallelWrite(uint8_t pw){
   switch(pw){
       case 0: cbi(PORTC,3);
            cbi(PORTC,4);
            cbi(PORTC,5);
          break;
       case 1: sbi(PORTC,3);
            cbi(PORTC,4);
            cbi(PORTC,5);
          break;
       case 2: cbi(PORTC,3);
            sbi(PORTC,4);
            cbi(PORTC,5);
          break;
       case 3: sbi(PORTC,3);
            sbi(PORTC,4);
            cbi(PORTC,5);
          break;
       case 4: cbi(PORTC,3);
            cbi(PORTC,4);
            sbi(PORTC,5);
          break;
       case 5: sbi(PORTC,3);
            cbi(PORTC,4);
            sbi(PORTC,5);
          break;
```

```
        case 6: cbi(PORTC,3);
                sbi(PORTC,4);
                sbi(PORTC,5);
              break;
        case 7: sbi(PORTC,3);
                sbi(PORTC,4);
                sbi(PORTC,5);
              break;
    }
}

/**
 * @brief Main.
 */
int main(void){
  Init_PORT();
  Init_PWM();
  Init_Interrupt();
  Init_USART(MYUBRR);
  parallelSetup();
  /*while(1){
    if(stFlag == 1){
      delay_ms(50);
      transmit(dataStacked);
      stFlag = 0;
      dataStacked = '0';
      transmit(dataStacked);
    }
  }*/
  /* Start Process */
  while(1){
    if((dataStacked<=47)&&(dataStacked>=55)){
      dataStacked = '0';
    }
    // Read when detect LOW
    if(!(PINB&0x04)){
        parallelWrite(dataStacked-48);
    }
    else{
```

```
        keep = parallelRead();
        if((keep==1)||(keep==2)||(keep==3)||(keep==4)){
          delay_ms(25);
          transmit(0x01);
          transmit(0x01);
          transmit('<');
          transmit(keep+48);

          delay_ms(500);
          transmit(0x01);
          transmit(0x01);
          transmit('<');
          transmit(keep+48);
          dataStacked = '0';
        }
      }

  }
  return 0;
}

/* Private Function -----------------------------------------*/
/**
  * @brief  Interrupt Service Routine of " USART1->RX ".
  *    Send Value From RX1[Infrared] to TX0[Computer].
  */
ISR(USART_RXC_vect){
    dataTemp = receive();
  if(dataTemp == '['){     // from car protocol
      stTemp = 1;
    stFlag = 0;
  }
  if((stTemp == 1)&&(dataTemp !='[')){
      stTemp = 0;
    stFlag = 1;
      dataStacked = dataTemp;
    //transmit(dataStacked);
  }
}
```

```
/**
 * @brief  Interrupt Service Routine of " External Interrupt ".
 *    Modulate Signal from USART Port.
 */
ISR(INT0_vect){
  if(toggle)
  {
    OCR1A = TOP_PWM/2;
    MCUCR = (1<<ISC01)|(1<<ISC00);   // INT0 Detect Rising edge
    GICR  = (1<<INT0);          // INT0 Enable
  }
  else if(!toggle)
  {
    OCR1A = TOP_PWM;
    MCUCR = (1<<ISC01)|(0<<ISC00);   // INT0 Detect falling edge
    GICR  = (1<<INT0);          // INT0 Enable
  }
  toggle = !toggle;
}
```

โปรแกรม VB

เขียนเพื่อให้แสดงหน้าจอส่วนสั่งการและแสดงผล

```
Imports System
Imports System.Threading
Imports System.IO.Ports
Public Class Form1
    Dim str_input As Integer, c As Integer
    Dim b As String
    Dim st As String
    Dim intnum As Integer = 0, turnnum As Integer = 0 'ระยะเวลาที่นับถอยหลังในหนึ่งรอบ
    Dim i As Integer = 0
    '############################  webcam ##############
    ' Create constant using attend in function of DLL file.
    ' กำหนดค่าคงที่ในการใช้ function ของ DLL file.

