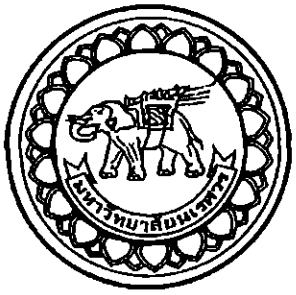


การออกแบบระบบแสดงสถานะการบินสำหรับอากาศยานปีกหมุน

FLIGHT SIMULATION DESIGN FOR A ROTARY-WING FLYING ROBOT

ธนพงษ์ อุดมรัตนศิริชัย รหัส 50610353

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2555

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน การออกแบบระบบแสดงสถานะการบินสำหรับอากาศยานปีกหมุน

ผู้ดำเนินโครงงาน นายธนพงษ์ อุดมรัตนศิริชัย รหัส 50610353

ที่ปรึกษาโครงงาน ดร.พนัส นัถฤทธิ์

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2555

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

..... ที่ปรึกษาโครงงาน
(ดร.พนัส นัถฤทธิ์)

..... กรรมการ
(ดร.สุรเดช จิตประไพกุลศาล)

..... กรรมการ
(อาจารย์รัฐภูมิ วรานุสาสน์)

..... กรรมการ
(อาจารย์เศรษฐา ตั้งค้าวานิช)

ชื่อหัวข้อโครงงาน การออกแบบระบบแสดงสถานะการบินสำหรับอากาศยานปีกหมุน
ผู้ดำเนินโครงงาน นายธนพงษ์ อุดมรัตนศิริชัย รหัส 50610353
ที่ปรึกษาโครงงาน ดร.พนัส นัถฤทธิ์
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา 2555

.....

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาโปรแกรมต้นแบบสำหรับจำลองสถานะการบินในรูปแบบของข้อมูลทิศทางและระดับความเอียงของเฮลิคอปเตอร์ในขณะที่ปฏิบัติงานอยู่ ระบบนี้สามารถแสดงมุมเอียงได้ทั้ง 3 แนวแกน คือ แกน X (Roll) แกน Y (Pitch) แกน Z (Yaw) โดยที่ในแกน X และ Y นั้นจะสามารถแสดงสถานะมุมเอียงได้ตั้งแต่ - 90 องศา จนถึง +90 องศา ในขณะที่ แกน Z สามารถแสดงสถานะมุมเอียงได้ที่ตั้งแต่มุม 0 องศา ถึง 360 องศา โดยที่มุม 0 องศานั้นจะอ้างอิงกับทิศเหนือและมีการหมุนแบบตามเข็มนาฬิกา โดยที่ข้อมูลที่ได้จากระบบนี้ใช้เป็นข้อมูลพื้นฐานสำหรับการสร้างระบบควบคุมการบินอัตโนมัติ คือ เมื่อมุมเอียงของระบบเอียงไปทางใดระบบต้องมีการควบคุมมุมเอียงเพื่อชดเชยมุมเอียงดังกล่าวเพื่อให้ระบบยังจะสามารถทรงตัวและบินอยู่ได้ด้วยความสมดุล

ผลที่ได้จากโครงงานนี้คือระบบที่สามารถจำลองสถานะการบินในรูปแบบของข้อมูลทิศทางและสถานะมุมเอียงโดยที่จะแสดงเป็นภาพจำลอง 3 มิติ บนหน้าจอแสดงผลของคอมพิวเตอร์ส่วนบุคคล

Project Title	Flight Simulation Design for a Rotary-Wing Flying Robot	
Name	Mr. Tanapong Udomrattanasirichai	ID. 50610353
Project advisor	Dr. Panus Nattharith	
Major	Computer Engineering	
Department	Electrical and Computer Engineering	
Academic year	2012	

.....

Abstract

The aim of the project is to develop the flight simulation design. The system can be performed in three directional axis consisting of the x-axis (Rolling), y-axis (Pitching) and z-axis (Yawing). The angle of x-axis and y-axis can be altered ranges between -90 to 90 degrees. The angle ranges of z-axis are between 0 to 360 degrees with clockwise and the zero degree is indicated to the north. The results of the simulation test are used as the basic data to create the control system of flying robot, which means that it will help to maintain the flight stability of the flying robot. For example, the flying robot has a tilt angle of rolling to the left then it will offset itself to the right to get back to the original motion or act stability.

The advantage outcome of the project is that the application can simulate the flight status in terms of the angle and directional motions of the aircraft flight, which will be demonstrated in 3D on the monitor screen of Personal Computer (PC).

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้เป็นเรื่องเกี่ยวกับการออกแบบระบบจำลองการแสดงสถานะการบิน ซึ่งจะสำเร็จไปไม่ได้เลยถ้าไม่ได้รับความช่วยเหลือจากบุคคลต่อไปนี้

ขอขอบพระคุณ ดร. พนัส นัถฤทธิ์อย่างสูงที่กรุณาเป็นอาจารย์ที่ปรึกษาให้แก่โครงงานนี้ อาจารย์ได้เสียสละเวลาในการให้คำปรึกษาพร้อมทั้งให้แนวคิดในการทำงาน รวมทั้งแนะนำและจัดหางบประมาณเพื่อจัดซื้ออุปกรณ์ทางฮาร์ดแวร์ต่างๆที่จำเป็นสำหรับการสร้างระบบขึ้นมา

ขอขอบคุณมหาวิทยาลัยนเรศวรที่ให้งบประมาณและสถานที่ในการดำเนินโครงงานในครั้งนี้

ขอขอบคุณบิดา มารดา ผู้ที่เลี้ยงดู อบรมบ่มนิสัยของผู้ดำเนินโครงงานมาเป็นอย่างดี ถ้าไม่มี 2 ท่านนี้แล้ว ทางผู้ดำเนินโครงงานคงไม่มีวันนี้ ซึ่งเป็นพระคุณที่หาใดเปรียบไม่ได้

ขอขอบคุณคณาจารย์ทุกท่าน ที่ได้มอบความรู้ต่างๆให้แก่ผู้ดำเนินโครงงาน ซึ่งเป็นส่วนช่วยผู้ดำเนินโครงงานอย่างมากในการทำโครงงานนี้

ขอขอบคุณทุกๆท่านที่ไม่ได้เอ่ยนามมา ณ ที่นี้ ที่มีส่วนช่วยเหลือในการดำเนินโครงงานนี้ให้สำเร็จลุล่วงไปด้วยดี

