การออกแบบและพัฒนาหุ่นยนต์แบบขนาน

DESIGN AND DEVELOPMENT OF PARALLEL ROBOT

นางสาวชนิกานต์ มุสิกทอง

MS. CHANIKAN MUSIKTHONG

นางสาวปัทมาพร แก้วกุดฉิม

MS. PATTHAMAPORN KAEWKUDCHIM

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

การออกแบบและพัฒนาหุ่นยนต์แบบขนาน

DESIGN AND DEVELOPMENT OF PARALLEL ROBOT

นางสาวชนิกานต์ มุสิกทอง
MS.CHANIKAN MUSIKTHONG
นางสาวปัทมาพร แก้วกุดฉิม
MS.PATTHAMAPORN KAEWKUDCHIM

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

DESIGN AND DEVELOPMENT OF PARALLEL ROBOT

MS.CHANIKAN MUSIKTHONG

MS.PATTHAMAPORN KAEWKUDCHIM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INDUSTRIAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2013

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์ การออกแบบและพัฒนาหุ่นยนต์แบบขนาน

DESIGN AND DEVELOPMENT OF PARALLEL ROBOT

นักศึกษา นางสาวชนิกานต์ มุสิกทอง รหัสประจำตัว 53010305

นางสาวปัทมาพร แก้วกุดฉิม รหัสประจำตัว 53010976

หลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมอุตสาหการ

อาจารย์ผู้ควบคุมปริญญานิพนธ์

_____

(ดร.อุดม จันทร์จรัสสุข)

หัวข้อปริญญานิพนธ์	การออกแบบและพัฒนาหุ่นยนต์แบบขนาน
นักศึกษา	นางสาวชนิกานต์ มุสิกทอง
	นางสาวปัทมาพร แก้วกุดฉิม
หลักสูตร	วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมอุตสาหการ
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา	2556
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ดร.อุดม จันทร์จรัสสุข

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้ทำการศึกษาการทำงานของหุ่นยนต์แบบขนานชนิดสามแกน โดยทำการออกแบบโครงสร้างของหุ่นยนต์รวมทั้งพัฒนาโปรแกรมควบคุมการทำงานทั้งหมดของหุ่นยนต์ หุ่นยนต์ที่พัฒนาขึ้นมีโครงสร้างเป็นอลูมิเนียม ประกอบด้วยแขนกลที่ขับเคลื่อนด้วยมอเตอร์แบบสเต็ปปิ้งจำนวน 3 แขน โดยยึดติดกับโครงของหุ่นยนต์จากด้านบนทำมุมกัน 120 องศา ที่ส่วนปลายของแขนด้านบนทั้งสามแขนจะถูกต่อกับแขนท่อนล่างซึ่งมีฐานยึดติดด้านล่าง การทำงานของหุ่นยนต์ถูกควบคุมโดยไมโครคอนโทรลเลอร์ (MCS-51) ซึ่งใช้เทคนิคการควบคุมตำแหน่งแบบเปิด (Open Loop) โดยรับคำสั่งจากเครื่องคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม (RS-232) โปรแกรมควบคุมการทำงานของหุ่นยนต์บนเครื่องคอมพิวเตอร์ถูกพัฒนาขึ้นโดยใช้ภาษา Visual C# โปรแกรมมีหน้าที่รับค่าตำแหน่งการเคลื่อนที่ของหุ่นยนต์และคำนวณหาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์แต่ละแกน โดยใช้สมการการเคลื่อนที่แบบผกผัน (Inverse Kinematics) และส่งคำสั่งไปให้ไมโครคอนโทรลเลอร์ จากการทดสอบการทำงานของหุ่นยนต์แบบขนานที่ได้พัฒนาขึ้น พบว่าหุ่นยนต์สามารถเคลื่อนที่ตามตำแหน่งที่ต้องการได้อย่างแม่นยำ

Thesis Title	Design and Development of Parallel Robot
Student	Ms.Chanikan Musikthong
	Ms.Patthamaporn Kaewkudchim
Degree	Bachelor of Engineering in Industrial Engineering
	King Mongkut's Institute of Technology Ladkrabang
Academic Year	2013
Thesis Advisor	Dr.Udom Janjarassuk

Abstract

This thesis studies the working principle of 3 axes parallel robot including its structure and control, then a prototype of a 3 axes parallel robot is built and developed. The robot arms are made of aluminum and driven by 3 stepping motors which adhered to the structure from above in 120 degree angle. The 3 upper arms of robot are connected with the axle of motors and the lower arms are adhered to the lower base. The robot is controlled by using microcontroller (MCS-51) with open loop control, and the command is received from a computer through serial port (RS-232). The control software is developed in Visual C# and is used to calculate the input position by using Inverse Kinematics equation in order to find the number of steps and movement of each motor. The results are sent to microcontroller. From the experiment, the robot can accurately move to the assigned positions.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ ด้วยความกรุณาของ ดร.อุดม จันทร์จรัสสุข อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งได้ให้คำปรึกษา ข้อชี้แนะ และความช่วยเหลือในหลายสิ่งหลายอย่างจนกระทั่งลุล่วงไปได้ด้วยดีคณะผู้จัดทำขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

ขอกราบขอบพระคุณ ดร.พลชัย โชติปรายนกุล และ ผศ.ดร. สกนธ์ คล่องบุญจิต กรรมการสอบปริญญานิพนธ์ ที่ให้ความกรุณาในการให้คำแนะนำที่ดี ชี้แนะข้อบกพร่องต่างๆ ของโครงงาน

ขอกราบขอบพระคุณ คณาจารย์และเจ้าหน้าที่ภาควิชาวิศวกรรมอุตสาหการทุกท่าน ที่ได้ให้ความรู้ คำแนะนำ ความช่วยเหลือและกำลังใจตลอดการศึกษาที่ผ่านมา

ขอขอบคุณและขอบใจ พี่บัณฑิต เพื่อน และน้องภาควิชาวิศวกรรมอุตสาหการและภาควิชาอื่นๆ ที่ให้ความช่วยเหลือ คำแนะนำ ความห่วงใยและให้กำลังใจเสมอมารวมถึงผู้มีพระคุณทุกท่านที่มิได้เอ่ยนามไว้ ณ ที่นี้

สุดท้ายนี้คณะผู้จัดทำขอกราบขอบพระคุณบิดามารดาและครอบครัวซึ่งคอยให้กำลังใจและเป็นผู้ให้ทุกสิ่งอย่างกับคณะผู้จัดทำ

นางสาวชนิกานต์ มุสิกทอง
นางสาวปัทมาพร แก้วกุดฉิม

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

สารบัญรูป(ต่อ)

หน้า

สารบัญรูป(ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันในวงการอุตสาหกรรม หุ่นยนต์ได้เข้ามามีบทบาทมากขึ้น โดยได้นำมาใช้แทนแรงงานมนุษย์ในงานที่อันตราย ได้แก่ งานยกเหล็กเข้าเตาหลอม งานที่เกี่ยวข้องกับสารเคมี รวมถึงงานที่ต้องการความแม่นยำและคุณภาพมาตรฐานสูง ได้แก่ งานทางด้านอิเล็กทรอนิกส์ งานประกอบอุปกรณ์ งานบรรจุภัณฑ์และคัดเลือกผลิตภัณฑ์ เป็นต้น ซึ่งเทคโนโลยีทางด้านหุ่นยนต์นี้มีความยืดหยุ่นสูงสามารถปรับเปลี่ยนการทำงานได้ในหลายรูปแบบ นอกจากนี้การนำหุ่นยนต์อุตสาหกรรมเข้ามาใช้ในกระบวนการผลิตยังส่งผลให้คุณภาพของผลิตภัณฑ์ที่ได้มีความสม่ำเสมอและเป็นมาตรฐานเดียวกัน

โดยเฉพาะอย่างยิ่งในอุตสาหกรรมด้านบรรจุภัณฑ์และการคัดเลือกผลิตภัณฑ์ ซึ่งเป็นลักษณะงานที่ต้องทำซ้ำต่อเนื่องและต้องการความถูกต้องแม่นยำ จึงมีการนำเอาหุ่นยนต์แบบขนานเข้ามามีส่วนช่วยในกระบวนการทำงาน เพื่อเป็นการเพิ่มขีดความสามารถในการผลิตและประสิทธิภาพในกระบวนการทำงานให้สูงขึ้น เนื่องจากหุ่นยนต์แบบขนานมีข้อดี คือ น้ำหนักเบา มีการทำงานที่รวดเร็ว แม่นยำเที่ยงตรง สามารถทำงานได้โดยไม่ต้องพักและสามารถรับภาระงานหนักได้ ถึงแม้หุ่นยนต์แบบขนานนี้จะมีข้อดีมากมายตามที่กล่าวไว้ข้างต้น แต่กลับพบว่าหุ่นยนต์ประเภทนี้ยังไม่เป็นที่นิยมอย่างแพร่หลายในวงการอุตสาหกรรมของประเทศไทยเท่าที่ควร เนื่องจากหุ่นยนต์ประเภทนี้มีราคาที่สูงมากและต้องนำเข้าจากต่างประเทศ ด้วยเหตุนี้คณะผู้จัดทำจึงเล็งเห็นถึงความสำคัญของการพัฒนางานวิจัยด้านหุ่นยนต์แบบขนาน โดยคำนึงถึงการออกแบบและพัฒนาหุ่นยนต์แบบขนานในขั้นต้นที่มีต้นทุนไม่สูงมากนัก เพื่อใช้เป็นต้นแบบในการพัฒนาต่อยอดและประยุกต์ใช้ในอุตสาหกรรมไทยในอนาคต

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษากระบวนการทำงาน โครงสร้างและส่วนประกอบต่างๆของหุ่นยนต์แบบขนาน

1.2.2 เพื่อสร้างหุ่นยนต์แบบขนานที่สามารถเคลื่อนที่ตามตำแหน่งที่กำหนดได้

1.2.3 เพื่อเป็นหุ่นยนต์แบบขนานในการพัฒนาต่อยอดต่อไป

1.3 ขอบเขตปริญญานิพนธ์

1.3.1 สร้างหุ่นยนต์แบบขนานที่มีโครงสร้างประกอบด้วยแกน 3 แกน

1.3.2 ใช้การควบคุมการทำงานของหุ่นยนต์แบบเปิด (Open loop) โดยขับเคลื่อนด้วยมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor)

1.3.3 ภาระของชิ้นงานที่หุ่นยนต์แบบขนานสามารถรองรับได้มีน้ำหนักไม่เกิน 100 g

1.3.4 การคำนวณหาสมการเพื่อใช้ในการควบคุมหุ่นยนต์ให้เคลื่อนที่ตามตำแหน่งที่ต้องการ

1.3.5 สร้างหุ่นยนต์แบบขนานที่มีความสูงไม่เกิน 750 mm

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถนำหุ่นยนต์แบบขนานไปพัฒนาต่อยอดและประยุกต์ใช้ในอุตสาหกรรม

1.4.2 สามารถนำหุ่นยนต์แบบขนานไปเป็นแนวทางพัฒนาทางด้านการศึกษาและวิจัยต่อไปในอนาคต

1.4.3 เป็นต้นแบบสำหรับใช้ในการศึกษากระบวนการทำงานของหุ่นยนต์แบบขนาน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

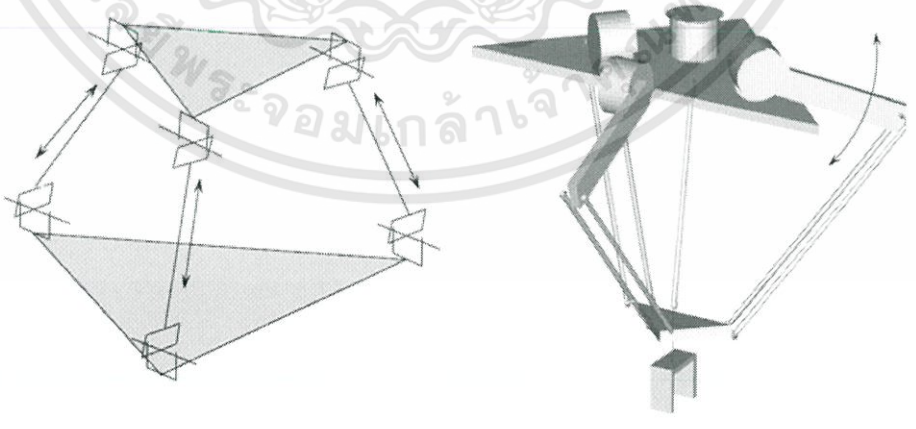
การออกแบบและพัฒนาหุ่นยนต์แบบขนาน (Delta Robot) ต้องอาศัยการศึกษาทฤษฎีในการกำหนดทิศทางการเคลื่อนที่และอุปกรณ์ที่นำมาใช้ในการสร้างและออกแบบตัวหุ่นยนต์ ซึ่งการศึกษาทฤษฎีเป็นตัวช่วยในการสร้างและการออกแบบหุ่นยนต์ให้มีประสิทธิภาพและเหมาะสมยิ่งขึ้น โดยทฤษฎีที่ได้ทำการศึกษามีดังต่อไปนี้

2.1 หุ่นยนต์แบบขนาน

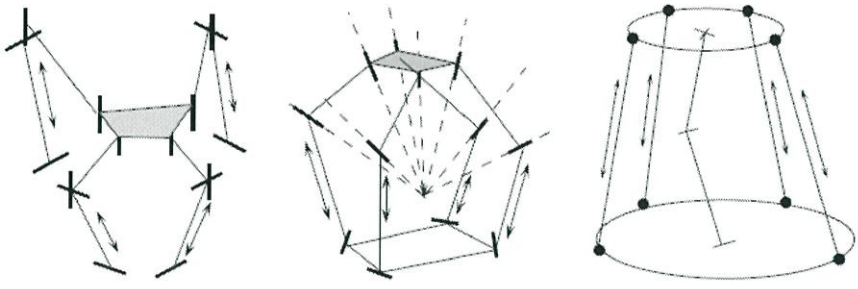
หุ่นยนต์แบบขนาน เป็นหุ่นยนต์ที่มีประสิทธิภาพในด้านของความถูกต้อง แม่นยำ สามารถจัดการกับโหลดที่มีขนาดใหญ่ได้ดี หุ่นยนต์แบบขนานประกอบด้วยแขนจำนวนสามแขนหรือมากกว่า โดยสามารถประยุกต์ใช้ในด้านการแพทย์ ดาราศาสตร์ งานวิจัย รวมถึงงานทางด้านอุตสาหกรรมในปัจจุบัน

2.2 ประเภทของหุ่นยนต์แบบขนาน

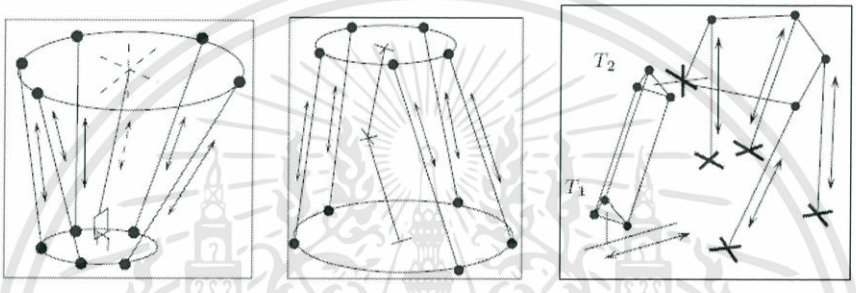
หุ่นยนต์แบบขนานสามารถแบ่งแยกได้หลากหลายรูปแบบตามลักษณะเฉพาะของการใช้งาน เช่น การจำแนกตามแกนการขับเคลื่อน ซึ่งมีจำนวน 3 แกนหรือมากกว่า โดยลักษณะการเคลื่อนที่ของแต่ละประเภทมีทิศทางการเคลื่อนที่ของแขนเป็นตามแนวลูกศร แบบขับเคลื่อน 3 แกน ดังแสดงในรูปที่ 2.1 แบบขับเคลื่อน 4 แกน ดังแสดงในรูปที่ 2.2 แบบขับเคลื่อน 5 แกน ดังแสดงในรูปที่ 2.3 แบบขับเคลื่อน 6 แกน ดังแสดงในรูปที่ 2.4



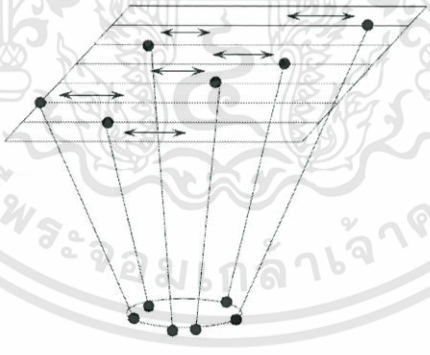
รูปที่ 2.1 โครงสร้างของหุ่นยนต์แบบขนานขับเคลื่อน 3 แกน [19]



รูปที่ 2.2 โครงสร้างของหุ่นยนต์แบบขนานขับเคลื่อน 4 แกน [19]



รูปที่ 2.3 โครงสร้างของหุ่นยนต์แบบขนานขับเคลื่อน 5 แกน [19]

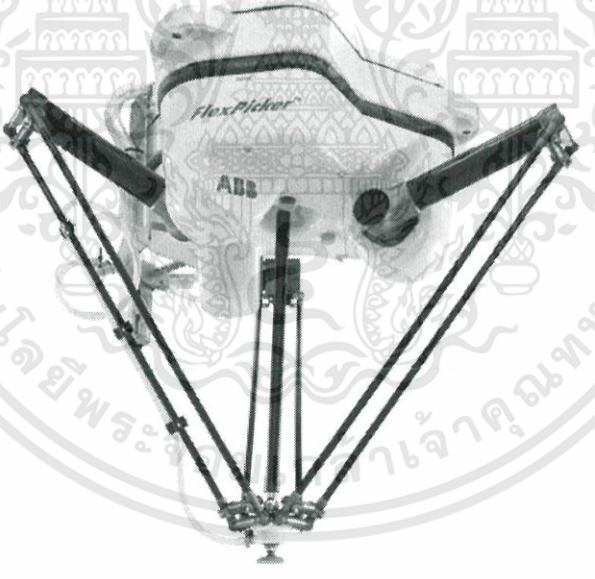


รูปที่ 2.4 โครงสร้างของหุ่นยนต์แบบขนานขับเคลื่อน 6 แกน [19]

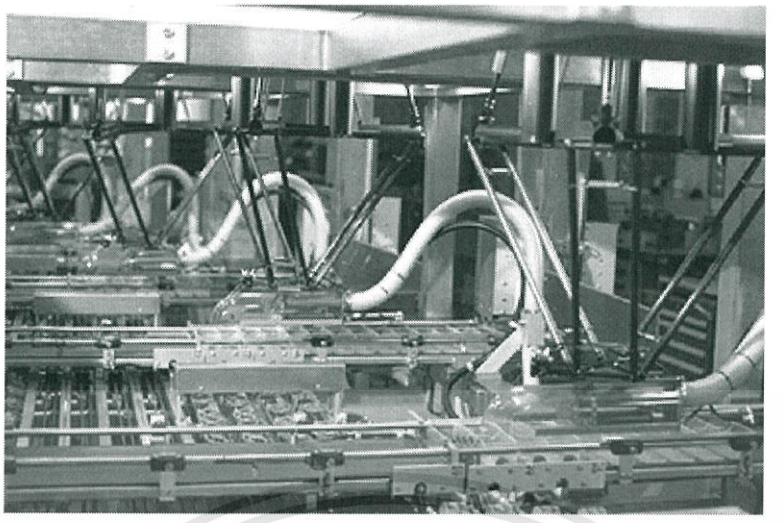
2.3 หุ่นยนต์แบบขนาน (Delta Robot) [14]

หุ่นยนต์แบบขนาน (Delta Robot) ถูกประดิษฐ์ขึ้นเมื่อต้นยุค ค.ศ.1980 โดยทีมนักวิจัยซึ่งนำโดยศาสตราจารย์เรย์มอนด์ คลาฟเวล (Prof. Raymond Clavel) จุดประสงค์หลักในการประดิษฐ์หุ่นยนต์แบบขนานนี้ก็เพื่อจัดการกับวัตถุที่มีน้ำหนักเบาและขนาดเล็กด้วยความเร็วสูง ซึ่งหุ่นยนต์ประเภทนี้มีข้อดี คือ มีความแม่นยำ เที่ยงตรง มีการทำงานที่รวดเร็ว สามารถรับภาระงานหนักได้ โดยหุ่นยนต์แบบขนาน (Delta Robot) นี้ประกอบด้วยแขน 3 แขน เชื่อมต่อกันโดยข้อต่ออ่อน (Universal Joints) การออกแบบที่สำคัญคือการใช้ กฎรูปสี่เหลี่ยมด้านขนาน (Parallelogram Law) มาเป็นการควบคุมทิศทางให้กับ เครื่องมือที่ติดกับปลายแขนหุ่นยนต์ (End Effector) เครื่องมือนี้สามารถเป็นได้หลายอย่าง เช่น เซนเซอร์ อุปกรณ์การยึดจับ หรือเครื่องเจาะ เป็นต้น ทั้งนี้ขึ้นอยู่กับผู้ใช้ว่าต้องการนำไปใช้งานในลักษณะใด

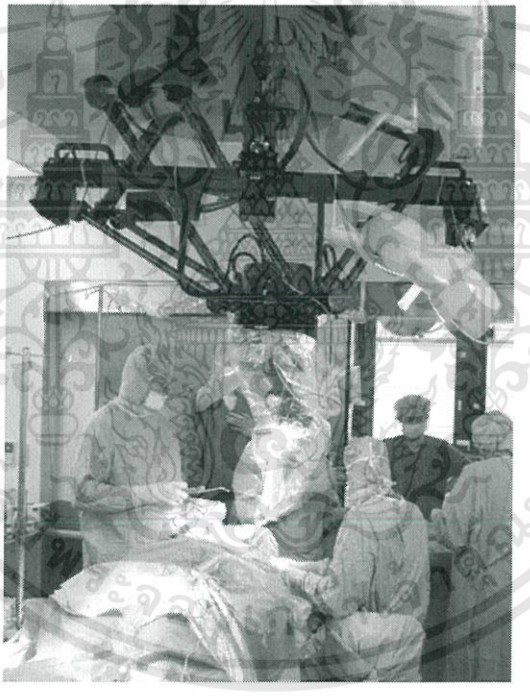
หุ่นยนต์แบบขนาน (Delta Robot) ที่ใช้ในปัจจุบันมีหลายแบบ เช่น หุ่นยนต์เดลต้ารุ่น ABB Flexible Automation ใช้งานในอุตสาหกรรมอาหาร ยา และอุปกรณ์อิเล็กทรอนิกส์ ซึ่งหุ่นยนต์นี้มีความสามารถในการคัดเลือกอย่างรวดเร็วและรับน้ำหนักได้ถึง 1 kg ดังตัวอย่างรูปที่ 2.5 หุ่นยนต์เดลต้ารุ่น Line-Placer ของบริษัท Demaurex ใช้งานในส่วนของการคัดเลือก และการบรรจุภัณฑ์ในอุตสาหกรรมเบเกอรี่ ดังตัวอย่างรูปที่ 2.6 และหุ่นยนต์เดลต้ารุ่น SurgiScope ของบริษัท Demaurex ใช้เป็นหุ่นยนต์ผ่าตัดในห้องปฏิบัติการของมหาวิทยาลัยฮุมโบลด์ทแห่งกรุงเบอร์ลิน ดังแสดงในรูปที่ 2.7



รูปที่ 2.5 หุ่นยนต์เดลต้ารุ่น ABB Flexible Automation [14]



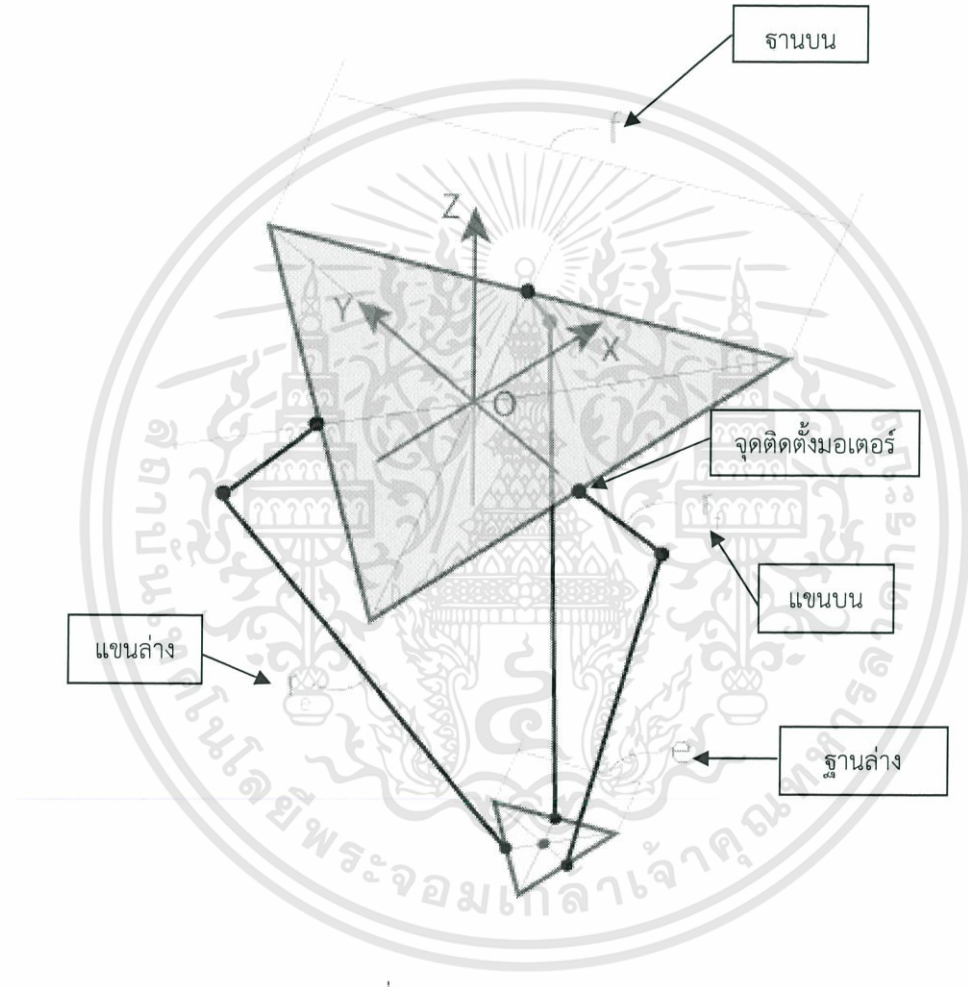
รูปที่ 2.6 หุ่นยนต์เดลต้ารุ่น Line-Placer [14]



รูปที่ 2.7 หุ่นยนต์เดลต้ารุ่น SurgiScope ของบริษัท Demaurex [14]

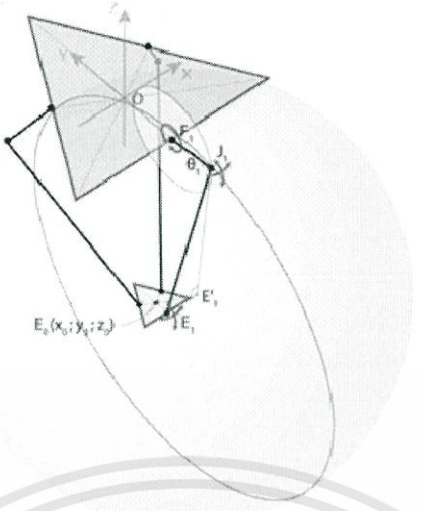
2.4 การศึกษาการเคลื่อนที่ของหุ่นยนต์โดยใช้หลักการของการเคลื่อนที่แบบผกผัน (Inverse Kinematics) [17]

การเคลื่อนที่แบบผกผัน (Inverse Kinematics) เป็นการศึกษาการเคลื่อนที่ของหุ่นยนต์โดยที่ผู้ควบคุมกำหนดตำแหน่ง แล้วนำไปคำนวณหามุมและจำนวนสเต็ปการเคลื่อนที่ของสเต็ปปิ้งมอเตอร์ (Stepping Motor) เพื่อให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งที่ผู้ควบคุมต้องการ โดยพิกัดตำแหน่งของหุ่นยนต์จะถูกกำหนดให้อยู่ในแกน X,Y,Z โดยแสดงดังรูปที่ 2.8 จุด O คือจุดที่เกิดจากการตัดกันของแกน X,Y,Z โดยจะกำหนดให้จุด O มีพิกัดเป็น (0,0,0) และเป็นจุดกำเนิดของปริภูมิสามมิติของหุ่นยนต์



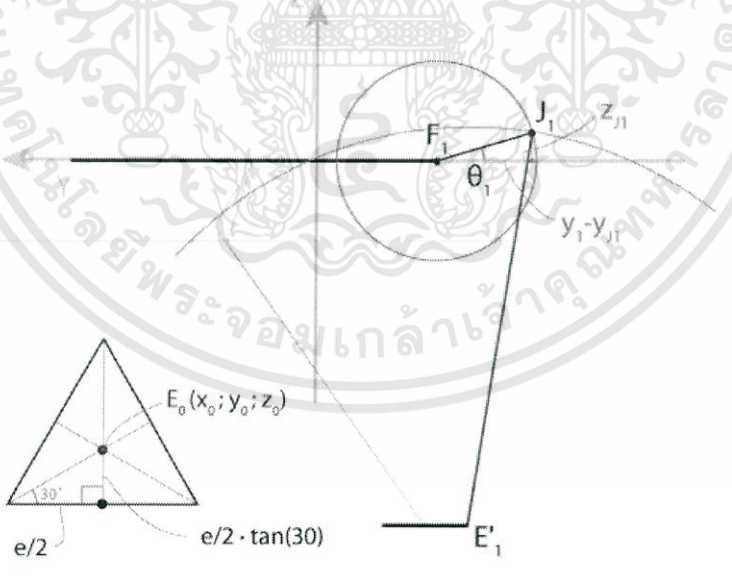
รูปที่ 2.8 การกำหนดระนาบ [17]

จากรูปที่ 2.8 เป็นโครงสร้างของหุ่นยนต์แบบขนาน ประกอบด้วยมอเตอร์แบบสเต็ปปิ้ง 3 ตัว ฐานบนยึดติดอยู่กับที่ (Fixed Base) คือ f ฐานล่างเคลื่อนที่อิสระ (End Effector) คือ e แขนบนเชื่อมต่อกับมอเตอร์แบบสเต็ปปิ้ง (Upper Arm) คือ r_f แขนล่างเชื่อมกับแขนบนกับฐานล่าง (Lower Arm) คือ r_e



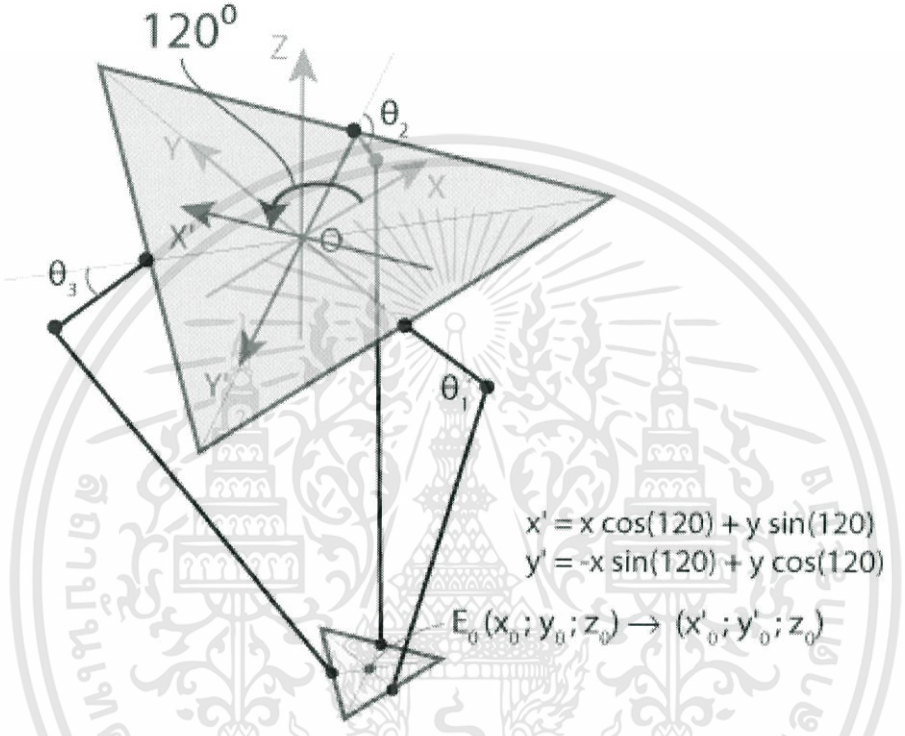
รูปที่ 2.9 ทรงกลมที่มีจุด E1 เป็นจุดศูนย์กลาง [17]