    Const WM_CAP As Short = &H400S
    Const WM_CAP_DRIVER_CONNECT As Integer = WM_CAP + 10
    Const WM_CAP_DRIVER_DISCONNECT As Integer = WM_CAP + 11
    Const WM_CAP_EDIT_COPY As Integer = WM_CAP + 30
    Const WM_CAP_SET_PREVIEW As Integer = WM_CAP + 50
    Const WM_CAP_SET_PREVIEWRATE As Integer = WM_CAP + 52
    Const WM_CAP_SET_SCALE As Integer = WM_CAP + 53
    Const WS_CHILD As Integer = &H40000000
    Const WS_VISIBLE As Integer = &H10000000
    Const SWP_NOMOVE As Short = &H2S
    Const SWP_NOSIZE As Short = 1
    Const SWP_NOZORDER As Short = &H4S
    Const HWND_BOTTOM As Short = 1
    Dim iDevice As Integer = 0 '  Normal device ID  device ID ปัจจุบันในเครื่อง
    Dim hHwnd As Integer ' Handle value to preview window ค่า Handle สำหรับการแสดงภาพในแตกละวิโดส์ว

    ' Delare function/ API function AVI capture DLL. เรียก API function จาก AVI capture DLL.

    Declare Function SendMessage Lib "user32" Alias "SendMessageA" ( _
        ByVal hwnd As Integer, ByVal wMsg As Integer, ByVal wParam As Integer, _
        ByVal lParam As Object) As Integer

    Declare Function SetWindowPos Lib "user32" Alias "SetWindowPos" (ByVal hwnd As Integer, _
        ByVal hWndInsertAfter As Integer, ByVal x As Integer, ByVal y As Integer, _
```

```
    ByVal cx As Integer, ByVal cy As Integer, ByVal wFlags As Integer) As Integer

Declare Function DestroyWindow Lib "user32" (ByVal hndw As Integer) As Boolean

Declare Function capCreateCaptureWindowA Lib "avicap32.dll" ( _
    ByVal lpszWindowName As String, ByVal dwStyle As Integer, ByVal x As Integer, _
    ByVal y As Integer, ByVal nWidth As Integer, ByVal nHeight As Short, ByVal hWndParent As Integer, _
    ByVal nID As Integer) As Integer

Declare Function capGetDriverDescriptionA Lib "avicap32.dll" (ByVal wDriver As Short, _
    ByVal lpszName As String, ByVal cbName As Integer, ByVal lpszVer As String, _
    ByVal cbVer As Integer) As Boolean

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load


    Label1.Text = "Credit by : Control and Mechatronics Engineering Of  KMITL....."
    SRP.Close()
    Dim name As String
    Dim portname As String()
    portname = IO.Ports.SerialPort.GetPortNames
    Cb_port.Sorted = True
    For Each name In portname '  เป็นการกำหนดค่าของport โดย portname คือ port IO ที่มีการเชื่อมต่ออยู่
        Cb_port.Items.Add(name) '  แล้วเอามาเก็บที่name จากนั้นให้add port จาก name มาใส่ในportที่จะทำการติดต่อ
    Next
    Lb_con.Text = ""
    Tm_data.Stop()
    tim_robot_go.Stop()
    Timer1.Stop()
    Tim_auto.Stop()
    Time_today.Interval = 100 'set ค่าระยะเวลาต่อรอบที่นับ การแสดงเวลาบอกหน้าจอ
    Time_today.Enabled = True 'แสดงเวลาที่หน้าจอ

    '############      tooltip      #######
    ToolTip1.SetToolTip(Btn_con, "ทำการเชื่อมต่อกับ main controller")
    ToolTip1.SetToolTip(Rd_auto, "สั่งให้ระบบทำงานอัตโนมัติ")
    ToolTip1.SetToolTip(Rd_man, "กำหนดค่าให้รถเคลื่อนที่ตามคำสั่ง")
    ToolTip1.SetToolTip(Btn_go, "สั่งให้รถมีการเคลื่อนไปยังstation ที่กำหนด")
    ToolTip1.SetToolTip(Btn_check, "check รถที่จอดที่ station")
```