ผู้ดำเนินโครงงาน
ธนพงษ์ อุดมรัตนศิริชัย

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....ก
บทคัดย่อภาษาอังกฤษ.....ข
กิตติกรรมประกาศ.....ค
สารบัญ.....ง
สารบัญตาราง.....ฉ
สารบัญรูป.....ช

บทที่ 1 บทนำ..... 1
1.1 ที่มาและความสำคัญของโครงงาน..... 1
1.2 วัตถุประสงค์..... 1
1.3 ขอบข่ายของโครงงาน..... 2
1.4 ขั้นตอนการดำเนินงาน..... 2
1.5 ตารางแสดงกิจกรรมการดำเนินงาน..... 2
1.6 ผลที่คาดว่าจะได้รับ..... 3
1.7 งบประมาณ..... 4

บทที่ 2 หลักการและทฤษฎี..... 5
2.1 กลศาสตร์ของอากาศยานปีกหมุน (เฮลิคอปเตอร์)..... 5
2.2 แกนอ้างอิงสำหรับวัดมุมเอียง..... 7
2.3 ความหมายและความสำคัญของ GUI..... 8
2.4 บอร์ดไมโครคอนโทรเลอร์ CP-PIC V3.0 (ICD2)..... 9
2.5 อุปกรณ์สำหรับตรวจจับมุมเอียงในแนวแกน X และ Y..... 12
2.6 อุปกรณ์สำหรับตรวจจับมุมเอียงในแนวแกน Z..... 16
2.7 ไลบรารีสำหรับสร้างภาพ 3 มิติในการแสดงผล..... 18
2.8 บทสรุป..... 25

บทที่ 3 วิธีการดำเนินการ..... 26
3.1 ภาพรวมของระบบ (System Overview)..... 26

สารบัญ (ต่อ)

หน้า

3.2 การเขียนโปรแกรมบนบอร์ดไมโครโปรเซสเซอร์เพื่อใช้ในการอ่านค่าจากเซนเซอร์ที่ใช้สำหรับวัดมุมเอียงทั้ง 2 ตัว 27
3.3 การเขียนโปรแกรมสร้างหน้าจอ GUI สำหรับแสดงผล 33
3.4 บทสรุป 41

บทที่ 4 การทดลองและผลการทดลอง 43
4.1 ต้นแบบของระบบ 43
4.2 ผลการทดลองการวัดมุมเอียงของเซนเซอร์ MEMSIC2125 44
4.3 ผลการทดลองการวัดมุมเอียงของเซนเซอร์ CMPS03 54
4.4 บทสรุป 59

บทที่ 5 สรุปผลการทดลอง 60
5.1 สรุปผลการทดลอง 60
5.2 วิจารณ์ผลการทดลอง 61
5.3 การพัฒนาโครงงานต่อไปในอนาคต 61
5.4 ข้อเสนอแนะ 62

เอกสารอ้างอิง 63

ภาคผนวก 66
ก. การติดตั้งโปรแกรม Microsoft Visual Studio 2010 Express 66
ข. การติดตั้ง Library TaoOpenGL 68
ค. คู่มือการใช้งานโปรแกรม Flight Simulator 71

ประวัติผู้ดำเนินโครงงาน 75

สารบัญตาราง

ตารางที่ หน้า

1. 1 แผนการดำเนินงาน 2

2. 1 ตารางแสดงค่าความเร่งที่ขา X_{out} หรือ Y_{out} ที่มุมเอียงค่าต่างๆ[12] 16

3. 1 รายละเอียดของรีจิสเตอร์ T1CON[11] 27

3. 2 รายละเอียดอัตราส่วนในการหารความถี่ของ Prescaler[11] 28

4. 1 แสดงผลการทดลอง ค่าเฉลี่ย และค่าความคลาดเคลื่อนของการทดลอง ของการวัดมุมเอียงในแนวแกน X 46

4. 2 แสดงผลการทดลอง ค่าเฉลี่ย และค่าความคลาดเคลื่อนของการทดลอง ของการวัดมุมเอียงในแนวแกน Y 50

4. 3 แสดงผลการทดลอง ค่าเฉลี่ย และค่าความคลาดเคลื่อนของการทดลอง ของการวัดมุมเอียงในแนวแกน Z 55

สารบัญรูป

รูปที่ หน้า

2. 1 เฮลิคอปเตอร์ต้องมีโรเตอร์ท้ายเพื่อทำหน้าที่ต้านแรงต้านที่เกิดขึ้น[2] 6
2. 2 เฮลิคอปเตอร์รุ่น KA 27 Helix [3] 6
2. 3 เฮลิคอปเตอร์รุ่น CH47 [4] 7
2. 4 แกนสำหรับวัดมุมเอียง[5] 8
2. 5 บอร์ดไมโครคอนโทลเลอร์ CP-PIC V3.0 (ICD2)[11] 10
2. 6 สถาปัตยกรรมภายในของ PIC 16F877[11] 11
2. 7 ตำแหน่งจาสัญญาณต่างๆของ PIC 16F877 [11] 12
2. 8 รูปร่างและการจัดขาของ MEMSIC2125[12] 12
2. 9 บล็อกไดอะแกรมสำหรับ MEMSIC 2125[12] 13
2. 10 แสดงหลักการทำงานของเซนเซอร์ [12] 14
2. 11 แสดงเอาต์พุทดิวตี้ไซเคิล 15
2. 12 การประยุกต์ใช้งานโมดูล MEMSIC2125 กับความลาดเอียง[12] 15
2. 13 แสดงรูปร่างและขาการต่อใช้งาน[14] 17
2. 14 แสดงวงจรบอร์ด ADX-CMPS03 และเชื่อมต่อกับ CMPS03[14] 18
2. 15 ภาพและตำแหน่งของปริมาตรการมองของ Projection เชิงตั้งฉากที่เป็นค่า Default[22] 23
2. 16 หลักการ Perspective เชิงสมมาตร [23] 24
3. 1 ภาพรวมของระบบ (System Overview) 26
3. 2 รายละเอียดของรีจิสเตอร์ T1CON[11] 27
3. 3 เปรียบเทียบการตั้งค่า y = 0 (ภาพซ้ายมือ) และ y = 1 (ภาพขวามือ) 34
3. 4 รูปลูกบาศก์ที่แต่ละด้านนั้นมีสีที่แตกต่างกัน 36
3. 5 โปรโตคอลสำหรับส่งข้อมูลจากบอร์ดไมโครคอนโทลเลอร์ไปยัง PC 37
4. 1 แสดงต้นแบบของระบบ 43
4. 2 หน้าจอ GUI ของระบบ 44
4. 3 วิธีการทดลองวัดมุมเอียงในแนวแกน X และ Y 45
4. 4 กราฟแสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน X (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 0 ถึง 80 องศา 47
4. 5 กราฟแสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน X (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง -5 ถึง - 80 องศา 48