จากรูปที่ 2.9 เนื่องจากข้อต่อ F_1 ขยับได้เพียงระนาบ YZ จึงทำการสร้างวงกลมที่มีจุด F1เป็นจุดศูนย์กลาง มีรัศมีเท่ากับ r_f ส่วนข้อต่อ J_1 และ E_1 เป็นแบบ Universal Joint สามารถขยับได้รอบทิศทาง ซึ่งสามารถสร้างทรงกลมที่มีจุด E_1เป็นจุดศูนย์กลาง มีรัศมีเท่ากับ r_e



รูปที่ 2.10 จุดตัดของทรงกลมและระนาบ YZ [17]

จากรูปที่ 2.10 จุดตัดของทรงกลมและระนาบ YZ จะได้เป็นรูปวงกลม มีจุดศูนย์กลางอยู่ที่ E'_1 มีรัศมี E'_1J_1 (จุด E'_1คือโปรเจ็คชันของจุด E_1 บนระนาบ YZ) จุด J_1 สามารถหาได้จากจุดตัดของวงกลม 2 วง คือวงที่มีจุดศูนย์กลางที่ F1และ E'_1 เลือกจุดตัดเพียงจุดเดียวคือจุดที่มีพิกัด Y น้อยๆ เมื่อรู้พิกัดของ J_1 แล้วจึงสามารถหาค่ามุมของแขนที่1ได้ ส่วนการหามุมที่ 2 และ 3 ใช้วิธีการเหมือนการหามุมที่ 1 แต่ต้องทำการหาพิกัด E_0 ใหม่ โดยการหมุนระนาบ XY รอบแกน Z 120 องศา ทวนเข็มนาฬิกาและตามเข็มนาฬิกา ดังแสดงในรูปที่ 2.11



รูปที่ 2.11 การหมุนระนาบ XY รอบแกน Z 120 องศา [17]

2.4.1 ขั้นตอนการคำนวณหาจำนวนสเต็ปของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor)

1. กำหนดจุด E_0 ให้มีพิกัด $= (x_0, y_0, z_0)$

2. หาระยะ E_0E_1 $= \frac{e}{2}\tan 30°$

3. หาพิกัด E_1 $= (x_0, y_0 - \frac{e}{2}\tan 30°, z_0)$

4. หาพิกัด E'_1 $= (0, y_0 - \frac{e}{2}\tan 30°, z_0)$

5. หาระยะ E'_1J_1 $= \sqrt{r_e^2 + x_0^2}$

6. หาพิกัด F_1 $= (0, -\frac{f}{2}\tan 30°, 0)$

7. จากสูตรตรีโกณมิติ (สามเหลี่ยมมุมฉาก $a^2+b^2 = c^2$)

จะได้สมการ $$(Y_{J_1} - Y_{F_1})^2 + (Z_{J_1} - Z_{F_1})^2 = r_f^2 \quad (2.1)$$

$$(Y_{J_1} - Y_{E'_1})^2 + (Z_{J_1} - Z_{E'_1})^2 = r_e^2 - x_0^2 \quad (2.2)$$

จากนั้นจะได้ค่าพิกัด J_1 เพื่อนำไปคำนวณหามุมระหว่างแขนบนกับระนาบ XY โดยสมการ

$$\theta_1 = \tan^{-1}\left(\frac{Z_{J_1}}{Y_{F_1} - Y_{J_1}}\right) \quad (2.3)$$

8. หาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์ โดยการนำค่ามุมที่ได้คูณด้วยจำนวนทดรอบของมอเตอร์แบบสเต็ปปิ้ง จากนั้นทำการแปลงเป็นจำนวนสเต็ปโดยสมการ

$$\text{จำนวนสเต็ปของมอเตอร์} = \frac{\text{มุมระหว่างแขนบนกับระนาบXY} \times \text{จำนวนทดรอบของมอเตอร์}}{\text{มุมองศาต่อสเต็ปของมอเตอร์}} \quad (2.4)$$

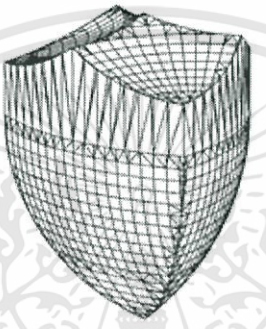
9. การหามุมระหว่างแขนบนกับระนาบ XY และจำนวนสเต็ปของมอเตอร์ที่ 2 และ 3 ใช้วิธีการเหมือนการหามุมที่ 1 แต่ต้องทำการหาพิกัด E_0 ใหม่ โดยการหมุนระนาบ XY รอบแกน Z 120 องศา ทวนเข็มนาฬิกาและตามเข็มนาฬิกา โดยสมการ

$$x'_0 = x_0 \cos 120° + y_0 \sin 120° \quad (2.5)$$

$$y'_0 = -x_0 \sin 120° + y_0 \cos 120° \quad (2.6)$$

2.5 การวิเคราะห์พื้นที่การทำงาน (Workspace Analysis) [17]

การวิเคราะห์พื้นที่การทำงานเป็นการศึกษาขอบเขตการเคลื่อนที่ของหุ่นยนต์ พื้นที่ทำงานของหุ่นยนต์ต้องมีการวิเคราะห์พื้นที่และรูปร่างของพื้นที่การทำงาน การคำนวณพื้นที่และ ขอบเขตเป็นสิ่งสำคัญ เนื่องจากส่งผลต่อการออกแบบสามมิติและการจัดวางตำแหน่งที่มีผลให้เกิดการทำงานอย่างแม่นยำในพื้นที่การทำงานที่ถูกจำกัดด้วยเงื่อนไขหลายประการ เช่น พื้นที่การทำงานขึ้นอยู่โครงสร้างของหุ่นยนต์และขอบเขตการเคลื่อนที่ของทั้ง 3 แกน ข้อจำกัดสำคัญคือขอบเขตที่เป็นไปได้ตามทฤษฎี Inverse kinematics นอกจากนี้ยังมีพื้นที่ที่ถูกจำกัดจากการเข้าถึงของ Drives และ Joints โดยหุ่นยนต์แบบขนานนั้นตระหนักถึงบริเวณพื้นที่การทำงานที่กว้างและครอบคลุม ดังนั้นการออกแบบพื้นที่การทำงานจึงเป็นสิ่งสำคัญของการวิเคราะห์ประสิทธิภาพการทำงาน ซึ่งพื้นที่การทำงานของหุ่นยนต์แบบขนานจะมีลักษณะคล้ายรูปแห ดังตัวอย่างรูปที่ 2.12

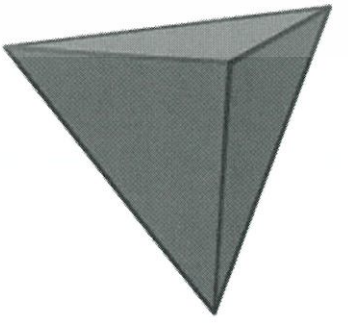


รูปที่ 2.12 พื้นที่การทำงานของหุ่นยนต์แบบขนานซึ่งมีลักษณะคล้ายรูปแห [17]

หลักในการคำนวณหาพื้นที่การทำงาน สามารถหาพื้นที่ได้โดยการแบ่งพื้นที่การทำงานทั้งหมดออกเป็นส่วนเล็กๆ ซึ่งมีรูปทรงเป็นพีระมิด ดังแสดงในรูปที่ 2.13 การหาพิกัดมุมของรูปพีระมิดสามารถหาได้จากการเคลื่อนที่แบบผันตรง (Forward Kinematics) ซึ่งเมื่อรู้ตำแหน่งมุมจะสามารถหาพื้นที่การทำงานและนำค่าพื้นที่การทำงานทั้งหมดที่คำนวณได้รวมกัน

ปริมาตรของรูปพีระมิดสามารถคำนวณได้จากสูตร $$V = \frac{1}{3} A_0 h \quad (2.7)$$

โดย V คือ ปริมาตรของรูปพีระมิด A_0 คือพื้นที่ฐาน และ h คือความสูงจากฐานถึงยอด

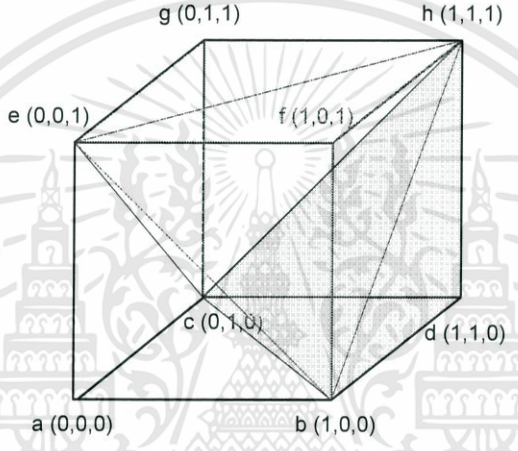


รูปที่ 2.13 พีระมิด [17]

การวิเคราะห์พื้นที่การทำงานของหุ่นยนต์แบบขนานโดยวิธีการคำนวณเวกเตอร์ เมื่อกำหนดให้ a,b,c,d คือ พิกัดตำแหน่งของมุมพีระมิด ซึ่ง a = (a1, a2, a3), b = (b1, b2, b3), c = (c1, c2, c3) และ d = (d1, d2, d3) จากนั้นทำการหาปริมาตรของรูปพีระมิดโดยสมการ

$$V = \frac{\left|(a-d)\cdot\left((b-d)\times(c-d)\right)\right|}{6} \tag{2.8}$$

ตัวอย่างการคำนวณหาพื้นที่การทำงานของหุ่นยนต์แบบขนาน โดยอ้างอิงจากลูกบาศก์ที่ประกอบด้วยรูปพีระมิด 5 รูป ดังแสดงในรูปที่ 2.14



รูปที่ 2.14 ลูกบาศก์ที่ประกอบด้วยรูปพีระมิด 5 รูป [17]

ปริมาตรทั้งหมดของรูปพีระมิด สามารถหาได้ ดังนี้

$$V = V(a,b,c,e) + V(b,c,d,h) + V(e,b,f,h) + V(c,e,h,g) + V(c,b,e,h)$$

จะได้

$$\begin{aligned} V = {} & V[(0,0,0),(1,0,0),(0,1,0),(0,0,1)] + V[(1,0,0),(0,1,0),(1,1,0),(1,1,1)] \\ & + V[(0,0,1),(1,0,0),(1,0,1),(1,1,1)] + V[(0,1,0),(0,0,1),(1,1,1),(0,1,1)] \\ & + V[(0,1,0),(1,0,0),(0,0,1),(1,1,1)] \end{aligned}$$

เมื่อทำการแสดงตำแหน่งพิกัดแต่ละจุดของลูกบาศก์ด้วย เวคเตอร์หนึ่งหน่วย (Unit Vector) i,j,k ดังนั้นปริมาตรทั้งหมดของรูปพีระมิดสามารถหาได้ ดังนี้

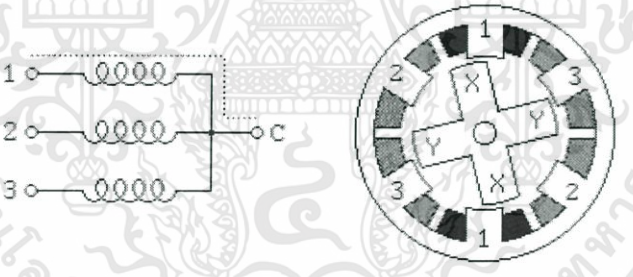
$$\begin{aligned} V = {} & V[(i,j,k),(i+1,j,k),(i,j+1,k),(i,j,k+1)] \\ & + V[(i+1,j,k),(i,j+1,k),(i+1,j+1,k),(i+1,j+1,k+1)] \\ & + V[(i,j,k+1),(i+1,j,k),(i+1,j,k+1),(i+1,j+1,k+1)] \\ & + V[(i,j+1,k),(i,j,k+1),(i+1,j+1,k+1),(i,j+1,k+1)] \\ & + V[(i,j+1,k),(i+1,j,k),(i,j,k+1),(i+1,j+1,k+1)] \\ = {} & 0.2+0.2+0.2+0.2+0.2 \\ = {} & 1 \quad \text{หน่วย}^3 \end{aligned}$$

2.6 มอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) [10]

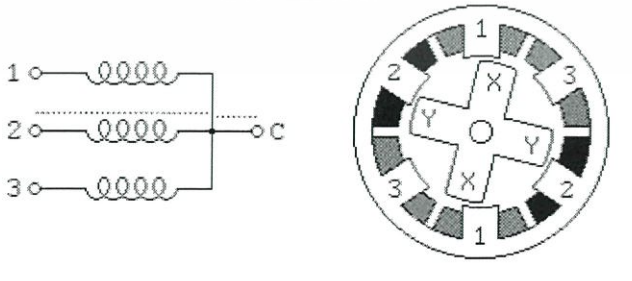
มอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) คือมอเตอร์ที่ขับเครื่องด้วยพัลส์ (Pulse) ลักษณะการขับเคลื่อน จะหมุนรอบแกนได้ 360 องศา มีลักษณะไม่ต่อเนื่อง แต่มีลักษณะเป็นสเต็ป โดยแต่ละสเต็ปจะขับเคลื่อนได้ 1, 1.5, 1.8 หรือ 2 องศา แล้วแต่ละโครงสร้างของมอเตอร์ ลักษณะของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) ภายนอกจะประกอบไปด้วยสายไฟที่จะต้องป้อนสัญญาณพัลส์ (Pulse) เข้าไปควบคุมมอเตอร์ ถ้ามอเตอร์แบบสเต็ปปิ้ง มีสายไฟ 3 สายจะมีสายหนึ่งที่ต่อเป็นสาย Ground ร่วมของทั้ง 3 สายนั้น ลักษณะแบบนี้คือเป็นแบบ 3 เฟส โดยมอเตอร์แบบสเต็ปปิ้ง ทั่วๆ ไปก็อาจจะมีหลายแบบ เช่น แบบ 4 เฟส แบบ 5 เฟส

2.6.1 หลักการทำงานของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor)

การทำงานของมอเตอร์แบบสเต็ปปิ้งประกอบด้วยส่วนสำคัญ 2 ส่วนคือ โรเตอร์ (Rotor) ซึ่งเป็นส่วนที่หมุนหรือเคลื่อนที่ได้ และสเตเตอร์ (Stator) ซึ่งเป็นส่วนที่อยู่กับที่ไม่สามารถเคลื่อนที่ได้ โดยที่ส่วนที่เป็นสเตเตอร์ของสเต็ปปิ้งมอเตอร์นี้จะเป็นส่วนซึ่งมีขดลวดพันล้อมอยู่บนแกนเหล็ก เมื่อจ่ายไฟเข้าที่ขดลวดที่พันอยู่บนสเตเตอร์ขดที่ 1 ก่อน จะทำให้เกิดอำนาจแม่เหล็กขึ้นที่ตำแหน่ง 1 และจะผลักให้โรเตอร์เกิดการเคลื่อนที่ได้ การเคลื่อนที่นี้เรียกว่าการเคลื่อนที่ไป 1 สเต็ป ดังแสดงในรูปที่ 2.15 ซึ่งการที่มอเตอร์จะเคลื่อนที่ไปเป็นมุมเท่าไหร่นั้นขึ้นอยู่กับคุณสมบัติของมอเตอร์ตัวนั้นว่าสามารถที่จะหมุนได้สเต็ปละกี่องศา และถ้าหยุดจ่ายไฟเข้าที่ขดลวด 1 และไปจ่ายไฟเข้าที่ขดลวด 2 แทน จะทำให้เกิดสนามแม่เหล็กขึ้นที่บริเวณ 2 บนสเตเตอร์และจะทำให้สนามแม่เหล็กนี้ผลักโรเตอร์ให้เคลื่อนที่ต่อไปได้ ดังแสดงในรูปที่ 2.16 และถ้าเราหยุดจ่ายกระแสเข้าที่ 2 และไปจ่ายเข้าที่ขด 3 จะทำให้สนามแม่เหล็กเกิดขึ้นที่บริเวณ 3 แทน และผลักให้โรเตอร์เคลื่อนที่ต่อไปได้



รูปที่ 2.15 การเคลื่อนที่ของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) เมื่อเฟส 1 ถูกกระตุ้นเฟสเข้ามาทำให้เกิดสนามแม่เหล็กที่สเตเตอร์ที่ขดลวด 1 [10]



รูปที่ 2.16 การเคลื่อนที่ของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) เมื่อเฟส 2 ถูกกระตุ้นเฟสเข้ามาทำให้เกิดสนามแม่เหล็กที่สเตเตอร์ที่ขดลวด 2 [10]

2.6.2 วิธีการขับมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) ให้หมุนโดยการกระตุ้นเฟส

ในการควบคุมมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) เพื่อที่จะให้ทำการหมุน มีวิธีการควบคุมกระแสไฟที่จ่ายให้กับขดลวดสเตเตอร์ (Stator) ในแต่ละเฟสของสเต็ปปิ้งมอเตอร์ อย่างเป็นลำดับที่แน่นอน โดยถ้าหากต้องการให้กระแสไหลในเฟสใดๆ จะทำให้สถานะของเฟสนั้นเป็นสถานะลอจิก "1" และในการกระตุ้นเฟสของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) มีอยู่ด้วยกัน 2 แบบ คือ

2.6.2.1 <u>การกระตุ้นเฟสแบบฟูลสเต็ปมอเตอร์ (Full Step Motor)</u> สามารถแบ่งการกระตุ้นเฟสออกได้เป็นอีก 2 วิธี ด้วยกันคือ การกระตุ้นเฟสแบบฟูลสเต็ป 1 เฟส (Single-Phase Driver) หรือแบบเวฟ แสดงดังตารางที่ 2.1(a) จะเป็นการป้อนกระแสไฟให้กับขดลวดของสเต็ปปิ้งมอเตอร์ทีละขด โดยจะป้อนกระแสเรียงตามลำดับกันไป ดังนั้นกระแสที่ไหลในขดลวดจะทำการไหลในทิศทางเดียวกันทุกขดลักษณะเช่นนี้จึงทำให้แรงขับของสเต็ปปิ้งมอเตอร์มีน้อย และการกระตุ้นเฟสแบบฟูลสเต็ป 2 เฟส (Two-Phase Driver) แสดงดังตารางที่ 2.1(b) เป็นการป้อนกระแสให้กับขดลวด 2 ขด ของสเต็ปปิ้งมอเตอร์พร้อมๆ กันไป และจะกระตุ้นเรียงถัดกับไปเช่นเดียวกับแบบหนึ่งเฟส ดังนั้นการกระตุ้นแบบนี้จึงต้องใช้กำลังไฟมากขึ้นและจะทำให้มีแรงบิดของมอเตอร์มากกว่าการกระตุ้นแบบ 1 เฟส

2.6.2.2 <u>การกระตุ้นเฟสแบบฮาล์ฟสเต็ปมอเตอร์ (Half Step Motor)</u> หรือ One-Two Phase Driver คือการกระตุ้นเฟสแบบฟูลสเต็ป 1 เฟส และ 2 เฟส เรียงลำดับกันไป แสดงดังตารางที่ 2.1(c) แรงบิดที่ได้จากการกระตุ้นเฟสแบบนี้จะมีเพิ่มมากขึ้น เพราะช่วงของสเต็ปมีระยะสั้นลง ในการกระตุ้นแบบนี้ เราจะต้องมีการกระตุ้นที่เฟสถึง 2 ครั้ง จึงจะได้ระยะของสเต็ปเท่ากับการกระตุ้นเพียงครั้งเดียวของแบบฟูลสเต็ป 2 แบบแรก ความละเอียดของการหมุนตำแหน่งองศาต่อสเต็ปเป็นสองเท่าของแบบแรก ความถูกต้องของตำแหน่งที่กำหนดจึงมีมากขึ้น

ตารางที่ 2.1(a) การกระตุ้นเฟสแบบฟูลสเต็ป 1 เฟส (Single-Phase Driver) [10]

สเต็ปที่	เฟสที่1	เฟสที่2	เฟสที่3	เฟสที่4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

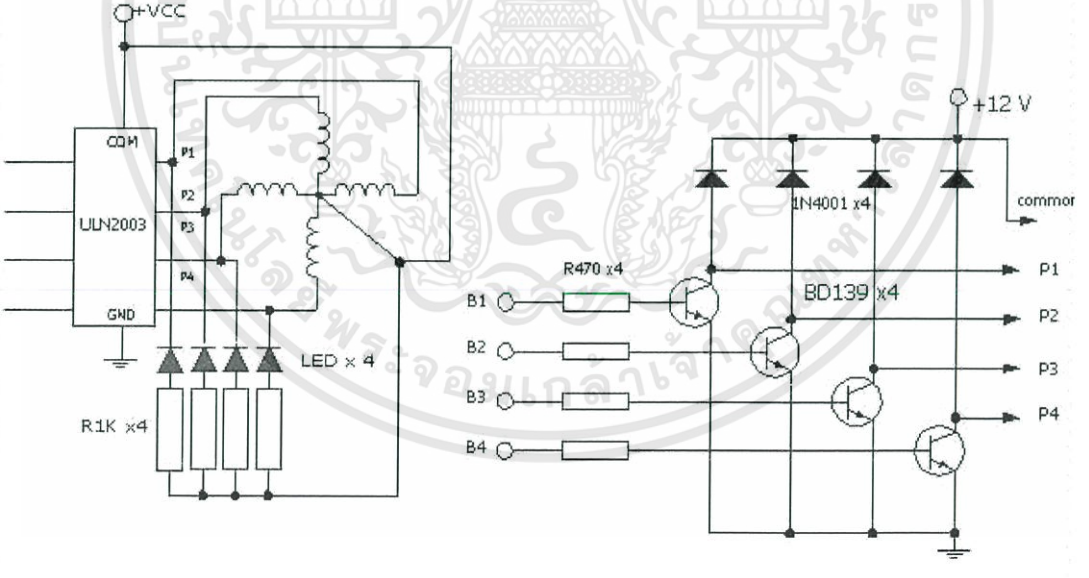
ตารางที่ 2.1(b) การกระตุ้นเฟสแบบฟูลสเต็ป 2 เฟส (Two-Phase Driver) [10]

สเต็ปที่	เฟสที่1	เฟสที่2	เฟสที่3	เฟสที่4
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

ตารางที่ 2.1(c) การกระตุ้นเฟสแบบฮาล์ฟสเต็ปมอเตอร์ (Half Step Motor) [10]

สเต็ปที่	เฟสที่1	เฟสที่2	เฟสที่3	เฟสที่4
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

วงจรที่ใช้ในการขับมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) โดยใช้ไอซีสำเร็จรูป และวงจรจากทรานซิสเตอร์ แสดงได้ในรูปที่ 2.17 โดยไอซีสำเร็จรูปเบอร์ ULN2003 จะมีคุณสมบัติเป็นไอซีไดร์ฟเวอร์กระแสสูงแบบคอลเล็คเตอร์เปิด สามารถเลือกแรงดันได้กว้าง 5-30 โวลท์ จ่ายกระแสได้สูงถึง 500 mA ต่อขา และมีไดโอดที่ป้องกันกระแสย้อนกลับอยู่ภายในไอซี ส่วนแอลอีดีที่ต่อในวงจรจะต่อไว้เพื่อแสดงการกระตุ้นแต่ละเฟสของแต่ละแบบ



รูปที่ 2.17 การต่อวงจรขับมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) โดยใช้ไอซีสำเร็จรูปและวงจรทรานซิสเตอร์ [10]

2.7 ไมโครคอนโทรลเลอร์ [2]

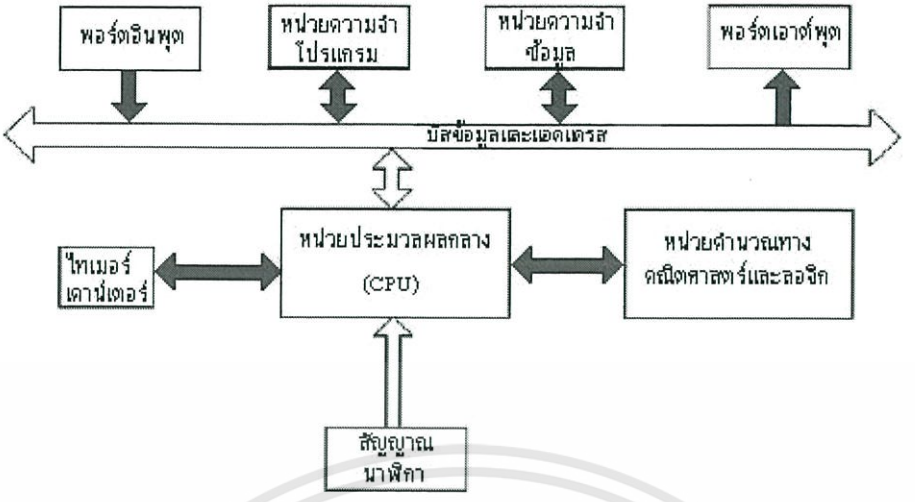
ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นอุปกรณ์ไอซี (IC: Integrated Circuit) ที่สามารถโปรแกรมการทำงานได้หลายครั้ง สามารถรับข้อมูลในรูปสัญญาณดิจิตอลเข้าไปทำการประมวลผล แล้วส่งผลลัพธ์ข้อมูลดิจิตอลออกมาเพื่อนำไปใช้งานตามที่ต้องการได้ ไมโครคอนโทรลเลอร์หรืออาจจะเรียกได้ว่าไมโครโพรเซสเซอร์ ชิปเดี่ยว (Single-Chip Microprocessor) เป็นไมโครโพรเซสเซอร์ชนิดหนึ่งเช่นเดียวกับหน่วยประมวลผลกลาง (CPU: Central Processing Unit) ที่ใช้ในคอมพิวเตอร์ แต่ได้รับการพัฒนาแยกออกมาภายหลังเพื่อนำไปใช้ในวงจรทางด้านงานควบคุม คือ แทนที่ในการใช้งานจะต้องต่อวงจรภายนอกต่างๆ เพิ่มเติม เช่น เดียวกับไมโครโพรเซสเซอร์ จะทำการรวมวงจรที่จำเป็น เช่น หน่วยความจำ ส่วนอินพุต/เอาท์พุต บางส่วนเข้าไปในตัวไอซีเดียวกัน และเพิ่มวงจรบางอย่างเข้าไปด้วย เพื่อให้มีความสามารถเหมาะกับการใช้ในงานควบคุม เช่น วงจรตั้งเวลา วงจรการสื่อสารอนุกรม เป็นต้น ดังนั้น ไมโครคอนโทรลเลอร์สามารถจะทำงานได้เสมือนกับเป็นคอมพิวเตอร์เล็กๆ เครื่องหนึ่ง ซึ่งในโครงงานนี้เลือกใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตที่มีอุปกรณ์สนับสนุนประกอบอยู่ภายในหลายอย่างได้แก่ หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บโปรแกรม ตัวตั้งเวลา ตัวนับ อุปกรณ์รับส่งข้อมูลแบบอนุกรม เนื่องจากโครงสร้างของไมโครคอนโทรลเลอร์มีอุปกรณ์สนับสนุนประกอบอยู่ภายในนี้เอง ทำให้การใช้งานง่ายขึ้นและมีประสิทธิภาพมากขึ้นโดยไม่ต้องมีการเชื่อมต่ออุปกรณ์ภายนอกเพิ่มเติมมากเหมือนกับตัวไมโครโปรเซสเซอร์ทั่วไป นอกจากนี้หากต้องการใช้งานไมโครคอนโทรลเลอร์ร่วมกับอุปกรณ์อื่นเพิ่มเติมเช่น ไอซี 8255 หรือหน่วยความจำภายนอก ยังสามารถนำมาเชื่อมต่อเพิ่มเติมเข้ากับไมโครคอนโทรลเลอร์ได้อีกด้วย

2.7.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

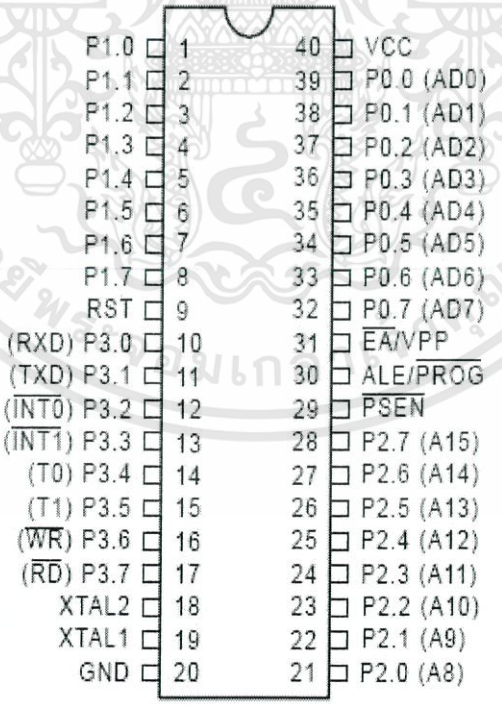
โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51 แสดงในรูปที่ 2.18 ประกอบด้วยอุปกรณ์ต่างๆ ดังนี้

- หน่วยประมวลผลกลางขนาด 8 บิต
- หน่วยประมวลผลสำหรับข้อมูลแบบบิต (Boolean Processor)
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์
- หน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์ แบบ อีพรอม (เบอร์ 8451)
- หน่วยความจำแบบ แรม ภายในจำนวน 128 ไบต์
- พอร์ตอินพุต เอาต์พุตแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ
- วงจรนับ จับเวลาขนาด 16 บิต จำนวนสองวงจร
- วงจรสื่อสารแบบอนุกรมแบบดูเพล็กเต็ม (Full Duplex)
- วงจรควบคุมการอินเตอร์รัปต์จากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนดลำดับ
- วงจรผลิตสัญญาณนาฬิกาภายในซึ่งโครงสร้างการทำงานทั้งหมดของไมโครคอนโทรลเลอร์จะอาศัยหลักการทำงานที่เกี่ยวข้องกันโดยอาศัยหลักการทำงานที่เป็นไปตามโครงสร้าง



รูปที่ 2.18 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51 [2]

โดยส่วนมากไมโครคอนโทรลเลอร์ตระกูลนี้มักจะมีรูปร่างของไอซีเป็นแบบขนาด 40 ขา ดังแสดงในรูปที่ 2.19 ซึ่งแต่ละขาสัญญาณจะมีหน้าที่ที่ระบุชัดเจนตามสัญลักษณ์ชื่อย่อที่กำกับในแต่ละขา แต่มีบางขาสัญญาณที่จะมีหน้าที่ได้มากกว่าหนึ่ง ซึ่งจะไม่สามารถใช้งานในเวลาเดียวกันได้