```
        ToolTip1.SetToolTip(Btn_reset, "ยกเลิกการเชื่อมต่อกับ main controller")
        ToolTip1.SetToolTip(Cb_a, "สถานีต้นทาง")
        ToolTip1.SetToolTip(Cb_b, "สถานีปลายทาง")
        'b = InputBox("กรุณากรอกชื่อ", "ลงชื่อเข้าใช้งาน", "")
        'MessageBox.Show("สวัสดีครับคุณ " + b, "ยินดีต้อนรับ", MessageBoxButtons.OK)
        Label6.Text = " "
        Label5.Text = " "
    End Sub


    '########## time_data ทำงาน แล้วทำการรับค่ามาจาก maincontroller ###
    Private Sub Tm_data_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Tm_data.Tick
        Dim sr2 As String 'เก็บค่าที่ได้รับมาจาก serail port
        If SRP.IsOpen = True Then
            Lb_con.Text = "Connecting..."
            sr2 = SRP.ReadExisting
            TXT_status.Text += sr2

        End If
    End Sub

    '##############   function การทำงานของserial port  ############

    Private Sub Btn_con_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btn_con.Click
        If Cb_port.Text = "" Then
            MessageBox.Show("เลือก comport ที่จะทำการเชื่อมต่อก่อนทำการ connect", "เลือก Comport ก่อนนะ",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        Else
            Tm_data.Enabled = True
            'Timer2.Enabled = True
            If Btn_con.Text = "Connect" Then
                Btn_con.Text = "Disconnect"
                With SRP 'set parameter serial port
                    .PortName = Cb_port.Text
                    .BaudRate = 9600
                    .DataBits = 8
                    .StopBits = IO.Ports.StopBits.One
                    .Parity = IO.Ports.Parity.None
```

```
            SRP.Open()
            ToolTip1.SetToolTip(Btn_con, "ยกเลิกการเชื่อมต่อกับ main controller")
        End With
    ElseIf Btn_con.Text = "Disconnect" Then

        If MessageBox.Show(" คุณต้องการยกเลิกการเชื่อมต่อกับ Process หรือ ไม่ ", "Worning", MessageBoxButtons.OKCancel, _
  MessageBoxIcon.Question) = Windows.Forms.DialogResult.OK Then
            SRP.Close()
            Lb_con.Text = ""
            Btn_con.Text = "Connect"
            Tm_data.Enabled = False
        Else
            Lb_con.Text = "Connecting..."
            Btn_con.Text = "Disconnect"
        End If
    End If
  End If
End Sub

Private Sub Btn_reset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btn_reset.Click
  TXT_status.Clear()
  Cb_a.Text = ""
  Cb_b.Text = ""
  Cb_c.Text = ""
  Cb_d.Text = ""
End Sub

Private Sub Btn_check_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btn_check.Click
  If SRP.IsOpen = True Then 'check หมายเลขรถแต่ละสถานี
    SRP.Write("c")
    SRP.Write("h")
    SRP.Write("e")
    SRP.Write("c")
    SRP.Write("k")
    SRP.Write(" ")
  Else
    TXT_status.Text = "Don't connect"
  End If
End Sub
```

```
Sub output(ByVal a_out As String, ByVal b_out As String) 'ตัวแปรส่งค่าไปผ่านserialportเพื่อส่งไปให้สถานีทำงาน
    SRP.Write(a_out) 'ค่าที่จะส่งไปยังสถานีต้นทาง
    SRP.Write(b_out) 'ค่าที่จะสั่งให้รถเคลื่อนที่ไปยังปลายทาง
    SRP.Write(" ")
End Sub

Sub arm(ByVal num_arm As String) 'สั่งการแขนกลเพื่อให้ทำงาน
    SRP.Write("R")
    SRP.Write("M")
    SRP.Write(num_arm) 'เลือกชุดคำสั่งที่อยู่บน maincontroller
    SRP.Write(" ")
End Sub