สารบัญรูป (ต่อ)

รูปที่ หน้า

4. 6 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน (แกน Y) กับค่ามุมเอียงในแนวแกน X (แกน X).....49

4. 7 กราฟแสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Y (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 0 องศา ถึง 80 องศา..... 51

4. 8 แสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Y (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง -5 องศา ถึง -80 องศา 52

4. 9 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน (แกน Y) กับค่ามุมเอียงในแนวแกน Y (แกน X)..... 53

4. 10 วิธีการทดลองวัดมุมเอียงในแนวแกน Z..... 54

4. 11 แสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Z (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 0 - 170 องศา 56

4. 12 แสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Z (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 180 - 350 องศา 57

4. 13 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน (แกน Y) กับค่ามุมเอียงในแนวแกน Z (แกน X) 58

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

การเดินทางโดยอากาศยานเป็นสิ่งที่มีสะดวกสบายและมีความรวดเร็วมากกว่าการเดินทางในรูปแบบอื่นๆ โดยที่ในปัจจุบันได้มีการออกแบบและพัฒนาอากาศยานแบบไร้นักบินขึ้น เพื่อใช้ในงานด้านต่างๆ ซึ่งระบบนี้มีความจำเป็นที่จะต้องรู้สถานะต่างๆของเครื่องบิน เช่น เครื่องบินหันไปในทิศทางใด เอียงไปทางซ้ายหรือขวา ก้มหน้าลงหรือเงยหน้าขึ้นมากน้อยเพียงใด เพื่อที่จะทำให้ไม่เกิดการหลงทิศทางและป้องกันการตกของเครื่องบิน

โครงงานการออกแบบระบบแสดงสถานการณ์บินสำหรับอากาศยานปีกหมุน (Flight Simulation Design for a Rotary-Wing Flying Robot) นี้เป็นโครงงานที่มีวัตถุประสงค์เพื่อพัฒนาต้นแบบสำหรับระบบแสดงสถานะการบินของเฮลิคอปเตอร์ในรูปแบบของข้อมูลทิศทางและระดับความเอียงของเฮลิคอปเตอร์ในขณะปฏิบัติงานอยู่ โดยข้อมูลที่วัดได้จะถูกส่งไปที่เครื่องคอมพิวเตอร์ส่วนบุคคล(PC) เพื่อแสดงผลออกทางหน้าจอ GUI (Graphic User Interface) ที่พัฒนาขึ้น

โดยปกติมุมที่ใช้อ้างอิงว่าเฮลิคอปเตอร์อยู่ในสถานะใดจะประกอบด้วย 3 มุม ได้แก่มุมตามแกน X (Roll) มุมตามแกน Y (Pitch) และมุมตามแกน Z (Yaw) อย่างไรก็ตามโครงงานนี้เป็นเพียงโครงงานต้นแบบจึงทำการพัฒนาในขั้นต้นเพียง 2 แกนก่อนคือแกน X (Roll) และแกน Z (Yaw)

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาเซ็นเซอร์ที่ใช้ในการวัดมุมในแนวแกนทั้ง 3 แกน โดยที่การวัดมุมเอียงในแนวแกน X และ Y จะใช้โมดูล Memsic 2125 ส่วนในแนวแกน Z จะใช้โมดูล CMPS 03

1.2.2 เพื่อศึกษาการเขียนโปรแกรมไมโครคอนโทลเลอร์สำหรับการประมวลผลค่าที่ได้รับมาจากเซ็นเซอร์โดยโปรแกรมจะทำการแปลงสัญญาณที่ได้รับจากเซนเซอร์ให้เป็นค่ามุมเอียงในแนวแกนต่างๆและส่งผลที่ได้ไปยังเครื่องคอมพิวเตอร์ส่วนบุคคล (PC)

1.2.3 เพื่อศึกษาการเขียนโปรแกรมการแสดงผลบนหน้าจอคอมพิวเตอร์

1.3 ขอบข่ายของโครงงาน

1.3.1 สร้างระบบแสดงสถานะการบินบนหน้าจอคอมพิวเตอร์ ประกอบด้วยการเคลื่อนที่รอบแกน 3 แกน คือ แกน X (Roll) , แกน Y (Pitch) และ แกน Z (Yaw) ที่ใช้บนระบบปฏิบัติการ window

1.3.2 ทดสอบระบบบนอุปกรณ์จำลองที่สามารถปรับทิศทางและมุมเอียงได้อย่างอิสระ

1.3.3 ความคลาดเคลื่อนเฉลี่ยของระบบที่พัฒนาอยู่ในระดับไม่เกิน 5% ของค่าที่ถูกต้อง

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาการเขียนโปรแกรมควบคุมไมโครคอนโทลเลอร์

1.4.2 ศึกษาการทำงานของเซ็นเซอร์ทั้ง 2 ชนิด

1.4.3 เขียนโปรแกรมควบคุมการทำงานของเซนเซอร์พร้อมทำการทดสอบการทำงาน

1.4.4 ศึกษาการเขียนหน้าต่างแสดงผลบนคอมพิวเตอร์

1.4.5 เขียนโปรแกรมหน้าต่างแสดงผลพร้อมทดสอบการทำงาน

1.4.6 ทดสอบการทำงานร่วมกันของไมโครคอนโทลเลอร์และหน้าต่างสำหรับแสดงผล

1.4.7 สรุปผลการทดลองและจัดทำรูปเล่มรายงาน

1.5 ตารางแสดงกิจกรรมการดำเนินงาน

ตารางที่ 1. 1 แผนการดำเนินงาน

กำหนดการดำเนินงาน	2555							2556	
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
1.ศึกษาการเขียนโปรแกรมควบคุมไมโครคอนโทล-เลอร์									
2.ศึกษาการทำงานของเซ็นเซอร์ทั้ง 2 ชนิด									

