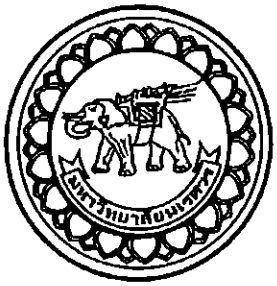


การออกแบบและสร้างอินฟราเรดสแกนเนอร์สำหรับตรวจจับสิ่งกีดขวาง

Design and Construction of Infrared Scanner for obstacle detection

นายญาณวัฒน์ เตชา รหัสนิสิต 52362571

นางสาวเยาวภา โทนะบุตร รหัสนิสิต 52362830

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2555

ใบรับรองปริญญานิพนธ์

หัวข้อโครงงาน การออกแบบและสร้างสแกนเนอร์อินฟราเรดสำหรับตรวจจับสิ่งกีดขวาง

ผู้ดำเนินโครงงาน นายญาณวัฒน์ เตชา รหัส 52362571

นางสาวเยาวภา โทนะบุตร รหัส 52362830

อาจารย์ที่ปรึกษา ดร. พนัส นัถฤทธิ์

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2555

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....ที่ปรึกษาโครงงาน

(ดร. พนัส นัถฤทธิ์)

.....กรรมการ

(ดร. สุรเดช จิตประไพกุลศาล)

.....กรรมการ

(อาจารย์รัฐภูมิ วรานุสาสน์)

.....กรรมการ

(อาจารย์เศรษฐา ตั้งค้าวานิช)

Project Title	Design and Construction of Infrared Scanner for obstacle detection		
Name	Mr. Yanavat	Tacha	ID. 52362571
	Miss Yaowapa	Tonabut	ID. 52362830
Project Advisor	Dr. Panus	Nattharith	
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic Year	2012		

Abstract

This project describes design and construction of the Infrared scanner for obstacle detection. In the system, an infrared sensor, SHARP 2Y0A02, is employer. It is installed on the servo motor to detect the distance of obstacle between 20 -150 centimeters within 180 degree in front of the robot. The Infrared sensor provides analog signal to Microcontroller. It is then converted to digital signal using A/D converter. The results of A/D converter are sent to the personal computer for further process in order to display all data on Graphic User Interface (GUI)

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้เกี่ยวกับการออกแบบและสร้างสแกนเนอร์อินฟราเรดสำหรับตรวจจับสิ่งกีดขวางซึ่งจะสำเร็จลุล่วงไปไม่ได้ด้วยความเมตตาและช่วยเหลือจากบุคคลต่อไปนี้

ขอขอบพระคุณ ดร. พนัส นัฉฤทธิ์ อย่างสูงที่ท่านได้กรุณามาเป็นที่ปรึกษาแก่โครงงานชิ้นนี้ และสละเวลาในการให้คำปรึกษาพร้อมทั้งแนะนำแนวทางการทำงานโดยตลอดรวมทั้งจัดสรรอุปกรณ์ทางฮาร์ดแวร์ที่ใช้ในการศึกษา

ขอขอบพระคุณบิดา มารดา ที่เป็นผู้เลี้ยงดู อบรมสั่งสอนมาเป็นอย่างดีและให้คำปรึกษาในเรื่องต่างๆ พร้อมทั้งให้กำลังใจที่ดีเสมอมา

ขอขอบคุณมหาวิทยาลัยนเรศวรที่ให้งบประมาณในการศึกษาและสถานที่ดำเนินการในครั้งนี้ และขอบคุณคณาจารย์ เจ้าหน้าที่ทุกคน ที่ให้ความรู้และคำแนะนำที่ดีเสมอมา รวมทั้งผู้ที่ไม่ได้เอ่ยนามที่มีส่วนช่วยในการดำเนินงานจนสำเร็จลุล่วงไปได้ด้วยดีมา ณ ที่นี้ด้วย

ผู้ดำเนินโครงงาน
นายญาณวัฒน์ เตชา
นางสาวเยาวภา โทนะบุตร

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในปัจจุบันเทคโนโลยีเข้ามามีบทบาทกับชีวิตประจำวันมากขึ้นโดยเฉพาะอย่างยิ่งอุปกรณ์อำนวยความสะดวกภายในบ้าน ไม่ว่าจะเป็นอุปกรณ์ไฟฟ้าและคอมพิวเตอร์ประเภทต่างๆ รวมถึงหุ่นยนต์ที่มีความฉลาดสามารถทำงานตามโปรแกรมที่กำหนดไว้ได้ โดยระบบภายในของหุ่นยนต์นั้น จะประกอบไปด้วยอุปกรณ์ที่สำคัญหลายชนิด ซึ่งช่วยให้หุ่นยนต์สามารถรับรู้สภาวะภายนอกของบริเวณที่หุ่นยนต์ทำงานอยู่ทำการตัดสินใจโดยอาศัยข้อมูลที่ได้รับเข้ามาและเคลื่อนที่เข้าสู่เป้าหมายเพื่อทำภาระกิจที่ถูกกำหนดไว้ให้สำเร็จลุล่วงได้อย่างมีประสิทธิภาพ

ในการเคลื่อนไหวของมนุษย์ใช้อวัยวะเพื่อให้เคลื่อนไหวได้ เช่น แขน-ขา ใช้ในการเคลื่อนที่ไปในที่ต่างๆ และอวัยวะที่สำคัญได้แก่ ตา ใช้ในการมองเห็นภาพต่างๆ และนำไปประมวลผลร่วมกับสมองทำให้รู้ว่าข้างหน้ามีสิ่งกีดขวางหรือไม่ หากมีอยู่ในระยะเท่าไร เพื่อนำไปสั่งงานให้แขน-ขาเคลื่อนที่ไปยังจุดหมายที่ต้องการ

หุ่นยนต์ก็เช่นเดียวกันหากไม่มีอุปกรณ์ที่ทำหน้าที่เป็นดวงตา อาจทำให้หุ่นยนต์เคลื่อนที่ไปชนกับสิ่งกีดขวางส่งผลให้เกิดความเสียหาย จึงจำเป็นต้องมีอุปกรณ์ที่ทำหน้าที่เหมือนกับดวงตาเพื่อให้ตรวจสอบว่า ทางข้างหน้ามีสิ่งกีดขวางหรือไม่ หากมีอยู่ในระยะเท่าไร เพื่อนำไปเป็นข้อมูลที่ใช้ในการเคลื่อนที่ไปให้ถึงจุดหมายอย่างปลอดภัย

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 สามารถควบคุมการหมุนของเซอร์โวมอเตอร์ได้

1.2.2 สามารถควบคุมอุปกรณ์ที่พัฒนาขึ้นให้บอกระยะห่างระหว่างเซนเซอร์อินฟราเรดกับสิ่งกีดขวางได้

1.2.3 สามารถแสดงค่าระยะห่างของสแกนเนอร์อินฟราเรดผ่านทางหน้าจอแสดงผล Graphic User Interface (GUI) ได้

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 มีความรู้เกี่ยวกับเซนเซอร์อินฟราเรด

1.3.2 มีความรู้เกี่ยวกับบอร์ดในตระกูล PIC

1.3.3 มีความรู้ในการเชื่อมต่ออุปกรณ์ต่างชนิดให้ทำงานร่วมกันอย่างมีระบบได้

1.3.4 ได้อุปกรณ์ที่สามารถนำไปใช้งานร่วมกับหุ่นยนต์เคลื่อนที่ (Mobile Robot) เพื่อบอกระยะของสิ่งกีดขวางได้จริง

1.3.5 มีความรู้ในการเขียนโปรแกรมเพื่อควบคุมอุปกรณ์ภายนอกและควบคุมการแสดงผลออกทางหน้าจอ

1.4 ขอบเขตการดำเนินโครงงาน

1.4.1 อุปกรณ์ที่สร้างขึ้นสามารถบอกระยะและทิศทางระหว่างเซนเซอร์กับสิ่งกีดขวางได้ในช่วง 20-150 เซนติเมตร ในบริเวณ 180 องศา ทางด้านหน้าในแนวระนาบ

1.4.2 ความละเอียดของการสแกนอยู่ที่อย่างน้อย 5 องศาต่อการเคลื่อนที่ 1 ครั้ง

1.4.3 ความคาดเคลื่อนของระยะห่างขึ้นอยู่กับระยะห่างของเซนเซอร์กับสิ่งกีดขวางตามคุณลักษณะของเซนเซอร์ตามรูปที่ 2.8 ในหน้าที่ 11 โดยในช่วง 20 เซนติเมตร ถึง 60 เซนติเมตร จะมีความคลาดเคลื่อนที่น้อยกว่าช่วง 60 เซนติเมตร เป็นต้นไป

1.4.4 การทำงานต้องอยู่ภายในอาคาร ไม่มีแสงแดดเข้าถึง และไม่มีแสงอินฟราเรดรบกวน

1.5 ขั้นตอนการดำเนินการ

1.5.1 ศึกษาการทำงานของมอเตอร์กระแสตรงและศึกษาการทำงานของเซนเซอร์อินฟราเรด

1.5.2 ศึกษาการหลักการทำงานและการใช้งานบอร์ดไมโครคอนโทรลเลอร์

1.5.3 เขียนโปรแกรมเพื่อควบคุมการทำงานของเซนเซอร์อินฟราเรดกับ มอเตอร์กระแสตรง

1.5.4 ศึกษาการเขียนโปรแกรมในการติดต่อกับฮาร์ดแวร์

1.5.5 เขียนโปรแกรมเพื่อวัดระยะทางเพื่อแสดงผลทางหน้าจอและทดสอบโปรแกรม

1.5.6 สรุปและวิเคราะห์ผล

1.6 แผนการดำเนินงาน

รายการ	พ.ศ.2555							พ.ศ. 2556				
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
1.5.1ศึกษาการทำงานของมอเตอร์และศึกษาการทำงานของเซนเซอร์อินฟราเรด	↔											
1.5.4 ศึกษาการหลักการทำงานและการใช้งานบอร์ดไมโครคอนโทรลเลอร์	↔	↔										
1.5.3 ออกแบบโปรแกรมควบคุมการทำงานของมอเตอร์กระแสตรง	↔	↔	↔	↔								
1.5.4เขียนโปรแกรมเพื่อควบคุมการทำงานของเซนเซอร์อินฟราเรดกับมอเตอร์กระแสตรง				↔	↔	↔	↔					
1.5.5 ศึกษาการเขียนโปรแกรมในการติดต่อกับ ฮาร์ดแวร์								↔				
1.5.6 ออกแบบโปรแกรมแสดงระยะทางโดยรับค่าจากเซนเซอร์อินฟราเรด									↔			
1.5.7 เขียนโปรแกรมวัดระยะทางเพื่อแสดงผลทางหน้าจอและทดสอบโปรแกรม										↔	↔	
1.5.8 สรุปและวิเคราะห์ผล												↔

1.7 งบประมาณที่ใช้

1.7.1 ค่าหนังสือประกอบการทำโครงงานและรูปเล่ม	เป็นเงิน	1,000	บาท
1.7.2ค่าใช้จ่ายอื่นๆ	เป็นเงิน	1,000	บาท
รวมเป็นเงินทั้งสิ้น		2,000	บาท

บทที่2
หลักกการและทฤษฎี

โครงงานเรื่อง การออกแบบและสร้างเซนเซอร์อินฟราเรดสำหรับตรวจจับสิ่งกีดขวางเป็นการออกแบบและสร้างเซนเซอร์อินฟราเรดตรวจพบวัตถุแล้วจะทำการส่งค่าระยะทางที่วัตถุอยู่ห่างจากเซนเซอร์อินฟราเรดขึ้นแสดงออกทางหน้าจอ Graphic User Interface (GUI) เพื่อสะดวกต่อผู้ใช้ในการอ่านค่าระยะทาง ซึ่งโครงงานชิ้นนี้เปรียบเสมือนการทำดวงตาให้กับหุ่นยนต์เคลื่อนที่

2.1 หุ่นยนต์อัตโนมัติ (Autonomous robots)

หุ่นยนต์อัตโนมัติคือหุ่นยนต์ที่สามารถทำงานได้ด้วยตัวเองมีอิสระในการทำงานโดยปราศจากการควบคุมของมนุษย์และเมื่อเจออุปสรรคกีดขวางการทำงานสามารถหลบหลีกหรือแก้ปัญหาได้ หุ่นยนต์อัตโนมัติมีความฉลาดมากน้อยหรือไม่ขึ้นอยู่กับการออกแบบโปรแกรมการทำงานและเทคโนโลยีของหุ่นยนต์เองพร้อมกับการที่มีระบบเซ็นเซอร์ที่ดีจึงทำให้หุ่นยนต์นั้นฉลาดและรับรู้การทำงานขณะนั้นได้ในปัจจุบันหุ่นยนต์อัตโนมัติสามารถทำงานได้อย่างมีประสิทธิภาพในสภาพแวดล้อมที่หลากหลาย

ในการออกแบบและสร้างหุ่นยนต์เคลื่อนที่มีหลักการพื้นฐานและองค์ประกอบด้วยกันหลายส่วน เพื่อทำให้หุ่นยนต์ที่สร้างนั้น มีความสามารถในการทำงานที่ต้องการได้ พื้นฐานของหุ่นยนต์เคลื่อนที่ มีดังนี้

Path planning

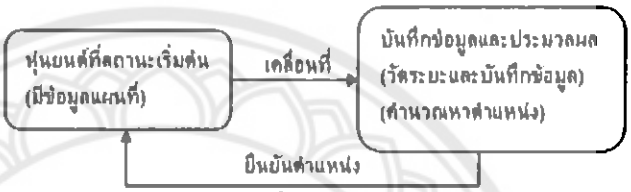
เป็นการวางแผนการเคลื่อนที่ของหุ่นยนต์คือการสร้างการเคลื่อนไหวจากจุดเริ่มต้นถึงจุดเป้าหมายได้อย่างรวดเร็วโดยไม่ชนสิ่งกีดขวางซึ่งเป็นเส้นทางที่ดีที่สุดและปลอดภัยที่สุดเช่นเมื่อได้รับคำสั่งให้เดินทางจากห้องหนึ่งไปยังห้องถัดไปในอาคารสำนักงานหุ่นยนต์จะมีวิธีการเดินทางอย่างไรเพื่อไปถึงจุดหมายโดยไม่ชนกับสิ่งกีดขวางและไม่ตกบันไดเป็นต้น[1]

Simultaneous localization and Mapping (SLAM)

เป็นเทคนิคที่ใช้ในหุ่นยนต์และยานพาหนะขับเคลื่อนอัตโนมัติโดยการทำให้หุ่นยนต์สามารถระบุตำแหน่งตัวเองและสร้างแผนที่ในขณะเดียวกันในสภาวะต่างๆหรือในสถานที่ที่ไม่เคยไปมาก่อนซึ่งนำไปประยุกต์ใช้ได้หลายสภาพแวดล้อมเช่นใต้ดินภายในหรือนอกอาคารใต้น้ำหรือบนดาวเคราะห์อื่นๆซึ่งหลักการของสแลมจะต้องอาศัยวิธีนำเอาข้อมูลจากเซนเซอร์มากัดกรองก่อนเช่นตัวกรองคาล

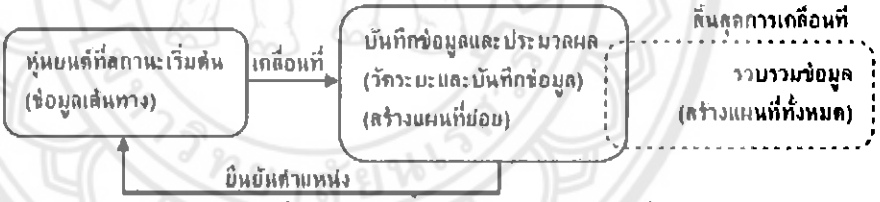
มาน ตัวกรองคาลมานแบบขยาย ตัวกรองอนุภาค (Particle filter) สแลมแบบเร็ว (Fast SLAM) เป็นต้น [2]

การระบุตำแหน่ง การระบุตำแหน่งนั้นหุ่นยนต์ต้องรับข้อมูลของแผนที่มาก่อนเมื่อหุ่นยนต์เคลื่อนที่จะทำการประมวลผลหาตำแหน่งของตัวเองโดยใช้ข้อมูลที่ได้จากเซ็นเซอร์และนำไปอ้างอิงตำแหน่งจากแผนที่ที่มีอยู่เดิมซึ่งส่วนใหญ่นำไปใช้ในการติดตามตำแหน่งหุ่นยนต์แผนภาพแสดงขั้นตอนระบุตำแหน่ง ดังแสดงในรูปที่ 2.1



รูปที่2.1แผนภาพแสดงขั้นตอนการระบุตำแหน่ง [2]

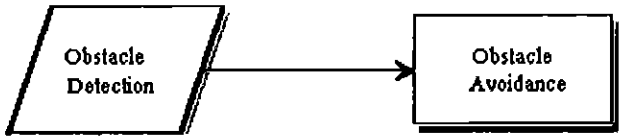
การสร้างแผนที่ หุ่นจะถูกกำหนดเส้นทางการเคลื่อนที่เป็นเส้นทางระหว่างเคลื่อนที่นั้นจะใช้เซนเซอร์ทำการเก็บบันทึกข้อมูลสภาพแวดล้อมนั้นๆ มาสร้างเป็นแผนที่ดังแผนภาพใน รูปที่ 2.2



รูปที่2.2แผนภาพแสดงการสร้างแผนที่ [2]

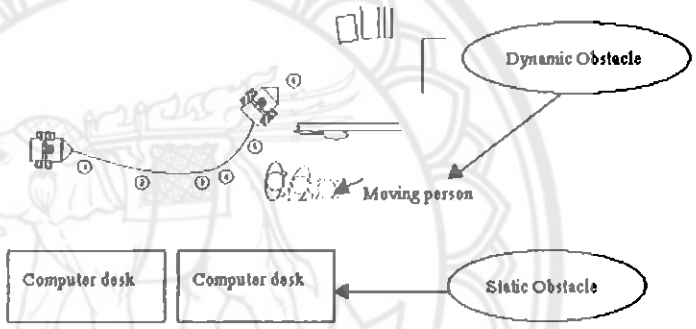
Obstacle Avoidance

เป็นการหลีกเลี่ยงสิ่งกีดขวางทั้งในสถานะที่เป็น สิ่งกีดขวางแบบสแตติก (Static Obstacle) คือวัตถุที่ไม่มีการเคลื่อนที่ เช่น โต๊ะ ตึก พนัง เป็นต้น และ สิ่งกีดขวางแบบไดนามิก (Dynamic Obstacle) คือวัตถุที่สามารถเคลื่อนที่ได้ เช่น คน เป็นต้นโดยการเคลื่อนที่ของหุ่นยนต์จะทำการเลี้ยวซ้ายและเลี้ยวขวาเพื่อเคลื่อนที่ไปยังพื้นที่ที่ว่างและหลีกเลี่ยงสิ่งกีดขวาง



รูปที่ 2.3Obstacle Avoidance [3]

จากรูปที่ 2.3 เป็นหลักการทำงานของการหลีกเลี่ยงสิ่งกีดขวางโดยจะมีการตรวจจับหาวัตถุที่เป็นสิ่งกีดขวางเมื่อตรวจเจอก็จะทำการหลีกเลี่ยงสิ่งกีดขวาง



รูปที่ 2.4 Obstacle Avoidance [3]

จากรูปที่ 2.4 เป็นตัวอย่างของหุ่นยนต์ที่ทำการหลีกเลี่ยงสิ่งกีดขวางทั้งที่เป็นแบบ สแตติก (Static) และไดนามิค (Dynamic) โดยในภาพกำหนดให้ Computer desk เป็น Static Obstacle หรือสิ่งกีดขวางที่อยู่กับที่ และ Moving Person เป็น Dynamic Obstacle หรือสิ่งกีดขวางที่เคลื่อนที่ได้ โดยที่ตัวหุ่นยนต์ได้มีการเคลื่อนที่ไปยังพื้นที่ที่ว่างเมื่อหุ่นยนต์ไปถึงพื้นที่ที่มีสิ่งกีดขวางอยู่ก็จะทำการหมุนตัวออกไปเพื่อเป็นการหลีกเลี่ยงสิ่งกีดขวางและเคลื่อนที่ไปยังพื้นที่ว่างต่อไป

เมื่อมีพื้นฐานของหุ่นยนต์เคลื่อนที่แล้ว องค์ประกอบที่สำคัญของหุ่นยนต์เคลื่อนที่อีกอย่างหนึ่งก็คือ เซนเซอร์ที่ใช้ในหุ่นยนต์เคลื่อนที่ ซึ่งทำหน้าที่เสมือนประสาทสัมผัสของหุ่นยนต์ซึ่งใช้การตรวจจับหรือรับรู้การเปลี่ยนแปลงทางกายภาพต่างๆ เช่น ความร้อน แสง สี เสียง การเคลื่อนที่ ความดัน การไหล ระยะทาง เป็นต้น แล้วเปลี่ยนให้อยู่ในรูปของสัญญาณหรือข้อมูลที่สอดคล้องและความเหมาะสมกับการกำหนดเงื่อนไข โดยตัวเซนเซอร์ที่ใช้ในหุ่นยนต์เคลื่อนที่มีด้วยกันหลายประเภท ไม่ว่าจะเป็น

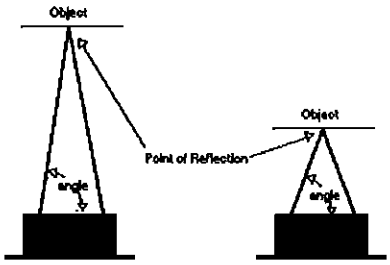
เซนเซอร์ประเภทตรวจจับอุณหภูมิ เซนเซอร์ประเภทนี้จะเปลี่ยนระดับอุณภูมิ เป็นระดับแรงดันไฟฟ้า ซึ่งทำหน้าที่เป็นส่วนรับความรู้สึกของหุ่นยนต์ เช่น เทอมิสเตอร์ เซนเซอร์ประเภทเหนี่ยวนำ เซนเซอร์ประเภทนี้จะมีแม่เหล็กอยู่ที่ส่วนหัวซึ่งมีความถี่สูง เมื่อมีวัตถุที่เป็นโลหะอยู่ในบริเวณที่สนามแม่เหล็กสามารถส่งไปถึง จะทำให้เกิดการเปลี่ยนแปลงความเหนี่ยวนำ เช่น พร็อกซิมิตี้เซนเซอร์เซนเซอร์ประเภทเก็บประจุจะมีลักษณะการทำงานเดียวกันกับเซนเซอร์ประเภทเหนี่ยวนำแต่จะสามารถตรวจจับวัตถุที่ไม่ใช่โลหะได้ เซนเซอร์ประเภทเสียง เซนเซอร์ประเภทนี้จะเปลี่ยนแปลงความถี่เสียงให้กลายเป็นสัญญาณไฟฟ้า ทำหน้าที่เป็นส่วนหูของหุ่นยนต์ เช่น อัลตราโซนิคเซนเซอร์ เซนเซอร์ประเภทสัมผัส เซนเซอร์ประเภทนี้จะเปลี่ยนการสัมผัส ให้เป็นสัญญาณทางไฟฟ้า ซึ่งทำหน้าที่เป็นส่วนผิวหนังของหุ่นยนต์ เช่น เซนเซอร์เม็คคานิค (Mechanical sensor) เซนเซอร์ประเภทแสง เซนเซอร์ประเภทนี้ทำหน้าที่เปลี่ยนสัญญาณอินพุตแบบต่างๆ (แล้วแต่ลักษณะ input) ให้กลายเป็นสัญญาณไฟฟ้า ทำหน้าที่เป็นตาของหุ่นยนต์ เช่น อินฟราเรดเซนเซอร์ โดยอินฟราเรดเซนเซอร์นี้ใช้แสงในการวัดระยะทางได้ จึงเลือกใช้เซนเซอร์ชนิดนี้ในการทำโครงงาน โดยรายละเอียดของเซนเซอร์ประเภทนี้จะอยู่ในหัวข้อถัดไป