Private Sub Btn_go_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btn_go.Click
    If SRP.IsOpen = True Then  'checkเงื่อนไขว่าport มีการเชื่อมต่อหรือไม่
        If Rd_man.Checked = True Then    'check condition ว่ามีการเลือกที่ manual controlหรือไม่
            If Ckb_allarm.Checked = True Then
                arm("0")
            End If
            If Ckb_arm1.Checked = True Then
                arm("3")
            End If
            If Ckb_arm2.Checked = True Then
                arm("4")
            End If
            If Ckb_arm3.Checked = True Then
                arm("1")
            End If
            If Ckb_arm4.Checked = True Then
                arm("2")
            End If
            If Ckb_stall.Checked = True Then
                output("A", "B") '5
                output("B", "C") '6
                output("C", "D") '7
                output("D", "A") '8
                'Ckb_allarm.CheckState = CheckState.Unchecked
            End If
            If Ckb_stA.Checked = True Then
```

```
            Select Case Cb_a.Text    'เลือกเงื่อนไขการป้อนตัวอักษรในช่องที่จะให้รถเคลื่อนไป
              Case "A"
                output("A", "A") '1
              Case "B"
                output("A", "B") '5
              Case "C"
                output("A", "C") '9
              Case "D"
                output("A", "D") '13
            End Select
            If Cb_a.Text = "" Then
              MessageBox.Show("ยังไม่ได้ทำการเลือกสถานีปลายทางของสถานี A", "Select the final station",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            End If
          End If
          If Ckb_stB.Checked = True Then
            Select Case Cb_b.Text
              Case "B"
                output("B", "B") '2
              Case "C"
                output("B", "C") '6
              Case "D"
                output("B", "D") '10
              Case "A"
                output("B", "A") '14
            End Select
            If Cb_b.Text = "" Then
              MessageBox.Show("ยังไม่ได้ทำการเลือกสถานีปลายทางของสถานี B", "Select the final station",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            End If
          End If
          If Ckb_stC.Checked = True Then
            Select Case Cb_c.Text
              Case "C"
                output("C", "C") '3
              Case "D"
                output("C", "D") '7
              Case "A"
                output("C", "A") '11
```

```
                Case "B"
                    output("C", "B") '15
                End Select
                If Cb_c.Text = "" Then
                    MessageBox.Show("ยังไม่ได้ทำการเลือกสถานีปลายทางของสถานี C", "Select the final station",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
                End If
            End If
            If Ckb_stD.Checked = True Then
                Select Case Cb_d.Text
                    Case "D"
                        output("D", "D") '4
                    Case "A"
                        output("D", "A") '8
                    Case "B"
                        output("D", "B") '12
                    Case "C"
                        output("D", "C") '16
                End Select
                If Cb_d.Text = "" Then
                    MessageBox.Show("ยังไม่ได้ทำการเลือกสถานีปลายทางของสถานี D ", "Select the final station",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
                End If
            End If
        ElseIf Rd_auto.Checked = True Then
            tim_robot_go.Interval = 60000
            Timer1.Interval = 1000
            tim_robot_go.Enabled = True
            Timer1.Enabled = True
        End If
    Else
        TXT_status.Text = "Don't connect"
    End If
End Sub
Private Sub tim_robot_go_Tick_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tim_robot_go.Tick
    If Rd_auto.Checked = True Then
        If SRP.IsOpen = True Then
            If turnnum < NumericUpDown1.Value Then
                intnum = intnum + 1
```

```
            SRP.Write("A")
            SRP.Write("U")
            SRP.Write("T")
            SRP.Write(" ")
            If intnum Mod 4 = 0 Then
              turnnum = turnnum + 1
              intnum = 0
            End If
          Else
            tim_robot_go.Enabled = False
            Timer1.Enabled = False
            NumericUpDown1.Value = 0
            intnum = 0
            turnnum = 0
          End If
        End If
      End If
    End Sub

    Private Sub Time_today_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Time_today.Tick
      Label4.Text = "To day :" & Today & " " & TimeOfDay 'show time of today
      If Ckb_stall.CheckState = CheckState.Checked Then
        If Ckb_stA.Checked = True Then
          Ckb_stA.CheckState = CheckState.Unchecked
          'MessageBox.Show("กรุณาเอาเครื่องหมายถูกออกก่อนทำการเลือก", "เลือกสั่งได้ทีละแบบนะ", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)