กำหนดการดำเนินงาน	2555								2556	
	มิ.ย.		ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
3.เขียนโปรแกรมควบคุมการทำงานของเซนเซอร์พร้อมทำการทดสอบ										
4.ศึกษาการเขียนหน้าต่างแสดงผลบนคอมพิวเตอร์										
5.เขียนโปรแกรมหน้าต่างแสดงผลพร้อมทดสอบการทำงาน										
6.ทดสอบการทำงานร่วมกันของไมโครคอนโทลเลอร์และหน้าต่างสำหรับแสดงผล										
7.สรุปผลการทดลองและจัดทำรูปเล่มรายงาน										

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 ได้รับระบบต้นแบบสำหรับบอกสถานะการบินคือมุมเอียงและทิศทางการบินของเฮลิคอปเตอร์โดยแสดงผลบนหน้าจอคอมพิวเตอร์

1.7 งบประมาณ

1.7.1 ค่าเอกสารและจัดทำรูปเล่ม	700 บาท
1.7.2 ค่าหนังสือสำหรับค้นคว้าข้อมูล	300 บาท
รวมทั้งสิ้น	1000 บาท

บทที่ 2
หลักการและทฤษฎี

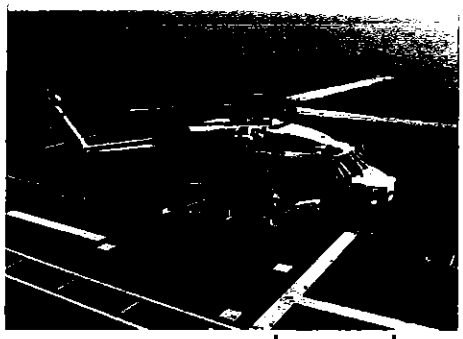
ในบทนี้จะกล่าวถึงกลศาสตร์ของอากาศยานปีกหมุนเบื้องต้น แกนอ้างอิงสำหรับวัดมุมเอียง ความสำคัญของหน้าจอแสดงผล (GUI) และอุปกรณ์ทาง HARDWARE ที่นำมาใช้ในการสร้างระบบ และสุดท้ายจะกล่าวถึง ไลบรารีที่นำมาสร้างรูป 3 มิติในการแสดงผลบนหน้าจอคอมพิวเตอร์ส่วนบุคคล (PC)

2.1 กลศาสตร์ของอากาศยานปีกหมุน (เฮลิคอปเตอร์)

เฮลิคอปเตอร์มีส่วนประกอบที่สำคัญคือ ใบพัด(โรเตอร์) , เครื่องยนต์และกลไกในการส่งผ่านกำลังไปสู่ใบพัด , ใบพัดส่วนหาง ,โครงสร้างของลำตัวและระบบอื่นๆที่ใช้ประกอบกัน ในปัจจุบันนั้นมีความก้าวหน้าทางเทคโนโลยีเป็นอย่างมากทำให้เฮลิคอปเตอร์รุ่นใหม่นั้นสามารถบินได้ในทุกสภาพอากาศ

ใบพัดโรเตอร์หลักเป็นส่วนสำคัญที่สัดในการสร้างแรงยกให้กับใบพัด ซึ่งจะทำหน้าที่เช่นเดียวกันกับปีกของเครื่องบิน โดยที่ปีกของเครื่องบินจะมีกลไกลคือจะต้องเคลื่อนที่ไปข้างหน้าเพื่อให้มีกระแสไหลผ่านปีกของเครื่องบินทำให้เกิดแรงยกขึ้น แต่ใบพัดหลักของเฮลิคอปเตอร์เมื่อเกิดการหมุนรอบแกนโรเตอร์จะทำให้เกิดกระแสไหลผ่านและเกิดแรงยกขึ้น โดยที่ใบพัดหลักของโรเตอร์นั้นจะมีตั้งแต่ 2 ใบขึ้นไป ในปัจจุบันเฮลิคอปเตอร์มีเครื่องยนต์มากถึง 3 เครื่องเช่น EH101 Cormorant, EH 101 Heliliner และ CH 53E Super Stallion ทั้ง 3 แบบเป็นอากาศยานปีกหมุนขนาดใหญ่ที่ใช้ในการบรรทุกสัมภาระจำนวนมาก

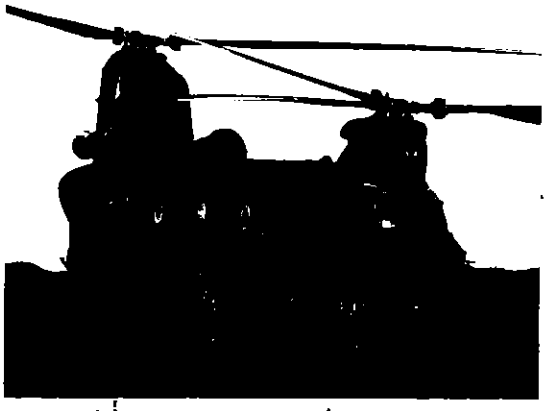
ด้วยเหตุที่ใบพัดหลักหมุนรอบแกนหนึ่งๆนั้น ความเร็วที่ได้จะแตกต่างกันตามระยะห่างจากแกน จุดที่อยู่ใกล้แกนหมุนมากจะใช้ระยะทางน้อยกว่าจุดที่อยู่ปลายสุดของแกนใบพัดในการเคลื่อนที่ไปในรอบเดียวกัน เนื่องจากความเร็วที่ปลายจะมากกว่าความเร็วที่โคนใบพัดส่งผลให้เกิดความแตกต่างในการสร้างแรงยกจากโคนถึงปลายใบพัดตามไปด้วย หากต้องการให้เฮลิคอปเตอร์มีแรงยกมากจะต้องออกแบบให้ใบพัดหลักมีพื้นที่มากเพื่อเพิ่มแรงยกสามารถทำได้โดยการเพิ่มทั้งขนาดและความยาวของใบพัดหลัก หรือเพิ่มจำนวนใบพัดในการหมุนของใบพัดหลักซึ่งจะสร้างแรงควบคู่รอบแกนโรเตอร์ทำให้ลำตัวของเฮลิคอปเตอร์หมุนตามไปด้วย จึงมีความจำเป็นอย่างยิ่งที่เฮลิคอปเตอร์จะต้องมีใบพัดหางหรือโรเตอร์ท้าย(ดังแสดงในรูปที่ 2.1)เพื่อทำหน้าที่ต้านแรงหมุนควบคู่ที่จะเกิดขึ้น[1]