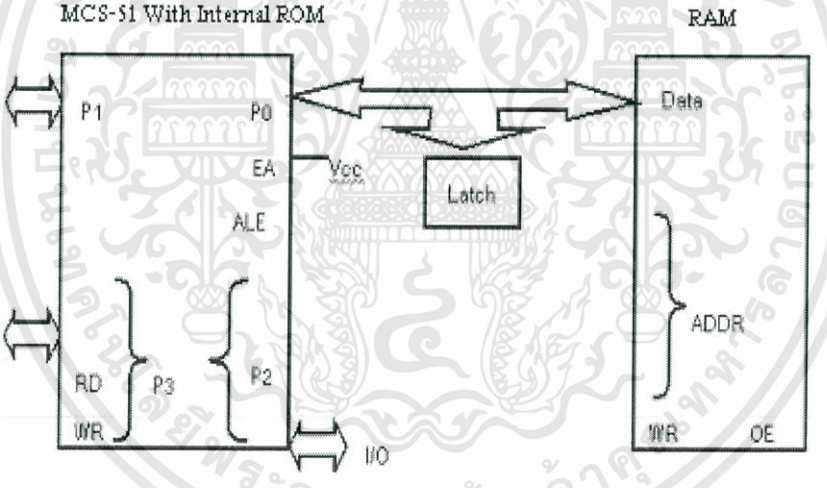
รูปที่ 2.19 รูปร่างและการจัดวางขาต่างๆ ของไมโครคอนโทรลเลอร์ MCS-51 [2]

2.7.2 โครงสร้างหน่วยความจำภายในไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 แยกการจัดการหน่วยความจำออกเป็น 2 ส่วนอย่างชัดเจนคือ หน่วยความจำโปรแกรม (Program Memory) และหน่วยความจำข้อมูล (Data Memory) หน่วยความจำทั้งสองนี้ มีหน้าที่ที่แตกต่างไปจากกัน และใช้วิธีการอ้างแอดเดรสสัญญาณการติดต่อแยกออกจากกัน

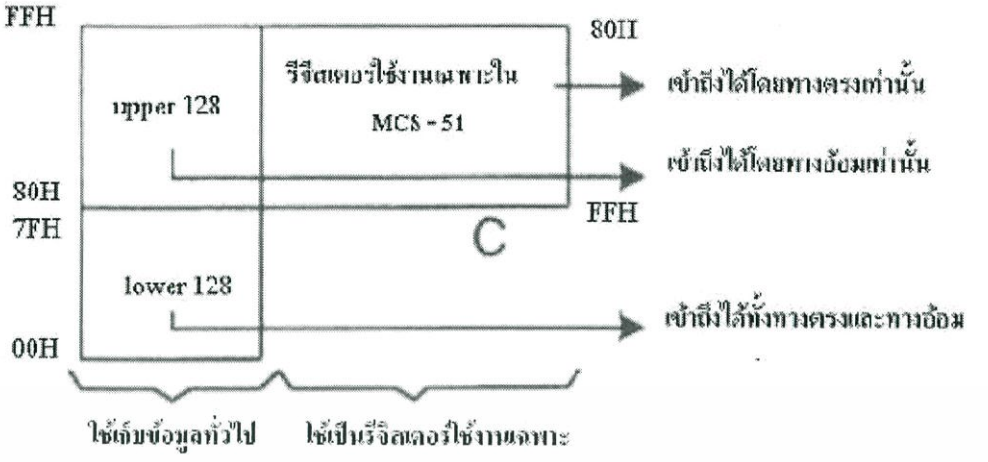
2.7.2.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 เป็นบริเวณหน่วยความจำสำหรับเก็บข้อมูลและคำสั่งใช้งานต่างๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบข้อมูลเหล่านี้ก็ยังคงอยู่ไม่สูญหายโครงสร้างของหน่วยความจำโปรแกรม มีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไมโครคอนโทรลเลอร์ MCS-51 ของหน่วยความจำประเภทต่างๆ เช่น หน่วยความจำแบบรอม (Read Only Memory) หรือ อีพรอมในไมโครคอนโทรลเลอร์ MCS-51 สามารถอ่านข้อมูลหน่วยความจำโปรแกรมนี้ได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทของหน่วยความจำโปรแกรมเป็น 2 ลักษณะ ตามตำแหน่งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน ซึ่งเป็นหน่วยความจำรวมหรืออีพรอม ที่อยู่ภายในตัวไอซีของไมโครคอนโทรลเลอร์ และหน่วยความจำโปรแกรมภายนอก ซึ่งเป็นการใช้ไอซีหน่วยความจำมาทำหน้าที่เป็นหน่วยความจำโปรแกรมของระบบ ดังแสดงในรูปที่ 2.20



รูปที่ 2.20 การใช้หน่วยความจำสำหรับเก็บโปรแกรม [2]

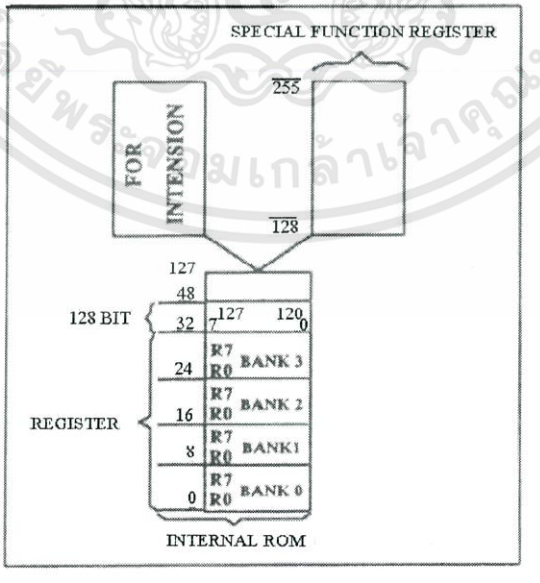
ไมโครคอนโทรลเลอร์เบอร์ต่างๆ ของตระกูล 8051 นี้ สามารถขยายให้ใช้งาน ในหน่วยความจำภายนอกได้ทั้งสิ้น โดยกรณีที่มีหน่วยความจำโปรแกรมภายในอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มีทั้งในหน่วยความจำโปรแกรมภายในและภายนอกนั้นจะต้องทำการควบคุมระดับลอจิกของสัญญาณในขณะนั้นด้วย ขนาดของหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์เบอร์ต่างๆ ภายในตระกูล 8051 จะแตกต่างกันออกไป เพื่อความเหมาะสมกับการนำไปใช้งานลักษณะต่างๆ ดังแสดงในรูปที่ 2.21



รูปที่ 2.21 หน่วยความจำสำหรับเก็บข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51 [2]

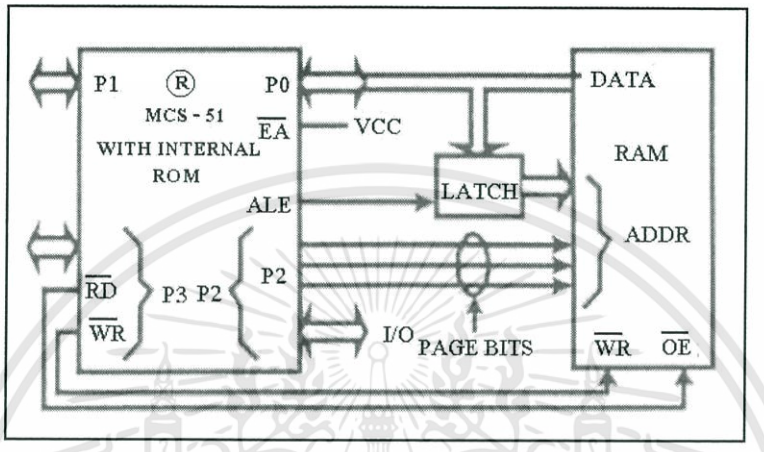
2.7.2.2 หน่วยความจำข้อมูล

หน่วยความจำข้อมูล (Data Memory) ซึ่งโดยพื้นฐานแล้วเป็นหน่วยความจำแรมสามารถเขียนหรืออ่านข้อมูลได้ (Read or Write Memory) ใช้สำหรับเก็บข้อมูลหรือตัวแปร ที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว ซึ่งโดยพื้นฐานแล้วหน่วยความจำข้อมูลจัดเป็นหน่วยความจำแรมแบบสแตติก ดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบก็จะมีผลทำให้ข้อมูลที่จัดเก็บไว้ภายในหน่วยความจำนี้สูญไป พื้นที่ของหน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 มีได้สูงสุดไม่เกิน 64 กิโลไบต์และแยกประเภทออกเป็นสองลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น ตามลักษณะของหน่วยความจำโปรแกรมภายในซึ่งก็เป็นแรมที่อยู่ภายในตัวไอซีในตระกูลของไมโครคอนโทรลเลอร์ และหน่วยความจำข้อมูลภายนอกซึ่งเป็นการใช้ไอซีหน่วยความจำแรมมาเพิ่มเติมเข้าไปในวงจรลักษณะเดียวกับการนำไอซีอีพรอมมาใช้งานเป็นหน่วยความจำโปรแกรม ดังแสดงในรูปที่ 2.22



รูปที่ 2.22 การจัดหน่วยความจำข้อมูล [2]

โดยที่หน่วยความจำสำหรับเก็บข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 นี้สามารถแบ่งออกเป็น 2 ส่วน คือ ในส่วนที่เป็นหน่วยความจำสำหรับเก็บข้อมูลภายใน และหน่วยความจำสำหรับเก็บข้อมูลภายนอก ไมโครคอนโทรลเลอร์ MCS-51 ทุกๆ เบอร์จะมีหน่วยความจำเก็บข้อมูลทั่วๆ ไปภายในไอซีอย่างน้อยคือ 128 ไบต์ ไปจนถึง 256 ไบต์ ทั้งนี้ขึ้นกับเบอร์ของไอซี หน่วยความจำสำหรับเก็บข้อมูลภายในไอซีในบริเวณ 128 ไบต์เรียกว่า LOWER 128 และในบริเวณ 128 ไบต์หลัง ที่มีเพิ่มในบางเบอร์มีชื่อเรียกว่า UPPER 128 แสดงดังรูปที่ 2.23



รูปที่ 2.23 การต่อกับหน่วยความจำข้อมูลภายนอกไอซี [2]

2.8 พอร์ต RS-232 [8]

2.8.1 คุณสมบัติของมาตรฐาน RS-232

RS-232 เป็นมาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรมที่มีคนนิยมใช้มากที่สุด กำหนดโดย EIA (Electronics Industry Association) หรือสมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ตั้งแต่ปี 1969 โดยมีจุดเริ่มต้นจากความต้องการที่จะกำหนดมาตรฐานการเชื่อมต่อระหว่างคอมพิวเตอร์กับโมเด็มในสมัยนั้น ตัวมาตรฐานจะกำหนดสิ่งที่เกี่ยวข้องกับการเชื่อมต่อนี้มีทั้งหมด 4 หัวข้อหลักๆ คือ

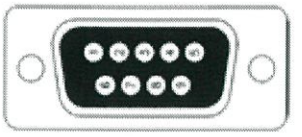
- คุณสมบัติทางไฟฟ้าของสัญญาณ
- คุณสมบัติทางกลของการเชื่อมต่อ
- หน้าที่การทำงานของวงจรสำหรับแลกเปลี่ยนข้อมูล
- มาตรฐานการเชื่อมต่อสำหรับระบบสื่อสารเฉพาะอย่าง

RS-232-C เป็นมาตรฐาน RS-232 ที่มีการปรับปรุงแก้ไขจากมาตรฐานเดิม ซึ่งเป็นที่นิยมมากกว่า RS-232-A หรือ RS-232-B นอกจากนี้ยังมีมาตรฐาน RS-232-D ที่ใหม่กว่า RS-232-C โดยที่มีการเพิ่มข้อกำหนดของคอนเน็คเตอร์แบบ DB เข้าไปด้วย เช่น DB-25 ซึ่งในขณะนั้นสิทธิบัตรของตัวคอนเน็คเตอร์แบบนี้ได้หมดอายุลง จึงสามารถรวมข้อกำหนดเข้าไว้ได้ ลักษณะโดยทั่วไปของการเชื่อมต่อข้อมูลแบบอนุกรมตามมาตรฐาน RS-232 คือเป็นการสื่อสารข้อมูลแบบจุดต่อจุด ซึ่งเดิมทีเป็นการสื่อสารข้อมูลระหว่างคอมพิวเตอร์กับโมเด็ม โดยที่ทั้งสองฝั่งจะเป็นอะไรก็ได้ การสื่อสารเป็นแบบสองทางพร้อมกัน (Full-Duplex) โดยอาจใช้สายสัญญาณอื่นร่วมเพื่อทำแฮนด์เชค (Hand-Shake) หรือไม่ก็ได้ มาตรฐาน RS-232 จำกัดความยาวสายไว้ที่ 50 ฟุต (ประมาณ 15 เมตร) สำหรับการส่งสัญญาณที่ความเร็ว 19,200 บิตต่อวินาที โดยที่ความยาวสายจะต้องสั้นลงถ้าต้องการสื่อสารที่ความเร็วสูงขึ้น และถ้ามีสัญญาณรบกวนมากๆ เช่น ในโรงงาน หรือบริเวณใกล้เครื่องจักรที่เป็นแบบมีการสวิตช์สัญญาณไฟฟ้าที่กระแสสูงๆ ก็จะทำให้ต้องมีการลดความเร็วในการส่งสัญญาณลงหรือใช้สายที่สั้นลง

2.8.2 การทำงานของ RS-232

มาตรฐาน RS-232 ใช้สัญญาณเพียงเส้นเดียวในการส่งสัญญาณ โดยสัญญาณจะส่งไปในทิศทางเดียวกัน ในกรณีที่อัตราเร็วในการส่งข้อมูลมีค่าเท่ากับ 20 กิโลบิทต่อวินาที ซึ่งค่านี้เป็นค่าสูงสุดที่ใช้ในการสื่อสารข้อมูล ซึ่งในปัจจุบันพัฒนาให้สามารถส่งข้อมูลได้มากขึ้น ระยะทางในการส่งข้อมูลไม่ควรเกิน 50 ฟุตตามข้อกำหนดในมาตรฐาน สำหรับการแทนแรงดันของระดับสัญญาณจะแทนระดับสัญญาณของลอจิก “0” ด้วยค่าแรงดัน +3 โวลต์ ถึง +12 โวลต์ ส่วนลอจิก “1” จะแทนระดับสัญญาณด้วยค่าแรงดันระหว่าง –3 โวลต์ ถึง –12 โวลต์

การเชื่อมต่อกับพอร์ตสื่อสารของคอมพิวเตอร์ส่วนบุคคลจะเลือกใช้พอร์ตสื่อสารแบบอนุกรม 9 ขา (DB-9) โดยแต่ละขามีสัญญาณที่แตกต่างกันออกไปดังตารางที่ 2.2 ซึ่งสามารถทำการรับส่งข้อมูลได้แบบอนุกรม โดยลักษณะของสัญญาณจะเป็นไปตามมาตรฐาน RS-232 โดยลักษณะของการเชื่อมต่อของพอร์ตสื่อสารสำหรับคอนเน็คเตอร์แบบ DB-9 แสดงในรูปที่ 2.24



รูปที่ 2.24 การจัดขาของคอนเน็คเตอร์พอร์ตอนุกรมแบบ DB-9 [8]

ตารางที่ 2.2 ตำแหน่งขาของพอร์ตอนุกรรมแบบ DB-9 [8]

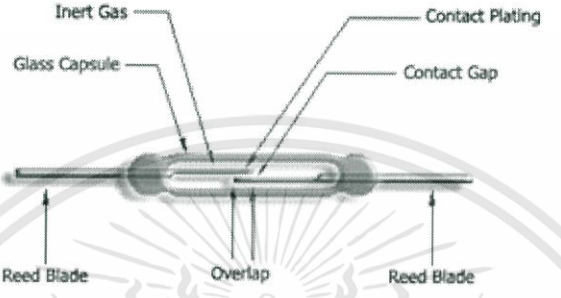
ตำแหน่งขา DB-9	สัญญาณ
1	Data Carrier Detect : DCD
2	Received Data : RxD
3	Transmitted Data : TxD
4	Data Terminal Ready : DTR
5	Signal Ground : GND
6	Data Set Ready : DSR
7	Request To Send : RTS
8	Clear To Send : CTS
9	Ring Indicator : RI

2.8.2.1 การทำงานของขาสัญญาณ DB9

1. TXD เป็นขาที่ใช้ส่งข้อมูล
2. RXD เป็นขาที่ใช้รับข้อมูล
3. DTR แสดงสภาวะพอร์ตว่าเปิดใช้งาน DSR ตรวจสอบว่าพอร์ต ที่ติดต่อด้วย เปิดอยู่หรือไม่
   - เมื่อเปิดพอร์ตอนุกรม ขา DTR จะ ON เพื่อให้อุปกรณ์ได้รับทราบว่าต้องการติดต่อด้วย
   - ในขณะเดียวกันตรวจสอบขา DSR ว่าอุปกรณ์พร้อมหรือไม่
4. RTS แสดงสภาวะพอร์ตว่าต้องการส่งข้อมูล CTS ตรวจสอบว่าพอร์ตที่ติดต่ออยู่ ต้องการส่งข้อมูลหรือไม่
   - เมื่อต้องการส่งข้อมูลขา RTS จะ ON และจะส่งข้อมูลออกที่ขา TXD เมื่อส่งเสร็จจะ OFF
   - ในขณะเดียวกันตรวจสอบขา CTS ว่าอุปกรณ์ต้องการส่งข้อมูลหรือไม่
5. GND ขา Ground

2.9 รีดสวิตช์ (Reed Switch) [15]

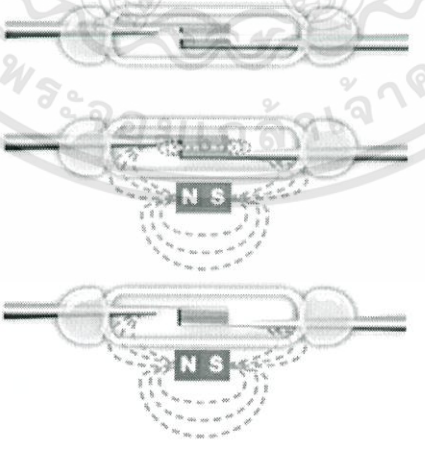
รีดสวิตช์ (Reed Switch) คือ แม็กเนติกเซนเซอร์ที่มีลักษณะเป็นแบบหน้าสัมผัส ซึ่งโดยปกติทั่วไปแล้ว จะเป็นหน้าสัมผัสแบบปกติเปิด (Normally Open : NO) สวิตช์นี้จะทำงานโดยอาศัยสนามแม่เหล็ก ซึ่งอาจจะเห็นแม่เหล็กถาวร หรือแม่เหล็กไฟฟ้าก็ได้ แผ่นหน้าสัมผัสจะทำมาจากสารที่มีผลต่อสนามแม่เหล็ก (Ferromagnetic) และติดตั้งอยู่ภายในกระเปาะแก้วเล็กๆ ที่มีการเติมก๊าซเฉื่อยเพื่อทำให้การตัดต่อการส่งกระแสไฟฟ้าได้เร็วยิ่งขึ้น ซึ่งมีองค์ประกอบดังรูปที่ 2.25



รูปที่ 2.25 องค์ประกอบรีดสวิตช์ (Reed Switch) [15]

2.9.1 การทำงานของรีดสวิตช์

รีดสวิตช์ คือสวิตช์ที่ควบคุมการทำงานโดยใช้แม่เหล็ก ซึ่งรีดสวิตช์มีความสะดวกในเรื่องของการติดตั้งที่ง่ายกว่าลิมิตสวิตช์ทั่วไป การทำงานเมื่อแม่เหล็กเคลื่อนที่เข้าใกล้กับตัวรีดสวิตช์ อำนาจแม่เหล็กที่จะไปดึงดูดให้หน้าคอนแทคของรีดสวิตช์ต่อกัน ซึ่งปกติหน้าคอนแทคจะเป็นหน้าคอนแทคปกติเปิด เมื่อแม่เหล็กเคลื่อนที่มาตรงกับตำแหน่งของรีดสวิตช์ รีดสวิตช์จะปิดวงจร ดังแสดงในรูปที่ 2.26



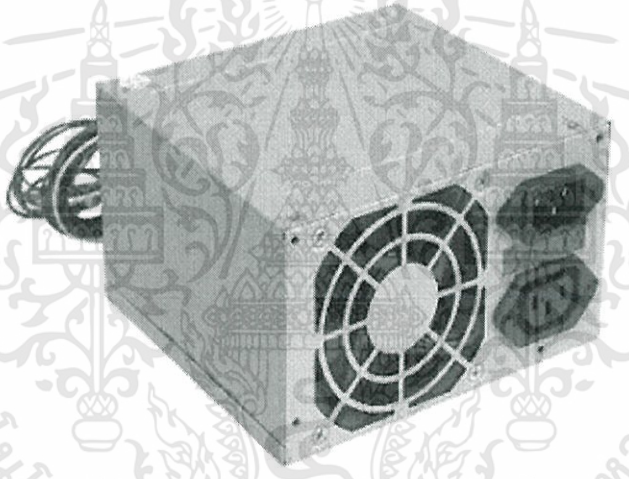
รูปที่ 2.26 การทำงานของรีดสวิตช์ [15]

2.9.2 ประเภทของรีดสวิตช์

ประเภทพื้นฐานคือแบบเอ ซึ่งจะเป็นแบบปกติเปิด ส่วนบีจะเป็นประเภทปกติปิด จะมีลักษณะคล้ายคลึงกับเอ แต่ทำงานตรงข้ามกับเอโดยใช้แม่เหล็กถาวรและแม่เหล็กที่มีขั้วตรงข้ามอีกอันสำหรับควบคุมการทำงานแบบซีเกิดจากการนำหน้าสัมผัสของแบบเอมารวมกับบีในคอยล์ (Operating Coil) เดียวกัน

2.10 เพาเวอร์ซัพพลาย (Power Supply) [1]

เป็นอุปกรณ์หลักที่คอยจ่ายไฟให้กับชิ้นส่วนและอุปกรณ์ต่างๆ ทั้งหมดภายในเครื่อง มีรูปร่างเป็นกล่องสี่เหลี่ยมติดตั้งอยู่ภายในตัวเคส สามารถถอดเปลี่ยนได้ ทำหน้าที่แปลงแรงดันไฟฟ้ากระแสสลับ (AC) ตามบ้านจาก 220 โวลต์ให้เหลือเพียงแรงดันไฟฟ้ากระแสตรง (DC) 3 ชุดคือ 3.3 และ 5 โวลต์ เพื่อจ่ายไฟให้กับวงจรชิ้นส่วนอุปกรณ์ต่างๆ และ 12 โวลต์ เพื่อจ่ายไฟให้กับมอเตอร์ของอุปกรณ์ดิสก์ไดร์ฟต่างๆ รวมถึงพัดลมระบายอากาศด้วย [1] แสดงในรูปที่ 2.27

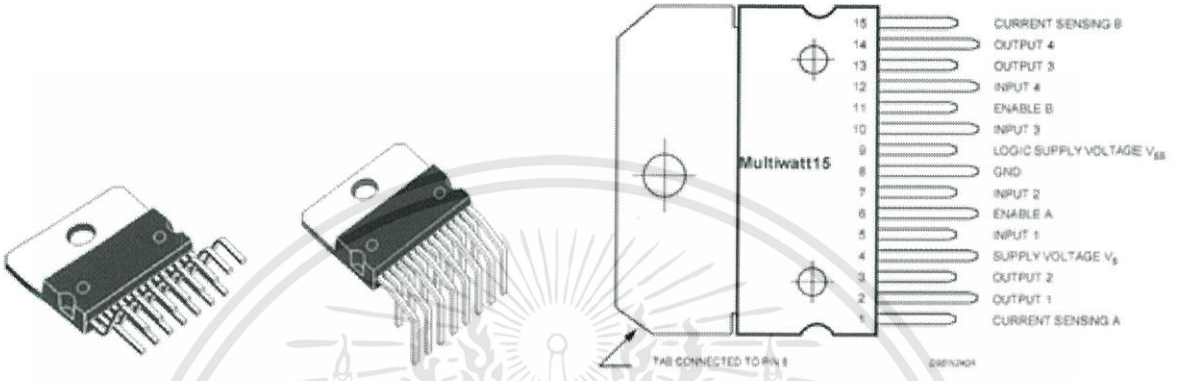


รูปที่ 2.27 ภาพตัวอย่างเพาเวอร์ซัพพลาย (Power Supply) [1]

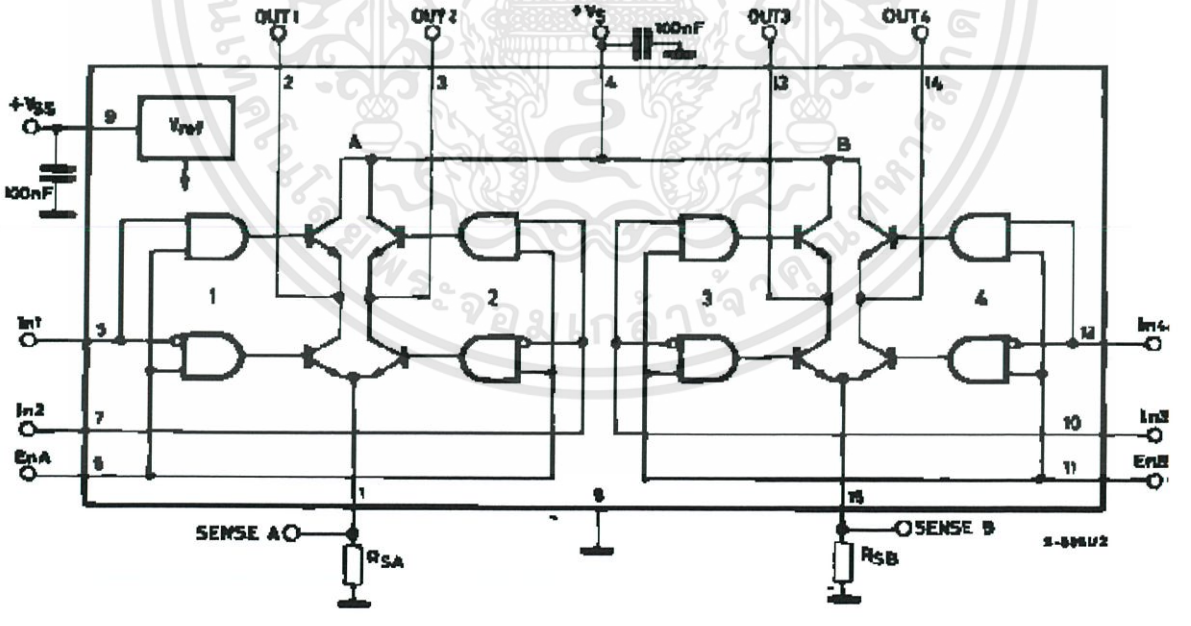
ปัจจุบันเพาเวอร์ซัพพลายที่จะนำมาใช้ควรมีกำลังไฟตั้งแต่ 400 วัตต์ขึ้นไป ทั้งนี้เพื่อให้เพียงพอกับความต้องการของชิ้นส่วนอุปกรณ์ต่างๆทั้งหมดที่อยู่ภายในเครื่องคอมพิวเตอร์ สำหรับแรงดันไฟฟ้ากระแสสลับ (AC) ตามบ้านในประเทศไทย โดยทั่วไปจะอยู่ที่ 200-250 VAC พร้อมกระแสไฟประมาณ 3.0-6.0 A และความถี่ที่ 50Hz ดังนั้นเพื่อให้ชิ้นส่วนอุปกรณ์คอมพิวเตอร์สามารถทำงานได้ เพาเวอร์ซัพพลายจะต้องแปลงแรงดันไฟ AC ให้เป็น DC แรงดันต่ำในระดับต่างๆ รวมถึงปริมาณความต้องการของกระแสไฟฟ้าที่จะต้องจ่ายให้กับชิ้นส่วนอุปกรณ์ต่างๆด้วย โดยระดับของแรงดันไฟ (DC Output) ที่ถูกจ่ายออกมาจากเพาเวอร์ซัพพลายแต่ละรุ่นและยี่ห้อจะใกล้เคียงกัน แต่ปริมาณสูงสุดของกระแสไฟ (Max Current Output) ที่ถูกจ่ายออกมานั้นอาจไม่เท่ากันแล้วแต่รุ่นและยี่ห้อ ซึ่งมีผลต่อการนำไปคำนวณค่าไฟโดยรวม (Total Power) ที่เพาเวอร์ซัพพลาย เพื่อจะสามารถจ่ายไฟให้กับอุปกรณ์ต่างๆ

2.11 วงจรขับมอเตอร์ [9]

การเปลี่ยนทิศทางการหมุนของดีซีมอเตอร์แบบแม่เหล็กถาวรทำได้โดยสลับขั้วของแหล่งจ่ายไฟฟ้าที่ต่อเข้ากับมอเตอร์ และในทางปฏิบัติจะใช้วงจรอิเล็กทรอนิกส์ที่เรียกว่า H Bridge เป็นตัวขับ การทำงานในส่วนของโครงงานนี้ได้เลือกใช้ชุดมอเตอร์แบบ Dual Full-Bridge Driver L298 ซึ่งมีแสดงดังรูปที่ 2.28

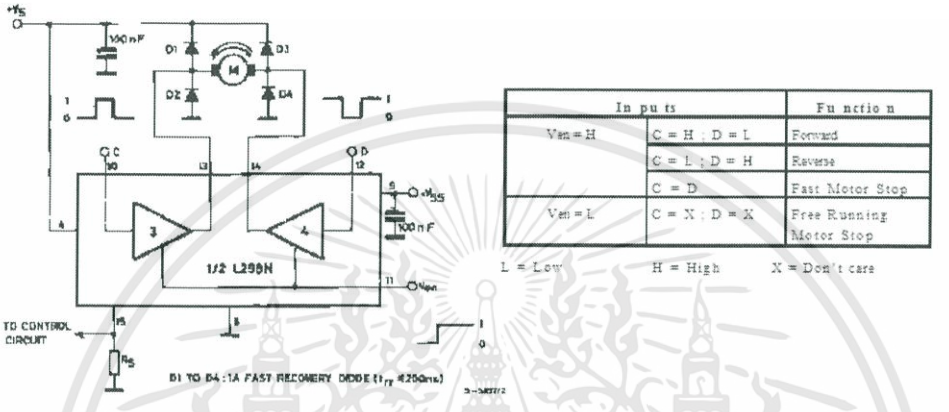


รูปที่ 2.28 การจัดขาของ IC L298 [9]



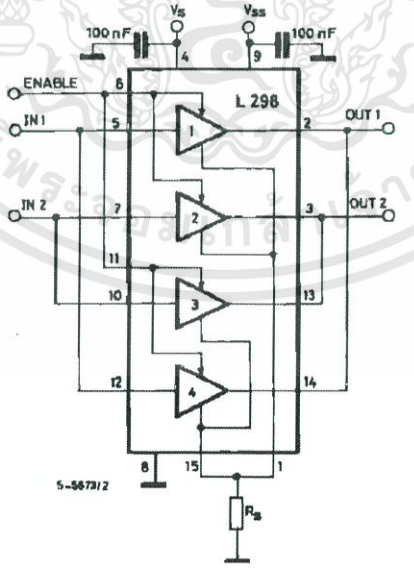
รูปที่ 2.29 วงจรภายในของ IC L298 [9]

จากรูปที่ 2.29 จะเห็นได้ว่าวงจรภายใน L298 สามารถขับโหลดได้ 2 ช่อง และสามารถรับสัญญาณควบคุมแบบ TTL (Transistor Transistor Logic) เพื่อที่จะควบคุมทิศทางการไหลของกระแส และมีขา Enable เพื่อป้อนสัญญาณ Pulse Width Modulation เพื่อควบคุมความเร็วขา Emitter ของ Transistor ทั้งสองข้างของแต่ละ Bridge จะต่อกับตัวต้านทานภายนอกเพื่อที่จะใช้ในการกำหนดค่ากระแสไฟฟ้าที่ไหลได้โดยหากเกินกว่าที่วงจรและโหลดต่ออยู่สามารถที่จะรับได้ก็อาจจะมีวงจรเพื่อที่ทำการ Disable การทำงานของโหลดได้ เนื่องจากโหลดที่ใช้เป็นตัวเหนี่ยวนำค่ากระแสไฟฟ้า ไม่สามารถเปลี่ยนเป็นศูนย์ได้ในทันที คล้ายกับตัวเก็บประจุที่ไม่สามารถเปลี่ยนค่าความต่างศักย์ได้อย่างทันที จึงต้องมีการต่อไดโอดอยู่ภายนอกเพื่อให้กระแสไฟฟ้าไหลได้ ดังแสดงในรูปที่ 2.30