2.2 เซนเซอร์อินฟราเรด (Infrared Sensor)

อินฟราเรดเป็นแสงที่ไม่สามารถเห็นได้ด้วยตาเปล่ามีความถี่อยู่ในช่วง 1011 – 1014 Hz [4] เนื่องจากแสงอินฟราเรดมีความยาวคลื่นที่สั้นมีคุณสมบัติที่เด่นคือจะเดินทางเป็นแนวเส้นตรงและสามารถเดินทางผ่านสิ่งกีดขวางหรือวัตถุได้จึงเป็นที่นิยมนำมาใช้ในการสื่อสารในระยะทางใกล้ๆเช่น รีโมทควบคุมวิทยุโทรทัศน์เป็นต้นหรือตรวจจับสิ่งกีดขวางต่างๆ

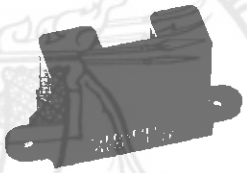
หลักการทำงานของเซนเซอร์อินฟราเรด

เซนเซอร์อินฟราเรด จะมีหลักการทำงานคือจะส่งแสงอินฟราเรดจากเครื่องรับไปยังเครื่องส่งซึ่งจะทำงานเป็นกระบวนการสามเหลี่ยม (Triangulation) โดยการส่งแสงอินฟราเรดออกไปแล้วสะท้อนกลับมา แสงอินฟราเรดจะกระทบกลับมาที่มุมที่จะขึ้นอยู่กับระยะทางที่เซนเซอร์อยู่ห่างจากวัตถุทำให้เห็นถึงระยะของวัตถุ กระบวนการสามเหลี่ยมทำงานโดยการตรวจหามุมของแสงอินฟราเรดเมื่อรู้มุมก็จะรู้ระยะทางได้ ดังแสดงใน รูปที่ 2.5



รูปที่ 2.5 การทำงานของ Infrared Sensor [5]

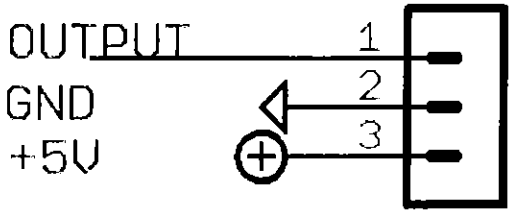
เซนเซอร์อินฟราเรดที่มีอยู่ในปัจจุบันมีหลายชนิดแต่ละชนิดมีคุณสมบัติของเซนเซอร์ที่เหมือนและแตกต่างกัน แต่ในการทำโครงงานชิ้นนี้เลือกใช้เซนเซอร์อินฟราเรดของ Sharp รุ่น 2Y0A02 ดังแสดงในรูปที่ 2.6



รูปที่2.6 Sharp 2Y0A02 Infrared Sensor [6]

Sharp2Y0A02 Infrared Sensor สามารถวัดระยะได้ตั้งแต่ 20 -150 เซนติเมตร [9] ในส่วนของเอาท์พุตจะเป็นแรงดันอนาล็อกที่ได้จากระยะที่วัดโดยจะมีสายเชื่อมต่อระหว่างเซนเซอร์กับอุปกรณ์ 3 เส้นประกอบไปด้วย ดังแสดงในรูปที่ 2.7

- สาย Output
- สาย Ground
- สายแรงดันไฟฟ้า 5 โวลต์ (+5 V)



รูปที่ 2.7 สายสัญญาณของ Sharp 2Y0A02 Infrared Sensor [7]

Sharp 2Y0A02 Infrared Sensor เป็นชนิดเซนเซอร์วัดระยะทาง (Distance Infrared) ระยะวัดที่ใกล้ที่สุด 20 เซนติเมตร ระยะวัดที่ไกลที่สุด 150 เซนติเมตร และเวลาที่ใช้ในการตอบสนอง 50 มิลลิวินาที คุณสมบัติด้านไฟฟ้าและคุณสมบัติทางกายภาพ สามารถดูได้จากตารางที่ 2.1 และตารางที่ 2.2 [6]

- คุณสมบัติด้านไฟฟ้าของ Sharp 2Y0A02 Infrared Sensor มีดังนี้

Supply Voltage Min แรงดันไฟฟ้าที่ใช้ในการทำงานน้อยสุด	4.5 V DC 4.5 โวลต์กระแสตรง
Supply Voltage Max แรงดันไฟฟ้าที่ใช้ในการทำงานมากสุด	5.5 V DC 5.5 โวลต์กระแสตรง
Current Consumption Max กระแสไฟฟ้าที่ใช้ในการทำงาน	50 mA 50 มิลลิแอมป์แปร์

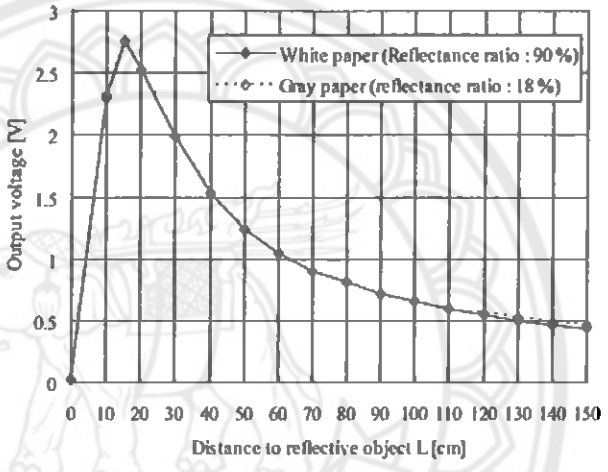
ตารางที่ 2.1 แสดงคุณสมบัติด้านไฟฟ้า [6]

- คุณสมบัติทางกายภาพของ Sharp 2Y0A02 Infrared Sensor มีดังนี้

Weight น้ำหนัก	6 g 6 กรัม
Operating Temperature Min อุณภูมิในการทำงานน้อยสุด	-10 °C -10 องศาเซลเซียส
Operating Temperature Max อุณภูมิในการทำงานมากสุด	60 °C 60 องศาเซลเซียส

ตารางที่ 2.2 แสดงคุณสมบัติทางกายภาพ [6]

จากรูปที่ 2.8 เป็นกราฟแสดงความสัมพันธ์ระหว่างระยะทาง (Distance) กับแรงดันเอาท์พุต (Output Voltage) ของ Sharp 2Y0A02 Infrared Sensor จากรูปจะเห็นว่าเอาท์พุตที่แสดงในระยะวัตถุที่ใกล้มากในระยะ 0 – 20 เซนติเมตร ยังไม่น่าเชื่อถือเนื่องจากมีความไม่แน่นอนของค่าความสัมพันธ์ระหว่างระยะทาง (Distance) กับแรงดันเอาท์พุต (Output Voltage) โดยจากกราฟจะเริ่มมีแนวโน้มที่มีความสัมพันธ์กันระหว่างค่าของ ระยะทาง (Distance) กับแรงดันเอาท์พุต (Output Voltage) ที่ช่วงตั้งแต่ 20 เซนติเมตร เป็นต้นไปดังนั้นจึงเริ่มใช้ค่าในระยะที่ 20 เซนติเมตร ถึง 150 เซนติเมตร



รูปที่ 2.8 กราฟแสดงความสัมพันธ์ระหว่างความสัมพันธ์ระหว่างระยะทางกับOutput Voltage ของ Sharp2Y0A02 Infrared Sensor [8]

ในการทำโครงงานนี้ใช้ Sharp 2Y0A02 Infrared Sensor เป็นตัวสแกนเนอร์เนื่องจากมีระยะการวัดของเซนเซอร์ที่ต้องการคือ ระหว่าง 20 – 150 เซนติเมตร และเซนเซอร์ตัวนี้ง่ายต่อการโปรแกรมเพื่อวัดค่าหาระยะทาง เช่น เมื่อเซนเซอร์ได้รับแรงดัน 2.5 โวลต์ จะให้ระยะทางเป็น 20 เซนติเมตร เป็นระยะที่ใกล้ที่สุด และที่แรงดัน 0.4 โวลต์ จะให้ระยะทางเป็น 150 เซนติเมตร เป็นระยะที่ไกลที่สุด เป็นต้น

ในการทำงานของอุปกรณ์เซนเซอร์สายทั้ง 3 เส้น ที่ประกอบไปด้วย สายสัญญาณ สายแรงดันไฟฟ้า และสายกราวน์ จำเป็นต้องเชื่อมต่อกับ บอร์ดไมโครคอนโทรลเลอร์ เพื่อเป็นแหล่งจ่ายไฟและต่อเชื่อมกับอุปกรณ์ประเภทอื่นๆ รายละเอียดของ บอร์ดไมโครคอนโทรลเลอร์จะแสดงในหัวข้อถัดไป

2.3 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ควบคุมขนาดเล็ก มีความสามารถคล้ายคอมพิวเตอร์ ภายในบรรจุด้วยหน่วยประมวลผล หน่วยความจำ และ พอร์ต ซึ่งเป็นส่วนประกอบหลักที่สำคัญรวมเข้าไว้ด้วยกันโครงสร้างของไมโครคอนโทรลเลอร์โดยทั่วไป สามารถแบ่งได้เป็น 5 ส่วนใหญ่ๆ ดังนี้ [18]

1). หน่วยประมวลผลกลาง หรือ CPU

2). หน่วยความจำ (Memory) แบ่งออกเป็น 2 ส่วน คือ หน่วยความจำสำหรับเก็บโปรแกรมหลัก (Program memory) และ หน่วยความจำข้อมูล (Data memory)

3). ส่วนติดต่อกับอุปกรณ์ภายนอกหรือพอร์ตมี 2 ลักษณะ คือพอร์ตอินพุต และ พอร์ตเอาท์พุต

4). ช่องทางเดินของสัญญาณ หรือ บัส (Bus) คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง CPU หน่วยความจำ และพอร์ต

5). วงจรกำเนิดสัญญาณนาฬิกา คือ ตัวกำหนดจังหวะการทำงานของ CPU

ทั้งหมดนี้เป็นโครงสร้างของไมโครคอนโทรลเลอร์ทั่วไป แต่ในการทำโครงงานชิ้นนี้ เลือกใช้ PIC 16F877 เป็นไมโครคอนโทรลเลอร์ เนื่องจากมีคุณสมบัติที่จำเป็นและเพียงพอต่อการใช้งาน เช่น การแปลงอนาล็อกเป็นดิจิตอล (A/D) เป็นต้นภาพแสดง PIC 16F877 ในรูปที่ 2.9

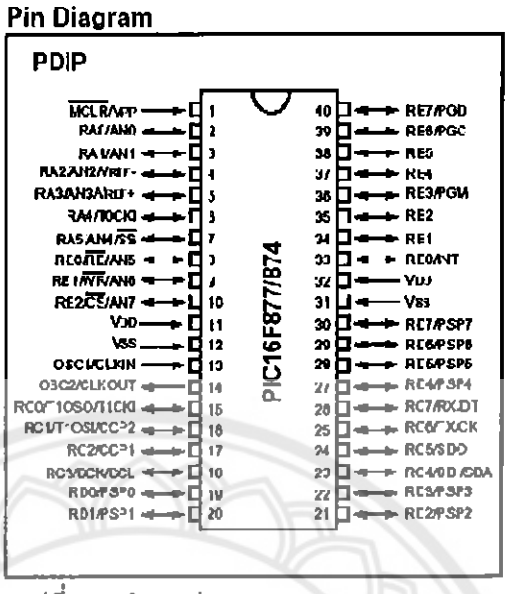
รูปที่ 2.9แสดงภาพPIC 16F877 [10]

2.5.1 ไมโครคอนโทรลเลอร์PIC 16F877

ไมโครคอนโทรลเลอร์ PIC 16F877 เป็น CPU ภายในตัวถังแบบ DIP ขนาด 40 ขา มีคุณสมบัติในการทำงาน ดังนี้

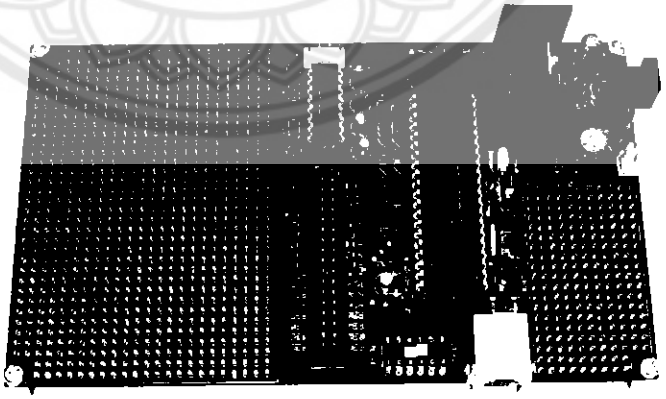
- มี 35 Instruction คำสั่ง
- ในการปฏิบัติงานคำสั่งต่างๆจะใช้ Cycle เดียวและ 2 Cycle ในคำสั่งที่เป็นการกระโดด
- ความถี่สูงสุดที่ทำงานได้คือ 20 MHz (16F877-20/P)
- การทำงานจะเป็นลักษณะ Pipeline ทำให้มีการทำงานที่เร็วขึ้น
- หน่วยความจำโปรแกรม FLASH Program Memory มีขนาด 8k (14-Bit Words)

- หน่วยความจำข้อมูล (RAM) 368 Bytes
-หน่วยความจำข้อมูล (EEPROM) 256 Bytes
-สามารถตอบสนองการอินเตอร์รัพได้ถึง 14 แหล่ง
- มี STACK 8 ระดับ
- มีเพาเวอร์ออนรีเซต (POR) เพาเวอร์อัปไทเมอร์ (PWRT) และ Oscillator Start-Up Timer
- Watchdog Timer
- สามารถเลือกการป้องกันข้อมูลได้ (Code Protection)
- มีโหมดประหยัดพลังงาน (Sleep Mode)
- เลือกโหมดของสัญญาณนาฬิกาได้หลายโหมด
- สามารถโปรแกรมโดยใช้แรงดัน +5V ได้
- มีฟังก์ชันการโปรแกรมแบบ ICSP (In-Circuit Serial Programming)
- ทำงานที่ไฟเลี้ยง 2.0V ถึง 5.5V
- กระแสทั้งซิงก์และซอร์สของพอร์ตคือ 25mA
- มี Timer/Counter จำนวน 3 ตัวคือ Timer0, Timer1 และ Timer2
- มีโมดูล Capture/Compare/PWM จำนวน 2 ชุด
- มี Analog to Digital Converter ความละเอียด 10 บิต 8 แชนเนลภายในตัว
- มีโมดูลการสื่อสาร USART
- มีโมดูลตรวจจับระดับไฟเลี้ยง Brown – out reset (BOR)
- มีพอร์ต I/O 5 พอร์ตประกอบด้วย A, B, C, D และ E แต่ละพอร์ตจะมีจำนวนบิตไม่เท่ากันซึ่งรวมแล้วจะมี I/O จำนวน 33 บิตดังแสดงโครงสร้างตำแหน่งของขาสัญญาณต่างๆ ในรูปที่ 2.10

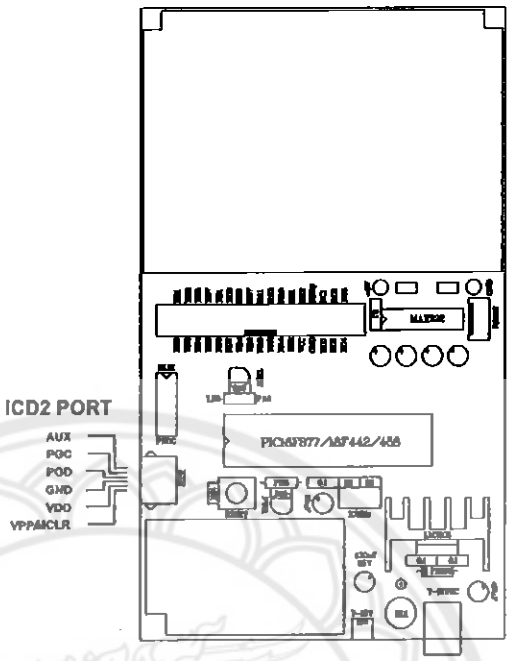


รูปที่ 2.10ตำแหน่งขาของ PIC 16F877 [11]

เนื่องจากตัวไมโครคอนโทรลเลอร์จำเป็นต้องเชื่อมต่อกับอุปกรณ์ประเภทอื่นๆเช่น แหล่งจ่ายไฟ เป็นต้น เพื่อให้สามารถทำงานได้ โดยการนำไมโครคอนโทรลเลอร์และอุปกรณ์นั้นๆไว้ภายในบอร์ดเดียวกัน แต่ในปัจจุบันมีบอร์ดสำเร็จรูปที่รวมเอาไมโครคอนโทรลเลอร์และอุปกรณ์ที่จำเป็นเข้าไว้ด้วยกันแล้ว ในการทำโครงงานชิ้นนี้จึงใช้บอร์ดไมโครคอนโทรลเลอร์ CP-PIC V3.0 (ICD2) ในการทำโครงงาน เนื่องจากมีอุปกรณ์ที่จำเป็นต่อการใช้งาน และยังมีพื้นอเนกประสงค์สำหรับการต่อวงจรเพิ่มเติมได้ รูปที่ 2.11แสดงรูปของบอร์ด CP-PIC V3.0 (ICD2) และ รูปที่ 2.12แสดงโครงสร้างของบอร์ด CP-PIC V3.0(ICD2)



รูปที่ 2.11แสดงบอร์ด CP-PIC V3.0 (ICD2) [12]



รูปที่ 2.12แสดงโครงสร้างของบอร์ด CP-PIC V3.0 (ICD2) [12]

2.5.2 บอร์ด CP-PIC V3.0 (ICD2)

บอร์ดไมโครคอนโทรลเลอร์ CP-PIC V3.0 (ICD2) มีการออกแบบเพื่อรองรับวงจรพื้นฐานที่จำเป็น เช่น แหล่งจ่ายไฟ วงจรรีเซ็ต วงจรกำเนิดความถี่สัญญาณนาฬิกา พอร์ตสำหรับดาว์นโหลดโปรแกรม และวงจรสื่อสารอนุกรม ส่วนสัญญาณ I/O จะต่อกับคอนเน็กเตอร์ (Connector) เพื่อติดต่อกับอุปกรณ์ภายนอก นอกจากนี้ยังมีพื้นที่อเนกประสงค์สำหรับต่อวงจร I/O เพิ่มเติมได้ ในอุปกรณ์ภายในบอร์ดที่สำคัญจะประกอบด้วย

- RS 232 I แชนเนล
- การจัดสรร I/O ของบอร์ด (ET BUS I/O 33PIN)
- แหล่งจ่ายไฟ (Power Supply)
- การกำหนดโหมดการทำงาน
- สัญญาณนาฬิกา (Xtal)

การทำงานของอุปกรณ์ภายในบอร์ด CP-PIC V3.0 (ICD2)

RS-232

ในส่วนของ RS-232 นี้มีไว้เพื่อเพิ่มความสามารถในการสื่อสารกันระหว่าง CPU กับ PC หรือกับบอร์ดอื่นๆซึ่งเป็นการสื่อสารแบบอนุกรมโดยในบอร์ดนี้ใช้ MAX232 เป็นสายไดรีเวอร์ (Line Driver) ซึ่งสามารถสื่อสารโดยใช้สายสัญญาณเพียง 3 เส้นได้ไกลประมาณ 15 เมตรสามารถทำได้โดยการต่อสายที่ออกจากคอนเนคเตอร์ RS-232 บนบอร์ดไปต่อกับ PC หรือบอร์ดตัวอื่นแล้วเขียนโปรแกรมควบคุมให้ CPU ทำงานรูปที่ 2.13เป็นการแสดงการสื่อสารระหว่าง RS 232

รูปที่ 2.13การต่อ RS232 [12]

การจัดสรร I/O ของบอร์ด CP-PIC V3.0 (ICD2)

บอร์ด CP-PIC V3.0 (ICD2) จะใช้ได้กับ CPU เบอร์ 16F877 และ 18F458 ในโครงงานชิ้นนี้จะเลือกใช้เพียง เบอร์ 16F877 จะมีขาสัญญาณที่นำมาใช้งานเป็น I/O พอร์ต ทั้งสิ้น 33 เส้นประกอบด้วย

- RA0-RA5 จำนวน 6 เส้นสัญญาณ
- RB0-RB7 จำนวน 8 เส้นสัญญาณ
- RC0-RC7 จำนวน 8 เส้นสัญญาณ
- RD0-RD7 จำนวน 8 เส้นสัญญาณ
- RE0-RE2 จำนวน 3 เส้นสัญญาณ

แหล่งจ่ายไฟ (Power Supply)

แหล่งจ่ายไฟของบอร์ด CP-PIC V3.0 (ICD2) สามารถใช้งานได้ทั้งกับไฟฟ้ากระแสตรงและกระแสสลับ เนื่องจากมีวงจรรีทิไฟเออร์ (RECTIFIER) แบบบริดก์ (BRIDGE) พร้อมวงจรฟิวเตอร์ (FILTER) และรีกูเลเตอร์ (REGULATOR) ขนาด +5 โวลต์ โดยสามารถป้อนไฟฟ้าแรงดัน 9 – 12 โวลต์ ต่อกับขั้วคอนเนคเตอร์เมื่อมีการทำงานของแหล่งจ่ายไฟจะมีหลอดแอลอีดี (LED) แสดงผลการทำงานด้วย

การกำหนดโหมดการทำงาน

การทำงานของบอร์ด CP-PIC V3.0 (ICD2) สามารถกำหนดโหมดการทำงานของบอร์ดได้ 2 โหมด คือ โหมดการโปรแกรม (PROG) เป็นการโปรแกรมข้อมูลลงบอร์ด CP-PIC V3.0 (ICD2) และโหมดการทำงานปกติ (RUN) เป็นการทำให้ CPU ทำคำสั่งต่างๆที่ได้โปรแกรมไว้