        ElseIf Ckb_stB.Checked = True Then
          Ckb_stB.CheckState = CheckState.Unchecked
          'MessageBox.Show("กรุณาเอาเครื่องหมายถูกออกก่อนทำการเลือก", "เลือกสั่งได้ทีละแบบนะ", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)

        ElseIf Ckb_stC.Checked = True Then
          Ckb_stC.CheckState = CheckState.Unchecked
          'MessageBox.Show("กรุณาเอาเครื่องหมายถูกออกก่อนทำการเลือก", "เลือกสั่งได้ทีละแบบนะ", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)

        ElseIf Ckb_stD.Checked = True Then
```

```
            Ckb_stD.CheckState = CheckState.Unchecked
            'MessageBox.Show("กรุณาเอาเครื่องหมายถูกออกก่อนทำการเลือก", "เลือกสั่งได้ทีละแบบนะ", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)

          End If
        End If
    End Sub

    Private Sub Rd_man_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Rd_man.CheckedChanged
        If SRP.IsOpen = True Then
          If Rd_man.Checked = True Then
            MessageBox.Show("กรุณาเลือกสถานีที่จะให้รถเคลื่อนที่ไป แล้วกดปุ่ม go", "เลือก Station", _
              MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
          End If
        Else
          TXT_status.Text = "Don't connect "

        End If
    End Sub
    Private Sub Rd_auto_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Rd_auto.CheckedChanged
        If SRP.IsOpen = True Then
          If Rd_auto.Checked = True Then
            MessageBox.Show("กรุณากรอกตัวเลขเป็นตัวเลขครับ แล้วกดปุ่ม go", "กรอกจำนวนรอบ", _
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
          End If
        Else
          TXT_status.Text = "Don't connect "
        End If
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        If Rd_auto.Checked = True Then
          If SRP.IsOpen() = True Then
            i = i + 1
            If i = 60 Then
              i = 0
            End If
```

```
            Label5.Text = " รอบที่ : " & turnnum.ToString & " รถเคลื่อนที่ไป " & intnum.ToString & " station"
        Else
            i = 0
        End If
        Label6.Text = i.ToString
    End If
End Sub
'*****************************************************************************************

" ขั้นตอนสุดท้าย, การปิดแสดงภาพใน โปรแกรมของเรา,เลิกทำการติดต่อ device และ destroy (ทำลาย Object) ที่ เราสร้างขึ้นมา
ตอนต้นการเขียนโปรแกรม ที่ให้แสดงภาพ และปิดโปรแกรม
' ติดต่อ device ของกล้อง.
Private Sub LoadDeviceList()
    Dim strName As String = Space(100)
    Dim strVer As String = Space(100)
    Dim bReturn As Boolean
    Dim x As Integer = 0
    ' เรียกรายชื่อของ device กล้องทั้งหมดในเครื่องคอมของเรา และโชว์ใน List Box ที่ชื่อ lstDevices .
    Do
        '  เรียกชื่อ Driver และ version
        bReturn = capGetDriverDescriptionA(x, strName, 100, strVer, 100)
        ' ถ้าใช่ device ให้เพิ่มชื่อเข้าไปใน  list box
        If bReturn Then lstDevices.Items.Add(strName.Trim)
        x += 1
    Loop Until bReturn = False

End Sub

' แสดงผลของรูปภาพเคลื่อนไหวออกที่หน้าฟอร์มโปรแกรมที่เราสร้าง
Private Sub OpenPreviewWindow()
    Dim iHeight As Integer = picCapture.Height
    Dim iWidth As Integer = picCapture.Width
    ' เปิดแสดงผลที่ picturebox .
    ' สร้าง window ลูก ขึ้นมาด้วย ฟังชั่น capCreateCaptureWindowA ซึ่งคุณสามารถเห็นใน picturebox.
    hHwnd = capCreateCaptureWindowA(iDevice, WS_VISIBLE Or WS_CHILD, 0, 0, 640, _
        480, picCapture.Handle.ToInt32, 0)
    ' ติดต่อกับ device
    If SendMessage(hHwnd, WM_CAP_DRIVER_CONNECT, iDevice, 0) Then
        ' ตั้งค่า preview scale
```