รูปที่ 2.30 การต่อใช้งานของ IC L298 แบบ 1 ช่อง [9]

ในทางปฏิบัตินั้นมอเตอร์ต้องการกระแสสูงเพื่อไปขับโหลดที่หนักๆซึ่งสามารถต่อวงจรแต่ละ Channel ขนานกันเพื่อให้วงจรขับมอเตอร์ได้จ่ายกระแสไปยังมอเตอร์ได้มากขึ้นดังแสดงในรูปที่ 2.31



รูปที่ 2.31 การต่อใช้งานแบบขนาน [9]

2.12 คาร์บอนไฟเบอร์ (Carbon Fiber) [5]

ใช้คาร์บอนไฟเบอร์ทำชิ้นส่วนประกอบในส่วนแขนล่างของหุ่น คาร์บอนไฟเบอร์หรือเส้นใยคาร์บอนเป็นวัสดุทางวิศวกรรมอีกชนิดหนึ่งที่ได้รับความสนใจจากหลายอุตสาหกรรมตั้งแต่สิ่งของที่ใช้เทคโนโลยีระดับสูงอย่างการผลิตอาวุธ และเครื่องบิน จนถึงสิ่งของที่พบได้ทั่วไปอย่างอุปกรณ์กีฬา เพราะวัสดุชนิดนี้มีสมบัติเด่นหลายอย่าง แต่ที่โดดเด่นมากคือ การเป็นวัสดุมีน้ำหนักเบา แต่มีความแข็งแรงสูงมาก ดังแสดงในตารางที่ 2.3 นอกจากนี้คาร์บอนไฟเบอร์ยังมีสมบัติเป็นฉนวนกันความร้อนที่ดี ทนต่อการกัดกร่อนจากสารเคมีต่างๆ และมีสมบัตินำไฟฟ้าได้ คาร์บอนไฟเบอร์มีองค์ประกอบเป็นคาร์บอนเหมือนกับถ่านและแกรไฟต์ โดยทั่วไปเนื้อของวัสดุชนิดนี้จะประกอบด้วยคาร์บอนอย่างน้อยร้อยละ 90

ตารางที่ 2.3 เปรียบเทียบสมบัติเชิงกลของคาร์บอนไฟเบอร์กับเหล็กกล้า [5]

วัสดุ	ความหนาแน่น (g/cm)	ความต้านทานแรงดึง (Tensile Strength (GPa))	ค่ามอดูลัสแรงดึง (Tensile Modulus (GPa))	ความแข็งแรงเฉพาะ (Specific Strength (GPa))
คาร์บอนไฟเบอร์	1.75	3.5	230.0	2.00
เหล็กกล้า	7.87	1.3	210.0	0.17

2.13 อลูมิเนียม (Aluminum) [11]

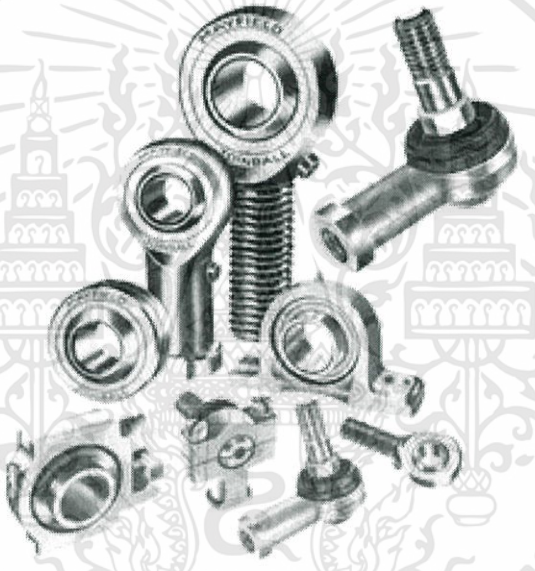
ใช้อลูมิเนียมทำชิ้นส่วนประกอบในส่วนแขนบนของหุ่นยนต์ โดยมีคุณสมบัติคือ อลูมิเนียมเป็นโลหะที่มีสีเงิน น้ำหนักเบาแต่มีความแข็งแรง นอกจากนี้ยังมีความยืนหยุ่นทำให้สามารถนำไปรีดเป็นแผ่นบางๆได้ อลูมิเนียมเป็นธาตุที่มีปริมาณมากเป็นอันดับที่ 3 ที่อยู่ในเปลือกโลก โดยมีประมาณ 8 % โดยน้ำหนัก ส่วน 2 อันดับแรกคือ ซิลิกอนและ ออกซิเจน อลูมิเนียมถูกนำไปใช้ได้มากมาย ทั้งในครัวเรือนและในอุตสาหกรรม และเป็นโลหะที่ใกล้ชิดกับผู้คนทั่วไป ประมาณ 85 % ของออกไซด์ที่ขุดมาทั้งโลกจะถูกนำไปผลิตอลูมินาสำหรับการสกัดเป็นโลหะอลูมิเนียมอีก 10% จะนำไปผลิตอลูมิเนียมสาหรับใช้ในผลิตภัณฑ์ที่เกี่ยวกับเคมี การขัด และอิฐทนไฟ ที่เหลืออีก 5% จะนำไปทำวัสดุที่ใช้ในการขัด วัสดุทนไฟ และสารประกอบอลูมิเนียม สำหรับโลหะอลูมิเนียมจะถูกนำไปใช้ในวงการ การขนส่ง บรรจุภัณฑ์ เช่น กระป๋องเครื่องดื่ม การก่อสร้างอาคาร เครื่องใช้ไฟฟ้า เป็นต้น คุณสมบัติที่เป็นข้อดีของโลหะอลูมิเนียมก็คือ มีน้ำหนักเบา มีความแข็งแรง มีความต้านทานการกัดกร่อนได้ดี และไม่เป็นพิษต่อสิ่งมีชีวิตทั้งพืชและสัตว์

2.14 อลูมิเนียมโปรไฟล์ (Aluminum Profile) [11]

ใช้ประกอบส่วนโครงของหุ่นยนต์แบบขนาน โครงสร้างอลูมิเนียมโปรไฟล์ เป็นวัสดุที่มีน้ำหนักเบา ไม่เป็นสนิม มีความสามารถในการรับแรงกดได้สูง สามารถถอดประกอบ และต่อเติมภายหลังได้สะดวก เหมาะสำหรับงานโครงสร้างในอุตสาหกรรมต่าง ๆ เช่น โครงสร้างเครื่องจักร สายการผลิต โครงห้องคลีนรูม โต๊ะทำงาน ชั้นวางของในโรงงานอุตสาหกรรม เป็นต้น

2.15 ข้อต่อแบบบอลจ้อย (Ball Joint)

ใช้สำหรับเป็นข้อต่อชิ้นส่วนที่เคลื่อนไหวต่าง ๆ สามารถเคลื่อนไหว เปลี่ยนมุมได้ บอลจ้อยเป็นชุดกลไกที่ใช้สำหรับการเชื่อมต่อของ 2 ส่วนที่ตั้งฉากกัน เป็นการช่วยให้การส่งกำลังสลับกับการเคลื่อนไหวเชิงมุมได้ ซึ่งบอลจ้อย มีหลากหลายแบบดังแสดงในรูปที่ 2.32



รูปที่ 2.32 ข้อต่อบอลจ้อย (Ball Joint) แบบต่างๆ [3]

2.16 นอตและสกรู

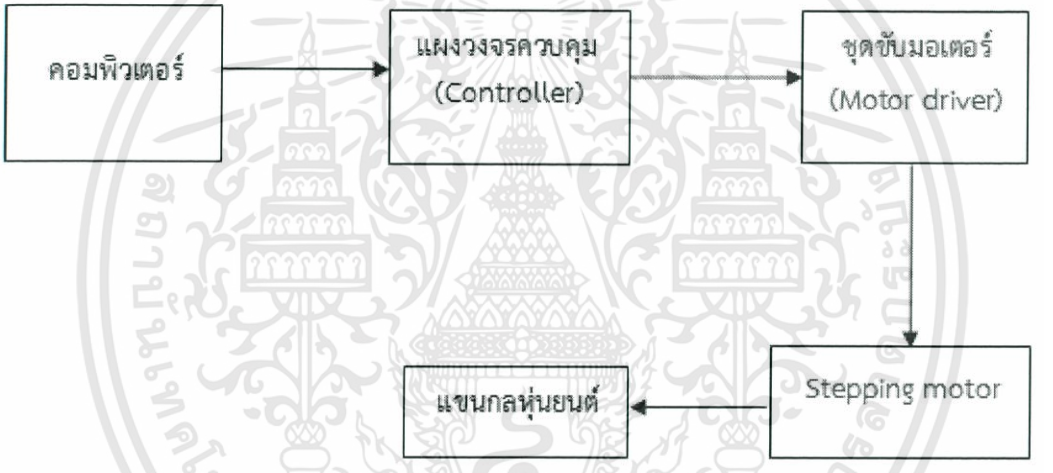
มีการใช้นอตในการยึดระหว่างชิ้นงาน โดยในการออกแบบส่วนใหญ่ใช้นอตขนาด M3 และขนาดนอตต่างๆ แสดงในตารางที่ 2.3

ตารางที่ 2.4 ขนาดของนอตในหน่วย mm [3]

Thread Size	dk	k	f	L	
				Type A	Type B
M3	5.6	1.65	0.75	6-22	25-30
M3.5	6.5	1.93	0.90	6-22	25-35
M4	7.5	2.20	1.00	8-25	28-40
M5	9.2	2.50	1.25	8-28	30-50
M6	11.0	3.00	1.50	10-35	40-60
M8	14.5	4.00	2.00	14-40	45-80
M10	18.0	5.00	2.50	18-45	50-100

บทที่ 3
วิธีการดำเนินงาน

จากการศึกษาทฤษฎีที่เกี่ยวข้องและหลักการทำงานของอุปกรณ์ต่างๆ ของหุ่นยนต์ ทางคณะผู้จัดทำจึงได้ออกแบบและพัฒนาหุ่นยนต์โดยมุ่งความสนใจไปที่หุ่นยนต์แบบขนานชนิด 3 แกนซึ่งมีโครงสร้างดังรูปที่ 2.1 เนื่องจากเป็นหุ่นยนต์ที่มีโครงสร้างไม่ซับซ้อน การออกแบบเริ่มจากการเขียนกระบวนการควบคุมการทำงานของหุ่นยนต์แบบขนาน (Delta Robot) ซึ่งสามารถแสดงโดยใช้ Block Diagram ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 Block Diagram ของระบบควบคุมการทำงานของหุ่นยนต์แบบขนาน (Delta Robot)

แผนการดำเนินงานถูกแบ่งออกเป็น 3 ส่วนหลักๆ ดังนี้

- การออกแบบเชิงกล
- การคำนวณการเคลื่อนที่แบบผกผัน (Inverse Kinematics)
- การออกแบบโปรแกรมควบคุมการทำงาน

3.1 การออกแบบเชิงกล

3.1.1 การคำนวณหาพื้นที่การทำงาน

ในการคำนวณหาพื้นที่การทำงาน กำหนดให้หุ่นยนต์สามารถเคลื่อนที่ในแกนต่างๆ โดยในแกน Z ให้เคลื่อนที่ลงมาได้ไม่เกิน 600 mm และมีรัศมีของการเคลื่อนที่ไม่เกิน 200 mm

3.1.2 การเลือกข้อต่อ

เนื่องจากข้อต่อมีผลต่อการออกแบบความยาวของแขนหุ่นยนต์ จึงต้องมีการพิจารณาถึงคุณสมบัติต่างๆ ของข้อต่อแต่ละชนิดให้เหมาะสมกับลักษณะการทำงานของหุ่นยนต์แบบขนาน โดยเลือกใช้ข้อต่อแบบบอลจ้อย (Ball Joint) ใช้เชื่อมต่อระหว่างแขนบนกับแขนล่าง และแขนล่างกับฐานตัวยกจับ ซึ่งมีลักษณะการใช้งานสำหรับเป็นข้อต่อชิ้นส่วนที่เคลื่อนไหวต่าง ๆ สามารถเคลื่อนไหว เปลี่ยนมุมได้ บอลจ้อยเป็นชุดกลไกที่ใช้สำหรับการเชื่อมต่อของ 2 ส่วนที่ตั้งฉากกัน เป็นการช่วยให้การส่งกำลังสลับกับการเคลื่อนไหวเชิงมุมได้ โดยมีข้อดี คือ ติดตั้งและใช้งานง่าย น้ำหนักเบา แต่มีข้อจำกัดในเรื่องมุมของการเคลื่อนที่ที่ไม่เกิน 20 องศา ซึ่งทำให้มีผลในการออกแบบความยาวของแขนหุ่นยนต์ให้มีความเหมาะสม ในการออกแบบเลือกใช้ข้อต่อแบบ Ball Joint แบบ AL 4D มีความยาว 24.5 mm เส้นผ่านศูนย์กลางยาว 13 mm และมีความสูง 20 mm สามารถรับแรงดึงได้ 1370 N ซึ่งเพียงพอกับความต้องการ ดังแสดงในรูปที่ ผ8

3.1.3 การสร้างแบบจำลองหุ่นยนต์แบบขนาน

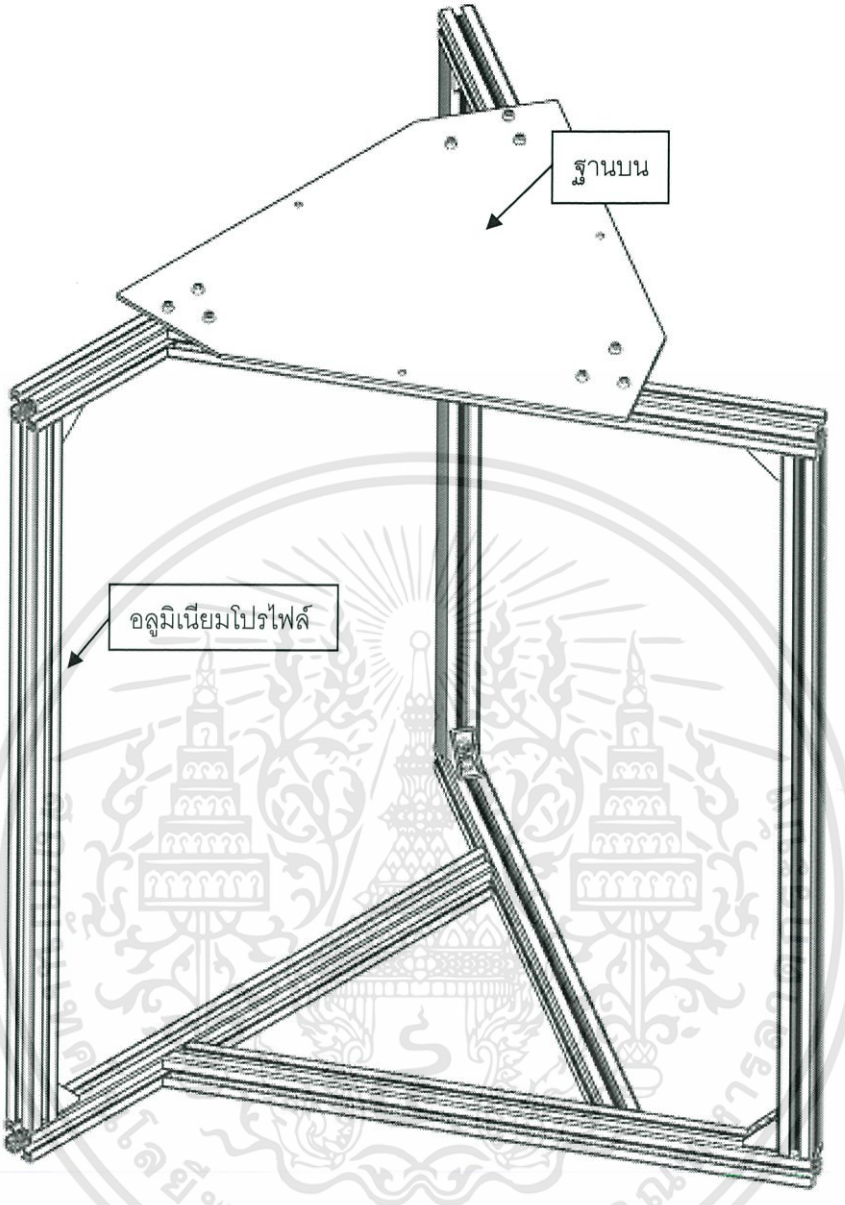
ก่อนการสร้างหุ่นยนต์แบบขนานคณะผู้จัดทำได้ศึกษาลักษณะของโครงสร้างต่างๆ รวมถึงลักษณะการเคลื่อนที่ของหุ่นยนต์ โดยการสร้างแบบจำลองของหุ่นยนต์เพื่อหาอัตราส่วนขนาดความยาวของแขนบนและแขนล่างที่เหมาะสมกับพื้นที่การทำงานที่กำหนด ซึ่งได้อัตราส่วนขนาดความยาวแขนที่เหมาะสม คือ 1:5 โดยแขนบนยาว 100 mm และแขนล่างยาว 505 mm

3.1.4 การออกแบบส่วนประกอบของหุ่นยนต์

ในส่วนของการออกแบบส่วนประกอบของหุ่นยนต์ ทำการออกแบบโครงสร้างเป็นอลูมิเนียมแขนกลขับเคลื่อนด้วยมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) จำนวน 3 ตัว โดยยึดติดกับโครงของหุ่นยนต์ด้านบนทำมุมกัน 120 องศา เพลาของมอเตอร์จะถูกต่อเข้ากับแขนบนของหุ่นยนต์จำนวนสามแขนและที่ปลายของแขนทั้งสามจะถูกต่อกับฐานตัวยกจับ ซึ่งจะใช้โปรแกรมโซลิดเวิร์ค (Solid Works) ช่วยในการออกแบบส่วนประกอบต่างๆ ของต้นแบบหุ่นยนต์แบบขนาน (Delta Robot) ซึ่งมีรายละเอียดในการออกแบบแต่ละส่วนดังนี้

3.1.4.1 <u>โครง Delta Robot</u>

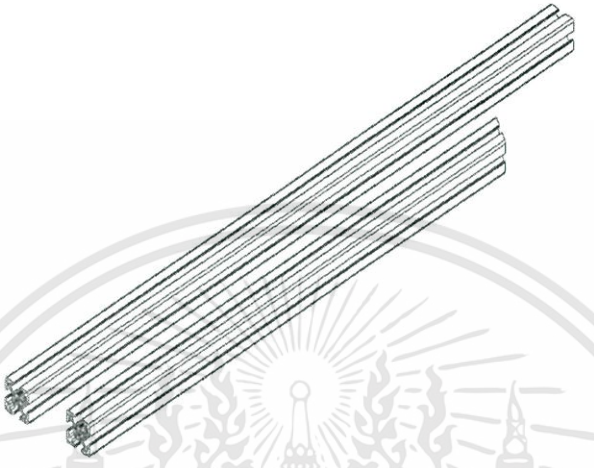
ทำจากอลูมิเนียมโปรไฟล์ ซึ่งเป็นวัสดุที่มีน้ำหนักเบา ไม่เป็นสนิม มีความสามารถในการรับแรงกดได้สูง สามารถถอดประกอบ และต่อเติมภายหลังได้สะดวก โดยมีการออกแบบให้มีโครงสร้างแบบสามเหลี่ยม เนื่องจากเป็นโครงสร้างสมมาตรที่แข็งแรงที่สุด ดังแสดงในรูปที่ 3.2 โดยมีส่วนประกอบดังนี้



รูปที่ 3.2 โครงหุ่นยนต์แบบขนาน (Delta Robot) ออกแบบโดยโปรแกรม Solid Works

- อลูมิเนียมโปรไฟล์

โดยทำการเลือกอลูมิเนียมโปรไฟล์ขนาดกว้าง 30x30 mm^2 ทั้งหมด 9 แท่ง ดังนี้ เสาจำนวน 3 แท่ง ความยาว 700 mm โครงด้านบนและล่างรวม 6 แท่ง ความยาว 550 mm และตัดปลายด้านหนึ่งให้เป็นมุม 60 องศา ดังแสดงในรูปที่ 3.3 และรูปที่ 3.4

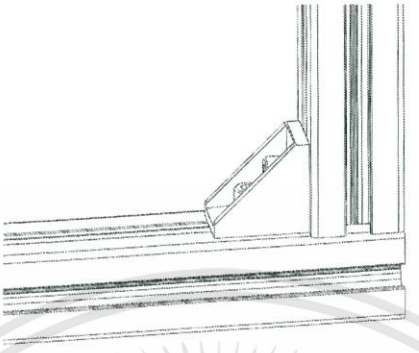


รูปที่ 3.3 อลูมิเนียมโปรไฟล์เสาและโครงบนล่าง

รูปที่ 3.4 การออกแบบอลูมิเนียมโปรไฟล์โครงด้านบนและด้านล่าง

- ชิ้นส่วนยึดระหว่างเสากับโครงอลูมิเนียมโปรไฟล์ด้านบนและด้านล่าง

เลือกใช้ขนาดกว้าง 30 mm เพื่อให้เหมาะสมกับขนาดอลูมิเนียมโปรไฟล์ ซึ่งใช้ชิ้นส่วนยึดระหว่างเสากับโครงอลูมิเนียมโปรไฟล์ด้านบนและด้านล่าง ดังแสดงในรูปที่ 3.5



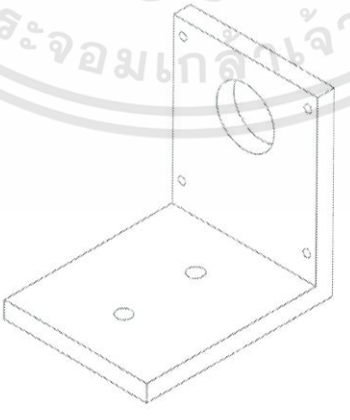
รูปที่ 3.5 การออกแบบอลูมิเนียมโปรไฟล์โครงด้านบนและด้านล่าง

- ฐานบน

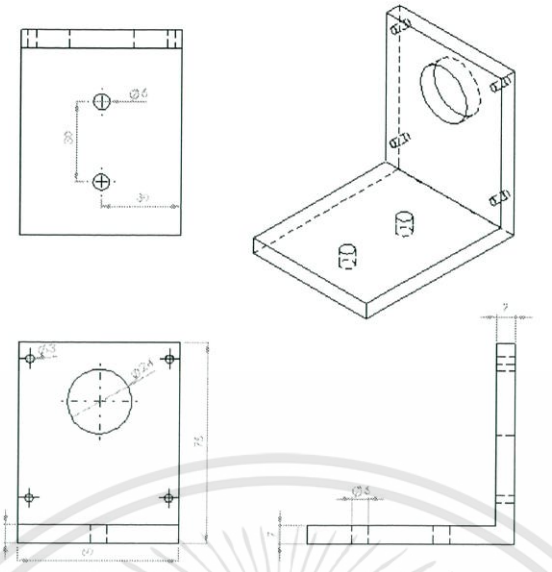
ทำจากแผ่นเหล็กบาง ซึ่งมีความหนา 1.7 mm ใช้ยึดโครงอลูมิเนียมโปรไฟล์ด้านบน และยึดกับส่วนที่ใช้ติดตั้งเซนเซอร์เพื่อป้องกันไม่ให้แขนขนกับโครงด้านบน

3.1.4.2 <u>ชิ้นส่วนเชื่อมต่อระหว่างมอเตอร์กับโครงอลูมิเนียมโปรไฟล์</u>

ใช้ยึดตัวมอเตอร์กับอลูมิเนียมโปรไฟล์เข้าด้วยกัน วัสดุทำจากอลูมิเนียม เนื่องจากอลูมิเนียมมีน้ำหนักเบา ระบายความร้อนได้ดี ไม่เกิดสนิม มีความเหนียว ไม่แตกร้าวหรือ หักได้ง่ายๆ โดยทำการยึดมอเตอร์ติดกับด้านล่างของโครงอลูมิเนียมโปรไฟล์ส่วนบน โดยฉากที่ยึดระหว่างมอเตอร์กับโครงอลูมิเนียมโปรไฟล์ มีขนาดความกว้าง 60 mm ความยาว 70 mm และความหนา 5 mm ดังแสดงในรูปที่ 3.6 และรูปที่ 3.7



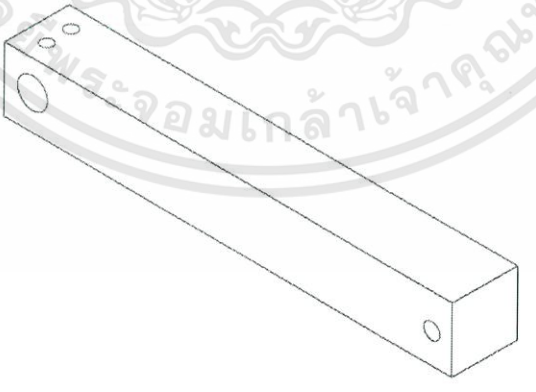
รูปที่ 3.6 ชิ้นส่วนเชื่อมต่อระหว่างมอเตอร์กับอลูมิเนียมโปรไฟล์



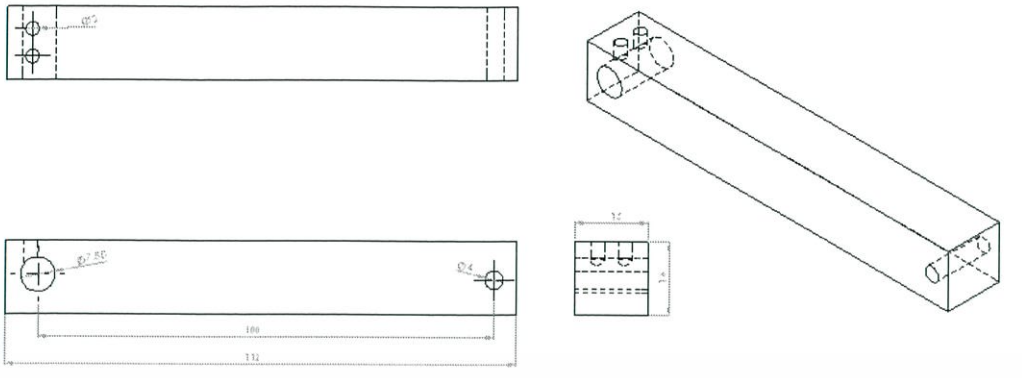
รูปที่ 3.7 การออกแบบชิ้นส่วนเชื่อมต่อระหว่างมอเตอร์กับอลูมิเนียมโปรไฟล์

3.1.4.3 แขนบน

แขนบนทำจากอลูมิเนียม เนื่องจากมีน้ำหนักเบาแต่แข็งแรงมีความต้านทานการกัดกร่อนได้ดี โดยมีการออกแบบให้มีขนาดพื้นที่หน้าตัด 16x16 mm^2 ความยาวทั้งหมด 112 mm และทำการเจาะรูเพื่อใช้เชื่อมต่อกับมอเตอร์และแขนล่าง โดยเจาะรูขนาดเส้นผ่านศูนย์กลาง 8 mm เพื่อเชื่อมต่อกับเพลาของมอเตอร์ เจาะรูขนาดเส้นผ่านศูนย์กลาง 3.2 mm เพื่อเชื่อมต่อกับแขนล่าง ซึ่งความยาวระหว่างจุดศูนย์กลางรูที่ใช้เชื่อมต่อกับมอเตอร์และจุดศูนย์กลางรูที่ใช้เชื่อมต่อกับแขนล่าง เท่ากับ 100 mm ส่วนสองรูด้านบนเป็นการเจาะเพื่อใส่เกลียวตัวหนอนสำหรับยึดเพลามอเตอร์กับติดแขนบน ดังแสดงในรูปที่ 3.8 และรูปที่ 3.9



รูปที่ 3.8 แขนบน

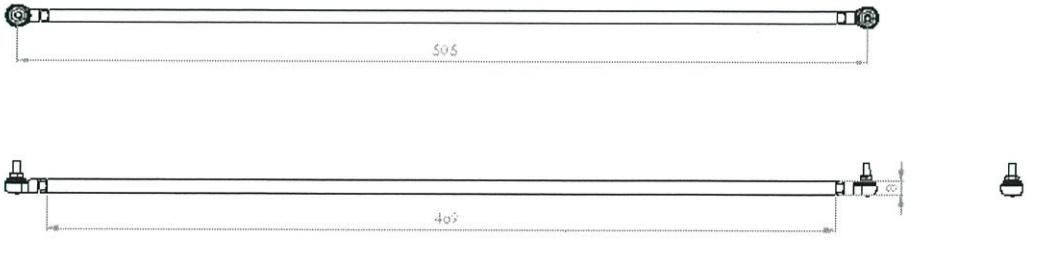


รูปที่ 3.9 การออกแบบแขนบน

3.1.4.4 <u>แขนล่าง</u>

แขนล่างเลือกใช้วัสดุคาร์บอนไฟเบอร์กลวงเนื่องจากมีคุณสมบัติที่ เบาแข็งแรงและมีความยืดหยุ่น โดยออกแบบให้มีความยาว 469 mm เมื่อต่อกับ Ball Joint แขนล่างจะมีความยาวทั้งหมด 505 mm ซึ่งวัดจากจุดหมุนของข้อต่อแบบ Ball Joint ทั้งสองด้าน ดังแสดงในรูปที่ 3.10 และรูปที่ 3.11

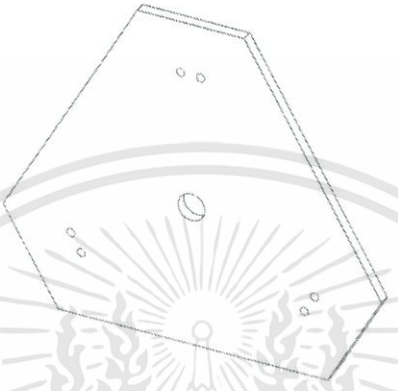
รูปที่ 3.10 แขนล่าง



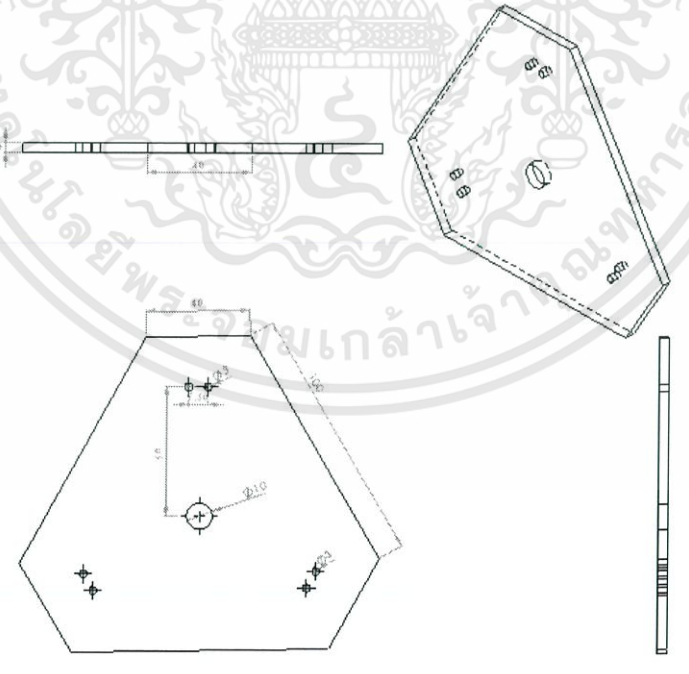
รูปที่ 3.11 การออกแบบแขนล่าง

3.1.4.5 ฐานตัวยกจับ

ทำจากอลูมิเนียมแผ่นบาง ความหนา 2 mm ตัดเป็นรูปสามเหลี่ยมด้านเท่ายาวด้านละ 180 mm และตัดมุมแหลมออกเพื่อลดความเสี่ยงในการชนขณะเคลื่อนที่ เจาะรูขนาดเส้นผ่านศูนย์กลาง 2.5 mm จำนวน 6 รู เพื่อยึดกับก้อนอลูมิเนียม โดยมีระยะห่างจากจุดศูนย์กลางของฐานตัวยกจับ 50 mm และเจาะรูขนาดเส้นผ่านศูนย์กลาง 8 mm จำนวน 1 รูเพื่อยึดกับตัวยึดจับสำหรับยกจับภาระ ดังแสดงในรูปที่ 3.12 และรูปที่ 3.13