สัญญาณนาฬิกา

PIC จะใช้สัญญาณนาฬิกาโดยมองเป็นลักษณะของวงรอบ (Cycle) ซึ่งระบุเอาไว้ว่า 1 คำสั่งนั้นจะประกอบไปด้วย1-2 วงรอบโดยแต่ละวงรอบนั้นจะแบ่งเป็น 4 ส่วนคือ Q1, Q2, Q3 และ Q4 ด้วยเหตุนี้ความเร็วโดยรวมของ PIC จึงเท่ากับค่าความถี่ของสัญญาณนาฬิกาหารด้วย 4 โดยสมการหาความเร็วโดยรวมของ PIC ดังสมการที่ 2.1 [13]

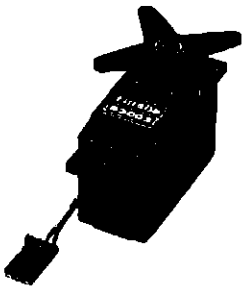
$$1cycle = Q1+Q2+Q3+Q4 = \frac{XTAL}{4} \quad (2.1)$$

ในการทำโครงงานชิ้นนี้นอกจากจะใช้เซนเซอร์ต่อเชื่อมกับบอร์ดไมโครคอนโทรลเลอร์แล้ว ยังมีอุปกรณ์ที่สำคัญอีกอย่างหนึ่งนั่นก็คือ เซอร์โวมอเตอร์มาเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ด้วย รายละเอียดของเซอร์โวมอเตอร์จะอยู่ในหัวข้อถัดไป

2.4 เซอร์โวมอเตอร์กระแสตรง

เซอร์โวมอเตอร์กระแสตรง ดังแสดงในรูปที่ 2.14 เป็นมอเตอร์กระแสตรงที่มีขนาดเล็กที่ถูกประกบเข้ากับส่วนประกอบต่างๆได้แก่ชุดเกียร์ทดชุดวงจรคุมตำแหน่งการหมุนโดยมีสายสำหรับต่อใช้งานทั้งสิ้น 3 สาย คือ

1. สายไฟ (สาย +V)
2. สายกราว์น (ground: GND)
3. สายควมคุม (Control line)



รูปที่ 2.14 Servo Motor [14]

สายควมคุมจะเป็นสายสำหรับควบคุมเซอร์โวมอเตอร์ให้หมุนซ้ายหมุนขวาโดยจะใช้สัญญาณ PWM เป็นตัวควบคุมส่วนแรงเคลื่อนแรงดัน ที่ป้อนสำหรับขับเซอร์โวมอเตอร์ใช้ประมาณ 4 ถึง6 โวลต์

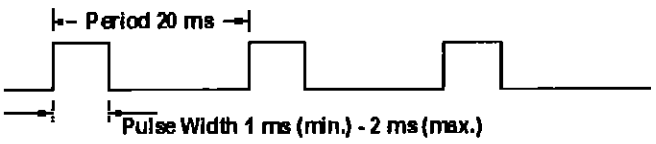
ข้อดีของเซอร์โวมอเตอร์คือมีขนาดเล็กน้ำหนักเบาแต่ให้แรงบิดสูงและกินพลังงานน้อยใช้ระดับสัญญาณควบคุมแบบ TTL level ซึ่งสามารถต่อกับไมโครคอนโทรลเลอร์ได้เลยไม่จำเป็นต้องมีวงจรขับโดยส่วนใหญ่จะมีช่วงการหมุนอยู่ที่180 – 210 องศา

หลักการทำงานของ Servo Motor

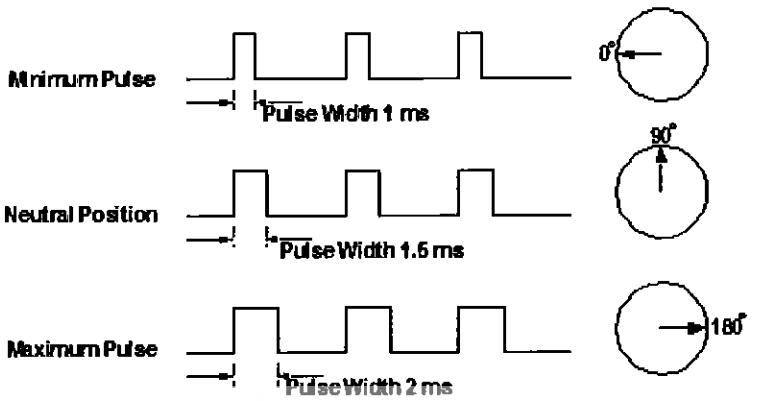
การควบคุมตำแหน่งการหมุนของเซอร์โวมอเตอร์สามารถทำได้โดยการป้อนสัญญาณความกว้างพัลซ์ของสัญญาณควบคุม

ประกอบด้วย

- Frame Period Pulse เป็นสัญญาณพัลซ์ต่อเนื่องจะเริ่มต้นห่างกันทุกๆ 20 มิลลิวินาทีตลอดเวลาเพื่อรักษาสภาพตำแหน่งการหมุนเอาไว้รูปที่ 2.15 แสดง Frame Period Pulse
- Position Pulse Width เป็นค่าความกว้างยอดพัลซ์ของ Frame Period width ใช้เป็นค่าควบคุมตำแหน่งและทิศทางการหมุนโดยจะมีค่าอยู่ระหว่าง 1.0 – 2.0 มิลลิวินาที ในรูปที่ 2.16 แสดงการกำหนดพัลซ์เพื่อควบคุมองศาในการหมุนของเซอร์โวมอเตอร์กระแสตรง



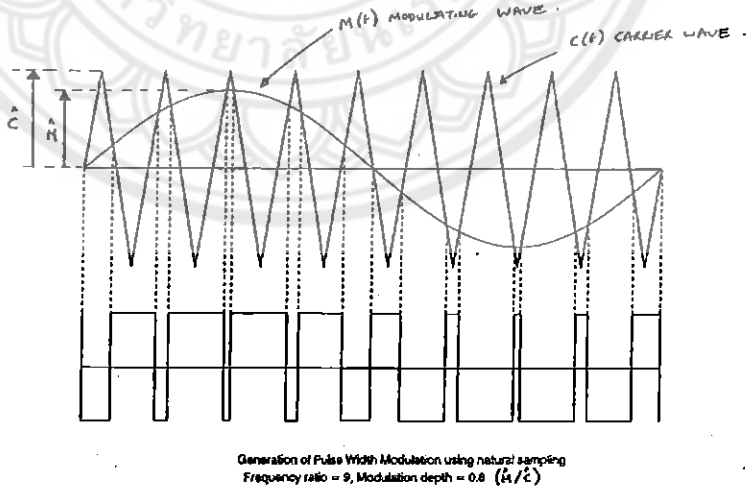
รูปที่ 2.15Frame Period Pulse [15]



รูปที่ 2.16 แสดงการกำหนดพัลซ์เพื่อควบคุมองศาในการหมุนของ DC Servo Motor [15]

สัญญาณ Pulse Width Modulation: **PWM**

สัญญาณ PWM มาจากชื่อเต็มว่า Pulse Width Modulation เป็นสัญญาณที่เกิดจากการผสมสัญญาณรูปสามเหลี่ยม (Triangle wave) กับระดับแรงเคลื่อนสัญญาณไฟดีซีหรือสัญญาณอื่นๆที่มีระดับไฟดีซีเป็นตัวรองรับผลที่ได้จะได้คลื่นสัญญาณพัลซ์ที่มีสองสถานะคือออน (ON) กับออฟ (OFF) เมื่อนำสัญญาณที่ได้ไปขับอุปกรณ์กำลังเช่นหลอดไฟมอเตอร์ดีซีจะได้ผลการควบคุมคือหลอดไฟจะติดสว่างเต็มที่ถ้าสถานะเป็น ON และเมื่อสัญญาณเป็น OFF หลอดไฟจะดับสัญญาณ PWM แสดงในรูปที่ 2.17



รูปที่ 2.17แสดงการเกิดสัญญาณ PWM จะการผสมระหว่างสัญญาณรูปสามเหลี่ยมและรูปไซน์ [16]

ในการผสมคลื่นสัญญาณคลื่นที่มาผสมต้องมีระดับความสูง (Amplitude) ไม่เกินยอดความสูงของคลื่นรูปสามเหลี่ยมซึ่งเป็นคลื่นพาหะหากแกนจะทำให้เกิดความเพี้ยนของสัญญาณ PWM เรียกว่า Over modulation

รายละเอียดของสัญญาณ PWM

1. ความถี่คือจำนวนรูปคลื่นต่อวินาที (Cycle/second) ในการใช้งาน PWM ความถี่จะต้องคงที่ไม่เปลี่ยนแปลงความถี่ที่เหมาะสมในการควบคุมอุปกรณ์กำลังต่างๆคือ 400 เฮิร์ต (Hz) – 10 กิโลเฮิร์ต (KHz) [17]
2. ดิวตี้ไซเคิล (Duty cycle) เป็นค่าคาบช่วงเวลาออนของไซเคิลโดยคิดเป็นเปอร์เซ็นของ คลื่นเต็ม (Full cycle)
3. ค่าแรงเคลื่อนเอาต์พุตเป็นค่าเฉลี่ยระหว่างแรงเคลื่อนช่วงเวลา ON กับ OFF คิดใน1 ลูกคลื่น ดังแสดงในสมการที่ 2.2 [17]

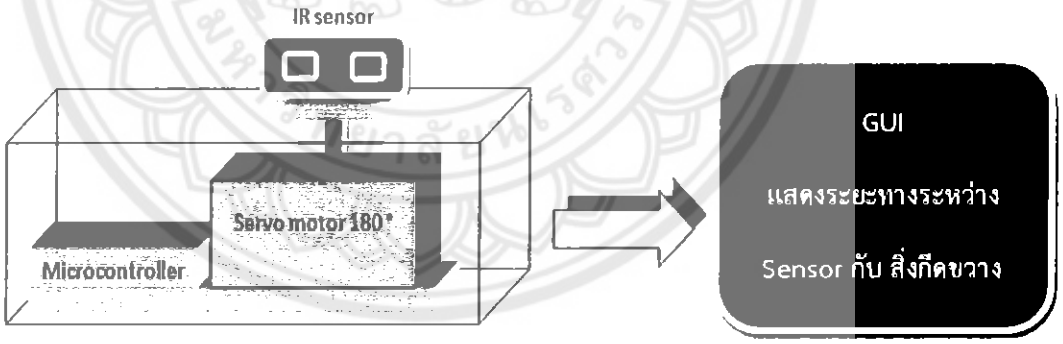
$$Vpw = \frac{\%\ dutycycleXVpower}{100} \qquad (2.2)$$

บทที่ 3
วิธีการดำเนินการ

จากทฤษฎีที่ได้กล่าวไว้ในบทที่ 2 สามารถนำมาประยุกต์ใช้ในการออกแบบวิธีการดำเนินการของสแกนเนอร์แบบอินฟราเรดสำหรับตรวจจับสิ่งกีดขวาง โดยในการออกแบบจะแบ่งส่วนการทำงานออกเป็น 2 ส่วน คือ ส่วนของสแกนเนอร์ ซึ่งเป็นอุปกรณ์ที่ทำหน้าที่ตรวจจับสิ่งกีดขวางเพื่อประมวลระยะห่างของสิ่งกีดขวาง และส่วนของการแสดงผลออกทางหน้าจอ ซึ่งเป็นส่วนของผู้ใช้งานในการรับทราบข้อมูลระยะห่างของสิ่งกีดขวาง หลังจากนั้นจะนำทั้ง 2 ส่วนมาเชื่อมต่อกัน เพื่อให้สามารถทำงานร่วมกันและแสดงค่าระยะห่างของสิ่งกีดขวางได้ตามต้องการ โดยในหัวข้อที่ 3.1 จะแสดงภาพรวมของระบบสแกนเนอร์อินฟราเรด

3.1ภาพรวมของระบบ

ภาพรวมของระบบจะเป็นการนำทั้งสองส่วนมาต่อเชื่อมเข้าด้วยกัน คือในส่วนของตัวสแกนเนอร์อินฟราเรดและส่วนการแสดงผลออกทางหน้าจอ (GUI) ซึ่งจะทำให้สามารถรับข้อมูลเข้ามาและส่งต่อข้อมูลเพื่อแสดงผลออกทางหน้าจอได้ทันที ดังแสดงภาพรวมของระบบในรูปที่ 3.1



รูปที่ 3.1 แสดงภาพรวมของระบบ

ในหัวข้อถัดไปเป็นการแสดงรายละเอียดทั้ง 2 ส่วน คือส่วนของตัวสแกนเนอร์และส่วนของการแสดงผลทางหน้าจอ GUI ดังนี้

3.2ส่วนสแกนเนอร์

3.2.1 หลักการทำงาน

ในส่วนของสแกนเนอร์อินฟราเรดจะเป็นตัวทำหน้าที่ตรวจจับสิ่งกีดขวางในในระยะห่างตั้งแต่ 20 เซนติเมตร ถึง 150 เซนติเมตร และในทิศทางตั้งแต่ -90 องศา ถึง +90 องศา ในแนวระนาบ เพื่อส่งค่าระยะห่างและทิศทางของตัวสแกนเนอร์อินฟราเรดกับสิ่งกีดขวางให้กับส่วนแสดงผลต่อไป

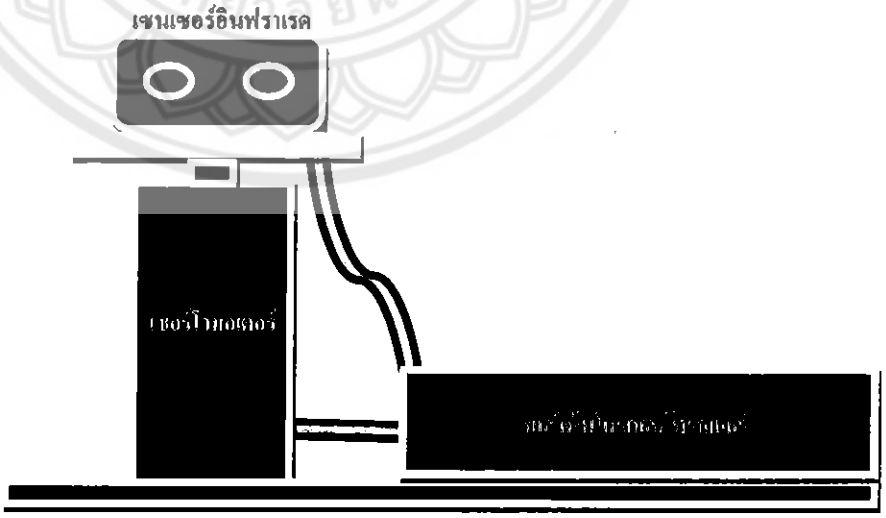
3.2.2 อุปกรณ์

1. เซอร์โวมอเตอร์ เป็นอุปกรณ์ที่ใช้ในการหมุนตัวสแกนเนอร์ โดยเซอร์โวมอเตอร์นี้สามารถควบคุมการหมุนแบบระบุเป็นองศาได้ตามความต้องการ ซึ่งในการทำงานจะกำหนดให้เซอร์โวมอเตอร์หมุนทีละ 5 องศา ตั้งแต่ -90 องศา จนถึง +90 องศา รวม 180 องศา เซอร์โวมอเตอร์ที่ใช้ในการทำงานคือ Futaba 3003
2. เซนเซอร์อินฟราเรด เป็นอุปกรณ์ที่ใช้ในการตรวจจับสิ่งกีดขวางของตัวสแกนเนอร์ เซนเซอร์อินฟราเรดที่ใช้คือ Sharp2Y0A02 Infrared Sensor ซึ่งสามารถตรวจจับสิ่งกีดขวางได้ตั้งแต่ระยะห่าง 20 เซนติเมตร ถึง 150 เซนติเมตร
3. ไมโครคอนโทรลเลอร์ เป็นอุปกรณ์ที่ใช้ในการควบคุม เซนเซอร์อินฟราเรด เซอร์โวมอเตอร์ และเชื่อมต่อกับเครื่องคอมพิวเตอร์เพื่อส่งข้อมูลต่างๆ ไมโครคอนโทรลเลอร์ที่ใช้คือ PIC 16F877
4. สายไฟต่อเชื่อม ใช้เชื่อมต่ออุปกรณ์กับบอร์ดไมโครคอนโทรลเลอร์
5. แบตเตอรี่ ใช้ในการจ่ายแรงดันไฟฟ้าปริมาณ 5 โวลต์ให้กับเซอร์โวมอเตอร์ แบตเตอรี่นี้ได้จาก ถ่านไฟฉายอัลคาไลน์ขนาด AA ปริมาณแรงดันไฟฟ้าก้อนละ 1.5 โวลต์ จำนวน 4 ก้อน ต่อแบบอนุกรม เท่ากับ 6 โวลต์ ซึ่งเพียงพอต่อความต้องการของเซอร์โวมอเตอร์

6. **โปรแกรม MiKroC** โปรแกรมนี้ใช้สำหรับเขียนคำสั่งควบคุมการทำงานของอุปกรณ์ฮาร์ดแวร์ ได้แก่ เซอร์โวมอเตอร์ อินฟราเรดเซนเซอร์ การรับส่งข้อมูลผ่านบอร์ดไมโครคอนโทรลเลอร์และเครื่องคอมพิวเตอร์ ภาษาที่ใช้เขียนคือ ภาษาซี
7. **สาย RS-232** เป็นสายที่ทำหน้าที่สื่อสารระหว่างบอร์ดไมโครคอนโทรลเลอร์กับเครื่องคอมพิวเตอร์ซึ่งเป็นการสื่อสารแบบอนุกรม

3.2.3 การออกแบบการทำงานของสแกนเนอร์

การออกแบบอินฟราเรดสแกนเนอร์นี้ ใช้เซนเซอร์อินฟราเรดเป็นตัวตรวจจับสิ่งกีดขวางซึ่งวัดระยะตั้งแต่ 20 เซนติเมตรถึง 150 เซนติเมตร โดยติดตั้งอยู่บนเซอร์โวมอเตอร์ที่กำหนดให้หมุนไปในบริเวณ180 องศาทิศทางด้านหน้าในแนวระนาบ ทีละ 5 องศาต่อ 1 สเต็ปจนครบ 180 องศาโดยที่สิ่งกีดขวางนั้นๆต้องมีความสูงไม่น้อยกว่าความสูงของตัวอินฟราเรดสแกนเนอร์และมีความกว้างไม่น้อยกว่าความกว้างของตัวอินฟราสแกนเนอร์โดยทั้งเซนเซอร์อินฟราเรดและเซอร์โวมอเตอร์จะเชื่อมต่อและถูกควบคุมโดยบอร์ดไมโครคอนโทรลเลอร์ PIC16F877 ภายในบอร์ดไมโครคอนโทรลเลอร์จะมีโปรแกรมควบคุมการหมุนของเซอร์โวมอเตอร์และรับแรงดันไฟฟ้าที่ได้จากการวัดของอินฟราเรดเซนเซอร์มาแปลงเป็นสัญญาณดิจิตอล แล้วทำหน้าที่ส่งต่อข้อมูลให้เครื่องคอมพิวเตอร์เพื่อใช้ในการแสดงผลออกทางหน้าจอ การออกแบบตัวสแกนเนอร์สามารถแสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 แสดงลักษณะการออกแบบตัวสแกนเนอร์

3.2.4 การควบคุมอุปกรณ์

การควบคุมการทำงานของอินฟราเรดสแกนเนอร์ใช้โปรแกรม MiKroC ในการเขียนคำสั่งต่างๆโดยจะอธิบายคำสั่งควบคุมของอุปกรณ์แต่ละตัว ดังนี้

3.2.4.1 เซอร์โวมอเตอร์

การสั่งงานควบคุมเซอร์โวมอเตอร์จะเป็นลักษณะคำสั่งที่ควบคุมตำแหน่งการหมุนและความเร็วซึ่งในโครงงานชิ้นนี้ต้องการให้เซอร์โวมอเตอร์หมุนทีละ 5 องศา ตั้งแต่ -90 องศาถึง +90 องศา โดยการทำงานของเซอร์โวมอเตอร์จะถูกควบคุมโดยสัญญาณพัลส์ซึ่งมีความกว้างอยู่ระหว่าง 1 มิลลิวินาที ถึง 2 มิลลิวินาที (พัลส์บวกหรือลอจิก1) การส่งสัญญาณพัลส์มีผลทำให้เซอร์โวมอเตอร์หมุนโดยทิศทางการหมุนนั้นจะขึ้นอยู่กับความกว้างของพัลส์บวก ส่วนความเร็วขึ้นอยู่กับขึ้นอยู่กับความกว้างของพัลส์ลบ ในรูปที่ 3.3 3.4 และ 3.5 แสดงตำแหน่งของเซอร์โวมอเตอร์ที่สัญญาณพัลส์ค่าต่างๆ

รูปที่ 3.3แสดงสัญญาณพัลส์ที่ 1.5 มิลลิวินาที ตำแหน่งของแซอร์โวมอเตอร์อยู่ที่ 0 องศา

รูปที่ 3.4แสดงสัญญาณพัลส์ที่ 1 มิลลิวินาที ตำแหน่งของแซอร์โวมอเตอร์อยู่ที่ -90 องศา

รูปที่ 3.5แสดงสัญญาณพัลส์ที่ 2 มิลลิวินาที ตำแหน่งของแซอร์โวมอเตอร์อยู่ที่ +90 องศา

Pseudo code สำหรับควบคุมตำแหน่งของเซอร์โวมอเตอร์

การควบคุมตำแหน่งของเซอร์โวมอเตอร์ใน โปรแกรม MiKroC จะต้องรู้**สัญญาณพัลส์เริ่มต้น**ของเซอร์โวมอเตอร์ที่ใช้ ซึ่งสัญญาณพัลส์เริ่มต้นของเซอร์โวมอเตอร์ Futaba 3003 **อยู่ที่** 2320 ไมโครวินาที ที่ตำแหน่ง -90 องศา สัญญาณพัลส์ 1420 ไมโครวินาที ที่ตำแหน่ง 0 **องศาและสัญญาณ**พัลส์ 520ไมโครวินาที ที่ตำแหน่ง -90 องศาโดยจะส่งสัญญาณเข้า PORTB.F1 **ภายในฟังก์ชั่น** angle_XXX() แล้วกำหนดให้วนรอบแบบไม่รู้จบ ดังนี้