รูปที่ 2. 1 เฮลิคอปเตอร์ต้องมีโรเตอร์ท้ายเพื่อทำหน้าที่ต้านแรงต้านที่เกิดขึ้น[2]

การสร้างแรงในทิศทางตรงกันข้ามกับแรงควบคู่ของใบพัดท้ายทำให้เฮลิคอปเตอร์สามารถลอยตัวนิ่งๆได้ แต่บางรุ่นของเฮลิคอปเตอร์ก็ไม่มีใบพัดหางหรือโรเตอร์ท้าย วิศวกรการบินผู้ออกแบบแก้ไขอาการหมุนรอบตัวเองด้วยการใช้แกนของใบพัดหลักแกนเดียวกัน แต่ใบพัดหลัก 2 ชุดซ้อนกันอยู่และหมุนสวนทางกัน[1] ลักษณะใบพัดหลัก 2 ชุดและหมุนสวนทางกันนี้จะมีอยู่ในเฮลิคอปเตอร์รุ่นใหม่ๆเช่น KA 27 Helix (ตามรูปที่ 2.2) , KA 25 Aligator และ KA32 หรือเฮลิคอปเตอร์บางรุ่นจะมีใบพัดหลัก 2 ชุดที่แยกออกจากกันในส่วนหน้าและหลัง โดยการทำงานของใบพัดหลัก 2 ชุดจะหมุนสวนทางกันเช่น เฮลิคอปเตอร์ลำเลียงยุทธวิธีขนาดใหญ่ CH47 ดังรูปที่ 2.3

รูปที่ 2. 2 เฮลิคอปเตอร์รุ่น KA 27 Helix [3]



รูปที่ 2. 3 เฮลิคอปเตอร์รุ่น CH47 [4]

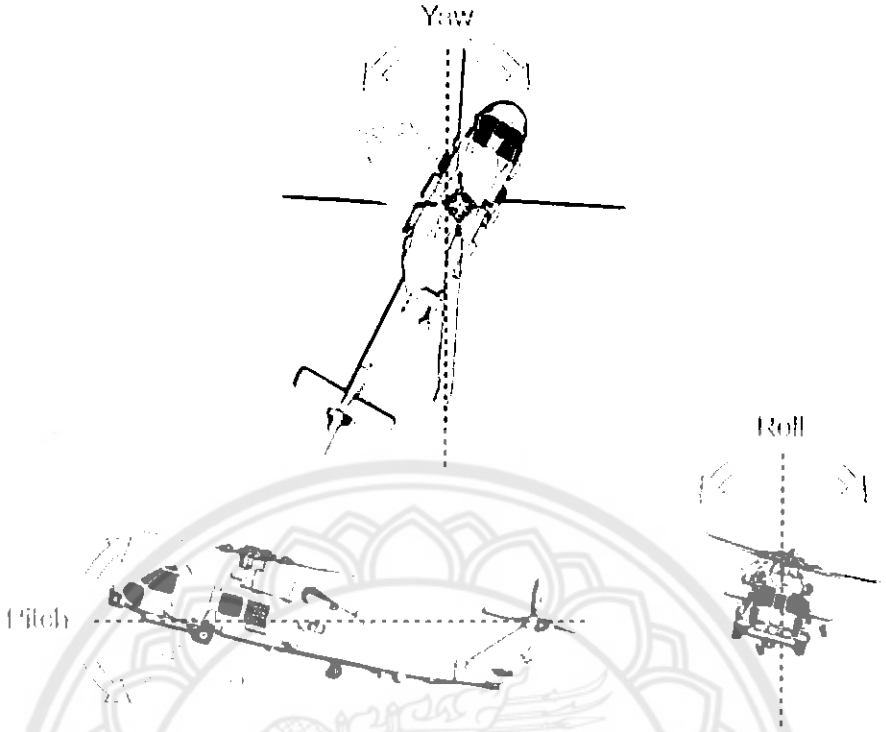
หัวใจสำคัญของอากาศยานทุกแบบก็คือเครื่องยนต์ ในสมัยแรกเฮลิคอปเตอร์จะใช้เครื่องยนต์แบบลูกสูบแต่ในปัจจุบันนั้นความก้าวหน้าทางเทคโนโลยีมีมากขึ้น เครื่องยนต์ส่วนใหญ่ของอากาศยานปีกหมุนจึงหันไปใช้เครื่องยนต์แบบ TurboShaft[1] ซึ่งเป็นหลักการเดียวกันกับเครื่องยนต์แบบ Turbopop[1] ของเครื่องบิน

หลักการทำงานของเครื่องยนต์ TurboShaft จะมีลักษณะการดูดอากาศมาผสมกับเชื้อเพลิงและจุดระเบิดใน Gas Generator เพื่อขับให้ใบพัดหมุน กำลังของเครื่องยนต์จะถูกส่งผ่าน Transmission Gear เข้าสู่เพลา (Shaft) เพื่อไปยังใบพัดหลักและใบพัดหาง โดยจะแบ่งอัตราส่วนกำลังที่ส่งไปดังนี้ 82% จะถูกส่งไปที่ใบพัดหลัก 10% จะถูกส่งไปที่ใบพัดหาง กำลังที่เหลืออีก 8% จะสูญเสียไปในระหว่างการหมุนและแรงเสียดทานในการขับดันเกียร์[1] นอกจากกลไกส่งผ่านกำลังส่วนหนึ่งของชุดกลไกจะต้องออกแบบให้มีชุดสำหรับหยุดการหมุนของใบพัดเมื่อเฮลิคอปเตอร์ลงจอดเรียบร้อยอีกด้วย

2.2 แกนอ้างอิงสำหรับวัดมุมเอียง

สาเหตุที่ต้องมีการวัดมุมเอียงของเฮลิคอปเตอร์เนื่องจากเมื่อเฮลิคอปเตอร์เอียงไปตามแนวแกนไหน ต้องมีการควบคุมเพื่อชดเชยมุมเอียงในแกนนั้นๆเพื่อที่จะทำให้เฮลิคอปเตอร์สามารถบินอยู่กลางอากาศได้ ไม่เกิดการหลงทิศ และไม่ทำให้เฮลิคอปเตอร์ตก

เครื่องบินจะมีแกนของการหมุนอยู่ 3 แกน ได้แก่ Longitudinal axis (Roll) (X) , Lateral axis (Pitch) (Y) และ Vertical axis (Yaw) (Z) ซึ่งแสดงในรูป 2.4



รูปที่ 2. 4 แกนสำหรับวัดมุมเอียง[5]