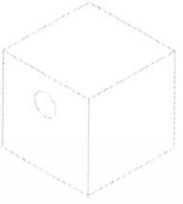
รูปที่ 3.12 ฐานตัวยกจับ



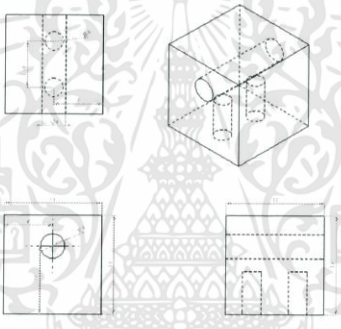
รูปที่ 3.13 การออกแบบฐานตัวยกจับ

3.1.4.6 ชิ้นส่วนเชื่อมต่อระหว่างฐานตัวยกจับและแขนล่าง

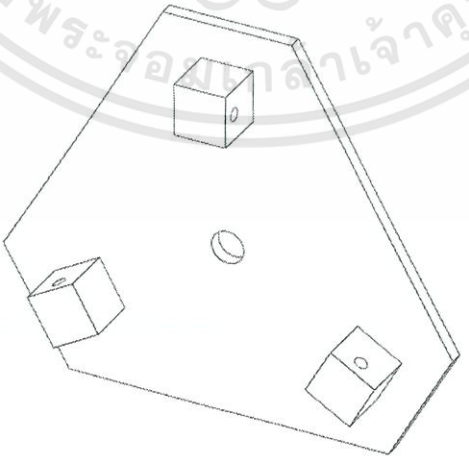
ทำจากอลูมิเนียม ออกแบบให้มีขนาด 16x16X16 mm^3 เจาะรูด้านข้างทั้งสองด้านขนาด 3.2 mm เพื่อต่อกับข้อต่อแบบ Ball Joint ซึ่งต่อกับแขนล่างและเจาะรูขนาดเส้นผ่านศูนย์กลาง 2.5 mm จำนวน 2 รู เพื่อยึดกับอลูมิเนียมแผ่นบาง ดังแสดงในรูปที่ 3.14 และรูปที่ 3.15 โดยเมื่อประกอบจะได้ดังรูปที่ 3.16



รูปที่ 3.14 ชิ้นส่วนเชื่อมต่อระหว่างฐานตัวยกจับและแขนล่าง



รูปที่ 3.15 การออกแบบชิ้นส่วนเชื่อมต่อระหว่างฐานตัวยกจับและแขนล่าง



รูปที่ 3.16 ฐานตัวยกจับ

3.1.5 การหาแรงบิดของมอเตอร์

ในการออกแบบได้เลือกใช้มอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) เป็นต้นกำลังสำหรับขับเคลื่อนแขนของหุ่นยนต์เนื่องจากมอเตอร์แบบสเต็ปปิ้งสามารถควบคุมได้ง่าย ในการใช้งานมอเตอร์แบบสเต็ปปิ้งจำเป็นต้องทำการคำนวณหาค่าแรงบิดสูงสุดของภาระโหลด (Maximum Torque) ที่มอเตอร์จะได้รับ ดังแสดงในรูปที่ 3.17 จากนั้นจึงทำการเลือกมอเตอร์ที่สามารถรองรับภาระโหลดสูงสุดได้ โดยวิธีการคำนวณหาแรงบิดสูงสุดสามารถทำได้ดังต่อไปนี้

วิธีคำนวณหาแรงบิดสูงสุด

กำหนดให้ แขนบนหนัก 50 g

แขนล่างหนัก 50 g

ภาระโหลดหนัก 100 g โดยใช้แขนกลสามแขนยกภาระขึ้นพร้อมกัน

ดังนั้นแขนทั้งสามจึงรับน้ำหนักเฉลี่ยเท่ากับ 33.33 g

ฐานยกจับหนัก 100 g โดยใช้แขนกลสามแขนยกภาระขึ้นพร้อมกัน

ดังนั้นแขนทั้งสามจึงรับน้ำหนักเฉลี่ยเท่ากับ 33.33 g

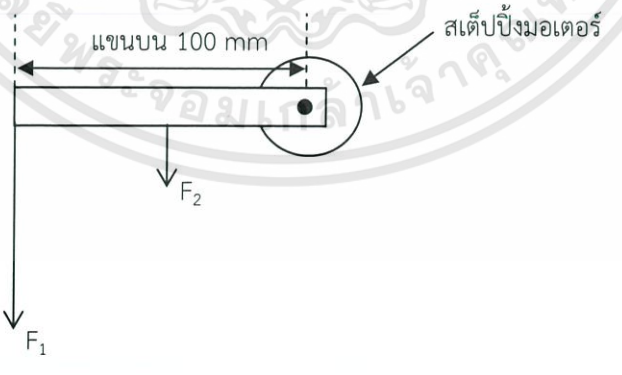
จากสูตร $F = mg$ (3.1)

เมื่อรวมน้ำหนักของแขนล่างภาระโหลด และ ฐานยกจับเข้าด้วยกัน จะได้

$$F_1 = (50+33.33+33.33) \times 10^{-3} \times 9.81$$
$$= 116.66 \times 10^{-3} \times 9.81$$
$$= 1.114 \ N$$

แรงจากน้ำหนักของแขนบน

$$F_2 = 50 \times 10^{-3} \times 9.81$$
$$= 0.490 \ N$$



รูปที่ 3.17 การคำนวณแรงบิดสูงสุดของมอเตอร์

จากสูตร $\tau = Fd$ (3.2)

เมื่อ F คือ แรงที่กระทำ และ d คือ ระยะที่ตั้งฉากกับแนวแรง

$$\tau = (1.114 \times 100 \times 10^{-3}) + (0.490 \times 50 \times 10^{-3})$$

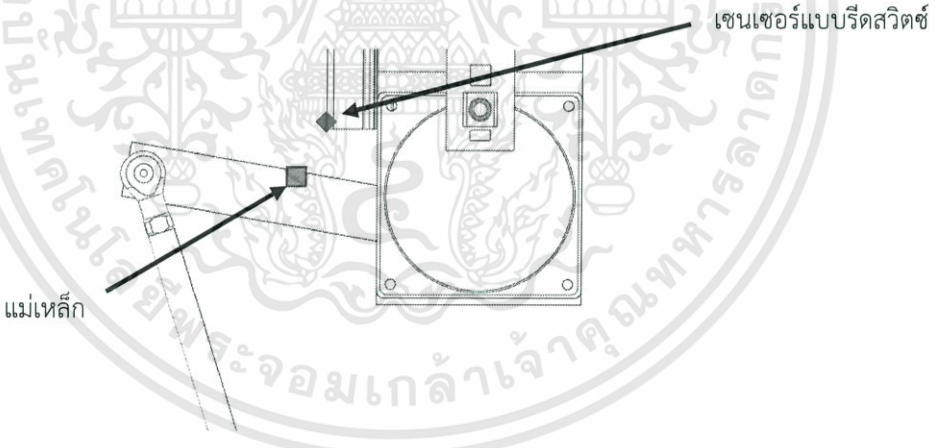
$$= 0.136 \text{ N.m} \quad \text{หรือ } 1.386 \text{ kg.cm}$$

ดังนั้นจึงเลือกใช้มอเตอร์แบบสเต็ปปิ้ง (Stepping motor) จำนวน 3 ตัว เนื่องจากคำนวณหาค่าแรงบิด ได้เท่ากับ 1.386 kg.cm จึงเลือกใช้มอเตอร์แบบสเต็ปปิ้ง (Stepping motor) รุ่น PH264-01-C97 แบบ 2 เฟส DC 4V 1.1A โดยแต่ละสเต็ปจะขับเคลื่อนได้ 1.8 องศา ทดเกียร์ 1:10 ขนาดเฟรม 60x60 mm^2 เพลามีเส้นผ่านศูนย์กลาง 8 mm และมีขนาด Maximum Holding Torque 3.97 kg.cm

3.1.6 การติดตั้งเซนเซอร์แบบรีดสวิตช์ (Reed Switch)

เป็นการป้องกันไม่ให้แขนบนของหุ่นยนต์ชนกับโครงอลูมิเนียมโปรไฟล์ ซึ่งเมื่อแขนบนเคลื่อนที่ขึ้นมาโดยมีระยะห่างจากบริเวณที่ติดตั้งเซนเซอร์แบบรีดสวิตช์ ประมาณ 1 cm แม่เหล็กที่ติดอยู่ด้านข้างของแขนบนจะทำให้เซนเซอร์ทำงาน โดยแขนบนทั้งสามแขนจะหยุดทำมุม -24 องศากับระนาบ XY และเป็นการกำหนดตำแหน่งเริ่มต้นของหุ่นยนต์ ดังแสดงในรูปที่ 3.18 โดยมีตำแหน่งของการติดตั้งรีดสวิตช์ ดังนี้

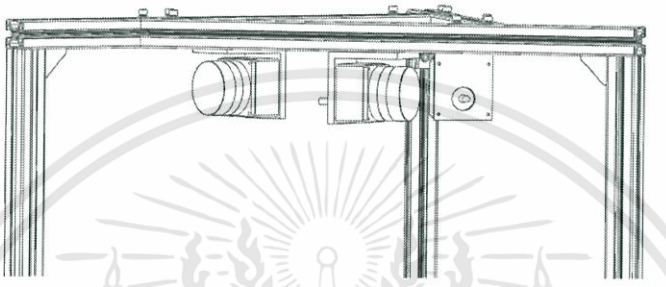
- ติดตั้งเซนเซอร์แบบรีดสวิตช์ในตำแหน่งเหนือจากแขนบนของแขนทั้งสาม
- ติดแม่เหล็กบริเวณด้านข้างของแขนบน โดยมีระยะห่างจากปลายแขนเป็นระยะ 4.4 cm



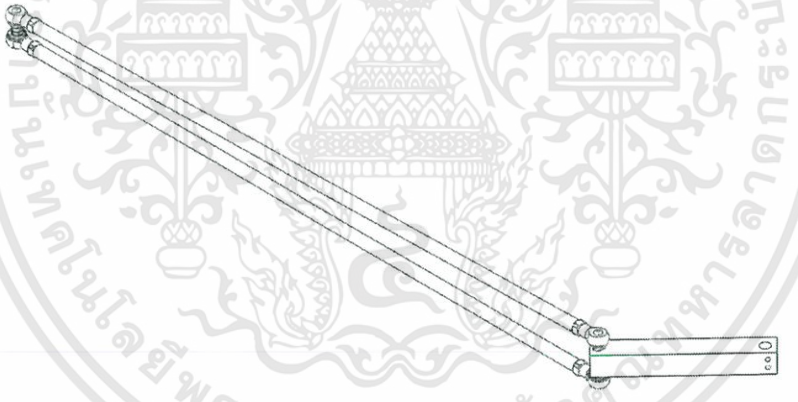
รูปที่ 3.18 การติดตั้งรีดสวิตช์

3.1.7 การประกอบหุ่นยนต์แบบขนาน

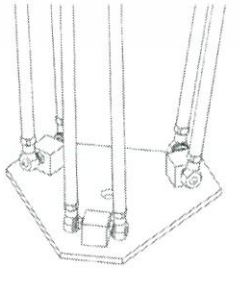
ในการประกอบหุ่นยนต์แบบขนาน ได้มีการออกแบบการประกอบดังต่อไปนี้ โดยเริ่มจากการติดตั้งมอเตอร์ทั้งสามตัวกับโครงอลูมิเนียมด้านบน ซึ่งในการติดตั้งมอเตอร์นั้นจะต้องทำมุมกัน 120 องศา ดังแสดงในรูปที่ 3.19 จากนั้นทำการประกอบแขนบนและแขนล่างเข้าด้วยกันโดยใช้ข้อต่อแบบ Ball Joint เป็นตัวเชื่อมระหว่างชิ้นงานทั้งสอง ดังแสดงในรูปที่ 3.20 เชื่อมต่อฐานตัวยกจับกับแขนของหุ่นยนต์ทั้ง 3 แขน ดังแสดงในรูปที่ 3.21 และสุดท้ายต่อแขนของหุ่นยนต์เข้ากับเพลาของมอเตอร์ ดังแสดงในรูปที่ 3.22 เมื่อประกอบส่วนประกอบทั้งหมดเสร็จสิ้นจะได้ต้นแบบหุ่นยนต์แบบขนานดังแสดงในรูปที่ 3.23



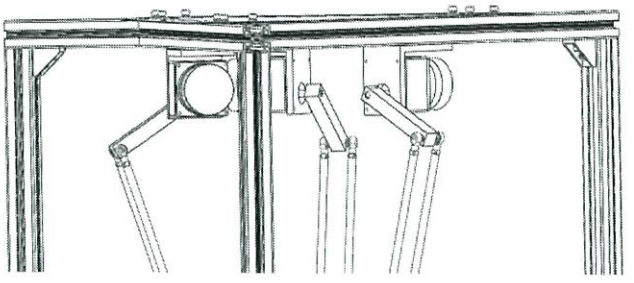
รูปที่ 3.19 การติดตั้งมอเตอร์กับโครงอลูมิเนียมด้านบน



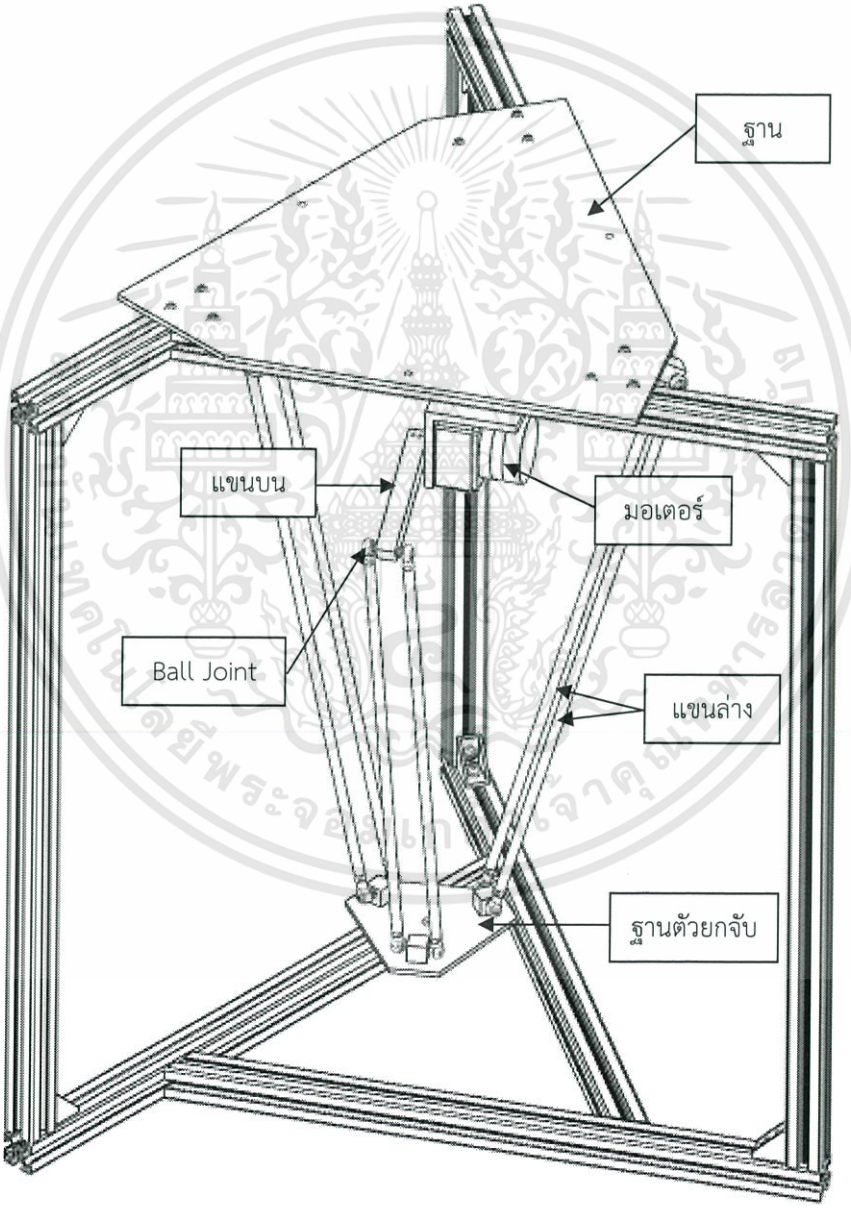
รูปที่ 3.20 การประกอบแขนบนกับแขนล่าง



รูปที่ 3.21 การประกอบแขนหุ่นยนต์กับฐานตัวยกจับ



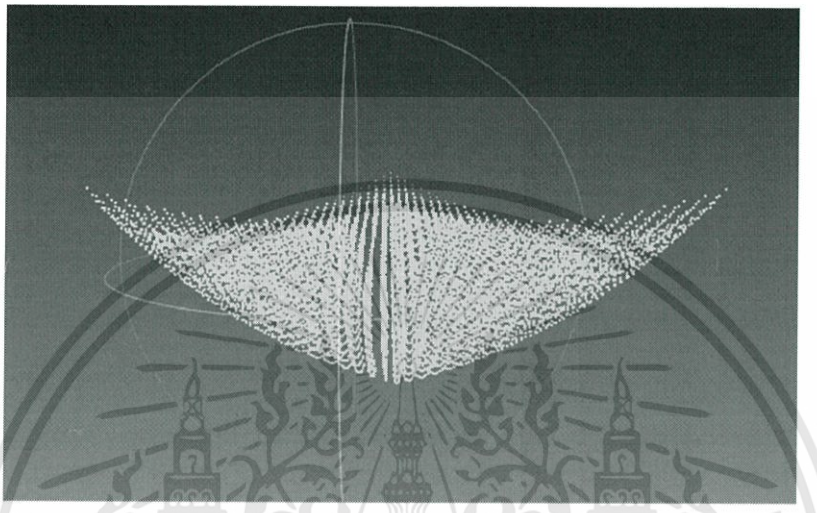
รูปที่ 3.22 การเชื่อมต่อแขนหุ่นยนต์เข้ากับเพลาของมอเตอร์



รูปที่ 3.23 หุ่นยนต์แบบขนาน (Delta Robot) ออกแบบโดยโปรแกรม Solid Works

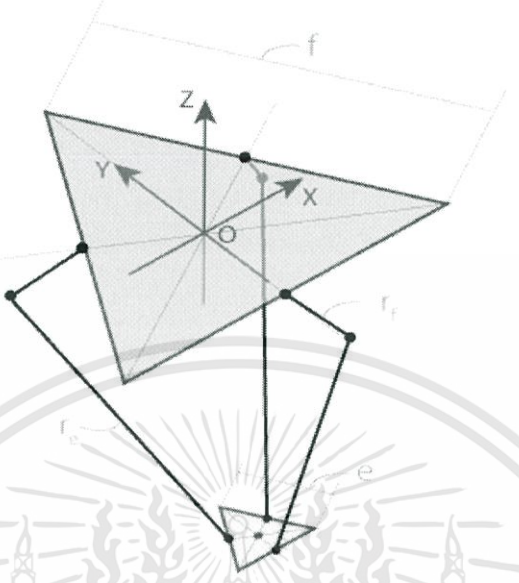
3.1.8 พื้นที่การทำงานของหุ่นยนต์แบบขนาน

ภายหลังการประกอบหุ่นยนต์ได้ทำการคำนวณพื้นที่การทำงานของหุ่นยนต์แบบขนาน โดยแขนบนสามารถเคลื่อนที่ขึ้นเป็นมุมไม่เกิน 24 องศาเคลื่อนที่ลงเป็นมุมไม่เกิน 80 องศา เทียบจากระนาบ XY จากวิธีการหาพื้นที่การทำงานที่ได้นำเสนอไปแล้วในบทที่ 2 สามารถหาพื้นที่การทำงานทั้งหมดเท่ากับ 12523.54 cm^3 และมีรูปร่างดังแสดงในรูปที่ 3.24



รูปที่ 3.24 พื้นที่การทำงานของต้นแบบหุ่นยนต์แบบขนาน

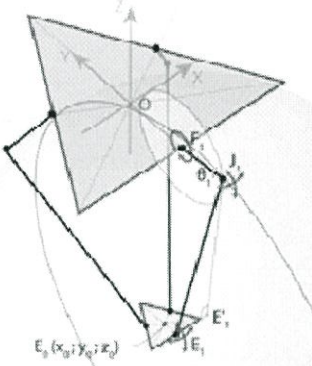
3.2 การคำนวณการเคลื่อนที่แบบผกผัน (Inverse Kinematics)



รูปที่ 3.25 การกำหนดพิกัดตำแหน่งของหุ่นยนต์ [17]

การควบคุมหุ่นยนต์โดยใช้หลักการของการเคลื่อนที่แบบผกผัน (Inverse Kinematics) เป็นการที่ผู้ควบคุมกำหนดตำแหน่ง แล้วนำไปคำนวณหามุมและจำนวนสเต็ปของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) เพื่อให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งที่ผู้ควบคุมต้องการ โดยพิกัดตำแหน่งของหุ่นยนต์จะถูกกำหนดให้อยู่ในแกน X,Y,Z โดยแสดงดังรูปที่ 3.25 จุด O คือจุดที่เกิดจากการตัดกันของแกน X,Y,Z โดยจะกำหนดให้จุด O มีพิกัดเป็น (0,0,0) และเป็นจุดกำเนิดของปริภูมิสามมิติของหุ่นยนต์ โดยจากการออกแบบส่วนประกอบต่างๆ ของหุ่นยนต์จะทำให้ได้ค่าพารามิเตอร์ต่างๆ ดังนี้

- ฐานบนยึดติดอยู่กับที่ (Fixed Base) : f	มีความยาวด้านเท่ากับ	398	mm
- ฐานล่างเคลื่อนที่อิสระ (End Effector) : e	มีความยาวด้านเท่ากับ	87	mm
- แขนบนเชื่อมต่อกับมอเตอร์แบบสเต็ปปิ้ง(Upper Arm) : r_f	มีความยาวเท่ากับ	100	mm
- แขนล่างเชื่อมกับแขนบนกับฐานล่าง (Lower Arm) : r_e	มีความยาวเท่ากับ	505	mm



รูปที่ 3.26 ทรงกลมที่มีจุด E1 เป็นจุดศูนย์กลาง [17]

3.2.1 สูตรที่ใช้ในการคำนวณหาจำนวนสเต็ปของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor)

1. กำหนดจุด E_0 ให้มีพิกัด (x_0, y_0, z_0)

2. หาระยะ E_0E_1 $= \frac{e}{2}\tan 30°$ $= 43.5\tan 30°$ $= 25.11$

3. หาพิกัด E_1 $(x_0, y_0 - \frac{e}{2}\tan 30°, z_0)$ $= (x_0, y_0 - 25.11, z_0)$

4. หาพิกัด E'_1 $(0, y_0 - \frac{e}{2}\tan 30°, z_0)$ $= (0, y_0 - 25.11, z_0)$

5. หาระยะ E'_1J_1 $= \sqrt{r_e^2 + x_0^2}$ $= \sqrt{505^2 + x_0^2}$

6. หาพิกัด F_1 $(0, -\frac{f}{2}\tan 30°, 0)$ $= (0, -114.89, 0)$

7. จากสูตรตรีโกณมิติ (สามเหลี่ยมมุมฉาก $a^2+b^2 = c^2$)

โดยใช้สมการ

$$\left(Y_{J_1} - Y_{F_1}\right)^2 + \left(Z_{J_1} - Z_{F_1}\right)^2 = r_f^2 \qquad (3.3)$$

$$\left(Y_{J_1} - Y_{E'_1}\right)^2 + \left(Z_{J_1} - Z_{E'_1}\right)^2 = r_e^2 - x_0^2 \qquad (3.4)$$

จากสมการที่ (3.3) จะได้ $$\left(Y_{J_1}-(-114.89)\right)^2+\left(Z_{J_1}\right)^2=100^2 \quad (3.5)$$

จากสมการที่ (3.4) จะได้ $$\left(Y_{J_1}-(y_0-25.11)\right)^2+\left(Z_{J_1}-z_0\right)^2=505^2-x_0^2 \quad (3.6)$$

จากการแก้สมการ (3.5) และ (3.6) จะได้

กำหนดให้ $$a=-114.89-\left(\frac{X_0^2+(Y_0-25.11)^2+Z_0^2-10768.71}{2Z_0^2}\right)\left(-114.89-(Y_0-25.11)\right)$$

$$b=\sqrt{-\left[\left(\frac{X_0^2+(Y-25.11)^2+Z_0^2-10768.71}{2Z_0}\right)+\left(\frac{13199.71+114.89(Y_0-25.11)}{Z_0}\right)\right]+\left(\frac{-11489-100(Y_0-25.11)^2}{Z_0}\right)+100^2}$$

$$c=\left(\frac{X_0^2+(Y_0-25.11)^2+Z_0^2-10768.71}{2Z_0}\right)+\frac{\left(-114.89-(Y_0-25.11)\right)}{Z_0}$$

จะได้

$$Y_{J_1}=\frac{a-b}{\left(\frac{-114.89-(Y_0-25.11)}{Z_0}\right)^2+1}$$

$$Z_{J_1}=\frac{c\times(a-b)}{\left(\frac{-114.89-(Y_0-25.11)}{Z_0}\right)+1}$$

จากนั้นจะได้ค่าพิกัด J_1 เพื่อนำไปคำนวณหามุมและจำนวนสเต็ปของมอเตอร์ โดยใช้สมการ

$$\theta_1=\tan^{-1}\left(\frac{Z_{J_1}}{Y_{F_1}-Y_{J_1}}\right)\qquad=\tan^{-1}\left[\frac{Z_{J_1}}{-114.89-Y_{J_1}}\right] \quad (3.7)$$

8. หาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์ โดยการนำค่ามุมที่ได้คูณด้วยจำนวนทดรอบของมอเตอร์แบบสเต็ปปิ้งจากนั้นทำการแปลงเป็นจำนวนสเต็ปโดยสมการ

$$\text{จำนวนสเต็ปของมอเตอร์} = \frac{\text{มุมระหว่างแขนบนกับระนาบXY} \times \text{จำนวนทดรอบของมอเตอร์}}{\text{มุมองศาต่อสเต็ปของมอเตอร์}} \qquad (3.8)$$

9. การหามุมระหว่างแขนบนกับระนาบ XY และจำนวนสเต็ปของมอเตอร์ที่ 2 และ 3 ใช้วิธีการเหมือนการหามุมที่ 1 แต่ต้องทำการหาพิกัด E_0 ใหม่ โดยการหมุนระนาบ XY รอบแกน Z 120 องศา ทวนเข็มนาฬิกาและตามเข็มนาฬิกา

โดยสมการ $x'_0 = x_0 \cos 120° + y_0 \sin 120°$ (3.9)

$y'_0 = -x_0 \sin 120° + y_0 \cos 120°$ (3.10)

ตัวอย่างการคำนวณหาจำนวนสเต็ปมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) ทั้งสามตัว

1. กำหนดจุด E_0 ให้มีพิกัด (x_0, y_0, z_0) $= (0, 0, -430)$

2. หาระยะE_0E_1 $= \frac{e}{2}\tan 30°$ $= 43.5\tan 30°$ $= 25.11$

3. พิกัด E_1 $(x_0, y_0 - \frac{e}{2}\tan 30°, z_0)$ $= (0, 0 -25.11, -430)$ $= (0, -25.11, -430)$

4. พิกัด E'_1 $(0, y_0 - \frac{e}{2}\tan 30°, z_0)$ $= (0, 0 -25.11, -430)$ $= (0, -25.11, -430)$

5. หาระยะ E'_1J_1 $= \sqrt{r_e^2 + x_0^2}$ $= \sqrt{505^2 + 0^2}$ $= 505$

6. พิกัด F_1 $(0, -\frac{f}{2}\tan 30°, 0)$ $= (0, -114.89, 0)$

7. จากสูตรตรีโกณมิติ (สามเหลี่ยมมุมฉาก $a^2+b^2 = c^2$)

แทนค่าในสมการที่ (3.3) จะได้ $\left(Y_{J_1} - (-114.89)\right)^2 + \left(Z_{J_1} - 0\right)^2 = 100^2$

แทนค่าในสมการที่ (3.4) จะได้ $\left(Y_{J_1} - (-25.11)\right)^2 + \left(Z_{J_1} - (-430)\right)^2 = 505^2 - 0^2$

ทำการแก้สมการจะได้ค่า $Y_{J_1} = -23.929$ และ $Z_{J_1} = 41.548$

ดังนั้น พิกัด $J_1 = (0, -23.929, 41.548)$

8. การหามุมและจำนวนสเต็ปของมอเตอร์ตัวที่ 1

หา θ_1 จากสมการที่ (3.7) จะได้ $\theta_1 = \tan^{-1}\left(\frac{Z_{J_1}}{Y_{F_1} - Y_{J_1}}\right) = \tan^{-1}\left[\frac{41.548}{-114.89 - (-23.929)}\right]$

$\therefore \theta_1$ = -24.549°

หาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์ โดยการนำค่ามุมที่ได้คูณด้วยจำนวนทดรอบของมอเตอร์แบบสเต็ปปิ้ง
จะได้ -24.549° x 10 (ทด 10 รอบ) = -245.49

ทำการแปลงองศาเป็นจำนวนสเต็ป $\frac{-245.49}{1.8}$ = -136.383

= -137 สเต็ป (ส่งเป็นจำนวนเต็มให้กับไมโครคอนโทรลเลอร์)

9. จากนั้นหมุนระนาบ xy 120° (ทวนเข็มนาฬิกา)
จะได้พิกัด E_0 ใหม่คือ (x'_0, y'_0, z'_0)

$x'_0 = x_0 \cos 120° + y_0 \sin 120°$

$= 0\cos 120° + 0\sin 120° = 0°$

$y'_0 = -x_0 \sin 120° + y_0 \cos 120°$

$= -0\sin 120° + 0\cos 120° = 0°$

$\therefore$ พิกัด $E'_0 (x'_0, y'_0, z'_0) = (0,0, -430)$

หาพิกัด J_2 ด้วยวิธีเหมือนหา J_1 แล้วแทนค่าลงสูตร
จะได้สมการ $\left(Y_{J_2} - Y_{F_2}\right)^2 + \left(Z_{J_2} - Z_{F_2}\right)^2 = r_f^2$

$\left(Y_{J_2} - Y_{E'_2}\right)^2 + \left(Z_{J_2} - Z_{E'_2}\right)^2 = r_e^2 - x_0^2$

แทนค่าในสมการที่ (3.3) จะได้ $\left(Y_{J_2} - (-114.89)\right)^2 + \left(Z_{J_2} - 0\right)^2 = 100^2$

แทนค่าในสมการที่ (3.4) จะได้ $\left(Y_{J_2} - (-25.11)\right)^2 + \left(Z_{J_2} - (-430)\right)^2 = 505^2 - 0^2$

ทำการแก้สมการจะได้ค่า Y_{J_2} = -23.929 และ Z_{J_2} = 41.548

ดังนั้นพิกัด J_2 = (0, -23.929, 41.548)

การหามุมและจำนวนสเต็ปของมอเตอร์ตัวที่ 2

หา θ_2 โดยแทนค่า $\theta_2 = \tan^{-1}\left(\frac{Z_{J_2}}{Y_{F_2} - Y_{J_2}}\right) = \tan^{-1}\left[\frac{41.548}{-114.89 - (-23.929)}\right]$

$\therefore \theta_2$ = -24.549°

หาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์ โดยการนำค่ามุมที่ได้คูณด้วยจำนวนทดรอบของมอเตอร์แบบสเต็ปปิ้ง