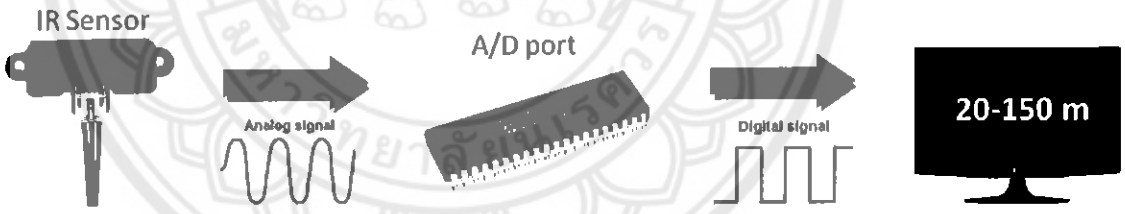
```
01:  TRISB.F1=0x00;          // ประกาศให้พอร์ต B.F1 เป็นพอร์ตเอาท์พุต
02:  while(1) {              // สั่งให้วนรอบแบบไม่รู้จบ
03:  void angle_XXX()        // ประกาศชื่อฟังก์ชั่น angle_XXX
04:  for(i=1;i<=50;i++)      // ควบคุมความถี่ภายในเซอร์โวมอเตอร์เท่ากับ 50 เฮิรตซ์
05:  PORTB.F1=1;             // พอร์ต B.F1=1 ทำงาน
06:  Delay_us(2320);         //สั่งให้เซอร์โวมอเตอร์หมุนไปที่ตำแหน่ง -90 องศาที่สัญญาณพัลส์ 2320 us
07:  PORTB.F1=0;             // พอร์ต B.F1=0หยุดการทำงาน
08:  Delay_ms(12); }         // หน่วงเวลา 12 มิลลิวินาที
09:  void angle_000()        // ประกาศชื่อฟังก์ชั่น angle_000
10:  for(i=1;i<=50;i++)      // ควบคุมความถี่ภายในเซอร์โวมอเตอร์เท่ากับ 50 เฮิรตซ์
11:  PORTB.F1=1;             // พอร์ต B.F1=1 ทำงาน
12:  Delay_us(1420);         //สั่งให้เซอร์โวมอเตอร์หมุนไปที่ตำแหน่ง 0 องศาที่สัญญาณพัลส์ 1420 us
13:  PORTB.F1=0;             // พอร์ต B.F1=0หยุดการทำงาน
14:  Delay_ms(12);           // หน่วงเวลา 12 มิลลิวินาที
15:  void angle_190()        // ประกาศชื่อฟังก์ชั่น angle_190
```

```
16:   for(i=1;i<=50;i++)      // ควบคุมความถี่ภายในเซอร์โวมอเตอร์เท่ากับ 50 เฮิรตซ์
17:   PORTB.F1=1;             // พอร์ต B.F1=1 ทำงาน
18:   Delay_us(520);          //สั่งให้เซอร์โวมอเตอร์หมุนไปที่ตำแหน่ง +90 องศาที่สัญญาณพัลส์ 520 us
19:   PORTB.F1=0;             // พอร์ต B.F1=0หยุดการทำงาน
20:   Delay_ms(12); }         // หน่วงเวลา 12 มิลลิวินาที
```

จาก Pseudo Code ข้างต้นเป็นเพียงการหมุน 3ครั้ง จากทั้งหมด 36 ครั้ง ค่าของสัญญาณพัลส์จะลดลงครั้งละ 50 ไมโครวินาทีจาก 2320 ไมโครวินาที จนถึง 520 ไมโครวินาที จากใน Pseudo Code ข้างต้น หากต้องการเปลี่ยนองศาของเซอร์โวมอเตอร์จะต้องเปลี่ยนตัวเลขสัญญาณพัลส์ที่ Delay_us() (บรรทัดที่ 06)

3.2.4.2 อินฟราเรดเซนเซอร์

การสั่งงานอินฟราเรดเซนเซอร์จะเป็นการสั่งให้อินฟราเรดเซนเซอร์ส่งค่าแรงดันไฟฟ้าที่รับเข้ามาไปยังไมโครคอนโทรลเลอร์เพื่อแปลงแรงดันไฟฟ้าเป็นสัญญาณดิจิตอลแล้วแปลงเป็นค่าระยะห่างระหว่างอินฟราเรดเซนเซอร์กับสิ่งกีดขวางแสดงในเครื่องคอมพิวเตอร์ ดังที่แสดงในรูปที่3.6



รูปที่ 3.6 แสดงการทำงานของอินฟราเรดเซนเซอร์

การรับ-ส่งข้อมูลผ่านสายอนุกรม RS-232

การส่งข้อมูลผ่าน RS-232 เป็นการสื่อสารระหว่างไมโครคอนโทรลเลอร์กับเครื่องคอมพิวเตอร์เป็นโปรโตคอลที่ง่าย รวดเร็ว เนื่องจากมีปริมาณข้อมูลไม่มาก ในการรับส่งข้อมูลระหว่างบอร์ดไมโครคอนโทรลเลอร์กับเครื่องคอมพิวเตอร์ มีโปรโตคอลดังนี้

ZERO (3-4 byte)	DATA (2-3 byte)	'/n'

โดยที่

- ZERO คือ เติมข้อมูลที่รับเข้ามาให้ครบ 6 ตัว
- DATA คือ ข้อมูลจากไมโครคอนโทรลเลอร์ที่ส่งให้กับเครื่องคอมพิวเตอร์
- '/n' คือ การเว้นบรรทัดใหม่

Pseudo Code สำหรับรับส่งค่าจากอินฟราเรดเซนเซอร์

ในการแปลงค่าจากแรงดันไฟฟ้าเป็นสัญญาณดิจิตอลจะเรียกใช้พอร์ต Analog to Digital (A/D) ที่อยู่ในบอร์ดไมโครคอนโทรลเลอร์ในรีจีสเตอร์ ADcon1 เมื่อพอร์ต A/D แปลงค่าแล้วจะส่งสัญญาณดิจิตอลเข้าสู่เครื่องคอมพิวเตอร์ โดยใช้โปรแกรม MiKroC ในการเขียนคำสั่งรับส่งค่าเพื่อติดต่อกับอินฟราเรดเซนเซอร์ บอร์ดไมโครคอนโทรลเลอร์และเครื่องคอมพิวเตอร์ ที่อยู่ในฟังก์ชั่น LLCD()โดยมีPseudo Code ดังนี้

```
01:  void LLCD() {
02:  unsigned int analog;          // ประกาศตัวแปร analog
03:  char txt[4];                  // เก็บค่าในรูปแบบตัวอักษร
04:  ADcon1 = 0;                   // รีจีสเตอร์ควบคุมเกี่ยวกับ A/D
05:  UART1_Init(9600);             // สื่อสารข้อมูลอนุกรมกับอุปกรณ์อื่น
06:  while(1)              // การวนรอบแบบไม่รู้จบ
07:  analog = ADC_Read(1);         // ตัวแปร analog รับค่าแรงดัน
08:  intTostr(analog,txt);         // แปลงสัญญาณอนาล็อกเป็นตัวอักษร
09:  UART1_Write_text(txt);        // ส่งข้อมูลออกทางพอร์ตอนุกรม
10:  delay_ms(500);          }     // หน่วงเวลา 500 มิลลิวินาที
```

Pseudo Code การทำงานของสแกนเนอร์อินฟราเรด

เมื่อโปรแกรมให้อุปกรณ์แต่ละตัวทำงานตามที่ต้องการได้แล้ว โดยโปรแกรมให้อุปกรณ์ทั้งสองตัวให้ทำงานร่วมกันซึ่งรวมเรียกว่าอินฟราเรดสแกนเนอร์ ให้เซอร์โวมอเตอร์รับอินพุตจาก PORTB ส่วนเซนเซอร์อินฟราเรดรับอินพุตจาก PORTA โดยฟังก์ชั่น main() จะมีการเรียกฟังก์ชั่นย่อย angle_XXX() และ LLCD() เพื่อเรียกใช้งานเซอร์โวมอเตอร์และเซนเซอร์อินฟราเรด มอเตอร์จะเริ่ม

หมุนตั้งแต่ -90 องศา ถึง +90 องศา ตามลำดับโดยมีเงื่อนไขว่า หาก UART1_Data_Ready = 0 เซอร์โวมอเตอร์จะกลับไปที่ angle_090() หรือตำแหน่งเริ่มต้นตาม Pseudo Code ด้านล่างนี้

```
01:  unsigned int analog              // ประกาศตัวแปร analog
02:  unsigned char receive            // ประกาศตัวแปร receive
03:  unsigned    int i                // ประกาศตัวแปร i
04:  char txt[4]                      // เก็บค่าในรูปแบบตัวอักษร
05:  int j                            // ประกาศตัวแปร j
06:  void main() {                    // ฟังก์ชั่น main()
07:  TRISB.F1=0x00;                   // ประกาศให้ PORTB.F1 รับเอาท์พุต
08:  ADcon1 = 0                       // รีจีสเตอร์ควบคุมเกี่ยวกับ A/D
09:  UART1_Init(9600)                 // สื่อสารข้อมูลอนุกรมกับอุปกรณ์อื่น
10:  while(1) {                       // การวนรอบแบบไม่รู้จบ
11:   if(UART1_Data_Ready())          // ตรวจสอบการทำงานรับข้อมูล
12:  receive = UART1_Read()           // ให้ตัวแปร receive เก็บค่าข้อมูลที่รับเข้ามา
13:  LLCD()                           // ประกาศชื่อฟังก์ชั่น LLCD()
14:  angle_XXX()                      // ประกาศชื่อฟังก์ชั่น angle_XXX()
15:  if(UART1_Data_Ready==0)          // ตรวจสอบการทำงานรับข้อมูล เท่ากับ 0 ไม่พร้อม
16:  angle_090() }                    // ประกาศชื่อฟังก์ชั่น angle_090
```

3.2.5 วิธีการทำงาน

1. นำตัวอินฟราเรดสแกนเนอร์ติดตั้งในบริเวณอาคารที่มีสภาวะควบคุม คือ บริเวณที่ทดลองต้องไม่มีแสงอาทิตย์และแสงอินฟราเรดรบกวนเพราะในแสงอาทิตย์มีแสงอินฟราเรดซึ่งอาจจะรบกวนการทำงานของให้อินฟราเรดเซนเซอร์
2. เมื่อตัวอินฟราเรดสแกนเนอร์ทำงานจะเริ่มสแกนที่มุม -90 องศา แล้วหมุนเพิ่มขึ้นทีละ 5 องศา จนถึง +90 องศา ตามลำดับรวมเป็น 180 องศา แล้วเริ่มกลับมาทำงานที่ -90 องศา อีกครั้ง โดยสแกนไปข้างหน้าในแนวระนาบ

3. อินฟราเรดสแกนเนอร์สามารถตรวจจับสิ่งกีดขวางเริ่มที่ระยะห่าง20 เซนติเมตร จนถึง 150 เซนติเมตร
4. เมื่ออินฟราเรดสแกนเนอร์ตรวจพบสิ่งกีดขวางแล้ว จะส่งค่าแรงดันไฟฟ้าให้กับไมโครคอนโทรลเลอร์เพื่อแปลงแรงดันไฟฟ้าเป็นสัญญาณดิจิตอล แล้วส่งค่าสัญญาณนั้นแสดงออกทางหน้าจอต่อไป

3.3 ส่วนแสดงผลทางหน้าจอ

3.3.1 หลักการทำงาน

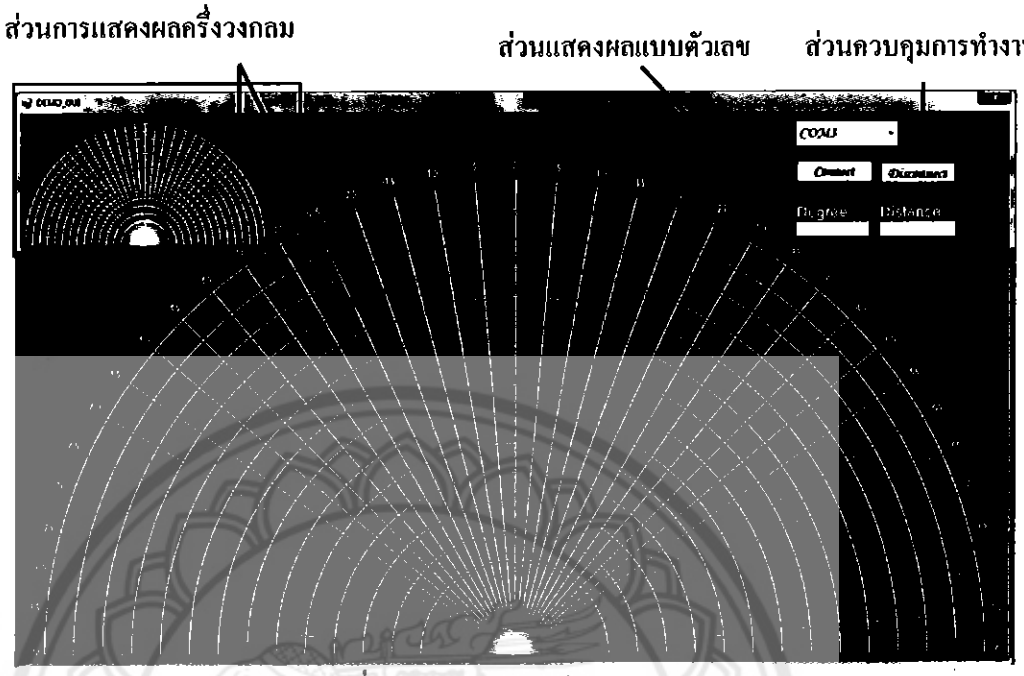
ส่วนการแสดงผลทางหน้าจอจะรับค่าจาก A/D แล้วทำการแปลงสัญญาณที่ได้รับเป็นระยะห่างของตัวสแกนเนอร์กับสิ่งกีดขวางแล้วแสดงออกผลทางหน้าจอ GUI ที่พัฒนาขึ้น เพื่อให้ผู้ใช้งานสะดวกในการตรวจสอบทิศทางและระยะห่างของสิ่งกีดขวางได้

3.3.2 อุปกรณ์

1. หน้าจอแสดงผล เป็นเครื่องมือที่เชื่อมต่อกับเครื่องคอมพิวเตอร์สำหรับแสดงผล
2. โปรแกรม Microsoft Visual Studio C# เป็นโปรแกรมที่ใช้สร้างส่วนแสดงผลทางหน้าจอและติดต่อรับค่าอินพุตจากไมโครคอนโทรลเลอร์

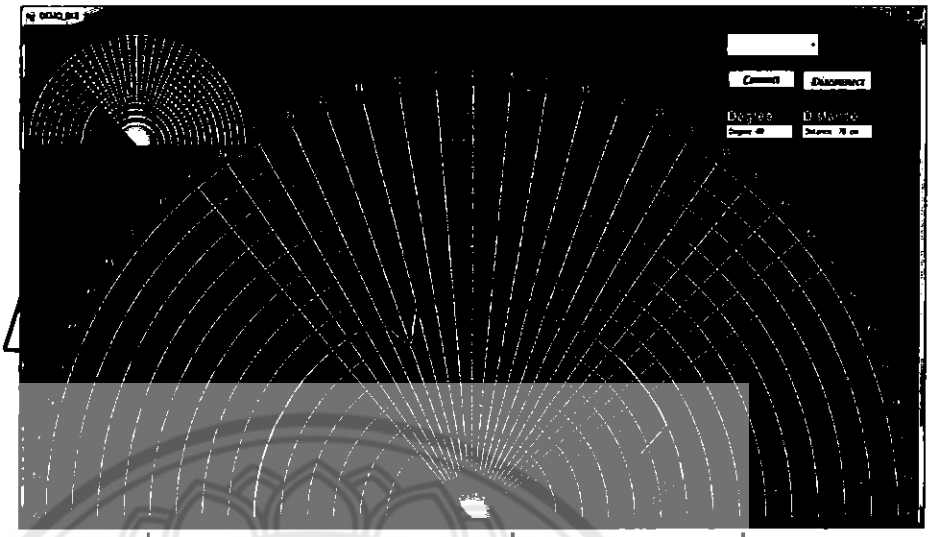
3.3.3 การออกแบบการทำงานของส่วนแสดงผลทางหน้าจอ

การออกแบบส่วนแสดงผลจะใช้โปรแกรม Microsoft Visual Studio C# เป็นเครื่องมือในออกแบบแสดงผลการทำงาน และติดต่อรับค่าจากไมโครคอนโทรลเลอร์ผ่านสายอนุกรม RS-232 ในส่วนหน้าจอแสดงผลจะแสดงระยะห่างของตัวสแกนเนอร์กับสิ่งกีดขวางในรูปแบบของกราฟครึ่งวงกลมและในรูปแบบตัวเลข โดยมีปุ่ม Connect เพื่อเริ่มการทำงาน ปุ่ม Disconnect เพื่อหยุดการทำงาน ช่อง Distance แสดงตัวเลขระยะห่าง และช่องเลือกพอร์ตการทำงาน ดังแสดงรูปที่ 3.7



รูปที่ 3.7 แสดงส่วนหน้าจอแสดงผล

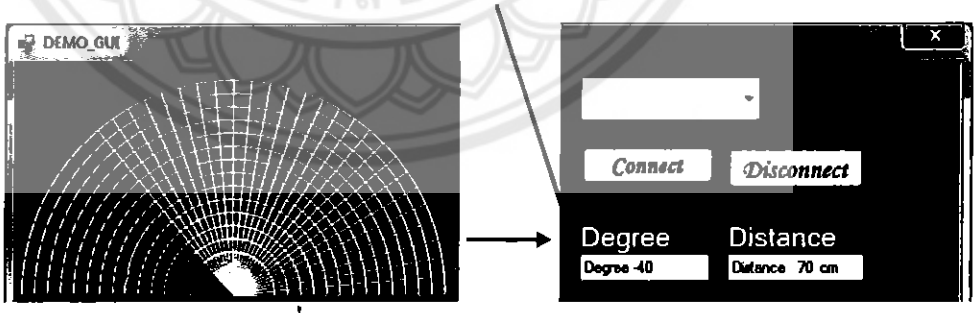
การแสดงในรูปแบบกราฟครึ่งวงกลมจะเริ่มที่ -90 องศา ไปจนถึง +90 องศาโดยมีเส้นแบ่งทุกๆ 5 องศาตามการหมุนของตัวสแกนเนอร์ รวมมีเส้นแบ่งองศา 36 เส้น และมีเส้นที่ใช้ในการแบ่งระยะห่างเริ่มที่ 20 เซนติเมตร จนถึง 150 เซนติเมตร ห่างกันเส้นละ 10 เซนติเมตร รวม 14 เส้นโดยรูปครึ่งวงกลมด้านบนแสดงการสแกนหาสิ่งกีดขวางในเวลาปัจจุบัน เมื่อครบรอบแล้วจะแสดงผลการสแกนไว้ที่ครึ่งวงกลมใหญ่ เมื่อกดปุ่ม Connect กราฟจะแสดงเส้นบอกตำแหน่งของสิ่งกีดขวาง โดยเริ่มสแกนที่ -90 องศา แล้วขยับไปทีละ 5 องศา จนถึง +90 องศาเพื่อบอกทิศทางและระยะห่างของสิ่งขวางและเมื่อกดปุ่ม Disconnect จะเคลียร์ภาพบนกราฟแล้วกลับไปจุดเริ่มต้น ดังรูปที่ 3.8



รูปที่ 3.8 แสดงเส้นบอกตำแหน่งของสิ่งกีดขวางบนกราฟครึ่งวงกลม

ส่วนรูปแบบการแสดงผลแบบตัวเลขจะรับค่าจาก A/D แล้วนำมาประมวลผล เพื่อกำหนดระยะห่างระหว่างตัวสแกนเนอร์กับสิ่งกีดขวางตั้งแต่ 20 เซนติเมตร ถึง 150 เซนติเมตร เมื่อเลือกพอร์ตสำหรับติดต่อสื่อสาร กดปุ่ม Connect จะแสดงค่าระยะห่างที่ตัวสแกนเนอร์วัดได้เป็นตัวเลขในช่อง Distance ตั้งแต่ 20 เซนติเมตร ถึง 150 เซนติเมตร หากต้องการจบการทำงานให้กดปุ่ม Disconnect ดังรูปที่ 3.9 แสดงส่วนแสดงผลแบบตัวเลขและมุม

การแสดงผลแบบตัวเลขสัมพันธ์กับกราฟครึ่งวงกลม



รูปที่ 3.9 แสดงส่วนแสดงผลแบบตัวเลขและมุม

3.3.4 ขั้นตอนการทำงาน

3.3.4.1 ไลบรารี่ของ Microsoft Visual Studio C# ที่ใช้งาน

Microsoft Visual Studio C# เป็นโปรแกรมที่ประกอบด้วยไลบรารี่มากมายเพื่อตอบสนองการทำงานในด้านต่างๆไม่ว่าจะเป็นด้านการออกแบบสร้างกราฟิก สามารถทำได้หลายรูปแบบ เช่นการออกแบบสร้าง GUI หรือการออกแบบหน้าเว็บไซต์ เนื่องจากมีไลบรารี่ที่จำเป็นต่อความต้องการ ไม่ว่าจะเป็นการวาดรูปพื้นฐาน รวมถึงการรับค่าจากภายนอก ซึ่งในการใช้เรียกใช้ไลบรารี่จะต้อง include ก่อนใช้งาน โดยการเรียก using namespace ตามด้วยชื่อไลบรารี่ที่ต้องการใช้งาน โดยมีไลบรารี่ที่ใช้ในการดำเนินโครงงาน ดังนี้

3.3.4.2 การสร้างหน้าจอแสดงผล Graphic User Interface (GUI)

ในการสร้างรูปครึ่งวงลกมที่ใช้ในการบอกตำแหน่งของวัตถุ ใช้โปรแกรม Illustrator [19] ในการวาดเนื่องจากเป็นโปรแกรมที่เหมาะกับการออกแบบ มีเครื่องมือในการแบ่งเส้นองศาตามที่ต้องการได้ เมื่อได้รูปครึ่งวงกลมแล้วจะนำรูปครึ่งวงกลมมาเข้าโปรแกรม Microsoft Visual Studio C# โดยใช้คำสั่ง BackgroundImage ในการนำรูปภาพเข้าเป็นพื้นหลังของส่วนแสดงผล โดยก่อนการวาดภาพต้องประกาศ using System.Drawing และ using System.Drawing2D ก่อนจึงจะใช้ฟังก์ชั่นวาดภาพได้ จากนั้นสร้างส่วนการควบคุมและส่วนแสดงแบบตัวเลข โดยการเพิ่มปุ่มและช่องต่างๆ ได้แก่ ปุ่ม Connect ปุ่ม Disconnect ช่องแสดงผล และปุ่มเลือกพอร์ตการทำงาน ซึ่งอยู่ในส่วน Toolbox โดยขนาดของส่วนแสดงผลจะขึ้นอยู่กับขนาดรูปภาพที่รับเข้ามาและการกำหนดขนาดหน้าจอด้วย