```
        SendMessage(hHwnd, WM_CAP_SET_SCALE, True, 0)
        ' ตั้งค่า preview rate ในระดับ milliseconds
        SendMessage(hHwnd, WM_CAP_SET_PREVIEWRATE, 66, 0)
        ' เริ่มต้นการแสดงภาพ จากกล้อง
        SendMessage(hHwnd, WM_CAP_SET_PREVIEW, True, 0)
        ' หรับขนาด window ให้เท่ากับใน picturebox
        SetWindowPos(hHwnd, HWND_BOTTOM, 0, 0, picCapture.Width, picCapture.Height, _
                     SWP_NOMOVE Or SWP_NOZORDER)
        BtnSave.Enabled = True
        btnStop.Enabled = True
        btnStart.Enabled = False
    Else
        ' การติดต่อdevice Error  ให้ ปิด window
        DestroyWindow(hHwnd)
        BtnSave.Enabled = False
    End If
End Sub

' ใช้ฟังก์ชั่นชื่อ SendMessage ไป copy ข้อมูลไว้ใน clipboard ซึ่งย้ายภาพไปที่ picture box.
Private Sub btnSave_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnSave.Click
    Dim data As IDataObject
    Dim bmap As Image
    ' Copy ภาพ ไป clipboard
    SendMessage(hHwnd, WM_CAP_EDIT_COPY, 0, 0)
    'นำ ภาพ จาก clipboard และ convert มันไปเป็น bitmap
    data = Clipboard.GetDataObject()
    If data.GetDataPresent(GetType(System.Drawing.Bitmap)) Then
        bmap = CType(data.GetData(GetType(System.Drawing.Bitmap)), Image)
        picCapture.Image = bmap
        ClosePreviewWindow()
        BtnSave.Enabled = False
        btnStop.Enabled = False
        btnStart.Enabled = True
        If sfdImage.ShowDialog = DialogResult.OK Then
            bmap.Save(sfdImage.FileName, Imaging.ImageFormat.Bmp)
        End If
        OpenPreviewWindow()
    End If
End Sub
```

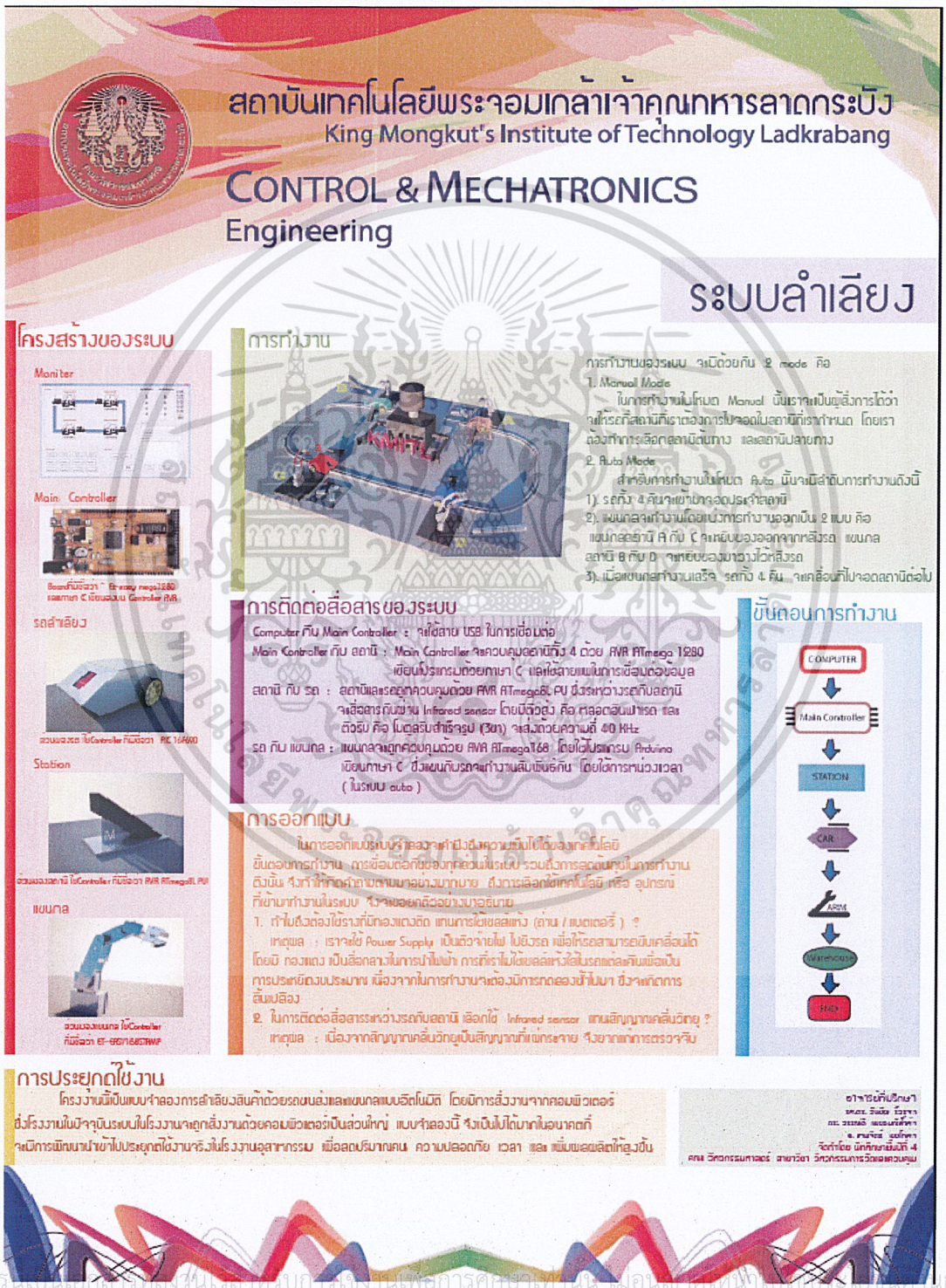
```
    ' ขั้นตอนสุดท้าย, การปิดแสดงภาพใน โปรแกรมของเรา,เลิกทำการติดต่อ deviceและ destroy (ทำลาย Object) ที่ เราสร้างขึ้นมาตอนต้น
การเขียนโปรแกรม ที่ให้แสดงภาพ และปิดโปรแกรม
    Private Sub ClosePreviewWindow()
        ' เลิกติดต่อกับ device
        SendMessage(hHwnd, WM_CAP_DRIVER_DISCONNECT, iDevice, 0)
        ' ปิด window
        DestroyWindow(hHwnd)
    End Sub