จากรูปที่ 2.4 แกน Longitudinal axis คือ แกนที่เริ่มตั้งแต่หัวเครื่องถึงหางเครื่อง เรียกการเคลื่อนที่รอบแกนนี้ว่า Roll โดยที่มุมตั้งต้นจะเริ่มที่ 0° , เมื่อเอียงไปทางขวาของแกนจะเป็นค่า + , เมื่อเคลื่อนที่ไปทางด้านซ้ายของแกนจะเป็นค่า - , ค่ามากที่สุดคือ +90° และค่าที่น้อยที่สุดคือ -90°

แกน Lateral axis คือ แกนที่เริ่มตั้งแต่ด้านข้างของเฮลิคอปเตอร์ฝั่งซ้ายมาถึงฝั่งขวา เรียกการเคลื่อนที่รอบแกน Lateral axis นี้ว่า Pitch โดยที่มุมตั้งต้นจะเริ่มที่ 0o , เมื่อก้มจะเป็นค่า + , เมื่อเงยจะเป็นค่า - , ค่ามากที่สุดคือ +90° และค่าที่น้อยที่สุดคือ -90°

แกน Vertical axis คือ แกนที่เริ่มตั้งแต่ใบพัดทางด้านบนจนถึงท้องเครื่องทางด้านล่าง เรียกการเคลื่อนที่รอบแกน Vertical axis นี้ว่า Yaw โดยจะอ้างอิงจากทิศเหนือเป็น 0° , ค่ามากที่สุดคือ 359° และมีทิศทางการเพิ่มขึ้นของมุมตามเข็มนาฬิกา

2.3 ความหมายและความสำคัญของ GUI

GUI คือประเภทของการติดต่อระหว่างผู้ใช้งานกับคอมพิวเตอร์อย่างหนึ่ง โดยการติดต่อกันระหว่างผู้ใช้งานกับคอมพิวเตอร์จะติดต่อกันผ่านสัญลักษณ์ภาพซึ่งมีประสิทธิภาพดีกว่าการติดต่อกันผ่านทางตัวอักษร โดยการใช้รูปภาพหรือสัญลักษณ์อื่นๆเพื่อแทนลักษณะต่างๆของโปรแกรมแทนการพิมพ์คำสั่งต่างๆในการสั่งให้ทำงาน จะช่วยให้ผู้ใช้งานสามารถทำงานได้ง่าย

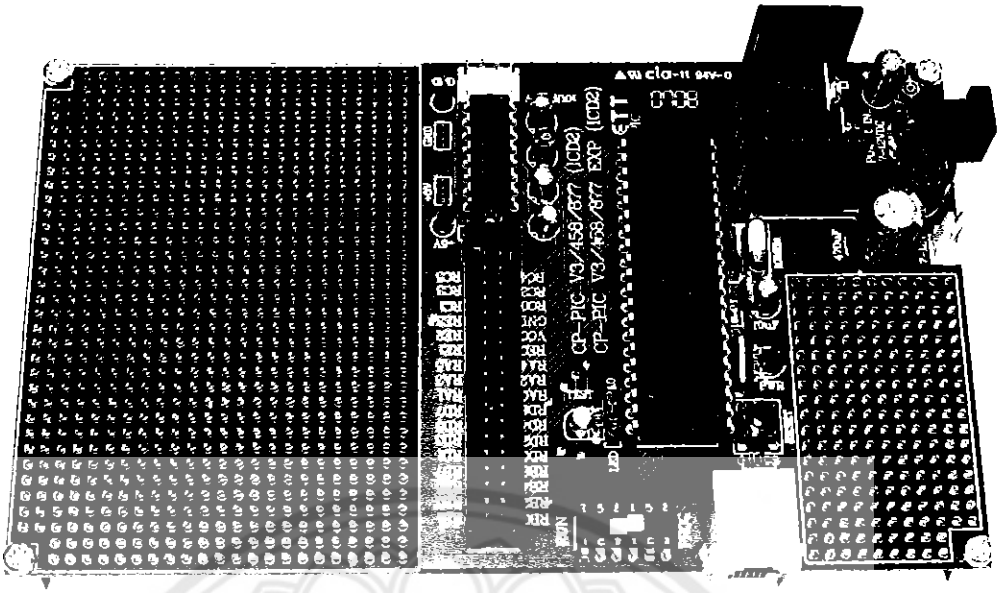
และมีความรวดเร็วมากขึ้น ไม่จำเป็นต้องจดจำคำสั่งต่างๆของโปรแกรมมากนัก ถือเป็นวิธีการเพิ่มความสะดวกและประสิทธิภาพการทำงานให้แก่ผู้ใช้คอมพิวเตอร์

GUI ถูกพัฒนาขึ้นโดยนักวิจัยจากสถาบันวิจัยสแตนฟอร์ดนำโดย ดัก เอนเกลบาร์ต (Doug Engelbart)[6] โดยการใช้งานร่วมกับไฮเปอร์ลิงก์และเมาส์ ซึ่งภายหลังได้นำมาวิจัยต่อที่ศูนย์วิจัยซีรีอกซ์พาร์ค (Xerox PARC) โดนใช้งานระบบกราฟิกแทนที่ระบบตัวอักษร โดยบางคนจะเรียกระบบนี้ว่า PARC User Interface หรือ PUI[7] ปลายคริสต์ศักราชที่ 1970 แอปเปิ้ลคอมพิวเตอร์ได้นำมาใช้กับเครื่องแมคอินทอชซึ่งภายหลังทางไมโครซอฟท์ได้นำมาใช้กับระบบปฏิบัติการวินโดวส์[8] ซึ่งในปัจจุบันนั้นระบบ GUI เป็นที่นิยมอย่างมากโดยสามารถเห็นได้ในทุกๆระบบปฏิบัติการไม่ว่าจะเป็นระบบปฏิบัติการแมคอินทอช[9] วินโดวส์[8] หรือลินุกซ์[10]ก็ตาม

ระบบ GUI สามารถนำเอามาประยุกต์ใช้กับการบอกสถานการณ์บินได้โดยการสร้างเป็นภาพจำลองของเฮลิคอปเตอร์ในการแสดงสถานะของมุมเอียงต่างๆตามจริงแบบ Real Time ซึ่งจะสามารถช่วยให้นักบินสามารถรู้สถานะของเฮลิคอปเตอร์ที่กำลังควบคุมอยู่ได้ ส่งผลให้นักบินสามารถควบคุมเฮลิคอปเตอร์ให้ไปในทิศทางที่ต้องการ โดยที่ไม่หลงทิศและป้องกันการตกของเฮลิคอปเตอร์ได้