3.3.4.3 ส่วนการติดต่อรับค่าจากภายนอก

หลังจากที่ได้สร้างส่วนแสดงผลแล้ว จากนั้นเป็นการรับค่าระยะห่างจากเซนเซอร์ที่ผ่านบอร์ดไมโครคอนโทรลเลอร์ เพื่อนำมาแสดงตำแหน่งของสิ่งกีดขวางบนหน้าจอแสดงผล GUI เมื่อต้องการรับค่าจากภายนอกต้องประกาศ using System.IO.Ports และ using System.IO ก่อน จากนั้นสร้างปุ่ม Serial Port แล้วรับค่าจากสายอนุกรม RS-232 ด้วยฟังก์ชั่น SerialPort.Read

Pseudo Code ของการติดต่อรับค่าจากภายนอก

การรับค่าจาก Serial Port ในภาษา C# นั้นจะต้องอยู่ในรูปแบบของออฟเจ็คและอีเวนต์ คือ โปรแกรมจะทำงานเองอัตโนมัติเมื่อมีค่าเข้ามายัง Serial Port ดังนี้

```
01:   str += this.serialPort1.ReadExisting()      // str เก็บข้อมูลที่รับจากเซนเซอร์ผ่านทางserialPort
02:   char[] spit = {' '}                         // กำหนดช่องว่างในการแยกข้อมูล
03:   string[] Word = str.Split(spit, StringSplitOptions.RemoveEmptyEntries)
                                                  // แยกข้อมูลโดยการใช้ช่องว่างในการแยก
04:   value = Int32.Parse(Word)                   // แปลงค่าข้อมูลจาก String เป็น int เพื่อใช้เทียบค่า
                                                  เป็นระยะห่าง เก็บไว้ในตัวแปร value
```

3.3.4.4 ส่วนการแสดงเส้นตำแหน่งบนหน้าจอจาก Serial Port

เมื่อสามารถรับค่าจากภายนอกได้แล้ว ขั้นตอนต่อไปจะเป็นการนำค่าที่ได้รับเข้ามาจากไมโครคอนโทรลเลอร์ผ่านสาย RS-232 มาแสดงเป็นในกราฟครึ่งวงกลมและตัวเลขการแสดงตำแหน่งของสิ่งกีดขวางในหน้าจอจะสัมพันธ์กับระยะจริงที่วัตถุอยู่ โดยเมื่อตัวสแกนเนอร์ตรวจพบสิ่งกีดขวางจะส่งค่าแรงดันไฟฟ้าเข้าสู่ไมโครคอนโทรลเลอร์เพื่อแปลงเป็นสัญญาณดิจิตอล จากนั้นเครื่องคอมพิวเตอร์ทำการรับค่าสัญญาณดิจิตอล แล้วตรวจสอบว่าสัญญาณที่รับเข้ามาอยู่ในช่วงระยะห่างเท่าใด แล้วนำค่าระยะห่างนั้นแสดงผลแบบตัวเลข ในขณะเดียวกันก็นำไปคำนวณเพื่อแสดงตำแหน่งบนกราฟครึ่งวงกลมตามตำแหน่งระยะห่างและทิศทางของวัตถุนั้นๆ

การวาดเส้นบอกตำแหน่งในกราฟครึ่งวงกลมก่อนอื่นจะต้องรู้ตำแหน่งต่างๆ (X, Y) ภายในกราฟ ซึ่งหาตำแหน่ง (X, Y) ภายในกราฟได้จาก สมการที่ 4.1และ 4.2 ตามลำดับ

$$Y_n = Y_1 - \left(\frac{Y_1 - Y_2}{130}\right) D \qquad (4.1)$$

และ

$$X_n = X_1 - \left(\frac{X_1 - X_2}{130}\right) D \qquad (4.2)$$

โดยที่ X_n คือ ตำแหน่งของจุดในแกน X

Y_n คือ ตำแหน่งของจุดในแกน Y

X_1, X_2 คือ ตำแหน่งเริ่มต้นและสุดท้ายในแกน X

Y_1, Y_2 คือ ตำแหน่งเริ่มต้นและสุดท้ายในแกน Y

D คือ ระยะห่างของวัตถุ

เมื่อทราบตำแหน่ง (X, Y) แล้วจึงนำมาวาดเป็นกราฟแสดงบนหน้าจอแสดงผล

Pseudo Code การแสดงตำแหน่งของวัตถุบนหน้าจอแสดงผล (GUI)

ในส่วนการแสดงตำแหน่งของวัตถุบนหน้าจอแสดงผลนี้จะแบ่ง Pseudo Code ออกเป็น 3 ส่วน ดังนี้

ส่วนที่ 1 Pseudo Code ในการเทียบช่วงระยะห่างจากสัญญาณดิจิตอลที่รับเข้ามาจาก Serial Port ดังนี้

```
01:  void checkAndShow (int value)          // ประกาศฟังก์ชั่น checkAndShow
02:  if (value >= 461 && value <= 524)      // เช็คช่วงค่าใน value
03:  if (value >= 511 && value < 524)       // เช็คช่วงค่าใน value
04:  distanceValue = 20                     // กำหนดค่าระยะ ไว้ใน distanceValue
05:  else if (value >= 500 && value <= 510) // เช็คช่วงค่าใน value
06:  distanceValue = 21                     // กำหนดค่าระยะ ไว้ใน distanceValue
```

ในการกำหนดระยะห่างของวัตถุในข้างต้น ใช้วิธีการเทียบค่าจากการวางสิ่งกีดขวางไว้ในระยะจริง แล้วสังเกตค่าสัญญาณดิจิตอลที่แสดงบนหน้าจอว่าอยู่ในช่วงใด เมื่อทราบแล้วจึงทำการกำหนดแต่ละช่วงระยะห่างดัง Pseudo Code ข้างต้น ตั้งแต่ 20 เซนติเมตร ถึง 150 เซนติเมตร

ส่วนที่ 2 Pseudo Code ในการแสดงผลแบบตัวเลขและมุม

```
showDistance.Text = "Distance " + distanceValue.ToString() + " cm" // แสดงผลระยะทางออกหน้าจอ
show_Angle.Text = "Degree "+angleShow.ToString();                  // แสดงผลองศาออกหน้าจอ
```

ส่วนที่ 3 Pseudo Code ในการแสดงผลแบบกราฟครึ่งวงกลม

```
01:   privatevoid addlist()                    // ฟังชันก์เก็บค่าตำแหน่ง (X,Y)
02:   index++;                                 // ตัวแปร index
03:   listP.Add(pointX,pointY));               // เพิ่มค่าตำแหน่ง (X,Y)เก็บไว้ในลิสต์
04:   Drawgraph();                             // เรียกใช้ฟังก์ชั่น Drawgraph()
05:   privatevoid Drawgraph()                  // ฟังชันDrawgraph()
06:   if(index==0){}                          // เช็คค่า index ถ้าเป็น 0 จะไม่ทำการใดๆ
07:   else {DrawLine(listP[index-1].X,listP[index-1].Y,listP[index].X,listP[index].Y);}
                                               // ถ้า index ไม่เท่ากับ 0 จะทำการวาดเส้นโดยจะใช้
                                                ค่าตำแหน่ง (X, Y) ของ index ก่อนหน้า
                                                และ index ปัจจุบันมาวาดเป็นเส้น
```

3.3.5 วิธีการทำงาน

1. เมื่อตัวอินฟราเรดสแกนเนอร์แปลงสัญญาณ A/D ที่ได้จากอินฟราเรดเซนเซอร์แล้ว จะส่งค่านั้นผ่านสายอนุกรม RS-232 เพื่อเชื่อมต่อกับคอมพิวเตอร์ ที่ทำหน้าที่แสดงผล
2. หากต้องการเริ่มการทำงานให้เลือกพอร์ตการสื่อสาร (COM3) แล้วกดปุ่ม Connect ในหน้าจอแสดงผลค่าระยะห่างจาก A/D ที่ได้รับจากไมโครคอนโทรลเลอร์เพื่อแสดงผลในรูปแบบกราฟและแบบตัวเลข
3. เมื่อต้องการจบการทำงานกดปุ่ม Disconnect

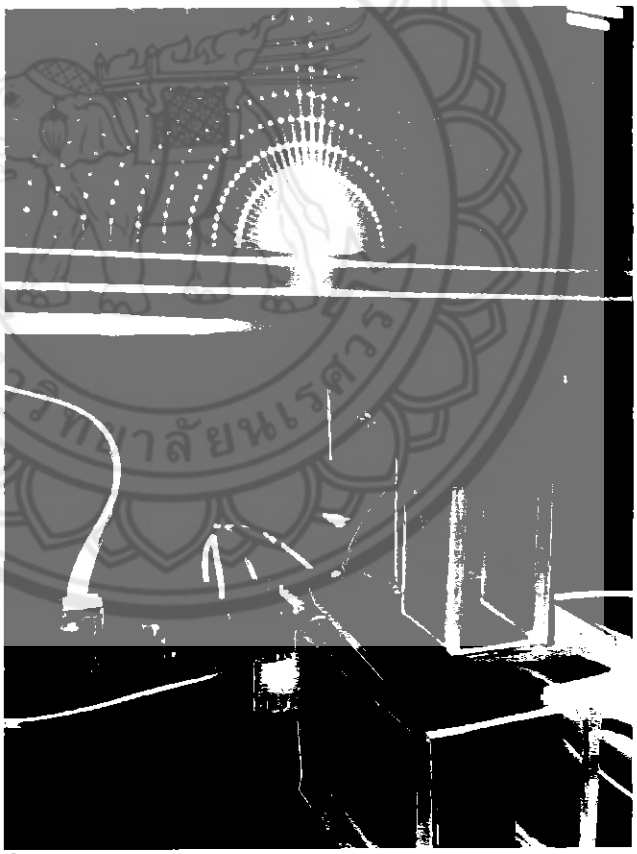
3.4 บทสรุป

ในบทที่ 3 นี้ได้กล่าวถึงการดำเนินงานทั้งหมดของระบบ ซึ่งอันดับแรกกล่าวถึงภาพรวมของระบบที่ได้พัฒนาขึ้น ว่าระบบนั้นมีองค์ประกอบอะไรบ้างและมีหลักการทำงานอย่างไร จากนั้นจะกล่าวถึงตัวสแกนเนอร์ที่ได้พัฒนาขึ้นว่าทำงานอย่างไรโดยอธิบายทีละส่วน เริ่มจากส่วนของเซอร์โวมอเตอร์ในด้านการทำงาน การออกแบบรวมถึงการโปรแกรมคำสั่งในการหมุนเซอร์โวมอเตอร์ ส่วนต่อไปเป็นเซนเซอร์อินฟราเรดโดยกล่าวถึงในด้านการทำงาน การออกแบบ การโปรแกรมคำสั่งในการรับส่งค่า รวมถึงการเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ในรูปแบบของ Pseudo Code ต่อมาจะ

กล่าวถึงการเขียนโปรแกรมแสดงผลออกทางหน้าจอ GUI โดยกล่าวตั้งแต่ไลบรารี่ที่จำเป็นในการใช้งาน การสร้างหน้าจอ การรับภาพ การรับส่งค่าจากบอร์ดไมโครคอนโทรลเลอร์ และการสร้างกราฟิกในการแสดงผลรวมทั้ง Pseudo Code ที่สำคัญในการโปรแกรมเพื่อสร้างหน้าจอแสดงผล

บทที่ 4
ผลการทดลอง

จากบทที่ 3 ที่ได้ทำการออกแบบตัวสแกนเนอร์อินฟราเรด ในขั้นตอนต่อไปจะเป็นการทดลองใช้งาน โดยในบทนี้จะเป็นการแสดงผลการทดสอบการทำงานในส่วนต่างๆของตัวสแกนเนอร์อินฟราเรด ระบบที่พัฒนาขึ้น คือระบบวัดระยะห่างของตัวสแกนเนอร์กับสิ่งกีดขวางทำหน้าที่บอกระยะห่างและทิศทางของวัตถุนั้นๆ เมื่อตัวสแกนเนอร์อินฟราเรดจับวัตถุได้แล้วจะส่งค่าแรงดันไฟฟ้าให้ไมโครคอนโทรลเลอร์เพื่อแปลงเป็นสัญญาณดิจิตอลผ่านสายอนุกรม RS-232 เข้าสู่เครื่องคอมพิวเตอร์ เพื่อแสดงตำแหน่งของวัตถุนั้นๆในหน้าจอแสดงผล GUI โดยจะแสดงผลเป็นแบบกราฟครึ่งวงกลมและแบบตัวเลข รูปที่ 4.1 แสดงภาพจริงของระบบอินฟราเรดสแกนเนอร์ที่พัฒนาขึ้น



รูปที่ 4.1 แสดงภาพจริงของระบบอินฟราเรดสแกนเนอร์

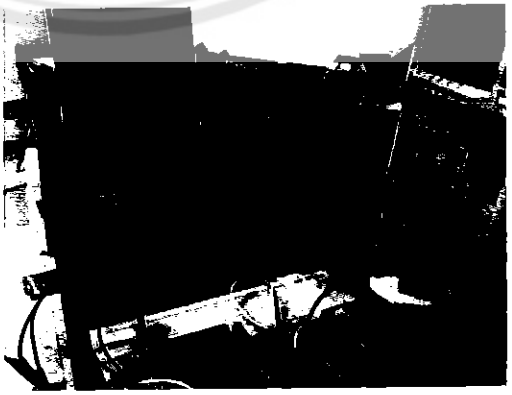
ในส่วนของหน้าจอแสดงผลจะประกอบไปด้วย 3 ส่วน ดังนี้

1. ส่วนควบคุมการทำงาน ในส่วนควบคุมการทำงานจะประกอบไปด้วย ปุ่มเลือกพอร์ตการทำงาน ปุ่ม Connect เพื่อเริ่มการทำงานของระบบ และปุ่ม Disconnect เพื่อหยุดการทำงานของระบบ
2. ส่วนแสดงผลแบบกราฟครึ่งวงกลม ในส่วนนี้จะแสดงตำแหน่งของวัตถุในหน้าจอ ตามตำแหน่งจริงที่วัตถุปรากฏอยู่ ซึ่งบอกทั้งทิศทางและระยะห่างของวัตถุนั้นๆ โดยจะมีเส้นบอกตำแหน่งสแกนในกราฟครึ่งวงกลมตามการสแกนของตัวสแกนเนอร์
3. ส่วนการแสดงผลแบบตัวเลข ในส่วนนี้จะแสดงระยะห่างระหว่างตัวสแกนเนอร์กับวัตถุเป็นตัวเลข ตั้งแต่ 20 เซนติเมตร ถึง 150 เซนติเมตร และแสดงทิศทางเป็นตัวเลขมุมตั้งแต่ -90 องศา ถึง +90 องศา ตามการสแกนของตัวสแกนเนอร์

4.1 อินฟราเรดเซนเซอร์

4.1.1 การทดลองวัดแรงดันไฟฟ้าของเซนเซอร์

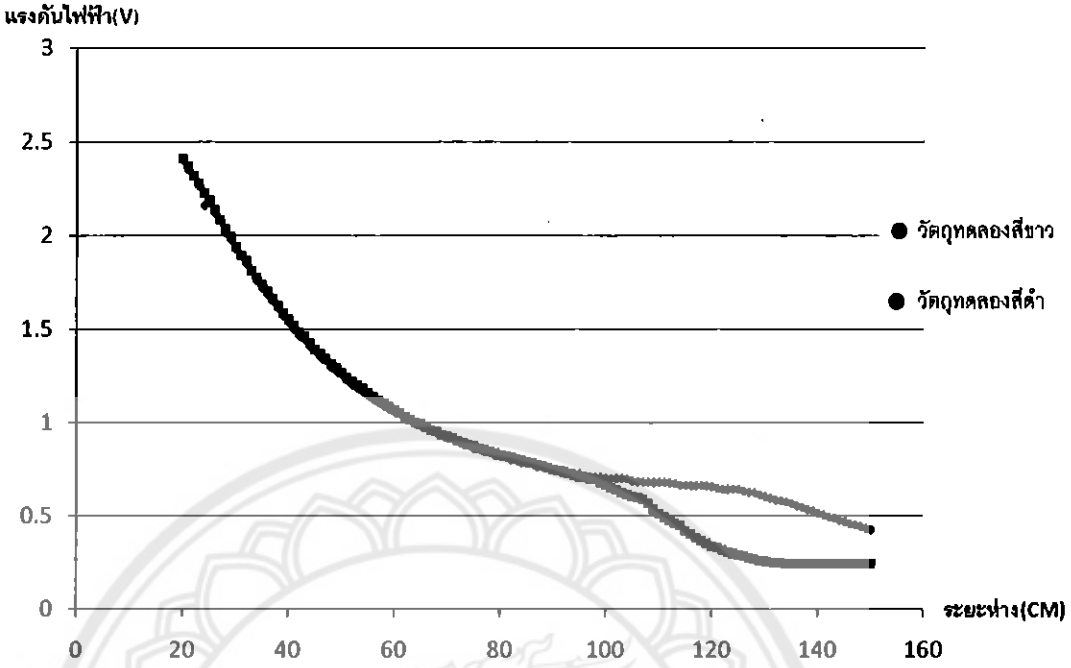
ในการทดสอบนี้จะทดสอบโดยการวัดระยะห่างระหว่างเซนเซอร์กับสิ่งกีดขวาง เพื่อหาช่วงแรงดันไฟฟ้าของเซนเซอร์ ตั้งแต่ 20 เซนติเมตร ถึง 150 เซนติเมตร โดยนำแรงดันไฟฟ้าที่ได้นั้นนำมาพล็อตกราฟเพื่อหาแนวโน้มแรงดันไฟฟ้าต่อระยะทางของเซนเซอร์ ในรูปที่ 4.3 แสดงวิธีการวัดแรงดันไฟฟ้าของเซนเซอร์อินฟราเรด และในตารางที่ 4.1 แสดงวิธีการวัดแรงดันไฟฟ้าจากเซนเซอร์อินฟราเรด



รูปที่ 4.2 แสดงวิธีการวัดแรงดันไฟฟ้าจากเซนเซอร์อินฟราเรด

ตารางที่ 4.1 แสดงค่าแรงดันไฟฟ้าเทียบกับระยะห่าง

ระยะห่าง (เซนติเมตร)	แรงดันไฟฟ้า (โวลต์)	ระยะห่าง (เซนติเมตร)	แรงดันไฟฟ้า (โวลต์)	ระยะห่าง (เซนติเมตร)	แรงดันไฟฟ้า (โวลต์)	ระยะห่าง (เซนติเมตร)	แรงดันไฟฟ้า (โวลต์)
20	2.399	55	1.1465	90	0.747	125	0.643
21	2.349	56	1.127	91	0.745	126	0.6325
22	2.3145	57	1.1085	92	0.743	127	0.6275
23	2.2615	58	1.0895	93	0.7285	128	0.625
24	2.1595	59	1.0715	94	0.7275	129	0.6145
25	2.1745	60	1.058	95	0.7275	130	0.605
26	2.119	61	1.047	96	0.716	131	0.596
27	2.0785	62	1.0335	97	0.7085	132	0.5855
28	2.015	63	1.014	98	0.7085	133	0.582
29	1.977	64	0.9945	99	0.708	134	0.575
30	1.927	65	0.9815	100	0.703	135	0.566
31	1.889	66	0.9745	101	0.7015	136	0.5545
32	1.847	67	0.9555	102	0.7015	137	0.5475
33	1.8115	68	0.945	103	0.7015	138	0.535
34	1.758	69	0.9365	104	0.699	139	0.528
35	1.7205	70	0.9185	105	0.686	140	0.5155
36	1.6845	71	0.915	106	0.683	141	0.5065
37	1.649	72	0.8995	107	0.682	142	0.4945
38	1.6125	73	0.8845	108	0.682	143	0.489
39	1.573	74	0.881	109	0.681	144	0.476
40	1.536	75	0.8625	110	0.681	145	0.47
41	1.4995	76	0.861	111	0.6805	146	0.457
42	1.4635	77	0.8445	112	0.6775	147	0.4525
43	1.4445	78	0.841	113	0.674	148	0.445
44	1.4075	79	0.827	114	0.664	149	0.4345
45	1.385	80	0.821	115	0.664	150	0.427
46	1.3535	81	0.818	116	0.6635		
47	1.3335	82	0.8025	117	0.6635		
48	1.2965	83	0.8025	118	0.6635		
49	1.2775	84	0.7865	119	0.6615		
50	1.2585	85	0.7835	120	0.657		
51	1.228	86	0.7835	121	0.649		
52	1.203	87	0.765	122	0.643		
53	1.1845	88	0.765	123	0.643		
54	1.1655	89	0.762	124	0.643		



รูปที่ 4.3 กราฟแสดงแนวโน้มระหว่างระยะห่างกับแรงดันไฟฟ้าของเซนเซอร์

จากรูปที่ 4.3 เส้นสีฟ้า คือ การวัดแรงดันไฟฟ้าที่สิ่งกีดขวางเป็นวัตถุสีขาว ส่วนเส้นสีแดงคือการวัดแรงดันไฟฟ้าที่สิ่งกีดขวางเป็นวัตถุสีดำ

ผลการทดลอง

จากตารางที่ 4.1 คือค่าแรงดันไฟฟ้าต่อระยะห่างของสิ่งกีดขวาง แล้วนำมาพล็อตเป็นกราฟดังรูปที่ 4.3 จะเห็นว่ากราฟที่ได้เป็นกราฟ non-linear จากการใช้วัตถุทดลองเป็นสีขาว (เส้นสีน้ำเงิน) พบว่าในช่วงระยะห่างตั้งแต่ 20 เซนติเมตร ถึง 60 เซนติเมตร กราฟจะโค้งลงอย่างต่อเนื่อง ส่วนช่วงระยะที่ 60 เซนติเมตร เป็นต้นไปกราฟจะลดลงอย่างไม่ต่อเนื่องเหมือนในช่วงแรก โดยมีค่าแรงดันไฟฟ้าซ้ำกันในบางค่าและมีความเปลี่ยนแปลงน้อยมากเมื่อเทียบกับช่วงแรก ทำให้การรับค่าของเซนเซอร์ที่ระยะห่างตั้งแต่ 60 เซนติเมตร เป็นต้นไปมีความไม่แน่นอน จึงทำให้เกิดความคลาดเคลื่อนในการแสดงผลทางหน้าจอ

ส่วนวัตถุที่ใช้ทดลองเป็นสีดำ (เส้นสีแดง) ในช่วงแรกที่ระยะห่างตั้งแต่ 20 เซนติเมตร ถึง 60 เซนติเมตร กราฟที่ได้จะคล้ายกับวัตถุทดลองสีขาว ในช่วงหลังที่ระยะห่างตั้งแต่ 60 เซนติเมตร เป็นต้นไป กราฟจะแสดงค่าแรงดันไฟฟ้าน้อยกว่าวัตถุทดลองสีขาว เนื่องจากในช่วง 60 เซนติเมตร เป็นต้นไป