    Private Sub btnStop_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnStop.Click
        ClosePreviewWindow()
        btnStart.Enabled = True
    End Sub

    Private Sub btnStart_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnStart.Click
        OpenPreviewWindow()
    End Sub
End Class
```

ภาคผนวก ข

โปสเตอร์ระบบลำเลียงสินค้า



เอกสารอ้างอิง

[1] ประจิน พลังสันติกุล. **เรียนรู้และใช้งาน CCS C คอมไพล์เลอร์ เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์ PIC.** พิมพ์ครั้งที่1. กรุงเทพนคร : อินโนเวตีฟ เอ็กเพอริเมนต์. 2547.

[2] นิรุธ อำนวยศิลป์. **คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0 ฉบับเพื่อการใช้งานจริง.** กรุงเทพนคร : ซัคเซส มีเดีย.

[3] Webmaster. " S3003 FUTABA SERVO ."[Online]. Available : http://www.etteam.com/product/1602.html.2553.

[4] Webmaster. "(EK2-0508) Digital Servo."[Online]. Available : http://www.helipal.com/ek2-0508-digital-servo-8g-for-esky-helicopters.html.2553..

[5] Webmaster. "**CNC**."[Online]. Available : http://pirun.ku.ac.th/~b4755376/link/CAD5.html.2553.

[6] Webmaster. "Computer-integrated manufacturing (CIM)."[Online]. Available : http://en.wikipedia.org/wiki/Computer-integrated_manufacturing.2553.