จะได้ -24.549° x 10 (ทด 10 รอบ) = -245.49

ทำการแปลงองศาเป็นจำนวนสเต็ป $\frac{-245.49}{1.8}$ = -136.383

= -137 สเต็ป (ส่งเป็นจำนวนเต็มให้กับไมโครคอนโทรลเลอร์)

10. จากนั้น หมุนระนาบ xy 120° (ตามเข็มนาฬิกา)

จะได้ E_0 ใหม่คือ (x''_0, y''_0, z''_0)

$x''_0 = x_0 \cos 120° + y_0 \sin 120°$

$= 0\cos 120° + 0\sin 120° = 0$

$y''_0 = -x_0 \sin 120° + y_0 \cos 120°$

$= -0 \sin 120° + 0\cos 120° = 0$

$\therefore$ พิกัด $E''_0(x''_0, y''_0, z''_0)$ = (0,0,-430)

หาพิกัด J_3 เหมือนหา J_1 แทนค่าลงสูตร

จะได้สมการ $(Y_{J_3} - Y_{F_3})^2 + (Z_{J_3} - Z_{F_3})^2 = r_f^2$

$(Y_{J_3} - Y_{E'_3})^2 + (Z_{J_3} - Z_{E'_3})^2 = r_e^2 - x_0^2$

แทนค่าในสมการที่ (3.3) จะได้ $(Y_{J_2} - (-114.89))^2 + (Z_{J_2} - 0)^2 = 100^2$

แทนค่าในสมการที่ (3.4) จะได้ $(Y_{J_2} - (-25.11))^2 + (Z_{J_2} - (-430))^2 = 505^2 - 0^2$

จะได้ค่า Y_{J_3} = -23.929 และ Z_{J_3} = 41.548

ดังนั้นพิกัด J_3 = (0, -23.929, 41.548)

การหามุมและจำนวนสเต็ปของมอเตอร์ตัวที่ 3

หา θ_3 โดยแทนค่า $\theta_3 = \tan^{-1}\left(\frac{Z_{J_3}}{Y_{F_3} - Y_{J_3}}\right) = \tan^{-1}\left[\frac{41.548}{-114.89 - (-23.929)}\right]$

$\therefore \theta_3$ = -24.549°

หาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์ โดยการนำค่ามุมที่ได้คูณด้วยจำนวนทดรอบของมอเตอร์แบบสเต็ปปิ้ง

จะได้ -24.549° x 10 (ทด 10 รอบ) = -245.49

ทำการแปลงองศาเป็นจำนวนสเต็ป $\frac{-245.49}{1.8}$ = -136.383

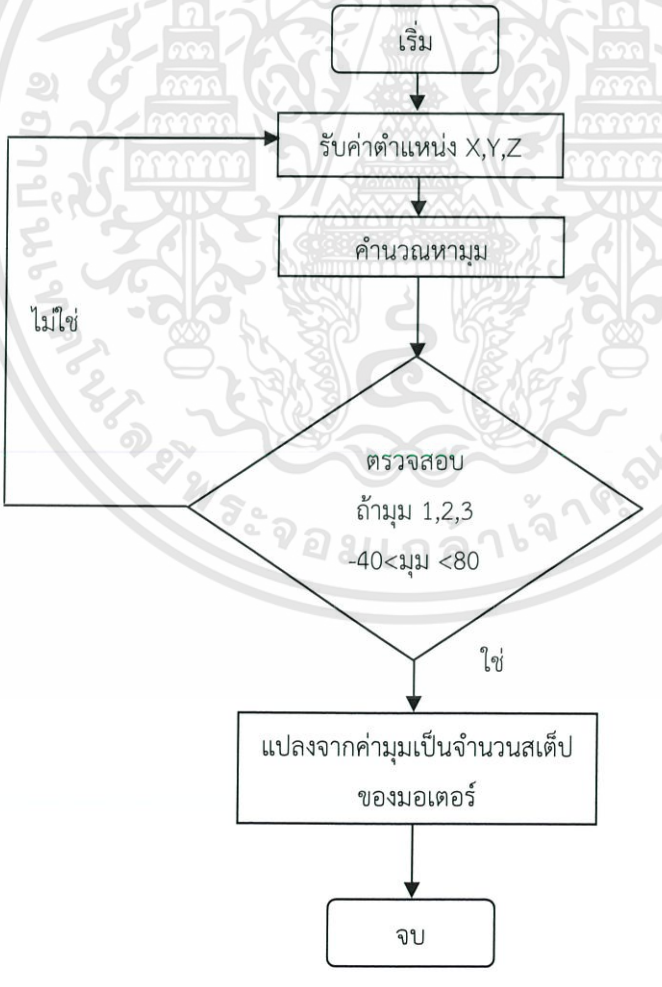
= -137 สเต็ป (ส่งเป็นจำนวนเต็มให้กับไมโครคอนโทรลเลอร์)

3.3 การออกแบบโปรแกรมควบคุมการทำงาน

ในการออกแบบโปรแกรมควบคุมการทำงานเริ่มต้นจากการเขียนแผนภาพการทำงานของโปรแกรม เพื่อกำหนดขั้นตอนการทำงานของโปรแกรมดังแสดงในรูปที่ 3.27 และทำการออกแบบหน้าต่างของโปรแกรมควบคุมการทำงานด้วยโปรแกรม Microsoft Visio เพื่อวางตำแหน่งของข้อความและรูปภาพให้เป็นไปตามต้องการ จากนั้นการใช้โปรแกรม Visual C# เพื่อพัฒนาโปรแกรมสำหรับควบคุมการทำงานของหุ่นยนต์แบบขนาน (Delta Robot) ซึ่งโปรแกรมที่พัฒนาขึ้นนั้นสามารถรับคำสั่งจากผู้ใช้งาน เพื่อประมวลผลและส่งค่าให้กับระบบควบคุมการทำงานของหุ่นยนต์โดยมีการแบ่งการทำงาน ดังนี้

- การศึกษาการเขียนแผนภาพการทำงานของหุ่นยนต์แบบขนาน
- การออกแบบหน้าต่างการทำงานโดยโปรแกรม Microsoft Visio
- การสร้างรูปแบบของโปรแกรม
- การตรวจสอบการทำงานของโปรแกรม
- การทดลองโปรแกรมกับต้นแบบหุ่นยนต์แบบขนาน (Delta Robot)

3.3.1 การเขียนแผนภาพการทำงานของหุ่นยนต์แบบขนาน (Delta Robot)

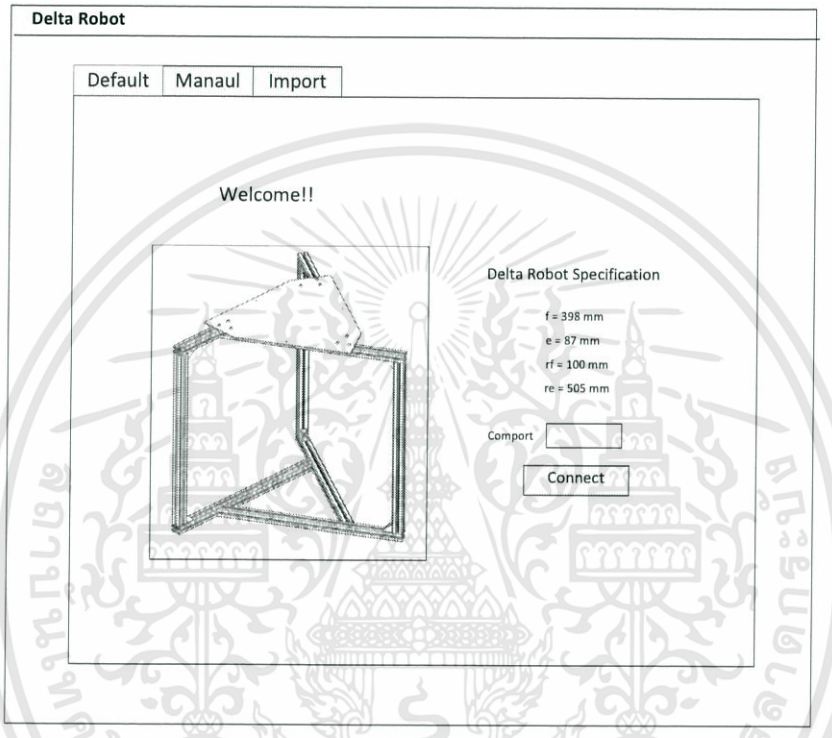


รูปที่ 3.27 แผนภาพการทำงานของต้นแบบหุ่นยนต์แบบขนาน (Delta Robot)

3.3.2 การออกแบบหน้าต่างโปรแกรม

ในการออกแบบได้กำหนดรูปแบบการทำงานของโปรแกรมไว้ 2 รูปแบบ คือ การใช้งานแบบ Manual เป็นการใช้งานโดยรับค่าพิกัดตำแหน่งที่ต้องการให้หุ่นยนต์เคลื่อนที่ไปทีละตำแหน่งและการใช้งานโดยสามารถ Import ไฟล์ในรูปแบบ .txt เพื่อให้หุ่นยนต์เคลื่อนที่ในรูปแบบต่างๆ ได้อย่างอัตโนมัติซึ่งมีขั้นตอนในการออกแบบดังนี้

1. ในหน้าต่าง Default ออกแบบให้มีการแสดงค่าพารามิเตอร์ต่างๆ ของหุ่นยนต์กำหนดคอมพอร์ต (Port) เพื่อเชื่อมต่อระหว่างคอมพิวเตอร์กับหุ่นยนต์แบบขนานให้ถูกต้องดังแสดงในรูปที่ 3.28



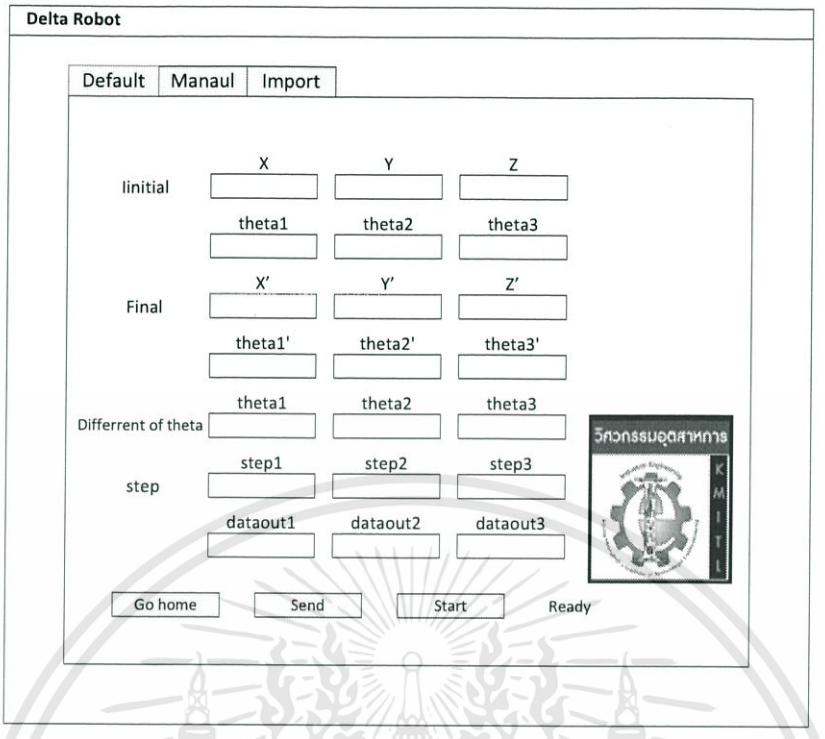
รูปที่ 3.28 หน้าต่าง Default ของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์

2. ในหน้าต่าง Manual ดังแสดงในรูปที่ 3.29 ได้ออกแบบโดยก่อนมีการสั่งงานให้หุ่นยนต์เคลื่อนที่ต้องทำการตั้งค่าให้แขนของหุ่นยนต์ทั้งสามอยู่ในตำแหน่งเริ่มต้น (Go Home)

3. กำหนดให้โปรแกรมแสดงจุดพิกัดเริ่มต้น (X,Y,Z) และสามารถป้อนค่าพิกัดตำแหน่ง (X',Y',Z') ที่ต้องการให้หุ่นยนต์เคลื่อนที่ไป จากนั้นกดปุ่ม Calculate เพื่อคำนวณตามสมการการเคลื่อนที่แบบผกผัน(Inverse Kinematics)และแสดงค่ามุมการเคลื่อนที่ของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) ทั้งสามตัวโดยค่ามุมจะต้องอยู่ระหว่าง -40 ถึง 80 องศา หากค่ามุมไม่อยู่ในช่วงดังกล่าวโปรแกรมจะไม่ทำงาน

4. จากนั้นเมื่อได้ค่ามุมการเคลื่อนที่ของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) นำมุมที่ได้มาทำการหาค่าผลต่างของมุมจากจุดเริ่มต้น แล้วทำการแปลงค่าผลต่างของมุมดังกล่าวเป็นจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์แต่ละตัว

5. ภายหลังจากที่ได้จำนวนสเต็ปการเคลื่อนที่ของมอเตอร์แต่ละตัวแล้ว และได้มีการออกแบบให้มีปุ่มส่งข้อมูล (Send) เพื่อทำการแปลงค่าที่ได้ให้อยู่ในรูปเลขฐานสิบ ซึ่งแสดงในช่อง Dataout และทำการส่งข้อมูลได้ไปยังระบบควบคุมการทำงานของหุ่นยนต์แบบขนาน จากนั้นสร้างปุ่มเริ่มต้น (Start) เพื่อเริ่มการทำงานของหุ่นยนต์

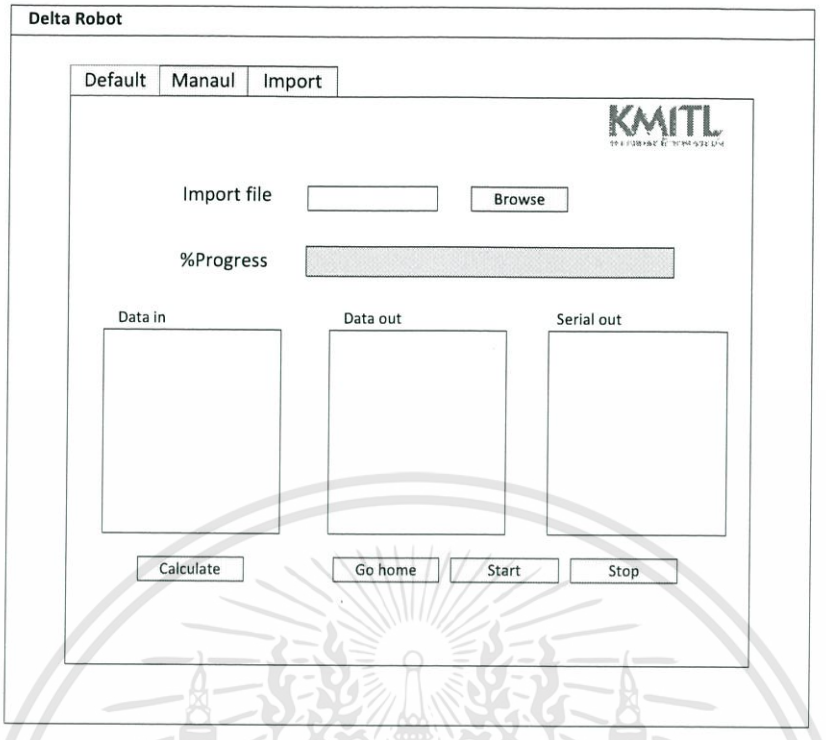


รูปที่ 3.29 หน้าต่าง Manual ของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์

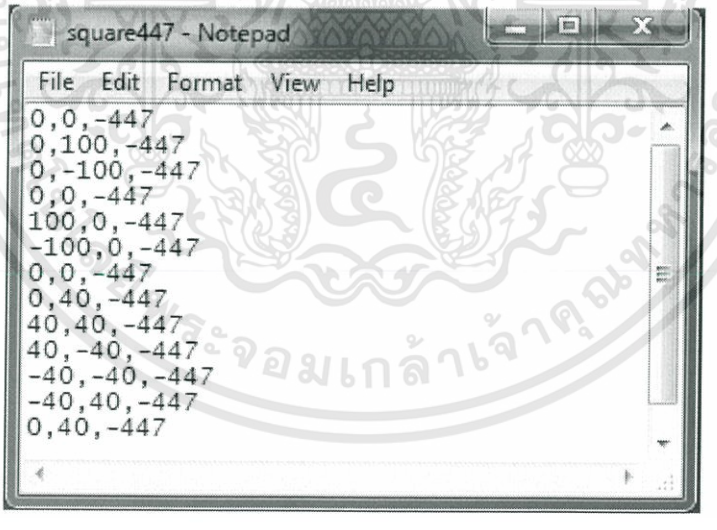
6. ในหน้าต่าง Import ดังแสดงในรูปที่ 3.30 ได้มีการออกแบบให้สามารถ Import ไฟล์ .txt ที่กำหนดรูปแบบการเคลื่อนที่ของหุ่นยนต์โดยพิกัดต่างๆ นั้นจะแสดงในช่อง Data in และไฟล์ .txt ต้องเป็นไฟล์ที่มีข้อมูลพิกัดตำแหน่งการเคลื่อนที่ของหุ่นยนต์ X,Y,Z ดังแสดงในรูปที่ 3.31

7. กำหนดปุ่มคำนวณ (Calculate) เพื่อคำนวณตามสมการการเคลื่อนที่แบบผกผัน (Inverse Kinematics) เช่นเดียวกับในหน้าต่าง Manual และแสดงค่าจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์แต่ละตัวในช่อง Data out

8. กำหนดปุ่มเริ่มต้น (Start) สำหรับการส่งข้อมูลจำนวนสเต็ปไปยังระบบควบคุมการเคลื่อนที่และเพื่อสั่งให้เริ่มการทำงานของหุ่นยนต์ โดยมีแถบแสดงสถานะการส่งข้อมูลและช่อง Serial Monitor เพื่อตรวจสอบการส่งข้อมูล และออกแบบปุ่ม Stop เพื่อหยุดการทำงานของหุ่นยนต์ทันทีเมื่อมีเหตุขัดข้องเกิดขึ้น



รูปที่ 3.30 หน้าต่าง Import ของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์



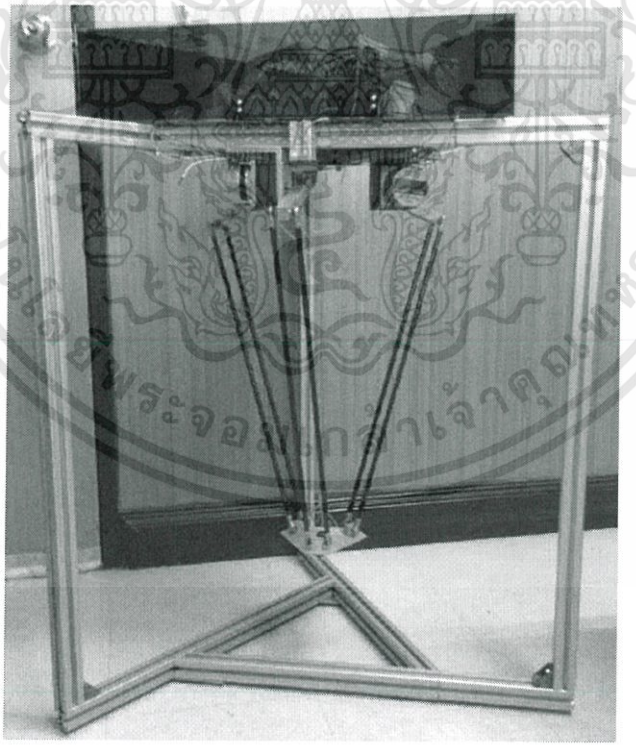
รูปที่ 3.31 ตัวอย่างไฟล์ .txt ในรูปแบบพิกัด X,Y,Z

บทที่ 4
ผลการดำเนินงาน

ในบทนี้เป็นการแสดงผลการดำเนินงานทางด้านเชิงกลและการออกแบบโปรแกรมความคุมการทำงาน รวมถึงทดสอบการทำงานของหุ่นยนต์แบบขนานชนิดสามแกนที่ได้ทำการสร้างขึ้นโดยจะแบ่งการทดสอบเป็นส่วนๆ ดังนี้ การวัดความเร็วเฉลี่ยของการเคลื่อนที่ การเคลื่อนที่ทีละตำแหน่งซึ่งจะเป็นการทดสอบเพื่อให้หุ่นยนต์เคลื่อนที่เป็นรูปร่างต่างๆ เช่น สามเหลี่ยม สี่เหลี่ยมและวงกลม เพื่อหาความแม่นยำของการเคลื่อนที่ การคำนวณหาค่าความผิดพลาดของตำแหน่งการเคลื่อนที่ รวมทั้งการเคลื่อนที่เพื่อทดสอบหาความสามารถในการทำซ้ำของหุ่นยนต์

4.1 ผลของการประกอบต้นแบบหุ่นยนต์แบบขนาน

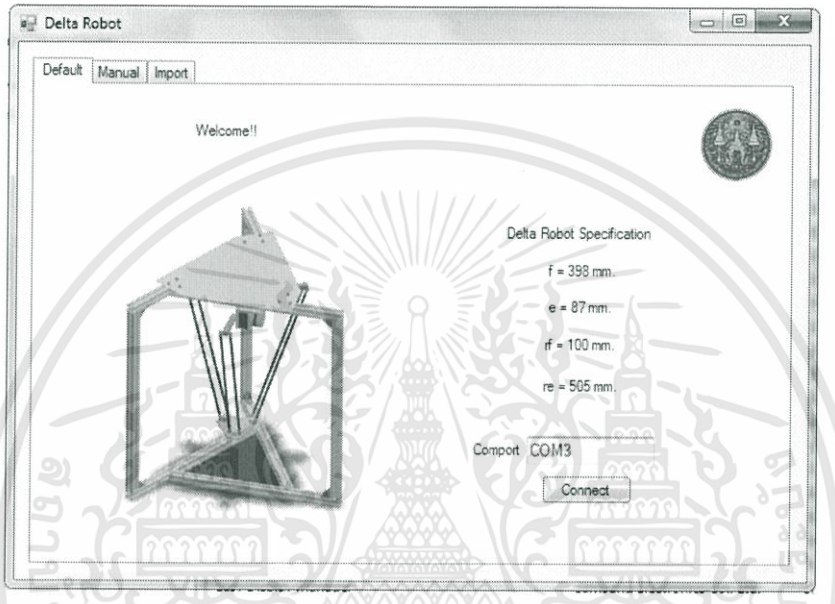
เมื่อทำการออกแบบหุ่นยนต์แบบขนานโดยโปรแกรมโซลิดเวิร์ค (Solid Works) ได้ทำการสร้างและประกอบชิ้นส่วนที่ออกแบบไว้ และได้ต้นแบบหุ่นยนต์แบบขนานที่สร้างขึ้นจริงดังแสดงในรูปที่ 4.1



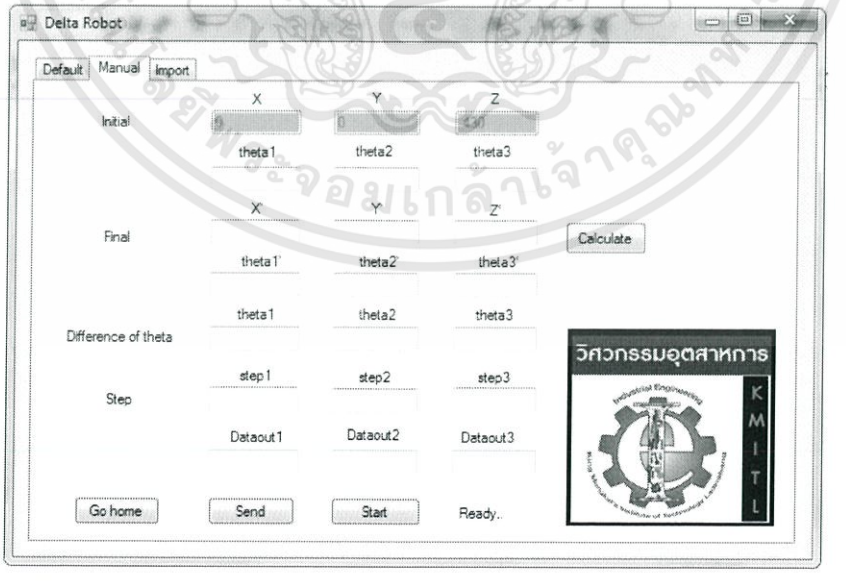
รูปที่ 4.1 หุ่นยนต์แบบขนานที่สร้างขึ้นจริง

4.2 ผลของการออกแบบโปรแกรมความคุมการทำงาน

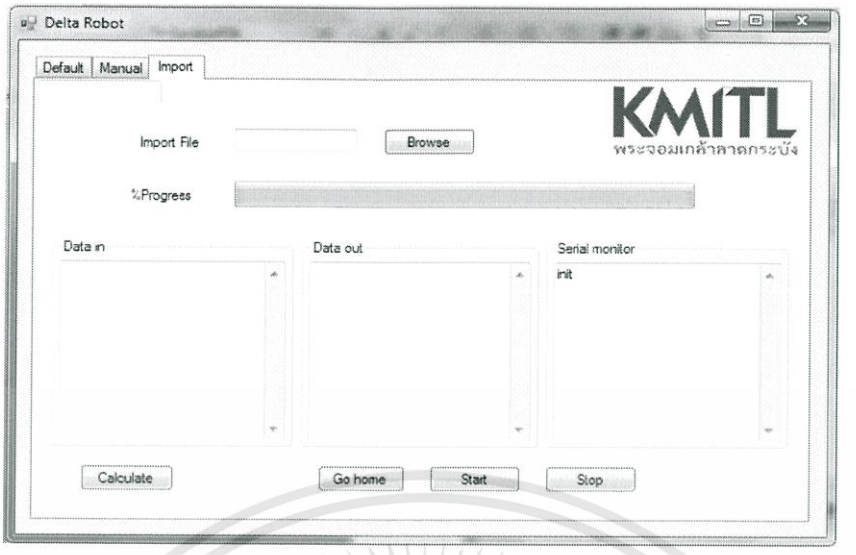
โปรแกรมที่พัฒนาขึ้นนี้สามารถเลือกรูปแบบการทำงานได้ 2 รูปแบบ คือ การใช้งานแบบ Manual เป็นการใช้งานโดยรับค่าพิกัดตำแหน่งที่ต้องการให้หุ่นยนต์เคลื่อนที่ไปทีละตำแหน่ง และการใช้งานโดยสามารถ Import ไฟล์ในรูปแบบ .txt เพื่อให้หุ่นยนต์เคลื่อนที่ในรูปแบบต่างๆ ได้อย่างอัตโนมัติ โดยมีหน้าจอแสดงผลดังนี้ หน้าต่าง Default ดังในแสดงในรูปที่ 4.2 หน้าต่าง Manual ดังในแสดงในรูปที่ 4.3 และหน้าต่าง Import ดังในแสดงในรูปที่ 4.4



รูปที่ 4.2 หน้าต่าง Default ของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์



รูปที่ 4.3 หน้าต่าง Manual ของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์



รูปที่ 4.4 หน้าต่าง Import ของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์

4.3 ผลการทดลองการวัดความเร็วเฉลี่ยของการเคลื่อนที่

การทดลองนี้เป็นการวัดความเร็วเฉลี่ยในการเคลื่อนที่ของหุ่นยนต์แบบขนาน ได้มีการออกแบบการทดลองโดยให้หุ่นยนต์เคลื่อนที่ในแกน x และแกน y รวมทั้งเคลื่อนที่สองแกนพร้อมกัน จากนั้นทำการจับเวลาและทำการคำนวณหาความเร็วเฉลี่ยของการเคลื่อนที่จากสมการ

$$\text{ความเร็วเฉลี่ยในการเคลื่อนที่} = \frac{\text{ระยะที่หุ่นยนต์เคลื่อนที่}}{\text{เวลาที่ใช้ในการเคลื่อนที่}} \qquad (4.1)$$

ตารางที่ 4.1 ผลการทดลองการวัดความเร็วเฉลี่ยในการเคลื่อนที่ของหุ่นยนต์แบบขนาน

ระยะที่หุ่นยนต์เคลื่อนที่ (mm)	เวลาที่ใช้ในการเคลื่อนที่ (วินาที)	ความเร็วเฉลี่ยในการเคลื่อนที่ (mm/วินาที)
160 (ตามแกน x)	5.1	31.37
160 (ตามแกน y)	5.82	27.49
226 (ตามแกน x และแกน y)	8.1	27.90

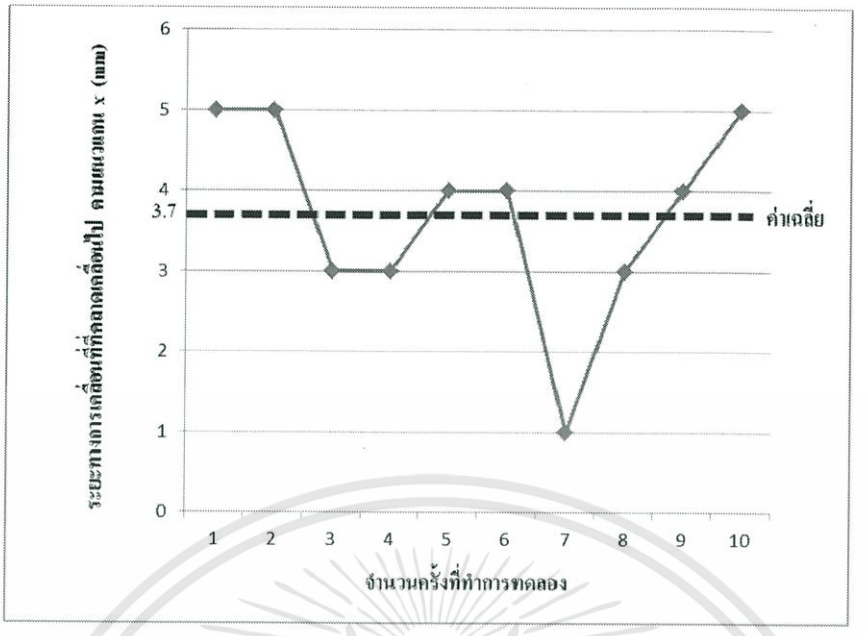
4.4 ผลการทดลองโดยการเคลื่อนที่ทีละตำแหน่ง

เพื่อทดสอบความแม่นยำในการเคลื่อนที่ของต้นแบบหุ่นยนต์แบบขนาน ได้มีการออกแบบการทดลองโดยให้หุ่นยนต์เคลื่อนที่ไปตามตำแหน่งที่ต้องการ เริ่มการทดสอบด้วยการรับค่าพิกัดตำแหน่งที่ต้องการให้หุ่นยนต์เคลื่อนที่ไปทีละตำแหน่งเป็นระยะทาง 50 มิลลิเมตร ตามแนวแกน xและแกนyโดยกำหนดค่า z คงที่จำนวน 10ครั้งแล้วทำการวัดระยะทางที่คลาดเคลื่อนไป จากนั้นนำค่าที่ได้ไปสร้างกราฟเส้นความสัมพันธ์ระหว่างจำนวนครั้งที่ทำการทดลองและระยะการเคลื่อนที่ที่คลาดเคลื่อนตามแนวแกน x ดังแสดงในรูปที่ 4.5 และกราฟเส้นความสัมพันธ์ระหว่างจำนวนครั้งที่ทำการทดลองและระยะการเคลื่อนที่ที่คลาดเคลื่อนตามแนวแกนyดังแสดงในรูปที่ 4.6 ซึ่งผลการทดลองแสดงตามตารางที่ 4.2 และตารางที่ 4.3