เซนเซอร์จะให้ค่าแรงดันไฟฟ้าที่น้อยกว่า ดังนั้นเซนเซอร์จะสามารถตรวจจับวัตถุสีขาวได้ดีกว่าวัตถุสีดำ

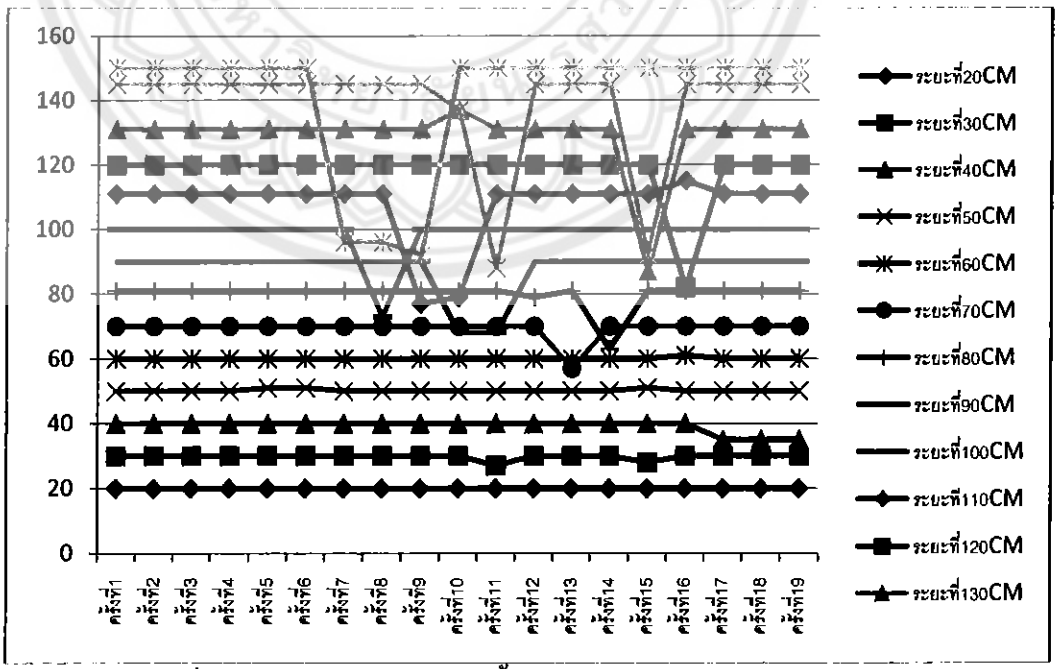
4.1.2 การทดลองวัดระยะของสิ่งกีดขวาง

ในการทดลองวัดระยะของสิ่งกีดขวางนี้ จะทดสอบโดยการวางวัตถุไว้ในระยะต่างๆ ตั้งแต่ 20 เซนติเมตร ถึง 150 เซนติเมตร ไปแนวตรง 0 องศา (เซนเซอร์อยู่กับที่ไม่กวาดไปมา) เปรียบเทียบกับระยะที่แสดงในกราฟที่แสดงผลทางหน้าจอ GUI ดังรูปที่ 4.4 และนำมาพล็อตกราฟเปรียบเทียบค่าระยะจริงกับระยะที่แสดงทางหน้าจอ ในตารางที่ 4.2

รูปที่ 4.4 แสดงการวัดระยะห่างของสิ่งกีดขวางในขณะเซนเซอร์คงที่

ตารางที่ 4.2 เปรียบเทียบระยะห่างของค่าจริงกับค่าที่แสดงทางหน้าจอ

ระยะห่างจริง	ระยะที่ได้จากกราฟ การทดลองครั้งที่																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
30	30	30	30	30	30	30	30	30	30	30	27	30	30	30	28	30	30	30	30
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	35	35	35
50	50	50	50	50	51	51	50	50	50	50	50	50	50	50	51	50	50	50	50
60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	61	60	60	60
70	70	70	70	70	70	70	70	70	70	70	70	70	57	70	70	70	70	70	70
80	81	81	81	81	81	81	81	81	81	81	81	79	81	63	81	81	81	81	81
90	90	90	90	90	90	90	90	90	90	68	68	90	90	90	90	90	90	90	90
100	100	100	100	100	100	100	100	73	100	100	100	100	100	100	100	100	100	100	100
110	111	111	111	111	111	111	111	111	77	79	111	111	111	111	111	115	111	111	111
120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	82	120	120	120
130	131	131	131	131	131	131	131	131	131	137	131	131	131	131	87	131	131	131	131
140	145	145	145	145	145	145	145	145	145	137	88	145	145	145	92	145	145	145	145
150	150	150	150	150	150	150	96	96	92	150	150	150	150	150	150	150	150	150	150



รูปที่ 4.5 กราฟแสดงการวัดผิดเพี้ยนของเซนเซอร์อินฟราเรดในบางค่า

ผลการทดลอง

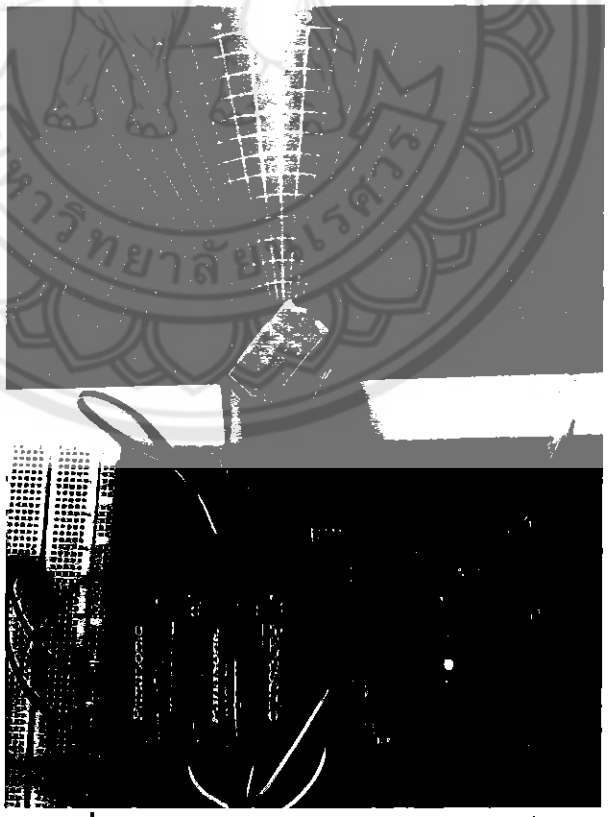
จากกราฟพบว่า ค่าที่แสดงออกทางหน้าจอมีความคลาดเคลื่อนมากในระยะตั้งแต่ 60 เซนติเมตร เป็นต้นไป เนื่องค่าจากค่าสัญญาณดิจิตอลที่รับเข้ามาจากไมโครคอนโทรลเลอร์มีความใกล้เคียงกันมากและในบางค่าซ้ำกัน ทำให้ค่าที่แสดงผลนั้นออกมาคลาดเคลื่อน

4.2 เซอร์โวมอเตอร์

ทดลองการหมุนของเซอร์โวมอเตอร์

การทำงานของสแกนเนอร์อินฟราเรดจะใช้เซอร์โวมอเตอร์ในการหมุนเป็นครึ่งวงกลม ตั้งแต่ -90 องศา ถึง +90 องศา รวมเป็น 180 องศาด้านหน้าของตัวสแกนเนอร์ โดยในการหมุนแต่ละครั้งจะกำหนดให้เซอร์โวมอเตอร์หมุนทีละ 5 องศา จนครบ แล้วกลับมาเริ่มทำงานใหม่

ในการทดสอบจะทดสอบว่าเซอร์โวมอเตอร์หมุนที 5 องศา จนครบ 180 องศาหรือไม่ โดยสร้างรูปครึ่งวงกลมที่มีเส้นช่องละ 5 องศา วางด้านล่างเซอร์โวมอเตอร์ จากนั้นสั่งให้เซอร์โวหมุน แล้วดูว่าเซอร์โซมอเตอร์หมุนตามเส้นด้านล่างหรือไม่ ดังรูปที่ทดลองที่ 4.6



รูปที่ 4.6 แสดงการทดสอบการหมุนของเซอร์โวมอเตอร์

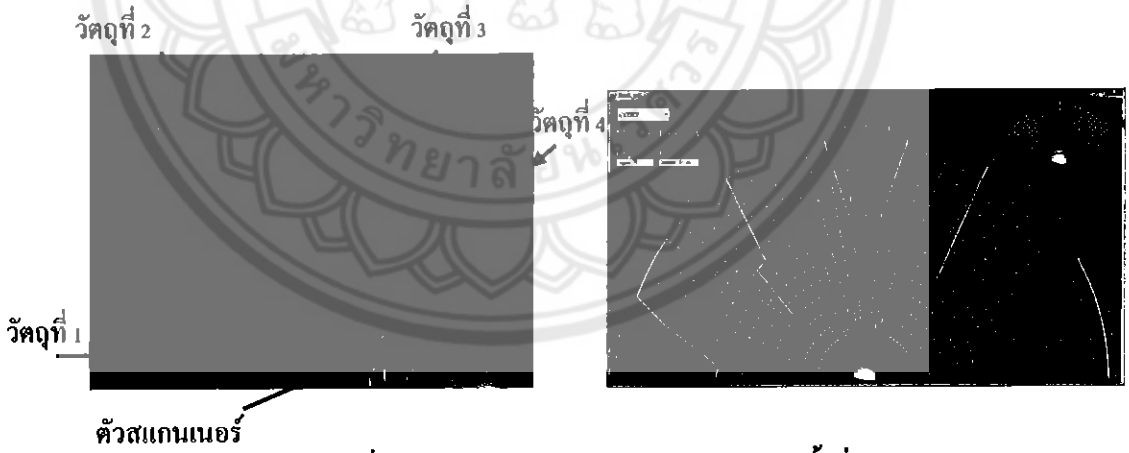
ผลการทดลอง

จากการทดลองพบว่า เซอร์โวมอเตอร์หมุนที่ละ 5 องศาได้จริง เนื่องจากในแต่ละครั้งที่เซอร์โวมอเตอร์ขยับตำแหน่งของอุปกรณ์ที่ติดอยู่กับเซอร์โวมอเตอร์ตรงกับรูปครึ่งวงกลมที่แบ่งช่องละ 5 องศาด้านล่างตรงกัน เป็นเพราะในการโปรแกรมคำสั่งเซอร์โวมอเตอร์ได้กำหนดสัญญาณพัลส์ให้ห่างกันสเต็ปละ 5 องศา

4.3 สแกนเนอร์อินฟราเรด

ในการทดสอบนี้จะเป็นการทดสอบโดยรวมของระบบที่พัฒนาขึ้น โดยการสุ่มวางวัตถุในบริเวณที่ตัวสแกนเนอร์อินฟราเรดสามารถตรวจสอบได้คือที่ระยะ 20 เซนติเมตร ถึง 150 เซนติเมตร ในทิศทาง 180 องศาด้านหน้าของตัวสแกนเนอร์ แล้วดูว่าที่หน้าจอแสดงผล GUI แสดงตำแหน่งของสิ่งกีดขวางได้ตรงตามตำแหน่งจริงหรือไม่ การทดลองสามารถตรวจสอบได้จากภาพการทดลอง ดังนี้

การทดลองครั้งที่ 1



รูปที่ 4.7 การทดลองสแกนเนอร์อินฟราเรดครั้งที่ 1

หมายเหตุ ที่กราฟแสดง 150 เซนติเมตร เท่ากับ ไม่พบสิ่งกีดขวาง

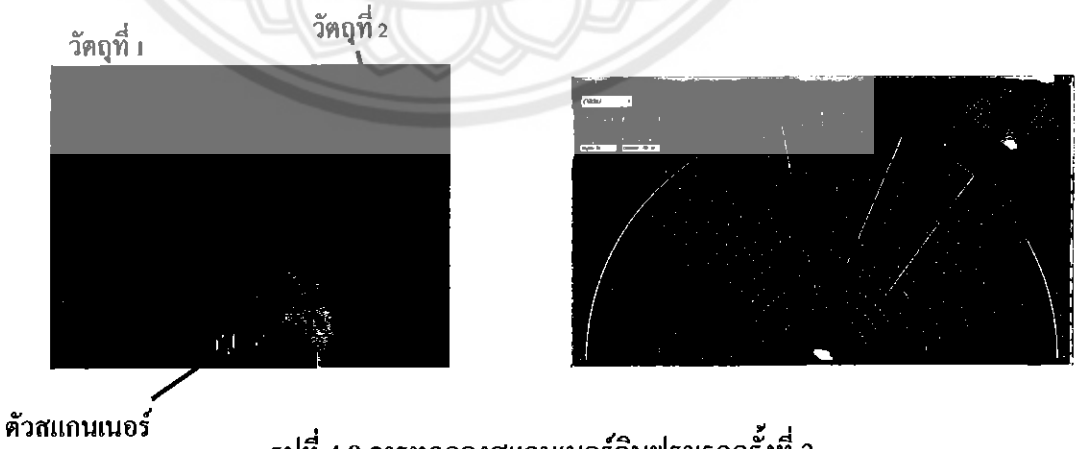
วัตถุที่	มุมที่ทำการวัด	ระยะจริงของวัตถุ	ระยะที่แสดงบนกราฟ
1	-90	90	88
2	-45	90	90
3	0	90	90
4	40	80	77

ตารางที่ 4.3 แสดงการวัดของสแกนเนอร์ที่ระยะต่างๆ ของการทดลองที่ 1

ผลการทดลองครั้งที่ 1

จากการทดลองครั้งที่ 1 พบว่าตัวสแกนเนอร์สามารถตรวจจับวัตถุได้ทั้งหมด แต่ตำแหน่งที่แสดงออกทางหน้าจอ ยังมีบางวัตถุที่ตัวสแกนนเนอร์วัดระยะผิดพลาด เช่นในวัตถุที่ 1 ในมุม -90 องศา ระยะจริงของวัตถุคือ 90 เซนติเมตร ระยะที่แสดงคือ 88 เซนติเมตร ซึ่งมีความคลาดเคลื่อน 2 เซนติเมตร เช่นเดียวกันกับในวัตถุที่ 4 ที่มีความคลาดเคลื่อน 3 เซนติเมตร ความคลาดเคลื่อนที่เกิดขึ้นอาจเกิดจากการวัดของเซนเซอร์ที่ไม่แน่นอนโดยอ้างอิงจากคุณลักษณะของเซนเซอร์ รูปที่ 2.8 หน้า 11 แต่เมื่อดูจากกราฟครึ่งวงกลมในรูปที่ 4.7 จะเห็นว่าในมุมที่ –50 องศา มีการบอกระยะที่ 40 เซนติเมตร ซึ่งไม่มีวัตถุอยู่ตำแหน่งนั้น อาจเกิดจากสิ่งแวดล้อมรอบข้างที่ทำการทดลอง การแก้ไขคือ เพิ่มการหน่วงเวลาเพื่อให้เซนเซอร์สามารถรับค่าได้นานขึ้น

การทดลองครั้งที่ 2



รูปที่ 4.8 การทดลองสแกนเนอร์อินฟราเรดครั้งที่ 2

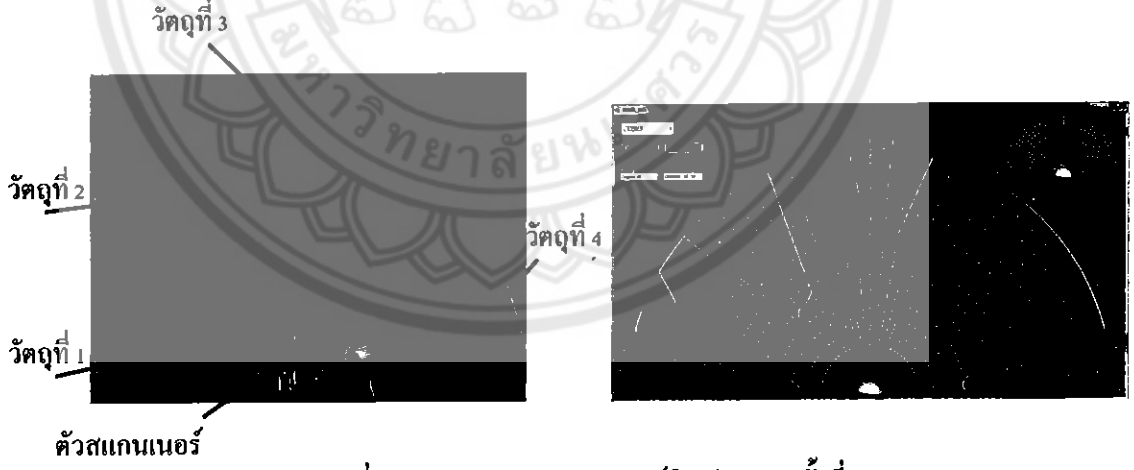
วัตถุที่	มุมที่ทำการวัด	ระยะจริงของวัตถุ	ระยะที่แสดงลนกราฟ
1	0	40	42
2	45	50	51

ตารางที่ 4.4 แสดงการวัดของสแกนเนอร์ที่ระยะต่างๆ ของการทดลองที่ 2

ผลการทดลองครั้งที่ 2

จากการทดลองครั้งที่ 2 พบว่าตัวสแกนเนอร์สามารถตรวจจับวัตถุได้ทั้งหมด แต่ตำแหน่งที่แสดงออกทางหน้าจอยังมีการวัดระยะผิดพลาด เช่นในวัตถุที่ 1 ในมุม 0 องศา ระยะจริงของวัตถุคือ 40 เซนติเมตร ระยะที่แสดงคือ 42 เซนติเมตร ซึ่งมีความคลาดเคลื่อน 2 เซนติเมตร เช่นเดียวกับในวัตถุที่ 2 ที่มีความคลาดเคลื่อน 2 เซนติเมตร ความคลาดเคลื่อนที่เกิดขึ้นอาจเกิดจากการวัดของเซนเซอร์ที่ไม่แน่นอนโดยอ้างอิงจากคุณลักษณะของเซนเซอร์ รูปที่ 2.8 หน้า 11 และอาจเกิดจากสิ่งแวดล้อมรอบข้างที่ทำการทดลอง การแก้ไขคืออาจเพิ่มการหน่วงเวลาเพื่อให้เซนเซอร์สามารถรับค่าได้นานขึ้น

การทดลองครั้งที่ 3



รูปที่ 4.9 การทดลองสแกนเนอร์อินฟราเรดครั้งที่ 3

หมายเหตุ ที่กราฟแสดง 150 เซนติเมตร เท่ากับ ไม่พบสิ่งกีดขวาง

วัตถุที่	มุมที่ทำการวัด	ระยะจริงของวัตถุ	ระยะที่แสดงบนกราฟ
1	-90	40	40
2	-45	60	61
3	0	90	88
4	90	60	54

ตารางที่ 4.9 แสดงการวัดของสแกนเนอร์ที่ระยะต่างๆ ของการทดลองที่ 3

ผลการทดลองครั้งที่ 3

จากการทดลองครั้งที่ 3 พบว่าตัวสแกนเนอร์สามารถตรวจจับวัตถุได้ทั้งหมด แต่ตำแหน่งที่แสดงออกทางหน้าจอยังมีการวัดระยะผิดพลาด เช่นในวัตถุที่ 2 ใน-45 องศา ระยะจริงที่วัตถุอยู่คือ 60 เซนติเมตร ระยะที่แสดงคือ 61 เซนติเมตร ซึ่งมีความคลาดเคลื่อน 1 เซนติเมตร เช่นเดียวกันกับในวัตถุที่ 3 และ 4 ที่มีความคลาดเคลื่อนที่ 2 และ 6 เซนติเมตร ความคลาดเคลื่อนที่เกิดขึ้นอาจเกิดจากการวัดของเซนเซอร์ที่ไม่แน่นอนโดยอ้างอิงจากคุณลักษณะของเซนเซอร์ รูปที่ 2.8 หน้า 11 แต่เมื่อดูจากกราฟครึ่งวงกลมในรูปที่ 4.7 จะเห็นว่าในมุมที่ -65 องศา มีการบอกระยะที่ 130 เซนติเมตร ซึ่งไม่มีวัตถุอยู่ตำแหน่งนั้น อาจเกิดจากสิ่งแวดล้อมรอบข้างที่ทำการทดลอง การแก้ไขคืออาจเพิ่มการหน่วงเวลาเพื่อให้เซนเซอร์สามารถรับค่าได้นานขึ้น

4.4 บทสรุป

ในบทนี้จะกล่าวถึงวิธีการทดลองและผลการทดลองของเซนเซอร์อินฟราเรด เซอร์โวมอเตอร์และระบบสแกนเนอร์อินฟราเรด ในส่วนเซนเซอร์อินฟราเรดในการวัดค่าแรงดันไฟฟ้าทำให้เห็นว่ากราฟที่ได้เป็นกราฟแบบ non-linear ค่าที่ได้จากเซนเซอร์จะไม่แน่นอน ทำให้ค่าที่วัดได้อาจมีความคลาดเคลื่อนได้โดยเฉพาะระยะห่างตั้งแต่ 60 เซนติเมตร เป็นต้นไป ส่วนต่อมาคือเซอร์โวมอเตอร์เป็นการวัดความเที่ยงตรงในการหมุนในแต่ละองศา ผลที่ได้คือเซอร์โวมอเตอร์นั้นสามารถหมุนได้ 180 องศาตามที่ต้องการ ส่วนสุดท้ายที่ทดลองนั่นคือตัวสแกนเนอร์อินฟราเรดที่พัฒนาขึ้น ทดสอบความแม่นยำในการตรวจจับสิ่งกีดขวางของระบบ แล้วแสดงขึ้นสู่หน้าจอได้อย่างถูกต้อง จากการทดสอบตัวสแกนเนอร์พบว่าสามารถจับวัตถุกีดขวางได้ แต่ในการบอกตำแหน่งยังมีข้อผิดพลาดอยู่บ้างบางตำแหน่ง สาเหตุอาจเกิดจากประสิทธิภาพการทำงานของเซนเซอร์ ที่ยังไม่แม่นยำพอ

บทที่ 5
สรุปผลการทดลอง

โครงการออกแบบและสร้างสแกนเนอร์อินฟราเรดสำหรับตรวจจับสิ่งกีดขวางนี้ มีวัตถุประสงค์เพื่อพัฒนาต้นแบบสำหรับแสดงผลทางหน้าจอในรูปแบบของข้อมูลทิศทางและระยะห่างของสิ่งกีดขวางในขณะที่ตัวสแกนเนอร์ทำงานอยู่ โดยข้อมูลที่ตรวจจับได้จะถูกส่งไปยังเครื่องคอมพิวเตอร์ส่วนบุคคล เพื่อแสดงผลออกทางหน้าจอ Graphic User Interface (GUI) ที่พัฒนาขึ้น