2.4 บอร์ดไมโครคอนโทรเลอร์ CP-PIC V3.0 (ICD2)

บอร์ด CP – PIC V3.0 (ICD2) นี้เป็นบอร์ดที่ออกแบบวงจรพื้นฐานที่จำเป็น เช่น แหล่งจ่ายไฟ วงจรรีเซ็ต วงจรกำเนิดคลื่นสัญญาณนาฬิกา พอร์ทสำหรับ Download โปรแกรม และวงจรสื่อสารอนุกรม ดังแสดงในรูป 2.5



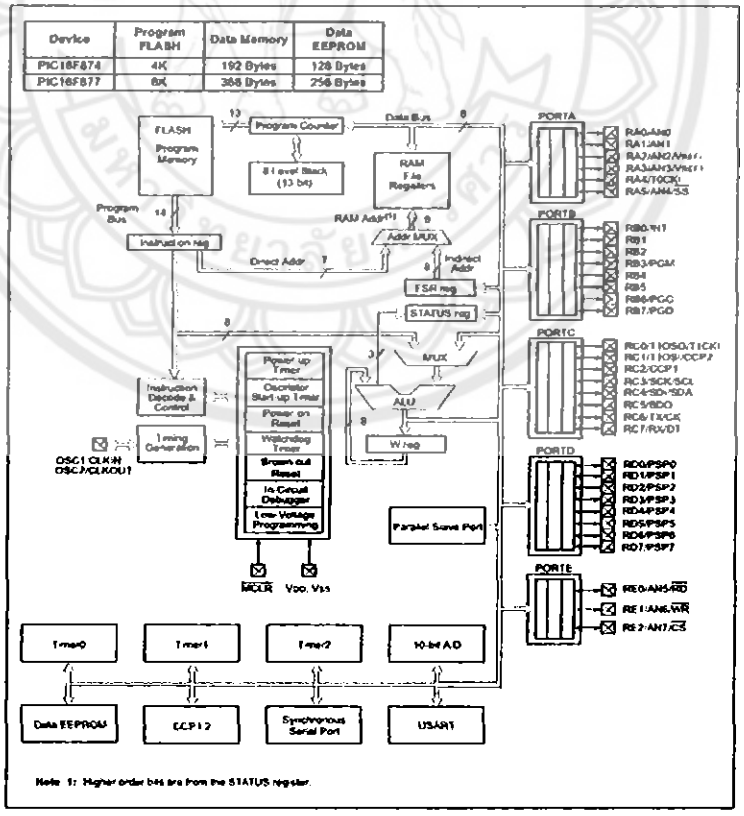
รูปที่ 2. 5 บอร์ดไมโครคอนโทลเลอร์ CP-PIC V3.0 (ICD2)[11]

2.4.1 สถาปัตยกรรมและโครงสร้างทางด้านฮาร์ดแวร์ของ PIC 16F877

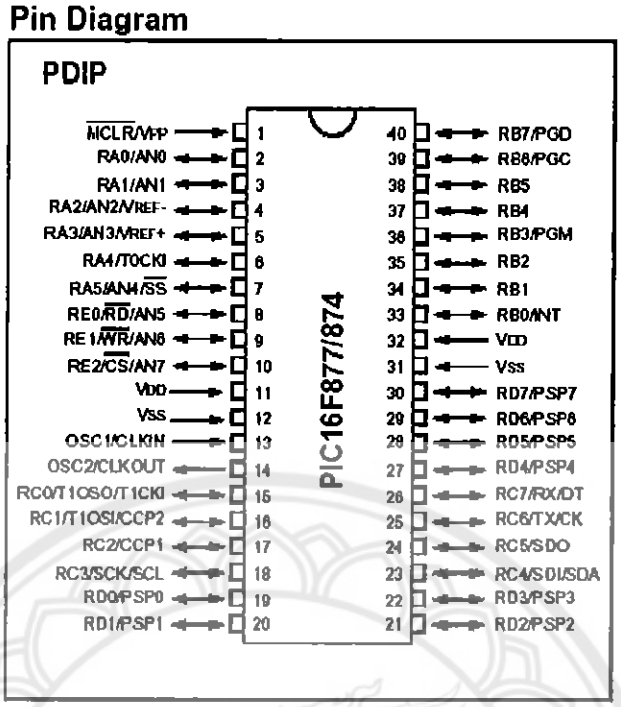
สถาปัตยกรรมของ PIC 16F877 ดังแสดงในรูป 2.6 เป็น CPU ของบริษัท Microchip Technology โดย IC เบอร์นี้เป็นเบอร์ที่มีความสามารถสูงปานกลาง ประกอบไปด้วยฟังก์ชั่นต่างๆ ดังนี้

- มี 35 Instruction คำสั่ง
- ในคำสั่งต่างๆจะใช้ 1 Cycle และใช้ 2 Cycle ในคำสั่งที่เป็นการกระโดด
- ความถี่สูงสุดที่ทำงานได้คือ 20 MHz
- การทำงานเป็นไปในลักษณะ Pipeline ทำให้มีการทำงานเร็วขึ้น
- หน่วยความจำโปรแกรม Flash Program Memory มีขนาด 8k (14-Bit Words)
- หน่วยความจำข้อมูล RAM 368Bytes
- หน่วยความจำข้อมูล (EEPROM) 256 Bytes
- สามารถตอบสนองการอินเตอร์รัพท์ได้ 14 แหล่ง
- มี Stack 8 ระดับ
- มี Power On Reset(POR) Power Up Timer (PWRT) และ Oscillator Start-Up Timer
- Watchdog Timer
- สามารถเลือกการป้องกันข้อมูลได้ (Code Protection)
- มีโหมดประหยัดพลังงาน (Sleep Mode)