ระยะที่คลาดเคลื่อนตามแนวแกน x

ตารางที่ 4.2 ผลการทดลองตามแนวแกน x

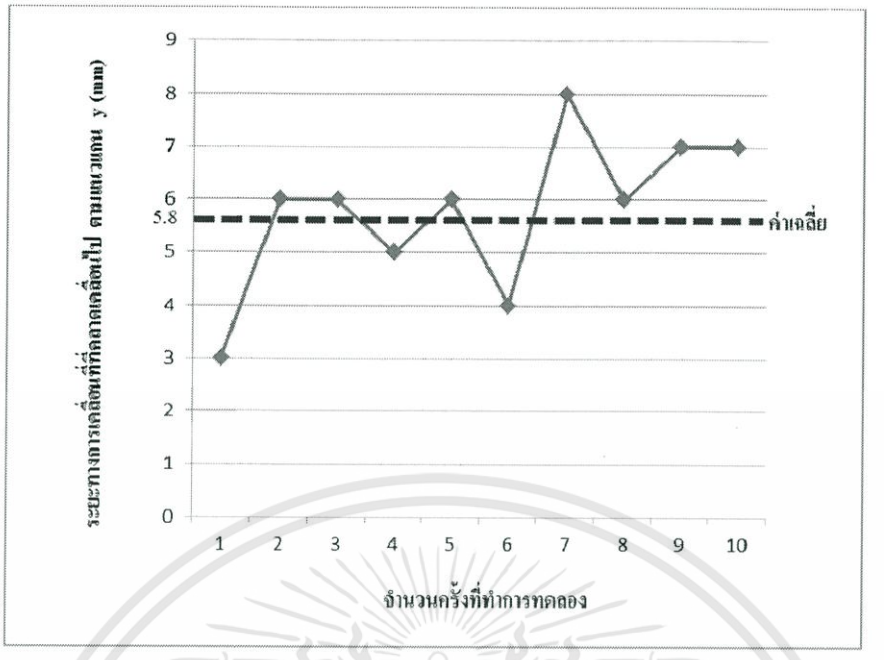
ครั้งที่ทำการทดลอง	ระยะทางการเคลื่อนที่ ที่คลาดเคลื่อนไปตามแนวแกน x (mm)
1	5
2	5
3	3
4	3
5	4
6	4
7	1
8	3
9	4
10	5
ค่าเฉลี่ย	3.7



รูปที่ 4.5 กราฟเส้นความสัมพันธ์ระหว่างจำนวนครั้งทำการทดลองและระยะการเคลื่อนที่ที่คลาดเคลื่อนตามแนวแกน x ระยะที่คลาดเคลื่อนตามแนวแกน y

ตารางที่ 4.3 ผลการทดลองตามแนวแกน y

ครั้งที่ทำการทดลอง	ระยะทางการเคลื่อนที่ ที่คลาดเคลื่อนไปตามแนวแกน y (mm)
1	3
2	6
3	6
4	5
5	6
6	4
7	8
8	6
9	7
10	7
ค่าเฉลี่ย	5.8



รูปที่ 4.6 กราฟเส้นความสัมพันธ์ระหว่างจำนวนครั้งทำการทดลองและระยะการเคลื่อนที่ที่คลาดเคลื่อนตามแนวแกน y

4.5 ผลการทดลองโดยการเคลื่อนที่ทีละหลายตำแหน่ง

เพื่อทดสอบความแม่นยำในการเคลื่อนที่ของต้นแบบหุ่นยนต์แบบขนาน ได้มีการออกแบบการทดลองโดยให้หุ่นยนต์เคลื่อนที่ไปตามตำแหน่งที่ต้องการทีละหลายตำแหน่ง และกำหนดพิกัดให้หุ่นยนต์เคลื่อนที่เป็นรูปต่างๆ เช่น รูปวงกลม รูปสามเหลี่ยม และรูปสี่เหลี่ยม เป็นต้นจากนั้นทำการวัดระยะทางที่คลาดเคลื่อนไปจากขนาดรูปที่กำหนดไว้ ตามแนวแกน x และแกน y แต่ละจุดแล้วนำมาหาค่าเฉลี่ย จากนั้นนำแล้วนำค่าที่ได้มาคำนวณร้อยละความผิดพลาด (Percentage Error) โดยทดลองที่ตำแหน่งแกน z เท่ากับ -447 และ -500 mm

$$\text{ร้อยละความผิดพลาด} = \frac{\left|\text{ค่าตำแหน่งที่ได้จากการทดลอง} - \text{ค่าตำแหน่งที่กำหนด}\right|}{\text{ค่าตำแหน่งที่กำหนด}} \times 100 \qquad (4.2)$$

ตารางที่ 4.4 ตารางเปรียบเทียบความคลาดเคลื่อนของรูปสามเหลี่ยมที่แกน z = -447 mm

พิกัดที่กำหนด	พิกัดจากการทดลอง	ความคลาดเคลื่อนแกน x	ความคลาดเคลื่อนแกน y
A(0,45)	A(0,49)	0 %	8.88 %
B(45,0)	B(48,0)	6.67 %	0 %
C(-45,0)	C(-47,0)	4.44 %	0 %
ค่าเฉลี่ยความคลาดเคลื่อน		3.70 %	2.96 %

ตารางที่ 4.5 ตารางเปรียบเทียบความคลาดเคลื่อนของรูปสี่เหลี่ยมที่แกน z = -447 mm

พิกัดที่กำหนด	พิกัดจากการทดลอง	ความคลาดเคลื่อนแกน x	ความคลาดเคลื่อนแกน y
A(35,35)	A(38,42)	8.57 %	20.00 %
B(35,-35)	B(38,-44)	8.57 %	25.71 %
C(-35,-35)	C(-41,-40)	17.14 %	14.29 %
D(-35,35)	D(-36,41)	2.86 %	17.14 %
ค่าเฉลี่ยความคลาดเคลื่อน		9.29 %	19.29 %

ตารางที่ 4.6 ตารางเปรียบเทียบความคลาดเคลื่อนของรูปวงกลมที่แกน z = -447 mm

พิกัดที่กำหนด	พิกัดจากการทดลอง	ความคลาดเคลื่อนแกน x	ความคลาดเคลื่อนแกน y
A(0,40)	A(0,49)	0 %	22.50 %
B(40,0)	B(53,0)	32.5 %	0 %
C(0,-40)	C(0,-50)	0 %	25.00 %
D(-40,0)	D(-48,0)	20.00 %	0 %
ค่าเฉลี่ยความคลาดเคลื่อน		13.13 %	11.88 %

ตารางที่ 4.7 ตารางเปรียบเทียบความคลาดเคลื่อนของรูปสามเหลี่ยมที่แกน z = -500 mm

พิกัดที่กำหนด	พิกัดจากการทดลอง	ความคลาดเคลื่อนแกน x	ความคลาดเคลื่อนแกน y
A(0,45)	A(0,47)	0 %	4.44 %
B(45,0)	B(46,0)	2.22 %	0 %
C(-45,0)	C(-44,0)	2.22 %	0 %
ค่าเฉลี่ยความคลาดเคลื่อน		1.48 %	1.48 %

ตารางที่ 4.8 ตารางเปรียบเทียบความคลาดเคลื่อนของรูปสี่เหลี่ยมที่แกน z = -500 mm

พิกัดที่กำหนด	พิกัดจากการทดลอง	ความคลาดเคลื่อนแกน x	ความคลาดเคลื่อนแกน y
A(35,35)	A(42,41)	20.00 %	17.14 %
B(35,-35)	B(40,-41)	14.29 %	17.14 %
C(-35,-35)	C(-38,-39)	8.57 %	11.43 %
D(-35,35)	D(-38,39)	8.57 %	11.43 %
ค่าเฉลี่ยความคลาดเคลื่อน		12.86 %	14.29 %

ตารางที่ 4.9 ตารางเปรียบเทียบความคลาดเคลื่อนของรูปวงกลมที่แกน z = -500 mm

พิกัดที่กำหนด	พิกัดจากการทดลอง	ความคลาดเคลื่อนแกน x	ความคลาดเคลื่อนแกน y
A(0,40)	A(0,48)	0 %	20.00 %
B(40,0)	B(46,0)	15.00 %	0 %
C(0,-40)	C(0,-44)	0 %	10.00 %
D(-40,0)	D(-44,0)	10.00 %	0 %
ค่าเฉลี่ยความคลาดเคลื่อน		6.25 %	7.50 %

ตารางที่ 4.10 ตารางเปรียบเทียบความคลาดเคลื่อนของรูปลักษณะต่างๆ ที่ค่า z ต่างกัน

รูปร่างในลักษณะต่างๆ	ค่าเฉลี่ยความคลาดเคลื่อน			
	Z= -447		Z= -500	
	แกน x	แกน y	แกน x	แกน y
รูปวงกลม	13.13 %	11.88 %	6.25 %	7.50 %
รูปสามเหลี่ยม	3.70 %	2.96 %	1.48 %	1.48 %
รูปสี่เหลี่ยม	9.29 %	19.29 %	12.86 %	14.29 %

(a) (b) (c)

รูปที่ 4.7 รูปร่างที่หุ่นยนต์เคลื่อนที่ (a) วงกลม (b) สามเหลี่ยม (c) สี่เหลี่ยม

จากรูปที่ 4.7 แสดงรูปร่างต่างๆ ที่หุ่นยนต์ทำการเคลื่อนที่ได้แก่ รูปวงกลม รูปสามเหลี่ยมและรูปสี่เหลี่ยมซึ่งเกิดจากการกำหนดพิกัดตำแหน่งให้หุ่นยนต์เคลื่อนที่ พบว่าหุ่นยนต์สามารถเคลื่อนที่ไปยังตำแหน่งที่กำหนดและเป็นไปตามรูปที่ต้องการแต่ก็ยังคงเกิดความคลาดเคลื่อนในตำแหน่งอยู่บ้างทำให้รูปร่างต่างๆ มีความผิดพลาดไปจากรูปที่ต้องการ

4.6 ผลการทดลองความสามารถในการทำซ้ำ (Repeatability)

ความสามารถในการทำซ้ำ(Repeatability) แสดงถึงระดับความถูกต้อง เที่ยงตรง ใกล้เคียงกันของผลการวัดที่ได้จากการวัดหลาย ๆ ครั้งในช่วงเวลาใกล้เคียงกัน โดยการวัดทั้งหมดต้องอยู่ภายใต้วิธีการวัด ผู้ทำการวัด และสภาวะแวดล้อมเดียวกัน โดยออกแบบการทดลองให้หุ่นยนต์มีการเคลื่อนที่ซ้ำในพิกัดเดิมจำนวน 3 ครั้ง ในรูปต่างๆ คือ วงกลม รูปสามเหลี่ยม รูปสี่เหลี่ยม แล้วทำการวัดระยะคลาดเคลื่อนสูงสุดที่เกิดขึ้น ซึ่งผลการทดลองที่ได้เป็นไปตามตารางที่ 4.11

ตารางที่ 4.11 ตารางความคลาดเคลื่อนสูงสุดของรูปลักษณะต่างๆ

รูปร่างในลักษณะต่างๆ	ระยะคลาดเคลื่อนสูงสุด (mm)
รูปวงกลม	4
รูปสามเหลี่ยม	2
รูปสี่เหลี่ยม	3

(a) (b) (c)

รูปที่ 4.8 ความสามารถในการทำซ้ำของหุ่นยนต์ในรูปร่างต่างๆ (a) วงกลม (b) สามเหลี่ยม (c) สี่เหลี่ยม

จากรูปที่ 4.8 แสดงความสามารถในการทำซ้ำของหุ่นยนต์ ซึ่งพบว่าหุ่นยนต์สามารถเคลื่อนที่ซ้ำตำแหน่งเดิมได้อย่างถูกต้อง แต่ก็ยังเกิดระยะคลาดเคลื่อนไปจากเดิมซึ่งระยะคลาดเคลื่อนได้แสดงดังตารางที่ 4.11

บทที่ 5
สรุปและวิเคราะห์ผลการดำเนินงาน

5.1 สรุปผลการดำเนินงาน

คณะผู้จัดทำโครงงานได้ดำเนินการออกแบบและพัฒนาหุ่นยนต์แบบสามแกน (Delta Robot) โดยเริ่มจากการสร้างส่วนประกอบซึ่งเป็นโครงสร้างของหุ่นยนต์อันได้แก่ 1).โครงหุ่นยนต์โดยใช้อลูมิเนียมโปรไฟล์ 2).แขนหุ่นยนต์ทั้งแขนบนและแขนล่าง 3).ฐานยึดจับด้านล่าง โดยได้ทำการออกแบบส่วนประกอบต่างๆ ของหุ่นยนต์โดยใช้โปรแกรม Solid Works และทำการประกอบต้นแบบหุ่นยนต์ขึ้น จากนั้นได้ทำการออกแบบและสร้างระบบควบคุมการเคลื่อนที่ของหุ่นยนต์ โดยใช้ไมโครคอนโทรลเลอร์ในการควบคุมทำงานร่วมกับเซนเซอร์แบบรีดสวิตช์ (Reed Switch) และสร้างสัญญาณขับส่งไปยังส่วนขับเคลื่อนมอเตอร์ รวมทั้งพัฒนาโปรแกรมบน Visual C# สำหรับควบคุมการทำงานของหุ่นยนต์แบบขนานเพื่อคำนวณหาค่าจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์แบบสเต็ปปิ้ง (Stepping Motor)

ในการดำเนินงานแบ่งออกเป็น 3 ส่วนหลักๆ ได้แก่ ส่วนโครงสร้าง ส่วนโปรแกรม และส่วนวงจรขับเคลื่อนมอเตอร์ในส่วนโครงสร้างนั้นได้ใช้อลูมิเนียมโปรไฟล์ขนาด 30X30 mm เป็นโครงของหุ่นยนต์ ใช้อลูมิเนียมความยาว 100 mm เป็นแขนบนจำนวน 3 แขน ใช้คาร์บอนไฟเบอร์ขนาดความยาว 505 mm เป็นแขนล่างจำนวน 3 แขน และใช้แผ่นอลูมิเนียมเป็นฐานยึดจับด้านล่าง โดยในโครงงานนี้ได้ใช้มอเตอร์แบบสเต็ปปิ้ง (Stepping Motor) จำนวน 3 ตัว ในการขับเคลื่อนแขนของต้นแบบหุ่นยนต์เพื่อให้หุ่นยนต์สามารถเคลื่อนที่ได้ทั้งสามแกน (แกน x,y,z) และในส่วนของวงจรขับเคลื่อนใช้ไอซี L298 ในการขับเคลื่อนมอเตอร์แบบสเต็ปทั้งสามตัว

การทดสอบการทำงานของต้นแบบหุ่นยนต์แบบสามแกนที่ได้สร้างขึ้น พบว่าหุ่นยนต์เคลื่อนที่ด้วยความเร็วเฉลี่ยอยู่ในช่วง 27.49-31.37 mm/วินาที เมื่อกำหนดพิกัดตำแหน่งให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งต่างๆ หุ่นยนต์สามารถเคลื่อนที่ได้ตามตำแหน่งที่กำหนดได้ โดยมีระยะการเคลื่อนที่ที่คลาดเคลื่อนเฉลี่ยตามแนวแกน x เท่ากับ 3.7 mm และระยะการเคลื่อนที่ที่คลาดเคลื่อนเฉลี่ยตามแนวแกน y เท่ากับ 5.8 mm รวมทั้งสามารถวาดเป็นรูปต่างๆ ตามที่ผู้ใช้งานป้อนข้อมูลให้ได้เช่นรูปสี่เหลี่ยม สามเหลี่ยม หรือวงกลม เป็นต้น แต่ยังมีความคลาดเคลื่อนของตำแหน่งในการเคลื่อนที่ของหุ่นยนต์อยู่บ้าง เช่น หุ่นยนต์ยังวาดเส้นได้ไม่เป็นเส้นตรงตามต้องการ หุ่นยนต์ยังไปตามตำแหน่งที่ต้องการได้ไม่แม่นยำเท่าที่ควร จากการทดลองที่ตำแหน่งแกน z เท่ากับ -447 mm พบว่าตำแหน่งของรูปสามเหลี่ยมมีความคลาดเคลื่อนตามแนวแกน x เท่ากับ 3.70 % และคลาดเคลื่อนตามแนวแกน y เท่ากับ 2.96 % ตำแหน่งของรูปสี่เหลี่ยมมีความคลาดเคลื่อนตามแนวแกน x เท่ากับ 9.29 % และคลาดเคลื่อนตามแนวแกน y เท่ากับ 19.29 % ตำแหน่งของรูปวงกลมมีความคลาดเคลื่อนตามแนวแกน x เท่ากับ 13.13 % และคลาดเคลื่อนตามแนวแกน y เท่ากับ 11.88 % จากการทดลองที่ตำแหน่งแกน z เท่ากับ -500 mm พบว่าตำแหน่งของรูปสามเหลี่ยมมีความคลาดเคลื่อนตามแนวแกน x เท่ากับ 1.48 % และคลาดเคลื่อนตามแนวแกน y เท่ากับ 1.48 % ตำแหน่งของรูปสี่เหลี่ยมมีความคลาดเคลื่อนตามแนวแกน x เท่ากับ 12.86 % และคลาดเคลื่อนตามแนวแกน y เท่ากับ 14.29 % ตำแหน่งของรูปวงกลมมีความคลาดเคลื่อนตามแนวแกน x เท่ากับ 6.25 % และคลาดเคลื่อนตามแนวแกน y เท่ากับ 7.50 %

โดยจากผลการทดลองพบว่าที่ตำแหน่งแกน z เท่ากับ -500 mm มีการคลาดเคลื่อนของรูปต่างๆน้อยกว่าที่ตำแหน่งแกน z เท่ากับ -447 mm เป็นผลมาจากพื้นที่การทำงานต่างกัน นอกจากนั้นในการทดสอบความสามารถในการทำซ้ำผลที่ได้คือหุ่นยนต์สามารถเคลื่อนที่ซ้ำตำแหน่งเดิมได้อย่างถูกต้อง แต่ยังเกิดระยะคลาดเคลื่อนสูงสุดเท่ากับ 2-4 mm

5.2 ปัญหาและแนวทางการแก้ไข

จากการดำเนินงานทั้งหมดที่ผ่านมานั้น คณะผู้จัดทำได้ประสบปัญหาหลายอย่างประการแรกคือปัญหาความไม่สมมาตรของแขนแต่ละแขนของหุ่นยนต์ ซึ่งเกิดขึ้นจากหลายสาเหตุ เช่น การวางตำแหน่งของมอเตอร์ทั้งสามตัวที่ไม่สมมาตร เนื่องจากมอเตอร์ทั้งสามตัวต้องวางในลักษณะที่ทำมุมกัน 120 องศา หากมีการวางตำแหน่งของมอเตอร์คลาดเคลื่อนไปจะทำให้แขนทั้งสามของหุ่นยนต์ไม่สมมาตร ซึ่งเมื่อเกิดปัญหาขึ้นก็ได้ทำการปรับแก้ตำแหน่งของมอเตอร์ให้อยู่ในตำแหน่งที่เหมาะสม ทำให้แขนของหุ่นยนต์มีความสมมาตรมากขึ้น อีกสาเหตุหนึ่งที่ทำให้แขนของหุ่นยนต์ทั้งสามไม่สมมาตร คือตำแหน่งการเจาะยึดระหว่างแขนบนและแขนล่าง หากมีการเจาะระยะคลาดเคลื่อนไปหรือไม่เท่ากันทั้งสามแขนก็เป็นสาเหตุทำให้เกิดความไม่สมมาตรขึ้นได้เช่นกัน คณะผู้จัดทำจึงได้ทำการปรับแก้ไขระยะการเจาะยึดของแขนให้มีความสมมาตรมากขึ้น นอกจากนี้ยังพบปัญหาที่เกิดจากการออกแบบได้แก่ ระยะ Backlash ซึ่งเกิดจากความไม่พอดีของชิ้นส่วนต่างๆ ของหุ่นยนต์เป็นผลทำให้เกิดระยะคลาดเคลื่อน สามารถลดปัญหานี้ลงได้โดยใช้สปริงช่วยทำให้กลไกที่ยังหลวมอยู่ชิดกันมากขึ้น ในเรื่องของความร้อนที่เกิดขึ้นที่ชุดขับเคลื่อนมอเตอร์ซึ่งเป็นความสูญเสียพลังงานในการขับเคลื่อนมอเตอร์ โดยแนวทางการแก้ปัญหานี้สามารถใช้ชุดขับสำเร็จรูปแทนได้แต่จะมีราคาค่อนข้างสูง และอีกประการหนึ่งคือความเร็วในการทำงานของหุ่นยนต์ค่อนข้างต่ำถ้าหากนำไปใช้งานจริงควรมีการปรับความเร็วในการทำงานให้มากกว่านี้และประการสุดท้ายที่เป็นปัญหาที่เกิดขึ้นกับคณะผู้จัดทำคือการพัฒนาโปรแกรมที่ใช้ในการประมวลผลและคำนวณหาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์แต่ละตัวนั้นมีความซับซ้อนเป็นอย่างมากทำให้เกิดปัญหาและส่งผลต่อการควบคุมการทำงานของมอเตอร์ ซึ่งใช้ระยะเวลานานในการแก้ไขโปรแกรมให้ทำงานได้ตามความต้องการ

5.3 ข้อเสนอแนะแนวและทางการพัฒนาต่อ

เนื่องจากหุ่นยนต์ที่ได้ทำการสร้างขึ้นนี้เป็นหุ่นยนต์ต้นแบบ ดังนั้นจึงยังมีข้อจำกัดและข้อผิดพลาดในการทำงานอยู่บ้าง เช่น ข้อจำกัดในเรื่องของน้ำหนักของโหลด ซึ่งสามารถแก้ไขได้โดยการเพิ่มพิกัดกำลังของมอเตอร์ไฟฟ้าให้สามารถรับน้ำหนักของโหลดได้มากขึ้น อีกทั้งยังสามารถลดน้ำหนักของแขนบนและแขนล่างของหุ่นยนต์ได้โดยการเลือกใช้วัสดุอุปกรณ์ที่มีน้ำหนักเบาและมีความแข็งแรง เป็นต้น ทั้งนี้ข้อผิดพลาดของตำแหน่งในการเคลื่อนที่ของหุ่นยนต์นั้นสามารถแก้ไขได้โดยใช้การควบคุมการเคลื่อนที่ของหุ่นยนต์เป็นแบบปิด (Close Loop) ซึ่งจะสามารถทำให้หุ่นยนต์มีความผิดพลาดของตำแหน่งน้อยลง นอกจากนี้คณะผู้จัดทำหวังว่าโครงงานนี้จะเป็นจุดเริ่มต้นของการพัฒนาในการประยุกต์ใช้หุ่นยนต์แบบขนานสามแกนในอุตสาหกรรมต่างๆ ของประเทศในอนาคต

บรรณานุกรม

[1] คลับอาร์โอ 95(clubro95),2553. พาวเวอร์ซัพพลาย (Power Supply). ค้นข้อมูล: 20 มกราคม 2557 จาก http://group.wunjun.com/clubro95/topic/164888-3796

[2] ดอนสัน ปงผาล, ทิพวัลย์ คำน้ำนอง ,2550. ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน. พิมพ์ครั้งที่ 1. กรุงเทพ: สำนักพิมพ์ ส.ส.ท.

[3] นายตฤณ ตู้จินดา นางสาวธัญชนก หุตังคบดีและนางสาวสุวพิชญ์ พีรดิศภีมพศ, หุ่นยนต์แบบสามแกนควบคุมด้วยเทคนิคการประมวลผลภาพ. ปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2555.

[4] บัญชา ปะสีละเตสัง, 2552. การพัฒนาแอปพลิเคชั่น Visual C# 2008. กรุงเทพฯ : สำนักพิมพ์ ซีเอ็ดยูเคชั่น ศูนย์รวมความรู้วิศวกรรม.

[5] บริษัท เอ็มเค โพลีเอสเตอร์เรซิ่น จำกัด, ใยคาร์บอน และใยเคฟลาร์.ค้นข้อมูล: 20 มกราคม 2557 จาก http://www.muikwang.com/index.php?lay=show&ac=article&Id=539148748

[6] มาร์ค 31055154, 2553.การใช้งานไมโครคอนโทรลเลอร์ MCS-51. ค้นข้อมูล: 25 ม.ค. 2557 จาก http://www.reocities.com/p_pirat/mcs51.htm

[7] สถาพร ลักษณะเจริญ, วิศวกรรมหุ่นยนต์ (Robotics Engineering), กรุงเทพฯ: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น).

[8] ศรีธร,2553. RS-232 คืออะไรและทำงานอย่างไร.ค้นข้อมูล: 25 ม.ค. 2557 จาก http://movableprojecter.wordpress.com/2010/07/22/32/

[9] อนันต์ สืบสำราญ,คมสัน มีสมบูรณ์และ กิตติศักดิ์ เพ็งสุข,การออกแบบและควบคุมหุ่นยนต์แบบ 4 ขา . วารสารวิชาการพระจอมเกล้าพระนครเหนือ ปีที่ 20 ฉบับที่ 2: 249-255

[10] อดิศักดิ์ ชัยนาวงษ์, 2545.สเตปปิ้งมอเตอร์ (Stepping Motor). ค้นข้อมูล: 31 ส.ค. 2556 จาก http://www.adisak51.com/page22.html

[11] สารานุกรมวิทยาศาสตร์ ของสมาคมวิทยาศาสตร์แห่งประเทศไทย,2553. อลูมิเนียม น้ำหนักเบาแต่แข็งแกร่ง.ค้นข้อมูล:31 ส.ค. 2556 จาก http://www.rmutphysics.com/charud/specialnews/5/aluminum/aluminum1.htm

[12] อภิชาติ ภู่พลับ, เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic, กรุงเทพฯ

[13] A. Asbeck, S. Dastoor, A. Parness, and L.Fullerton., 2009. Climbing_rough vertical surfaces with hierarchical directional adhesion. IEEE International Conference on Robotics and Automation. : 2675-2680.

[14] lian Bonev,2001. Delta Parallel Robot. Search Date: 8Th Jan 2014 From http://www.parallemic.org/Reviews/Review002.html

[15] Design World Staff, 2010. Reed Switch. Search Date: 5th Jan 2014 From http://www.designworldonline.com/choose-your-best-pneumatic-cylinder-sensor-here/#_

[16]John J. Craig, 2000. Introduction to Robotics Mechanics and Control, United State of America: Pearson Prentice Hall.

[17] Maxim Zavatsky,2010. Delta kinematics Robot. Search Date: 31th Aug 2013 From http://www.legomindstormsrobots.com/lego-minstorms/delta-kinematics-robot/

[18] M. Okada and S. Kino, 2008. Torque Transmission Mechanism with Nonlinear Passive Stiffness using Mechanical Singularity. IEEE International Conference on Robotics and Automation. : 1735-1740.

[19] P.Merlet,2006. Parallel Robots. The Netherlands: Springer

ภาคผนวก

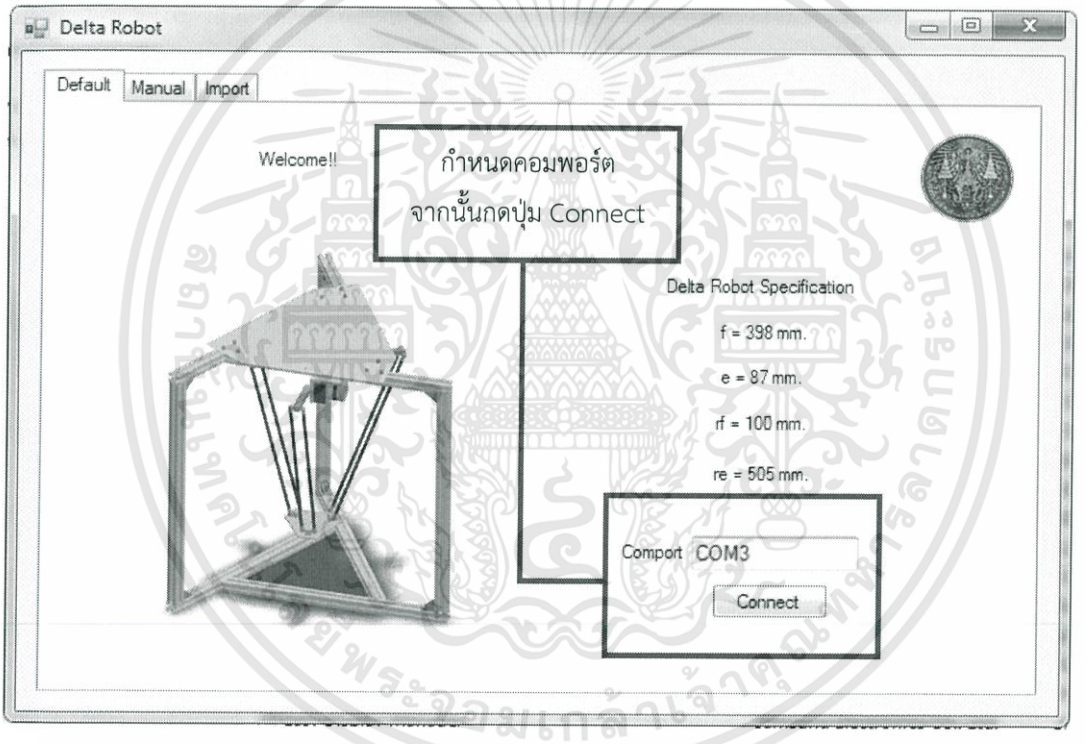
ภาคผนวก

1. คู่มือการใช้งานโปรแกรม

โปรแกรมที่พัฒนาขึ้นนี้สามารถเลือกรูปแบบการทำงานได้ 2 รูปแบบ ดังนี้

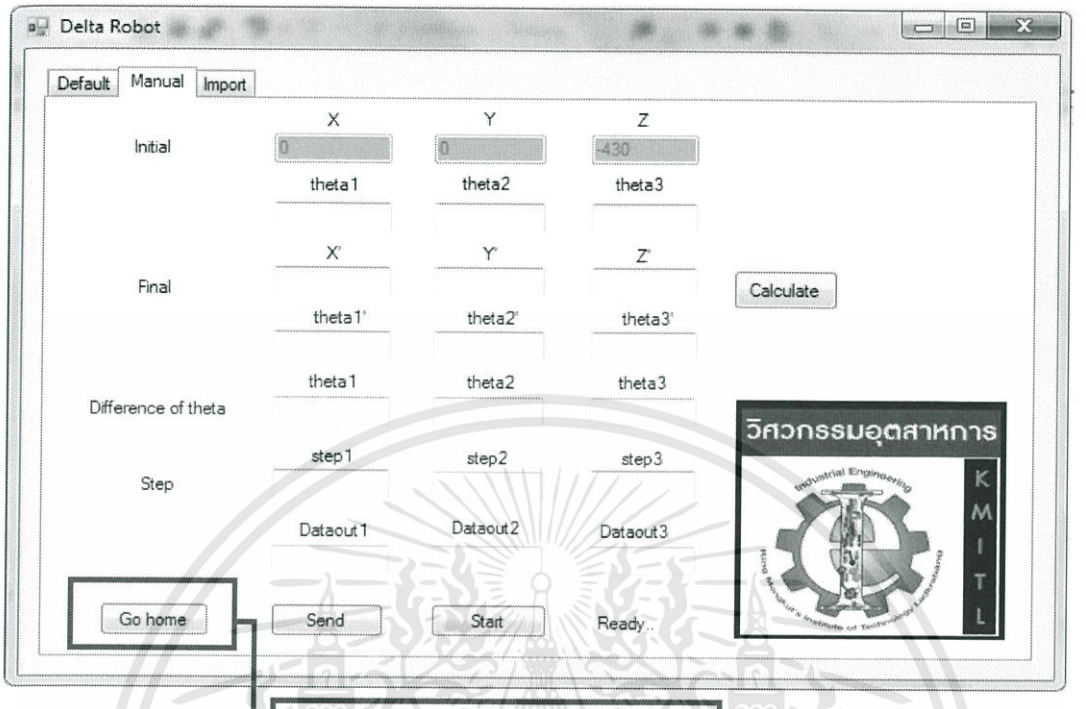
1. การใช้งานแบบ Manual เป็นการใช้งานโดยรับค่าพิกัดตำแหน่งที่ต้องการให้หุ่นยนต์เคลื่อนที่ไปทีละตำแหน่ง
2. การใช้งานโดยสามารถ Import ไฟล์ในรูปแบบ .txt เพื่อให้หุ่นยนต์เคลื่อนที่ในรูปแบบต่างๆ ได้อย่างอัตโนมัติ