5.1 สรุปผลการทดลอง

จากผลการทดลองในบทที่ 4 สามารถสรุปได้เป็น 2 ส่วนดังนี้

1. เซนเซอร์อินฟราเรด จากที่ทำการทดลองวัดค่า ค่าที่ให้มีความแม่นยำในระดับหนึ่ง กล่าวคือในช่วงแรกมีความผิดพลาดน้อย แต่เมื่อระยะไกลออกไปจะเริ่มให้ค่าที่ไม่แน่นอนทำให้ผลที่ได้คลาดเคลื่อน (เป็นไปตามกราฟคุณลักษณะของเซนเซอร์ ในรูปที่ 2.8 ซึ่งแสดงความสัมพันธ์ระหว่างระยะทางกับ Output Voltage ของ Sharp 2Y0A02 Infrared Sensor ในหน้าที่ 11)

2. ระบบสแกนเนอร์อินฟราเรด ในระบบนี้มีความแม่นยำในการวัดระดับไม่สูงมาก กล่าวคือ มีความผิดพลาดในการวัดตำแหน่งของวัตถุ โดยค่าความคลาดเคลื่อนมีผลมาจาก การใช้เซนเซอร์อินฟราเรดในการตรวจจับเนื่องจากเซนเซอร์ชนิดนี้ใช้แสงอินฟราเรดในการวัดซึ่งแสงอินฟราเรดสามารถถูกรบกวนได้ง่าย ประกอบกับต้องติดตั้งบนเซอร์โวมอเตอร์ที่ต้องหมุนตลอดเวลาทำให้ค่าที่ได้จากตัวสแกนเนอร์แกว่งขึ้นลงอย่างมากในการวัด นอกจากระบบสแกนเนอร์แล้วยังมีส่วนแสดงผลทางหน้าจอ GUI ที่แสดงภาพตามที่รับค่ามาจากตัวสแกนเนอร์ ที่สามารถแสดงผลตามค่าที่รับเข้ามาได้

5.2 วิจารณ์ผลการทดลอง

ความถูกต้องที่ได้จากการวัดของเซนเซอร์อินฟราเรด ในการตรวจจับระยะห่างของสิ่งกีดขวางนั้น จะมีความถูกต้องเมื่อมีระยะห่างไม่เกิน 20 เซนติเมตร ถึง 60 เซนติเมตร เมื่อระยะห่างมีค่าเกินกว่านี้จะทำให้ค่าของการวัดระยะห่างนั้นมีความถูกต้องลดลง ซึ่งตรงกับคู่มือของเซนเซอร์อินฟราเรดชนิดนี้

ในส่วนของตัวสแกนเนอร์อินฟราเรดที่มีความคลาดเคลื่อนในการวัดระยะห่างนั้น จะมีค่าความคลาดเคลื่อนสูงเนื่องจากเซนเซอร์ที่ใช้ตรวจจับระยะห่างนี้เป็นเซนเซอร์อินฟราเรดที่ถูกรบกวนได้ง่าย จึงต้องทดลองภายในบริเวณที่ถูกควบคุม นอกจากนั้นการที่เซอร์โวมอเตอร์หมุนไปด้วยก็มีส่วนทำ

ให้เกินความคลาดเคลื่อนในการวัด โดยเมื่อเปรียบเทียบค่าของการวัดระหว่างเซนเซอร์ที่ไม่ขยับกับเซนเซอร์ที่ขยับ พบว่าค่าที่เซนเซอร์ไม่ขยับให้ค่าที่แน่นอนกว่า

5.3 ปัญหาที่พบ

ปัญหาที่พบจากการทำงานคือ เซนเซอร์อินฟราเรดที่ใช้ ผลการวัดค่าแรงดันไฟฟ้าที่ได้จะเป็นกราฟแบบ Non-linear ซึ่งในการใช้งานจริงผลที่ได้ ช่วงแรกที่ระยะ 20 เซนติเมตร ถึง 60 เซนติเมตรจะมีความแน่นอนในการวัด ส่วนในช่วงหลังที่ระยะ 60 เซนติเมตร เป็นต้นไป จะมีความคลาดเคลื่อนในการวัด โดยสามารตรวจสอบค่าแรงดันไฟฟ้าของเซนเซอร์อินฟราเรดได้จากตารางที่ 4.1 หน้าที่ 39

5.4 การพัฒนาโครงงานต่อไปในอนาคต

1. ตัวสแกนเนอร์ สำหรับตรวจจับระยะห่างของสิ่งกีดขวาง จากเซนเซอร์อินฟราเรดที่ใช้งานอยู่นี้มีประสิทธิภาพในการวัดค่อนข้างน้อย สามารถแก้ไขโดยการเปลี่ยนเซนเซอร์อินฟราเรดที่มีคุณภาพสูงกว่านี้ จึงจะทำให้ค่าที่ได้นั้น มีความแม่นยำมากขึ้น สามารถนำไปประยุกต์ใช้กับอุปกรณ์อื่นๆได้

2. การเพิ่มเซนเซอร์อินฟราเรดเป็น 2 ตัว เพื่อเพิ่มความเร็วและความแม่นยำในการทำงาน โดยติดตั้งเซนเซอร์อินฟราเรดตัวที่ 1 ไว้ที่มุม -90 องศา และตัวที่ 2 ไว้ที่มุม 0 องศา แล้วสั่งให้เซอร์โวมอเตอร์หมุนเพียง 90 องศา ซึ่งเซนเซอร์ตัวที่ 1 จะหมุนไปที่ 0 องศา ส่วนเซนเซอร์ตัวที่ 2 จะหมุนไปที่มุม +90 องศา จะทำให้ระบบสามารถทำงานได้เร็วยิ่งขึ้น

เอกสารอ้างอิง

[1]: http://th.wikipedia.org/wiki/การวางแผนการเคลื่อนที่

[2]: http://mechatronics.ptwit.ac.th/seksan/menett25/DRC/DRC03.pdf

[3]: http://rambutan.net63.net/docs/robotics%202008.doc

[4]: http://www.pctelecom.co.th/know.php?id_s=3

[5]: http://www.acroname.com/robotics/info/articles/sharp/sharp.html

[6]: http://www.phidgets.com/products.php?product_id=3522_0

[7]: http://hackaday.com/2009/02/24/parts-analog-distance-sensors-sharp-gp2d122y0a02/

[8]: http://www.phidgets.com/documentation/Phidgets/3522_0_Datasheet.pdf

[9]: http://arduinomega.blogspot.com/2011/05/infrared-long-range-sensor-gift-of.html

[10]: http://www.circuitstoday.com/pic-tutorial-16f877

[11]: http://ww1.microchip.com/downloads/en/devicedoc/30292c.pdf

[12]: http://www.etteam.com/product/pic/man-Pic-v3.0&v4.0-ICD2.pdf

[13]: http://www.etteam.com/article/pic/pic009.html

[14]: http://www.etteam.com/product/1602.html

[15]: http://www.servocity.com/html/how_do_servos_work_.html

[16]:http://ampcircuitschematic.blogspot.com/2011/04/pwm-controller-with-555-timer-chip.html#.UCf7z50aPx0

[17]: หนังสือเรียนรู้และประยุกต์ใช้งานpic microcontroller สมบูรณ์เนียมกล่ำหน้า 218

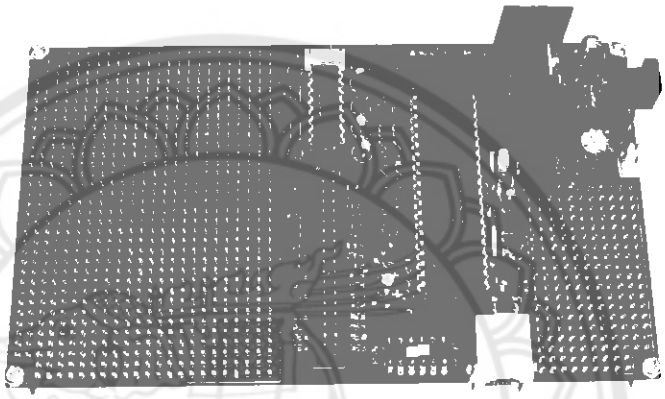
[18]: http://th.wikipedia.org/wiki/ไมโครคอนโทรลเลอร์

[19]: http://stang.sc.mahidol.ac.th/ait/training/PDF/AI.pdf

ภาคผนวก ก
อุปกรณ์ที่ใช้ในการทำโครงงาน

1. ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ที่ใช้คือ PIC 16F877



รูปที่ 1 แสดงบอร์ด CP-PIC V3.0 (ICD2)

ขาสัญญาณต่างๆ

RA0-RA3 และ RA5 ขาสัญญาณเหล่านี้นอกจากจะใช้งานเป็น I/O ปกติได้แล้วยังทำหน้าที่เป็นขาอินพุต ของสัญญาณอนาล็อก (AN0-AN4) อีกด้วย ดังนั้นจึงต่อสายสัญญาณเหล่านี้เข้ากับขั้วต่อ ADC/IO (CPU) เพื่อให้สะดวกต่อการนำไปใช้งาน

RA4 จะใช้งานในส่วนของ LCD ซึ่งจะต่อเข้าที่ขา 6 ของคอนเนคเตอร์ CLCD โดยทำหน้าที่เป็นขา Enable ให้กับ LCD

RA6/OSC2/CLKO เป็นขาสัญญาณที่ทำหน้าที่ในหลายส่วน คือ เป็นขา OSC2 และ CLKO จะนำมาใช้เป็นขาสัญญาณ I/O ได้ก็ต่อเมื่อใช้คริสตอลออสซิลเลเตอร์แบบที่เป็นโมดูลสำเร็จสามารถต่อเข้ากับขา OSC1/CLKIN ได้เลยโดยที่ไม่ต้องต่อกับขา RA6/OSC2 ทำให้ ขา RA6 ว่างและนำไปใช้เป็น I/O ได้ แต่ในบอร์ดจะใช้งานขา RA6/OSC2 ร่วมกับ OSC1 ในการรับสัญญาณนาฬิกาจากภายนอก ดังนั้น ขา RA6 นี้จึงไม่สามารถต่อออกไปใช้งานได้

RB0-RB7 สำหรับขาสัญญาณนี้จะสามารถใช้งานเป็น I/O ได้ปกติ แต่จะมีคุณสมบัติพิเศษคือจะมีวงจร Pull-Up ภายในและยังเป็นแหล่งกำเนิดสัญญาณอินเตอรัพท์ต่างๆ ดังนี้

- RB0/INT0 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 0

- RB1/INT1 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 1

- RB2/INT2 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 2

- RB3/INT3 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 3 (เฉพาะเบอร์ 18F442)

- RB4-RB7 เป็นขาที่สามารถกำเนิดสัญญาณอินเตอร์รัพท์ได้หากมีการเปลี่ยนแปลงในขาสัญญาณดังกล่าวและมีการ Enable อินเตอร์รัพท์ประเภทนี้ไว้ จึงเหมาะกับการนำไปใช้งานในส่วนของ สวิตช์คีย์บอร์ด เนื่องจากมีทั้ง อินเตอร์รัพท์และวงจร Pull-Up ในตัว

RC0 ขาสัญญาณนี้จะต่อเข้ากับขั้วต่อ แอลซีดี (CLCD) โดยจะต่อเข้าที่ขา 4 ของคอนเนคเตอร์ เพื่อทำหน้าที่เป็นขาสัญญาณ RS เพื่อควบคุมการทำงานของ LCD

RC1 เป็นขาสัญญาณที่ต่อเข้ากับขั้วต่อ OC1B เพื่อใช้งานในส่วนของ ขาสัญญาณอินพุตของ Timer 1 หรือใช้เป็นขาสัญญาณในส่วนของ Capture2 input /Compare2 Output/PWM2

RC2 เป็นขาสัญญาณที่ต่อเข้ากับขั้วต่อ OC1A เพื่อใช้เป็นขาสัญญาณในส่วนของ Capture1 input/Compare1 Output/PWM1

RC3 สำหรับขาสัญญาณ RC3 จะใช้ทำหน้าที่เป็นขาสัญญาณ SCL ในการติดต่อกับอุปกรณ์ I^2C Bus และจะต่อเข้ากับ ขั้วต่อ I^2C EXPAND เพื่อขยายพอร์ต I^2C BUS

RC4 สำหรับขาสัญญาณ RC4 จะใช้ทำหน้าที่เป็นขาสัญญาณ SDA ในการติดต่อกับอุปกรณ์ I^2C Bus และจะต่อเข้ากับ ขั้วต่อ I^2C EXPAND เพื่อขยายพอร์ต I^2C BUS

RC5 จะใช้เป็นสัญญาณควบคุมการรับส่งข้อมูลในการใช้งาน RS485 โดยจะควบคุมการทำงานของอุปกรณ์ที่เป็น Line Driver ก็คือ IC 75176

RC6 เป็นขาสัญญาณที่ทำหน้าที่ในการส่งข้อมูล (Tx) ในโหมดการสื่อสารอนุกรม RS232

RC7 เป็นขาสัญญาณที่ทำหน้าที่ในการรับข้อมูล (Rx) ในโหมดการสื่อสารอนุกรม RS232

RD0-RD3 สำหรับขาสัญญาณเหล่านี้จะต่อเข้ากับขั้วต่อ KBI/IO เพื่อใช้งานสำหรับการต่อ คีย์สวิตช์ 4x4 หรือ 4x3 ซึ่งเมื่อใช้งานเป็นคีย์บอร์ดดังกล่าวจะทำงานร่วมกับ พอร์ต RB4-RB7 หรือจะใช้งานเป็น I/O ก็ได้

RD4-RD7 ขาสัญญาณเหล่านี้จะทำหน้าที่เป็นขาสัญญาณ Data ที่ใช้ติดต่อกับ LCD โดยจะถูกต่อไปที่คอนเน็คเตอร์ CLCD ซึ่งขั้วต่อ LCD ที่ได้ออกแบบนี้จะเป็นแบบ 4 Bit Data ฉนั้นในการรับส่งข้อมูลจะทำผ่านสายสัญญาณทั้ง 4 เส้น คือ RD4-RD7

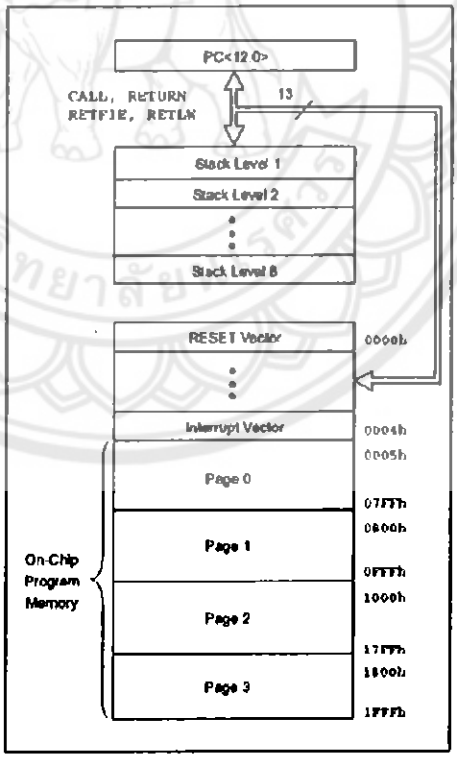
RE0-RE2 ขาสัญญาณเหล่านี้สามารถใช้งานเป็น I/O ได้ตามปกติ แต่จะมีคุณสมบัติพิเศษคือ ขาสัญญาณดังกล่าวจะทำหน้าที่เป็นขาอินพุตอนาลอก (AN5-AN7) เมื่ออยู่ในโหมดของ Analog to Digital โดยเราจะนำไปต่อกับขั้วต่อ ADC/IO (CPU) ทำให้สามารถต่อออกไปใช้งานได้สะดวก

การจัดสรรพื้นที่หน่วยความจำ และ รีจิสเตอร์ต่างๆ

การจัดหน่วยความจำของ PIC 16F877 นี้จะแบ่งออกเป็น 3 ส่วนคือ หน่วยความจำโปรแกรม, หน่วยความจำข้อมูล (RAM), และหน่วยความจำข้อมูลที่เป็น EEPROM

- หน่วยความจำโปรแกรม

PIC 16F877 นี้มี Program Counter ขนาด 13 บิต ซึ่งสามารถอ้างถึงตำแหน่งข้อมูลได้ถึง 8 กิโลเวิร์ด โดยจะมีตำแหน่ง Reset Vector ที่ 0000h และ Interrupt Vector ที่ 0004h ดังนั้นในการเขียนโปรแกรมจึงควรสงวนพื้นที่ส่วนนี้ไว้สำหรับการใช้งานอินเตอร์รัพท์ มีพื้นที่ของ Stack 8 ระดับ และหน่วยความจำโปรแกรมแบ่งออกเป็น 4 Page (8 kwords) ซึ่งพื้นที่ในส่วนนี้มีไว้สำหรับเก็บข้อมูลคำสั่งทั้งหมดโดยโครงสร้างจะเป็นแบบแฟลช (flash memory) ทำให้ลบและเขียนใหม่ได้หลายครั้ง



รูปที่ 2 แสดงการจัดสรรพื้นที่หน่วยความจำของ PIC 16F877

- **หน่วยความจำข้อมูล**

ใน PIC 16F877 นี้หน่วยความจำข้อมูลจะแบ่งออกเป็นพื้นที่ของ RAM หน่วยความจำใช้งานทั่วไป (General Purpose Register) ขนาด 368 Bytes และ พื้นที่ของ รีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Registers) ในการจัดวางพื้นที่จะแบ่งออกเป็น 4 แบงค์ ตั้งแต่แอดเดรส 00h ถึง 1FFh ในการเข้าถึงข้อมูลในแต่ละส่วน หรือ แต่ละแบงก์ สามารถทำได้โดยการกำหนดค่าในรีจิสเตอร์ STATUS ในบิตที่ 5 และ 6 (RP0, RP1) ซึ่งมีความหมายดังนี้

RP1	RP0	Bank Select
0	0	Bank0 : 00h – 7Fh
0	1	Bank1 : 80h – FFh
1	0	Bank2 : 100h – 17Fh
1	1	Bank3 : 180h – 1FFh

- **หน่วยความจำข้อมูล EEPROM**

PIC 16F877 มีหน่วยความจำแบบ EEPROM จำนวน 256 ไบต์ โดยสามารถอ่านและเขียนในขณะทำงานปกติได้แต่ต้องไม่มีการ Enable Code protect bit โดยการเข้าถึงนั้นจะต้องทำผ่านรีจิสเตอร์พิเศษ (Special Function Register) ซึ่งต้องใช้ถึง 4 ตัวดังนี้

EECON1: ควบคุมการเข้าถึงหน่วยความจำ

EECON2: จัดลำดับการเขียนข้อมูล

EEDATA: เป็นบัฟเฟอร์ใช้เก็บข้อมูล 8 บิต สำหรับการอ่านและเขียน

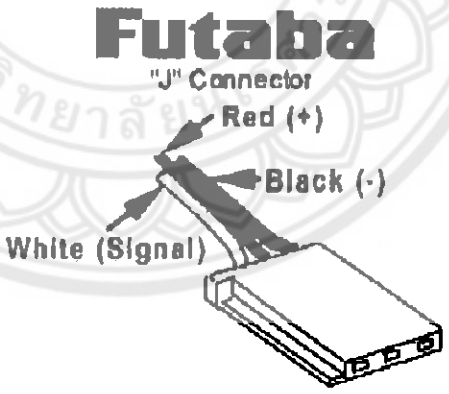
EEADR: รีจิสเตอร์ที่เก็บแอดเดรส 00h – FFh (256 ไบต์)

สาเหตุที่เลือกใช้ ชุดบอร์ดไมโครคอนโทรเลอร์ CP-PIC V3.0 (ICD3) และไมโครคอนโทรเลอร์ PIC 16F877 เพราะว่า มีฟังก์ชันที่จำเป็นต่อการใช้งาน เช่น ฟังก์ชั่น Analog to Digital Converter มีพื้นที่ต่อ วงจร I/O เพิ่มเติม เป็นต้น นอกจากจะมีฟังก์ชั่นที่เพียงพอต่อการใช้งานแล้ว ราคาก็ไม่สูงจนเกินไป

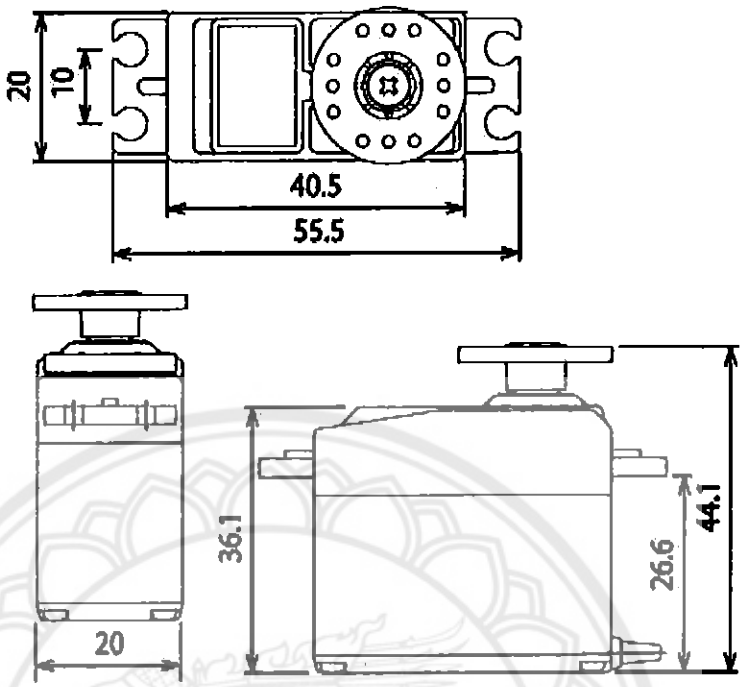
2. เซอร์โวมอเตอร์

เซอร์โวมอเตอร์ที่เลือกใช้คือ FUTABA 3003

รูปที่ 2 แสดงรูปเซอร์โวมอเตอร์



รูปที่ 4 สายควบคุมการทำงาน



รูปที่ 5โครงสร้างการทำงานของเซอร์โวมอเตอร์

คุณสมบัติของเซอร์โวมอเตอร์

Control System:	+Pulse Width Control 1520usec Neutral
Required Pulse:	3-5 Volt Peak to Peak Square Wave
Operating Voltage:	4.8-6.0 Volts
Operating Temperature Range:	-20 to +60 Degree C
Operating Speed (4.8V):	0.23sec/60 degrees at no load
Operating Speed (6.0V):	0.19sec/60 degrees at no load
Stall Torque (4.8V):	44 oz/in. (3.2kg.cm)
Stall Torque (6.0V):	56.8 oz/in. (4.1kg.cm)
Operating Angle:	45 Deg. one side pulse traveling 400usec
360 Modifiable:	Yes
Direction:	Counter Clockwise/Pulse Traveling 1520-1900usec
Current Drain (4.8V):	7.2mA/idle
Current Drain (6.0V):	8mA/idle

Motor Type:	3 Pole Ferrite
Potentiometer Drive:	Indirect Drive
Bearing Type:	Plastic Bearing
Gear Type:	All Nylon Gears
Connector Wire Length:	12"
Dimensions:	1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
Weight:	1.3oz. (37.2g)

ภาคผนวก ข

โปรแกรมการทำงาน

โปรแกรมแสดงผลทางหน้า ใช้โปรแกรม Microsoft Visual Studio C# ในการโปรแกรมคำสั่ง

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
////including
using System.IO.Ports;
using System.IO;
using System.Drawing.Drawing2D;
using System.Collections;
using System.Threading;

namespace IRScaner_Project_V2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            CheckForIllegalCrossThreadCalls = false;
            this.SetStyle(ControlStyles.OptimizedDoubleBuffer, true); //use Doublebuffer
            this.SetStyle(ControlStyles.AllPaintingInWmPaint, true);
            this.SetStyle(ControlStyles.UserPaint, true);
            this.SetStyle(ControlStyles.SupportsTransparentBackColor, true);

        }
        //Definition members
        internal static List<point_Positon> list_Point = new List<point_Positon>();
        static List<int> list_average = new List<int>();
        static GraphicsPath mainPath = new GraphicsPath();
        static int ADC2Distance;
        static int index = -1;
        static int index2 = 0;
        public int angle = 0; //for Draw degreeline
        public int degree = -5;
```

```
static int average;
int angleShow = -90;
string receiveA2D;
int splited_value = 0;




private void Form1_Load(object sender, EventArgs e)
{
    //get list_Port by ComBoBox
    String[] list_Port = SerialPort.GetPortNames();
    foreach (string N in list_Port)
    {
        select_PortCmB.Items.Add(N);
    }
    //comboBox1.SelectedIndex = 1;
}

private void startBtn_Click(object sender, EventArgs e)
{


     if(!serialPort1.IsOpen)
    {
        if (string.IsNullOrEmpty(select_PortCmB.Text))
        {
            MessageBox.Show("Please Select Port", "WARNING", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        }
        else
        {
            serialPort1.PortName = select_PortCmB.SelectedItem + "";
            serialPort1.Open();
            serialPort1.DataReceived += serialPort1_DataReceived;
            serialPort1.Write("s\n");
        }
    }
}

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    //receiveData from seraialport (string)
    receiveA2D = serialPort1.ReadExisting();

    //check length
    if (receiveA2D.Length == 7)
    {
        this.Invoke(new EventHandler(Splitaverage_fromA2D));
    }

}

private void Splitaverage_fromA2D(object sender, EventArgs e)
{
    int value = 0;
    char[] split = { ' ' };
    string[] word = receiveA2D.Split(split, StringSplitOptions.RemoveEmptyEntries);
    foreach (string ConvertToint in word)
    {
        value = Int32.Parse(ConvertToint);
    }


    splited_value = value;//////////
    }
```

```
private void Average(int splited_value)

{
    if (index2 % 1 == 0)
    {
        //add splited_value to list
        list_average.Add(splited_value);
        //avrage list_average
        average = Convert.ToInt32(Math.Round(list_average.Average()));
       // Console.WriteLine(average);
    }
    list_average.Clear();
    index2 = 0;
    Console.WriteLine(average);
}

private void lookUptable(int average)
{
    if (average >= 484 && average <= 516) { ADC2Distance = 20; }
    else if (average >= 477 && average <= 483) { ADC2Distance = 21; }
    else if (average >= 467 && average <= 476) { ADC2Distance = 22; }
    else if (average >= 456 && average <= 466) { ADC2Distance = 23; }
    else if (average >= 445 && average <= 455) { ADC2Distance = 24; }
    else if (average >= 437 && average <= 444) { ADC2Distance = 25; }
    else if (average >= 428 && average <= 436) { ADC2Distance = 26; }
    else if (average >= 420 && average <= 427) { ADC2Distance = 27; }
    else if (average >= 409 && average <= 419) { ADC2Distance = 28; }
    else if (average >= 397 && average <= 408) { ADC2Distance = 29; }
    else if (average >= 388 && average <= 396) { ADC2Distance = 30; }
    else if (average >= 380 && average <= 387) { ADC2Distance = 31; }
    else if (average >= 371 && average <= 379) { ADC2Distance = 32; }
    else if (average >= 363 && average <= 370) { ADC2Distance = 33; }
    else if (average >= 355 && average <= 362) { ADC2Distance = 34; }
    else if (average >= 347 && average <= 354) { ADC2Distance = 35; }
    else if (average >= 337 && average <= 346) { ADC2Distance = 36; }
    else if (average >= 330 && average <= 336) { ADC2Distance = 37; }
    else if (average >= 322 && average <= 329) { ADC2Distance = 38; }
    else if (average >= 315 && average <= 321) { ADC2Distance = 39; }
    else if (average >= 308 && average <= 313) { ADC2Distance = 40; }
    else if (average >= 300 && average <= 307) { ADC2Distance = 41; }
    else if (average >= 294 && average <= 299) { ADC2Distance = 42; }
    else if (average >= 288 && average <= 293) { ADC2Distance = 43; }
    else if (average >= 281 && average <= 287) { ADC2Distance = 44; }
    else if (average >= 275 && average <= 280) { ADC2Distance = 45; }
    else if (average >= 272 && average <= 276) { ADC2Distance = 46; }
    else if (average >= 266 && average <= 270) { ADC2Distance = 47; }
    else if (average >= 261 && average <= 265) { ADC2Distance = 48; }
    else if (average >= 256 && average <= 260) { ADC2Distance = 49; }
    else if (average >= 249 && average <= 254) { ADC2Distance = 50; }
    else if (average >= 246 && average <= 248) { ADC2Distance = 51; }
    else if (average >= 242 && average <= 245) { ADC2Distance = 52; }
    else if (average >= 238 && average <= 241) { ADC2Distance = 53; }
    else if (average >= 234 && average <= 237) { ADC2Distance = 54; }
    else if (average >= 231 && average <= 233) { ADC2Distance = 55; }
    else if (average >= 227 && average <= 230) { ADC2Distance = 56; }
    else if (average >= 222 && average <= 226) { ADC2Distance = 57; }
    else if (average >= 218 && average <= 221) { ADC2Distance = 58; }
    else if (average >= 215 && average <= 217) { ADC2Distance = 59; }
    else if (average >= 211 && average <= 214) { ADC2Distance = 60; }
    else if (average >= 206 && average <= 210) { ADC2Distance = 61; }
    else if (average >= 203 && average <= 203) { ADC2Distance = 62; }
    else if (average >= 199 && average <= 202) { ADC2Distance = 63; }
    else if (average >= 199 && average <= 202) { ADC2Distance = 64; }
    else if (average >= 195 && average <= 198) { ADC2Distance = 65; }
    else if (average >= 191 && average <= 194) { ADC2Distance = 66; }
    else if (average >= 191 && average <= 194) { ADC2Distance = 67; }
    else if (average >= 187 && average <= 190) { ADC2Distance = 68; }
    else if (average >= 187 && average <= 190) { ADC2Distance = 69; }
```

```
else if (average >= 184 && average <= 186) { ADC2Distance = 70; }
else if (average >= 181 && average <= 183) { ADC2Distance = 71; }
else if (average >= 181 && average <= 183) { ADC2Distance = 72; }
else if (average >= 176 && average <= 179) { ADC2Distance = 73; }
else if (average >= 176 && average <= 178) { ADC2Distance = 74; }
else if (average >= 172 && average <= 175) { ADC2Distance = 75; }
else if (average >= 172 && average <= 175) { ADC2Distance = 76; }
else if (average >= 168 && average <= 171) { ADC2Distance = 77; }
else if (average >= 168 && average <= 171) { ADC2Distance = 78; }
else if (average >= 164 && average <= 167) { ADC2Distance = 79; }
else if (average >= 164 && average <= 167) { ADC2Distance = 80; }
else if (average >= 159 && average <= 163) { ADC2Distance = 81; }
else if (average >= 159 && average <= 163) { ADC2Distance = 82; }
else if (average >= 156 && average <= 158) { ADC2Distance = 83; }
else if (average >= 156 && average <= 158) { ADC2Distance = 84; }
else if (average >= 156 && average <= 158) { ADC2Distance = 85; }
else if (average >= 152 && average <= 154) { ADC2Distance = 86; }
else if (average >= 152 && average <= 154) { ADC2Distance = 87; }
else if (average >= 148 && average <= 151) { ADC2Distance = 88; }
else if (average >= 148 && average <= 151) { ADC2Distance = 89; }
else if (average >= 145 && average <= 147) { ADC2Distance = 90; }
else if (average >= 145 && average <= 147) { ADC2Distance = 91; }
else if (average >= 140 && average <= 144) { ADC2Distance = 92; }
else if (average >= 140 && average <= 144) { ADC2Distance = 93; }
else if (average >= 140 && average <= 144) { ADC2Distance = 94; }
else if (average >= 134 && average <= 139) { ADC2Distance = 95; }
else if (average >= 134 && average <= 139) { ADC2Distance = 96; }
else if (average >= 134 && average <= 139) { ADC2Distance = 97; }
else if (average >= 134 && average <= 139) { ADC2Distance = 98; }
else if (average >= 134 && average <= 139) { ADC2Distance = 99; }
else if (average >= 128 && average <= 133) { ADC2Distance = 100; }
else if (average >= 128 && average <= 133) { ADC2Distance = 101; }
else if (average >= 128 && average <= 133) { ADC2Distance = 102; }
else if (average >= 128 && average <= 133) { ADC2Distance = 103; }
else if (average >= 128 && average <= 133) { ADC2Distance = 104; }
else if (average >= 120 && average <= 124) { ADC2Distance = 105; }
else if (average >= 120 && average <= 124) { ADC2Distance = 106; }
else if (average >= 120 && average <= 124) { ADC2Distance = 107; }
else if (average >= 120 && average <= 124) { ADC2Distance = 108; }
else if (average >= 120 && average <= 130) { ADC2Distance = 109; }
else if (average >= 117 && average <= 120) { ADC2Distance = 110; }
else if (average >= 117 && average <= 120) { ADC2Distance = 111; }
else if (average >= 117 && average <= 120) { ADC2Distance = 112; }
else if (average >= 117 && average <= 120) { ADC2Distance = 113; }
else if (average >= 117 && average <= 120) { ADC2Distance = 114; }
else if (average >= 113 && average <= 116) { ADC2Distance = 115; }
else if (average >= 113 && average <= 116) { ADC2Distance = 116; }
else if (average >= 113 && average <= 116) { ADC2Distance = 117; }
else if (average >= 113 && average <= 116) { ADC2Distance = 118; }
else if (average >= 113 && average <= 116) { ADC2Distance = 119; }
else if (average >= 108 && average <= 112) { ADC2Distance = 120; }
else if (average >= 108 && average <= 112) { ADC2Distance = 121; }
else if (average >= 108 && average <= 112) { ADC2Distance = 122; }
else if (average >= 108 && average <= 112) { ADC2Distance = 123; }
else if (average >= 108 && average <= 112) { ADC2Distance = 124; }
else if (average >= 104 && average <= 107) { ADC2Distance = 125; }
else if (average >= 104 && average <= 107) { ADC2Distance = 126; }
else if (average >= 104 && average <= 107) { ADC2Distance = 127; }
else if (average >= 104 && average <= 107) { ADC2Distance = 128; }
else if (average >= 104 && average <= 107) { ADC2Distance = 129; }
else if (average >= 99 && average <= 103) { ADC2Distance = 130; }
else if (average >= 99 && average <= 103) { ADC2Distance = 131; }
else if (average >= 99 && average <= 103) { ADC2Distance = 132; }
else if (average >= 99 && average <= 103) { ADC2Distance = 133; }
else if (average >= 99 && average <= 103) { ADC2Distance = 134; }
else if (average >= 95 && average <= 98) { ADC2Distance = 135; }
else if (average >= 95 && average <= 98) { ADC2Distance = 136; }
else if (average >= 95 && average <= 98) { ADC2Distance = 137; }
else if (average >= 95 && average <= 98) { ADC2Distance = 138; }
```

```
            else if (average >= 95 && average <= 98) { ADC2Distance = 139; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 140; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 141; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 142; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 143; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 144; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 145; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 146; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 147; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 148; }
            else if (average >= 91 && average <= 94) { ADC2Distance = 149; }
            else if (average >= 85 && average <= 90) { ADC2Distance = 150; }
            else { ADC2Distance = 150; }

            showDistance.Text = "Distance   " + ADC2Distance.ToString() + "  cm";
            send2CalculatePoint(ADC2Distance);

        }

        private void send2CalculatePoint(int Distance)
        {
            switch (degree)
            {
                case 0:
                    Point_Calculate(Distance,259, 358,16, 359);// distance, firstX , firstY, endX,
endY
                    break;
                case 5:
                    Point_Calculate(Distance,259, 353,17, 332);// distance, firstX , firstY, endX,
endY
                    break;
                case 10:
                    Point_Calculate(Distance,260, 348,20, 306);// distance, firstX , firstY, endX,
endY
                    break;
                case 15:
                    Point_Calculate(Distance,261, 343,26, 280);// distance, firstX , firstY, endX,
endY
                    break;
                case 20:
                    Point_Calculate(Distance,263, 338,34, 254);// distance, firstX , firstY, endX,
endY
                    break;
                case 25:
                    Point_Calculate(Distance,265, 333,44, 230);// distance, firstX , firstY, endX,
endY
                    break;
                case 30:
                    Point_Calculate(Distance,267, 328,56, 207);// distance, firstX , firstY, endX,
endY
                    break;
                case 35:
                    Point_Calculate(Distance,270, 324,71, 184);// distance, firstX , firstY, endX,
endY
                    break;
                case 40:
                    Point_Calculate(Distance,273, 319,87, 163);// distance, firstX , firstY, endX,
endY
                    break;
                case 45:
                    Point_Calculate(Distance,277, 316,105, 143);// distance, firstX , firstY,
endX, endY
                    break;
                case 50:
                    Point_Calculate(Distance,281, 312,124, 125);// distance, firstX , firstY,
endX, endY
                    break;
                case 55:
```

```
                    Point_Calculate(Distance,285, 309,146, 110);// distance, firstX , firstY,
endX, endY
                    break;
                case 60:
                    Point_Calculate(Distance,290, 306,168, 95);// distance, firstX , firstY, endX,
endY
                    break;
                case 65:
                    Point_Calculate(Distance,294, 303,191, 82);// distance, firstX , firstY, endX,
endY
                    break;
                case 70:
                    Point_Calculate(Distance,299, 301,216, 73);// distance, firstX , firstY, endX,
endY
                    break;
                case 75:
                    Point_Calculate(Distance,304, 300,241, 65);// distance, firstX , firstY, endX,
endY
                    break;
                case 80:
                    Point_Calculate(Distance,309, 299,267, 59);// distance, firstX , firstY, endX,
endY
                    break;
                case 85:
                    Point_Calculate(Distance,315, 298,293, 56);// distance, firstX , firstY, endX,
endY
                    break;
                case 90:
                    Point_Calculate(Distance,320, 298,320, 54);// distance, firstX , firstY, endX,
endY
                    break;
                case 95:
                    Point_Calculate(Distance,325, 298,346, 56);// distance, firstX , firstY, endX,
endY
                    break;
                case 100:
                    Point_Calculate(Distance,331, 299,373, 59);// distance, firstX , firstY, endX,
endY
                    break;
                case 105:
                    Point_Calculate(Distance,336, 300,399, 65);// distance, firstX , firstY, endX,
endY
                    break;
                case 110:
                    Point_Calculate(Distance,341, 301,424, 73);// distance, firstX , firstY, endX,
endY
                    break;
                case 115:
                    Point_Calculate(Distance,346, 303,449, 82);// distance, firstX , firstY, endX,
endY
                    break;
                case 120:
                    Point_Calculate(Distance,350, 306,472, 95);// distance, firstX , firstY, endX,
endY
                    break;
                case 125:
                    Point_Calculate(Distance,355, 309,495, 109);// distance, firstX , firstY,
endX, endY
                    break;
                case 130:
                    Point_Calculate(Distance,359, 312,515, 125);// distance, firstX , firstY,
endX, endY
                    break;
                case 135:
                    Point_Calculate(Distance,363, 315,535, 143);// distance, firstX , firstY,
endX, endY
                    break;
                case 140:
```

```
                    Point_Calculate(Distance,367, 319,553, 163);// distance, firstX , firstY,
endX, endY
                    break;
                case 145:
                    Point_Calculate(Distance,370, 324,569, 184);// distance, firstX , firstY,
endX, endY
                    break;
                case 150:
                    Point_Calculate(Distance,373, 328,583, 207);// distance, firstX , firstY,
endX, endY
                    break;
                case 155:
                    Point_Calculate(Distance,375, 333,596, 230);// distance, firstX , firstY,
endX, endY
                    break;
                case 160:
                    Point_Calculate(Distance,377, 338,606, 254);// distance, firstX , firstY,
endX, endY
                    break;
                case 165:
                    Point_Calculate(Distance,379, 343,614, 280);// distance, firstX , firstY,
endX, endY
                    break;
                case 170:
                    Point_Calculate(Distance,380, 348,620, 306);// distance, firstX , firstY,
endX, endY
                    break;
                case 175:
                    Point_Calculate(Distance,381, 353,623, 332);// distance, firstX , firstY,
endX, endY
                    break;
                case 180:
                    Point_Calculate(Distance,381, 358,624, 358);// distance, firstX , firstY,
endX, endY
                    break;
            }

        }
        //calculate point from postion x y
        private void Point_Calculate(int Distance, double firstX, double firstY, double endX,
double endY)
        {
            int finalX, finalY = 0;
            double CalX, CalY;
            double x, y;
            x = (firstX - endX) / 130.0;
            y = (firstY - endY) / 130.0;

            if (Distance == 20)
            {
                finalX = Convert.ToInt32(Math.Round(firstX));
                finalY = Convert.ToInt32(Math.Round(firstY));

            }
            {
                CalX = firstX - x * (Distance - 20);
                CalY = firstY - y * (Distance - 20);
            }
            finalX = Convert.ToInt32(Math.Round(CalX));
            finalY = Convert.ToInt32(Math.Round(CalY));
            Addlist(finalX, finalY);
        }


        //addlist
       private void Addlist(int pointX, int pointY)

       {
           switch (degree)
```

```
{
    case 0:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 5:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 10:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 15:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 20:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 25:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 30:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 35:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 40:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 45:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 50:
        index++;
        list_Point.Add(new point_Positon(pointX, pointY));
        Drawgraph();
        Console.Beep();
        break;
    case 55:
        index++;
```

```
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 60:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 65:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 70:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 75:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 80:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 85:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 90:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 95:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 100:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 105:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 110:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
```

```
            break;
        case 115:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 120:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 125:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 130:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 135:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 140:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 145:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 150:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 155:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 160:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 165:
            index++;
            list_Point.Add(new point_Positon(pointX, pointY));
            Drawgraph();
            Console.Beep();
            break;
        case 170:
            index++;
```

```
                    list_Point.Add(new point_Positon(pointX, pointY));
                    Drawgraph();
                    Console.Beep();
                    break;
                case 175:
                    index++;
                    list_Point.Add(new point_Positon(pointX, pointY));
                    Drawgraph();
                    Console.Beep();
                    break;
                case 180:
                    index++;
                    list_Point.Add(new point_Positon(pointX, pointY));
                    Drawgraph();
                    Console.Beep();
                    break;
            }

        }

        private void Drawgraph()
        {
            Graphics gr = panel_Top.CreateGraphics();
            Pen CurentPen = new Pen(Color.Red, 2);
            gr.SmoothingMode = SmoothingMode.AntiAlias;
            if (index == 0)
            {
                show_Degree.Text = degree.ToString() + " " + list_Point[0].X + ", " +
list_Point[0].Y;
            }
            else if (degree % 5 == 0 && degree > 0 && index <= 37)
            {
                mainPath.AddLine(list_Point[index - 1].X, list_Point[index - 1].Y,
list_Point[index].X, list_Point[index].Y);


                gr.DrawPath(CurentPen, mainPath);
                show_Degree.Text = degree.ToString() + " " + list_Point[index].X + ", " +
list_Point[index].Y;
            }
        }

        private void Lotated()
        {
            Graphics gr_Top = panel_Top.CreateGraphics();
            Graphics gr_Down = panel_Down.CreateGraphics();
            GraphicsPath path = new GraphicsPath();

            Point startPointA = new Point(320, 359);
            Point startPointC = new Point(16, 359);
            Color tran = Color.FromArgb(120, Color.White);
            Pen penD = new Pen(tran,2);
            Pen penD2 = new Pen(Color.Red,2);

            gr_Top.SmoothingMode = SmoothingMode.AntiAlias;
            gr_Down.SmoothingMode = SmoothingMode.AntiAlias;
            path.AddLine(startPointA, startPointC);
            Matrix mx = new Matrix();
            mx.RotateAt(angle, startPointA);
            if (serialPort1.IsOpen)
            {
                {
                    if (angle < 185)
                    {
                        path.Transform(mx);
                        gr_Down.DrawPath(penD, path);
                        gr_Top.DrawPath(penD, path);
                        angle += 5;
                        degree += 5;
```

```
                show_Degree.Text = "Degree " + angleShow.ToString();
            }
            else if (angle == 185)
            {


                serialPort1.Write("s\n");
                list_Point.Clear();
                list_average.Clear();
                panel_Down.Refresh();
                gr_Down.DrawPath(penD2, mainPath);
                mainPath.Reset();
                gr_Top.Dispose();
                gr_Down.Dispose();
                degree = -5;
                angle = 0;
                index = -1;
                index2 = 0;
                angleShow = -90;
                panel_Top.Invalidate();

            }
        }

    }
}
private void closePort()
{

    if (serialPort1 != null && serialPort1.IsOpen)
    {
        serialPort1.DataReceived -= serialPort1_DataReceived;
             serialPort1.DiscardInBuffer();
            serialPort1.DiscardOutBuffer();
             serialPort1.Close();
    }

}
private void timer1_Tick(object sender, EventArgs e)
{
    if (serialPort1.IsOpen) {
        Invalidate();
        lookUptable(average);
        Lotated();
        serialPort1.Write("s\n");
        angleShow = angleShow + 5;


    }
}
private void timer2_Tick(object sender, EventArgs e)
{
    if (degree >= 0)
    {
        Invalidate();
        index2++;
        Average(splited_value);
    }
}
private void reset_Btn_Click(object sender, EventArgs e)
{
    if (serialPort1.IsOpen)
    {
        closePort();
        show_Degree.Clear();
        showDistance.Clear();
        panel_Down.Invalidate();
        panel_Top.Invalidate();
        mainPath.Reset();
```

```
                index = -1;
                index2 = 0;
                angle = 0;
                degree = -5;
                list_Point.Clear();
                list_average.Clear();

            }
        }

    }
}
```