ขั้นตอนการใช้งาน

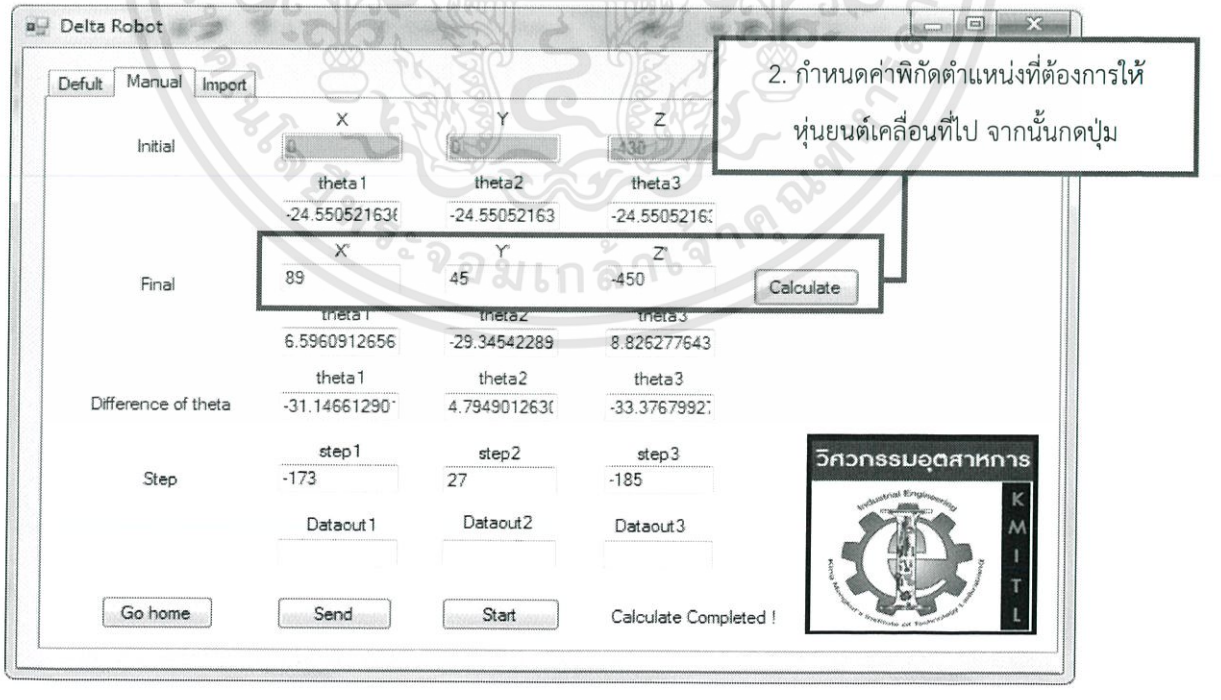


รูปที่ ผ1 ขั้นตอนในการกำหนดคอมพอร์ตเพื่อเชื่อมต่อกับระบบควบคุม

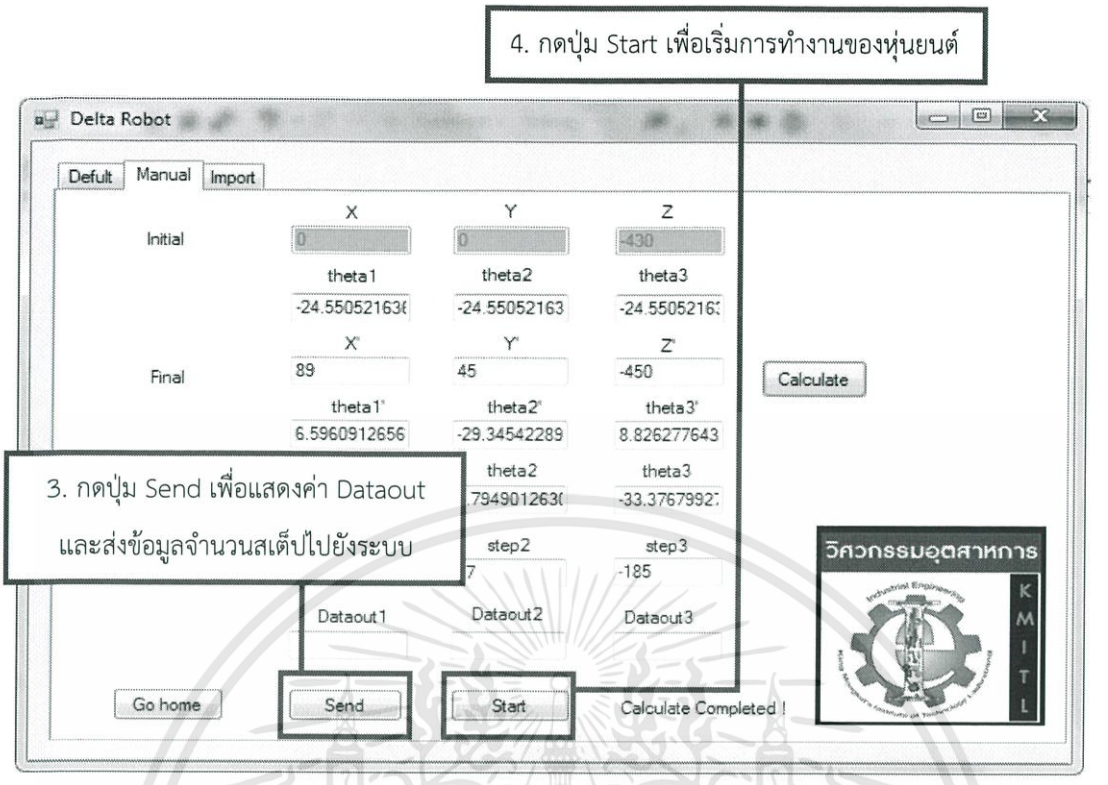
- การใช้งานแบบ Manual



รูปที่ ผ2 ขั้นตอนในการตั้งค่าเริ่มต้นของหุ่นยนต์

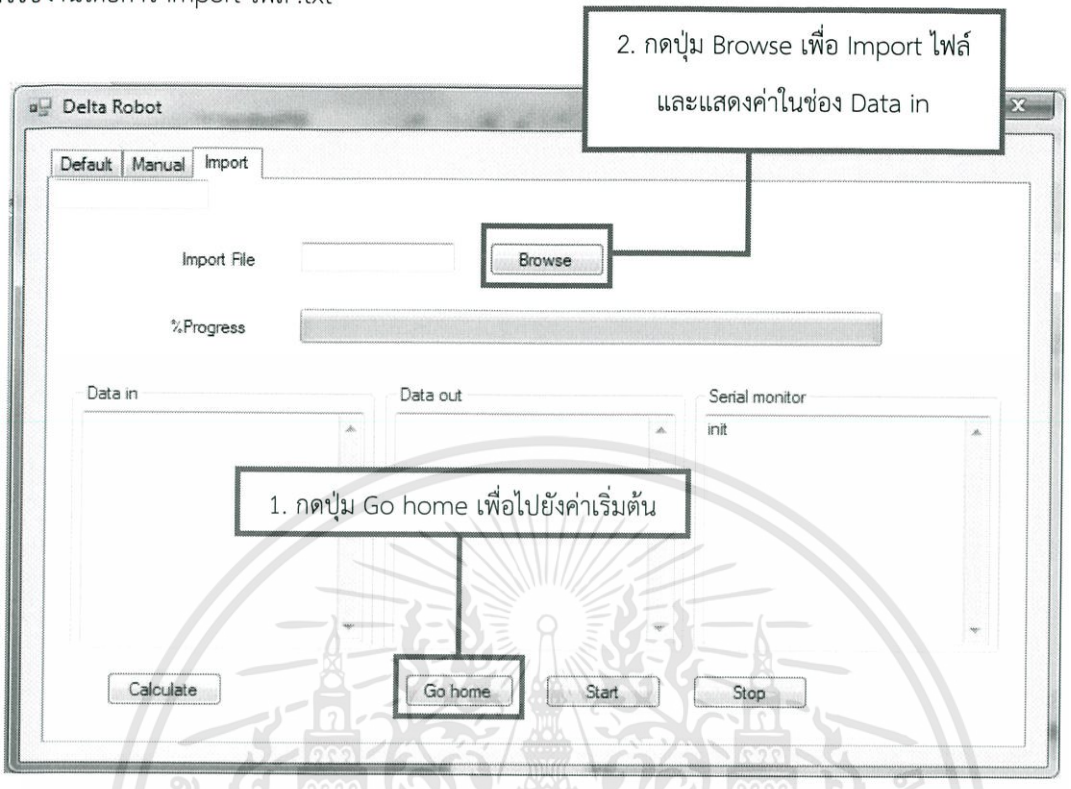


รูปที่ ผ3 ขั้นตอนในการคำนวณหาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์



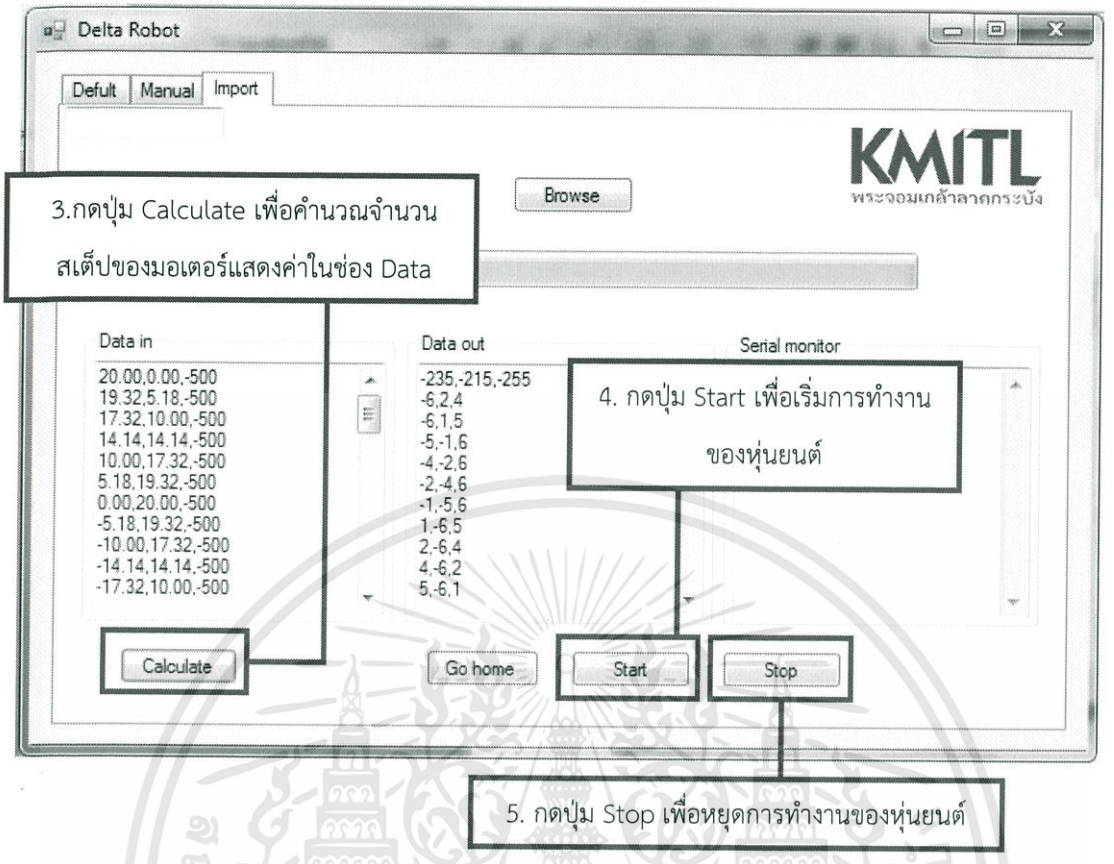
รูปที่ ผ4 ขั้นตอนการส่งข้อมูลจำนวนสเต็ปไปยังระบบควบคุมและเริ่มการทำงาน

- การใช้งานโดยการ Import ไฟล์ .txt



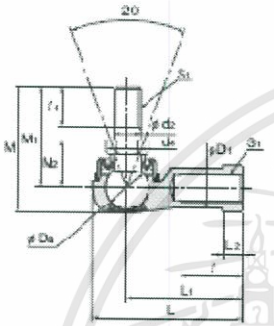
รูปที่ ผ5 ขั้นตอนการใช้งานแบบ Import ไฟล์

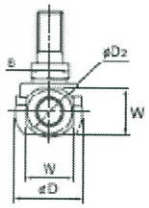
รูปที่ ผ6 ภายหลังกดปุ่ม Browse จะปรากฏหน้าต่างเพื่อเลือกไฟล์ที่ต้องการ



รูปที่ ผ7 ขั้นตอนในการคำนวณหาจำนวนสเต็ปการเคลื่อนที่ของมอเตอร์การเริ่มและหยุดการทำงานของหุ่นยนต์

Model AL





Unit: mm

Model No.	Outer dimensions Length L	Diameter D	Height M	Threaded S1 JIS Class 2	Holder dimensions L1	ℓ	L2	D	D1	W d +0.3	Ball shank dimensions d1 h9	M	M2 ±0.3	Z1	Hexagon B d −0.3	d	Ball diameter Da	Permissible tilt angles 2θ°	Applied static load C0 N	Yield-point strength P1 N	Mass g
AL 4D	24.5	13	20	M4×0.7	16	8	4	7.5	9.5	8	4	15	7	6	7	8.1	7.938	40	4510	1370	7
AL 6D	34.5	15	26.7	M6×0.8	27	15	4	9	12	10	6	21	10	8	8	9.2	9.525	40	6470	2250	12
AL 8D	38.5	17	32.6	M8×1	30	16	5	10	13	11	6	26	11	11	10	11.6	11.112	40	9900	3920	18
AL 8U	46	20	38.6	M8×1.25	36	19	6	10	16	14	8	31	14	12	12	13.8	12.7	40	12500	6570	32
AL 10D	66	26	46.3	M10×1.25	48	23	7	16.6	19	17	10	37	17	15	14	16.2	15.875	40	18300	11300	66
AL 10BD	56	26	52.3	M10×1.5	43	23	7	15.5	19	17	10	43	17	21	14	16.2	15.875	40	18300	11300	68

[Material]

Holder : Al alloy (see A-925)
Ball shank : Lightly Carburized Carbon Steel Ball 653 Hv or higher
Shank 935C (20 to 28 HRC)
Chromate treatment
Boot : NER special synthetic rubber

[Spherical Clearance]

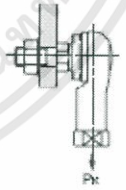
Perpendicular to the axis : 0.02 to 0.26mm
Axial direction : 0.5mm or less

[Tolerance of the Mating Hole of the Ball Shank]

H10 is recommended.

[Yield-Point Strength]

It indicates the strength in the direction shown in the figure below.



[Lubrication]

Lithium soap group grease No. 2 is contained in the boot and the cap.

[Identification of Left-hand Thread]

If the female threading is left-handed, its identification depends on the marking.

Threaded	Identification Cap marking
Right-hand	—
Left-hand	"L" mark

Model number — Without boot attached — Left-hand thread

AL6 D L

2. Code ที่ใช้พัฒนาโปรแกรมเพื่อควบคุมการทำงานของหุ่นยนต์แบบขนาน

ส่วนที่เป็นสมการ Inverse Kinematics

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Delta_Robot
{
classKinematicsDelta
   {
privatedouble e = 87.0;    // end effector
privatedouble f = 398.0;     // base
privatedouble re = 505.0;
privatedouble rf = 100.0;
privatedouble sqrt3 = Math.Sqrt(3);
privatedouble pi = Math.PI;    // PI
privatedouble sin120 = Math.Sqrt(3) / 2.0;
privatedouble cos120 = -0.5;
privatedouble tan60 = Math.Sqrt(3);
privatedouble sin30 = 0.5;
privatedouble tan30 = 1.0 / Math.Sqrt(3);
privateint delta_calcAngleYZ(double x0, double y0, double z0, outdouble theta)
       {
double y1 = -0.5 * 0.57735 * f; // YF1=-f/2 * tg 30
           y0 -= 0.5 * 0.57735 * e; // y0=y0-0.5*0.57735
*e shift center to edge
// z = a + b*y
double a = (x0 * x0 + y0 * y0 + z0 * z0 + rf * rf - re * re - y1 * y1) / (2.0 * z0);
double b = (y1 - y0) / z0;
// discriminant
double d = -(a + b * y1) * (a + b * y1) + rf * (b * b * rf + rf);
double yj = (y1 - a * b - Math.Sqrt(d)) / (b * b + 1.0); // choosing outer point
double zj = a + b * yj;
           theta = 180.0 * Math.Atan(-zj / (y1 - yj)) / pi + ((yj > y1) ? 180.0 : 0.0);
if (d < 0.0) { return -1; } // non-existing point
```

```
return 0;
        }
// inverse kinematics: (x0, y0, z0) -> (theta1, theta2, theta3)
// returned status: 0=OK, -1=non-existing position
publicint CalcInverse(double x0, double y0, double z0, outdouble theta1, outdouble theta2,
outdouble theta3)
        {
            theta1 = theta2 = theta3 = 0;
int status = delta_calcAngleYZ(x0, y0, z0, out theta1);
if (status == 0) status = delta_calcAngleYZ(x0 * cos120 + y0 * sin120, y0 * cos120 - x0 * sin120, z0, out
theta2);  // rotate coords to +120 deg
if (status == 0) status = delta_calcAngleYZ(x0 * cos120 - y0 * sin120, y0 * cos120 + x0 * sin120, z0, out
theta3);  // rotate coords to -120 deg
return status;
        }

publicint CalStep(double x0, double y0, double z0, double x1, double y1, double z1, outint ost1,
outint ost2, outint ost3)
        {
double th1, th2, th3;
double dth1, dth2, dth3;
double th1_2, th2_2, th3_2;
            ost1 = 0;
            ost2 = 0;
            ost3 = 0;

if (CalcInverse(x0, y0, z0, out th1, out th2, out th3) == -1)
            {
return -1;
            }

if (th1 < -40.0 || th1 > 80.0)
            {
return 1;
            }
if (th2 < -40.0 || th2 > 80.0)
```

```
            {
return 2;
            }
if (th3 < -40.0 || th3 > 80.0)
            {
return 3;
            }

/////////////// x2 y2 z2 /////////////////

if (CalcInverse(x1, y1, z1, out th1_2, out th2_2, out th3_2) == -1)
            {
return -2;
            }

if (th1_2 < -40.0 || th1_2 > 80.0)
            {
return 1;
            }
if (th2_2 < -40.0 || th2_2 > 80.0)
            {
return 2;
            }
if (th3_2 < -40.0 || th3_2 > 80.0)
            {
return 3;
            }

//////////////////////////////////////

            dth1 = th1 - th1_2;// find dev
            dth2 = th2 - th2_2;
            dth3 = th3 - th3_2;

            ost1 = (int)Math.Round(((dth1 * 10.0) / 1.8));
            ost2 = (int)Math.Round(((dth2 * 10.0) / 1.8));
```

```
            ost3 = (int)Math.Round(((dth3 * 10.0) / 1.8));
 return 0;
        }
    }
 }
```

ส่วนที่ใช้ในการออกแบบโปรแกรมควบคุมการทำงาน

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using Delta_Robot; // include Kinematic file

namespace projectdelta
{
publicpartialclassForm1 : Form
   {
privateThread sending_thread;
privatedouble xm, ym, zm;
privatedouble th1, th2, th3;
privatedouble th1_2, th2_2, th3_2;
privatedouble dth1, dth2, dth3;
privatedouble st1, st2, st3;
privatedouble[,] cmd_squ = newdouble[100,3];
privateint[,] ste_squ = newint[100, 3]; //calculated value
privateint rec_f = 0; //recived 'p' flag
privateint i_send;
privateint t_st = 0;
privateint stop_f = 0;
string tmpc;
```

```
string str_tmp;
staticint str_cnt;

KinematicsDelta kinemath = newKinematicsDelta();
staticbyte[] data_send = newbyte[3];


staticclassConstants
        {
publicconstchar CMD_SET_M1STP = 'a';
publicconstchar CMD_SET_M2STP = 'b';
publicconstchar CMD_SET_M3STP = 'c';
        }

public Form1()
        {
            InitializeComponent();
            sending_thread = newThread(send_function);
            sending_thread.Start();
            serial_t.Text += "init";
        }

privatevoid label6_Click(object sender, EventArgs e)
        {

        }

privatevoid RUN_Click(object sender, EventArgs e)
        {
            xm = Convert.ToDouble(inX.Text);
            ym = Convert.ToDouble(inY.Text);
            zm = Convert.ToDouble(inZ.Text);

if (kinemath.CalcInverse(xm, ym, zm, out th1, out th2, out th3) == -1) {
                error_text.Text = "Error";
return;
```

```
            }

            theta1.Text = th1.ToString();
            theta2.Text = th2.ToString();
            theta3.Text = th3.ToString();

 if (th1 < -40.0 || th1 > 80.0) {
                error_text.Text = "Error angle of th1 over limit";
 return;
            }
 if (th2 < -40.0 || th2 > 80.0)
            {
                error_text.Text = "Error angle of th2 over limit";
 return;
            }
 if (th3 < -40.0 || th3 > 80.0)
            {
                error_text.Text = "Error angle of th3 over limit";
 return;
            }

 /////////////// x2 y2 z2 /////////////////

            xm = Convert.ToDouble(inX2.Text);
            ym = Convert.ToDouble(inY2.Text);
            zm = Convert.ToDouble(inZ2.Text);

 if (kinemath.CalcInverse(xm, ym, zm, out th1_2, out th2_2, out th3_2) == -1)
            {
                error_text.Text = "Error";
 return;
            }

            theta4.Text = th1_2.ToString();
            theta5.Text = th2_2.ToString();
            theta6.Text = th3_2.ToString();
```

```
if (th1_2 < -40.0 || th1_2 > 80.0)
        {
            error_text.Text = "Error angle of th1 over limit";
return;
        }
if (th2_2 < -40.0 || th2_2 > 80.0)
        {
            error_text.Text = "Error angle of th2 over limit";
return;
        }
if (th3_2 < -40.0 || th3_2 > 80.0)
        {
            error_text.Text = "Error angle of th3 over limit";
return;
        }

//////////////////////////////////////

        dth1 = th1 - th1_2;// find dev
        dth2 = th2 - th2_2;
        dth3 = th3 - th3_2;

        difftheta1.Text = dth1.ToString();
        difftheta2.Text = dth2.ToString();
        difftheta3.Text = dth3.ToString();

        st1 = (dth1 * 10.0) / 1.8;
        st2 = (dth2 * 10.0) / 1.8;
        st3 = (dth3 * 10.0) / 1.8;

// rounding floating point//////
        st1 = Math.Round(st1);
        st2 = Math.Round(st2);
        st3 = Math.Round(st3);
//////////////////////////////////
```

```
            step1.Text = st1.ToString();
            step2.Text = st2.ToString();
            step3.Text = st3.ToString();

            error_text.Text = "Calculate Completed !";

        }

privatevoid Form1_Load(object sender, EventArgs e)
        {

        }

privatevoid textBox1_TextChanged(object sender, EventArgs e)
        {

        }

privatevoid label12_Click(object sender, EventArgs e)
        {

        }

privatevoid a_TextChanged(object sender, EventArgs e)
        {

        }

privatevoid button1_Click(object sender, EventArgs e)
        {
            inX.Text = inX2.Text;
            inX2.Text = "";
            inY.Text = inY2.Text;
            inY2.Text = "";
            inZ.Text = inZ2.Text;
            inZ2.Text = "";
```

```
            theta1.Text = "";
            theta2.Text = "";
            theta3.Text = "";
            theta4.Text = "";
            theta5.Text = "";
            theta6.Text = "";
            difftheta1.Text = "";
            difftheta2.Text = "";
            difftheta3.Text = "";
            step1.Text = "";
            step2.Text = "";
            step3.Text = "";



            data_send[0] = (byte)'a';
            data_send[1] = (byte)((int)(-st1) >> 8);
            data_send[2] = (byte)((int)(-st1) & 0xFF);
            dataout1.Text = data_send[1] + " " + data_send[2];
//serialPort1.Write(data_send, 0, 3);
            serialPort1.Write(data_send, 0, 1);
            serialPort1.Write(data_send, 1, 1);
            serialPort1.Write(data_send, 2, 1);

            data_send[0] = (byte)'b';
            data_send[1] = (byte)((int)(-st2) >> 8);
            data_send[2] = (byte)((int)(-st2) & 0xFF);
            dataout2.Text = data_send[1] + " " + data_send[2];
//serialPort1.Write(data_send, 0, 3);
            serialPort1.Write(data_send, 0, 1);
            serialPort1.Write(data_send, 1, 1);
            serialPort1.Write(data_send, 2, 1);

            data_send[0] = (byte)'c';
            data_send[1] = (byte)((int)(-st3) >> 8);
            data_send[2] = (byte)((int)(-st3) & 0xFF);
            dataout3.Text = data_send[1] + " " + data_send[2];
```

```
//serialPort1.Write(data_send, 0, 3);
            serialPort1.Write(data_send, 0, 1);
            serialPort1.Write(data_send, 1, 1);
            serialPort1.Write(data_send, 2, 1);
        }

privatevoid textBox1_TextChanged_1(object sender, EventArgs e)
        {

        }

privatevoid connectb_Click(object sender, EventArgs e)
        {
if (serialPort1.IsOpen)
            {
                serialPort1.Close();
                connectb.Text = "Connect";
            }
else {
                serialPort1.PortName = Serialname.Text;
                serialPort1.Open();
                connectb.Text = "Disconnect";
            }
        }

privatevoid button2_Click(object sender, EventArgs e)
        {
            data_send[0] = (byte)'h';

            serialPort1.Write(data_send, 0, 1);
        }

privatevoid button3_Click(object sender, EventArgs e)
        {
            data_send[0] = (byte)'s';
```

```
            serialPort1.Write(data_send, 0, 1);
        }

 privatevoid tabPage2_Click(object sender, EventArgs e)
        {

        }

 privatevoid groupBox1_Enter(object sender, EventArgs e)
        {

        }

 privatevoid button4_Click(object sender, EventArgs e)
        {
            openFileDialog1.ShowDialog();
            filename.Text = openFileDialog1.FileName;
            datain_file.Text = System.IO.File.ReadAllText(filename.Text);
        }

 privatevoid button9_Click(object sender, EventArgs e)
        {
            progressBar1.Value = 0;
            str_tmp = datain_file.Text;
            cmd_squ[0, 0] = 0;
            cmd_squ[0, 1] = 0;
            cmd_squ[0, 2] = -430;
string W_tmp = "";
int j = 0;
            str_cnt = 0;
int e_code;
int ost1,ost2,ost3;

////////////////////////////// read value from file ////////////////////////////

for(int i=0;i<datain_file.TextLength;i++){
```

```
if(str_tmp.ElementAt(i) == ','){
                cmd_squ[str_cnt+1,j] = double.Parse(W_tmp); //convert string to double
                j++;
                W_tmp = "";
            }
elseif (str_tmp.ElementAt(i) == '\n' || i + 1 == datain_file.TextLength)
            {
if (i + 1 == datain_file.TextLength) W_tmp += str_tmp.ElementAt(i);
                cmd_squ[str_cnt+1,j] = double.Parse(W_tmp); //convert string to double
                str_cnt++;
                j=0;
                W_tmp = "";
            }
else{
                W_tmp+=str_tmp.ElementAt(i);
            }
        }

////////////////////////////// calculate each value //////////////////////////////

        dataout_file.Text ="";
for (int i = 1; i <= str_cnt; i++) {

            e_code = kinemath.CalStep(cmd_squ[i-1,0],cmd_squ[i-1,1],cmd_squ[i-
1,2],cmd_squ[i,0],cmd_squ[i,1],cmd_squ[i,2], out ost1, out ost2, out ost3);
if (e_code != 0) {
                dataout_file.Text += "error in line " + i.ToString() + " : " ;
if (e_code == -1) dataout_file.Text += "calculate error ";
elseif (e_code == 1) dataout_file.Text += "Error angle of X limit ";
elseif (e_code == 2) dataout_file.Text += "Error angle of Y limit ";
elseif (e_code == 3) dataout_file.Text += "Error angle of Z limit ";
break;
            }
else{

            ste_squ[i-1, 0] = ost1;
```

```
            ste_squ[i-1, 1] = ost2;
            ste_squ[i-1, 2] = ost3;

            dataout_file.Text += ost1.ToString() + "," + ost2.ToString() + "," + ost3.ToString() +
Environment.NewLine;
            }

        }

////////////////////////////// print out calculated value //////////////////

    }

void send_function() {
while (true)
        {
while(t_st == 0);

            progressBar1.Value = 0;
for (i_send = 0; i_send < str_cnt; i_send++)
            {

this.Invoke(newEventHandler(send_st));
                rec_f = 1;
while (rec_f == 1) //wait 'p' from Rx
                {
if (stop_f == 1)
                    {
                        stop_f = 0;
                        i_send = 1000;
break;
                    }
                }

            }
            t_st = 0;
```

```
        }
    }

 privatevoid send_st(object sender, EventArgs e)
    {
        data_send[0] = (byte)'a';
        data_send[1] = (byte)((int)(-ste_squ[i_send, 0]) >> 8);
        data_send[2] = (byte)((int)(-ste_squ[i_send, 0]) & 0xFF);
        serial_t.Text += " a " + data_send[1] + " " + data_send[2];
 //serialPort1.Write(data_send, 0, 3);
        serialPort1.Write(data_send, 0, 1);
        serialPort1.Write(data_send, 1, 1);
        serialPort1.Write(data_send, 2, 1);

        data_send[0] = (byte)'b';
        data_send[1] = (byte)((int)(-ste_squ[i_send, 1]) >> 8);
        data_send[2] = (byte)((int)(-ste_squ[i_send, 1]) & 0xFF);
        serial_t.Text += " b " + data_send[1] + " " + data_send[2];
 //serialPort1.Write(data_send, 0, 3);
        serialPort1.Write(data_send, 0, 1);
        serialPort1.Write(data_send, 1, 1);
        serialPort1.Write(data_send, 2, 1);

        data_send[0] = (byte)'c';
        data_send[1] = (byte)((int)(-ste_squ[i_send, 2]) >> 8);
        data_send[2] = (byte)((int)(-ste_squ[i_send, 2]) & 0xFF);
        serial_t.Text +=" c " + data_send[1] + " " + data_send[2];
 //serialPort1.Write(data_send, 0, 3);
        serialPort1.Write(data_send, 0, 1);
        serialPort1.Write(data_send, 1, 1);
        serialPort1.Write(data_send, 2, 1);

        data_send[0] = (byte)'s';
        serialPort1.Write(data_send, 0, 1);
        serial_t.Text += " s " + Environment.NewLine;
        progressBar1.Value = (i_send+1) * 100 / str_cnt;
```

```
        }

 privatevoid DisplayText(object sender, EventArgs e)
        {
            serial_t.Text += tmpc;
        }

 privatevoid serialPort1_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            tmpc = serialPort1.ReadExisting();
 if (tmpc.ElementAt(0) == 'p') rec_f = 0;
 this.Invoke(newEventHandler(DisplayText));   //Display//
        }

 privatevoid button7_Click(object sender, EventArgs e)
        {
 //sending_thread.Abort();
 if (t_st == 0)
            {
                rec_f = 0;
                t_st = 1;
            }
        }

 privatevoid button8_Click(object sender, EventArgs e)
        {
            progressBar1.Value = 0;
            stop_f = 1;
        }

 privatevoid button6_Click(object sender, EventArgs e)
        {
            data_send[0] = (byte)'h';
            serialPort1.Write(data_send, 0, 1);
            serial_t.Text += " h " + Environment.NewLine;
        }
```

```
privatevoid label23_Click(object sender, EventArgs e)
        {

        }
    }
}
```