- เลือกโหมดของสัญญาณนาฬิกาได้หลายโหมด
- สามารถโปรแกรมโดยใช้แรงดัน 5 V ได้
- มีฟังก์ชั่นการโปรแกรมแบบ In-Circuit Serial Programming (ICSP)
- ทำงานที่ไฟเลี้ยง 2.0-5.0 V
- การะแสทั้งซิงก์และซอร์สของพอร์ตคือ 25 mA
- มี Timer/Counter จำนวน 3 ตัว คือ Timer0 Timer1 และ Timer2
- มีโมดูล Capture/Compare/PWM จำนวน 2 ชุด
- มี Analog to Digital Converter แบบ 10 บิต 8 แชลแนล ภายในตัว
- มีโมดูลสื่อสารแบบ USART
- มีโมดูลตรวจจับระดับไฟเลี้ยง Brown – Out Reset (BOR)
- มีพอร์ต I/O 5 พอร์ต ประกอบไปด้วย A,B,C,D และ E ซึ่งรวมแล้วจะมี I/O จำนวน 33 Bit ดังแสดงในรูปที่ 2.7



รูปที่ 2. 6 สถาปัตยกรรมภายในของ PIC 16F877[11]



รูปที่ 2. 7 ตำแหน่งจาสัญญาณต่างๆของ PIC 16F877 [11]

2.5 อุปกรณ์สำหรับตรวจจับมุมเอียงในแนวแกน X และ Y

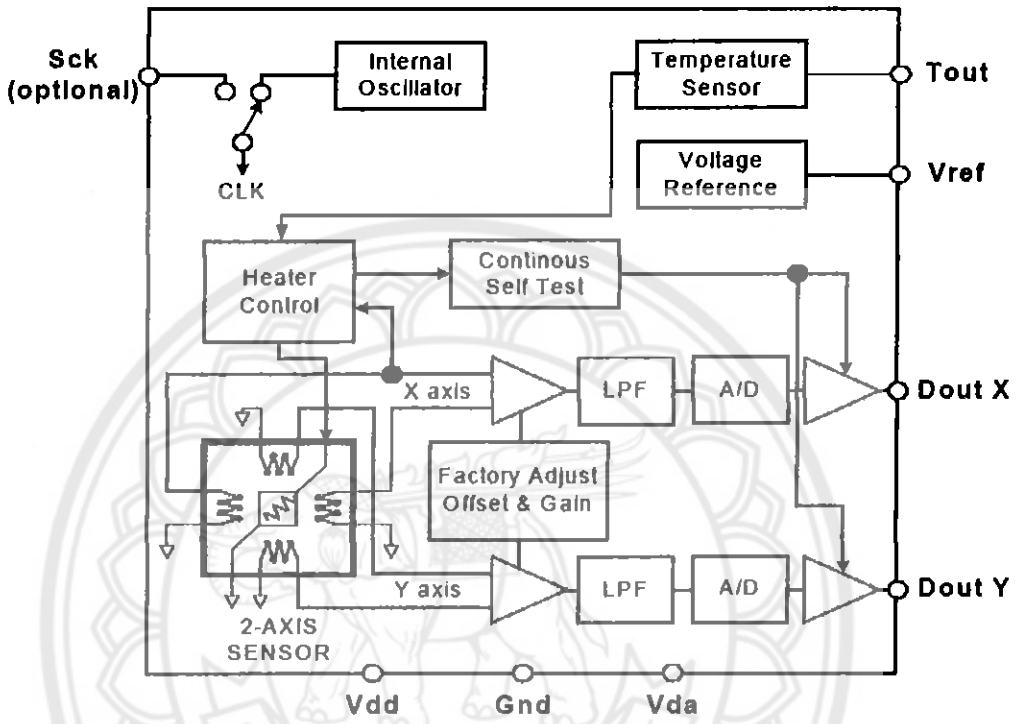
ตัวเซนเซอร์สำหรับตรวจจับมุมในแนวแกน x และ y ที่นำมาใช้งานมีชื่อว่า MEMSIC 2125[12] เป็นเซนเซอร์สำหรับตรวจจับความเร่งแบบ 2 แกน ซึ่งค่าที่ได้ออกมานั้นจะเป็นความเร่งในแนวแกนX และ Y ซึ่ง เราสามารถนำค่านี้มาคำนวณหาค่ามุมเอียงได้ ซึ่งจะมีรูปร่างดังแสดงในรูป 2.8

รูปที่ 2. 8 รูปร่างและการจัดขาของ MEMSIC2125[12]

2.5.1 รายละเอียดทั่วไป

โมดูล MEMSIC2125 นี้มีลักษณะใช้วัดความเร่งแบบ 2 แกน คือแกน X และแกน Y สร้างบนมาตรฐาน CMOS ที่มีระบบการตรวจจับเซนเซอร์สมบูรณ์แบบพื้นฐาน SoC (System on Chip)[13] สามารถวัดค่าได้ในย่านความเร่ง -2g ถึง +2g (โดยที่ค่า g คือค่าความเร่งที่เกิดจากแรง

โน้มถ่วงของโลก มีค่าประมาณ 9.81 m/s^2) มีความละเอียดในการวัดสูงกว่า 1mg มีความเร็วในการวัดมากกว่า 12.5% ต่อ g และสามารถทนทานต่อแรงสั่นสะเทือนถึง 5000g โดยที่เซนเซอร์ตัวนี้จะรับแรงดันไฟเลี้ยงได้ 3 ถึง 5.25 V[12] โดยจะมีบล็อกไดอะแกรมดังแสดงในรูป 2.9



รูปที่ 2. 9 บล็อกไดอะแกรมสำหรับ MEMSIC 2125[12]

2.5.2 ขาสำหรับการใช้งาน ดังแสดงในรูป 2.8

VDD: ขาแหล่งจ่ายแรงดันสำหรับวงจรแบบดิจิตัลที่ใช้สำหรับเซนเซอร์ความร้อนภายในตัวไอซีเป็นไฟดีซีระหว่าง 3.0 ถึง 5.25 โวลต์

VDA: ขาแหล่งจ่ายแรงดันสำหรับตัวขยายสัญญาณแบบอะนาลอกภายในไอซี โดยปกติแล้วจะเชื่อมต่อกันกับ VDD อยู่แล้วนั่นก็คือจะมีแรงดันค่าเท่ากัน

GND: ขากราวด์

DoutX: เป็นขาสำหรับการแสดงผลลัพธ์ดิจิตัลจากการตรวจสอบความโน้มถ่วงในแนวแกน X มีการกำหนดจากโรงงานให้ใช้ได้ในช่วงความถี่ระหว่าง 100 – 400 เฮิร์ต

DoutY: เป็นขาสำหรับการแสดงผลลัพธ์ดิจิตัลจากการตรวจสอบความโน้มถ่วงในแนวแกน Y มีการกำหนดจากโรงงานให้ใช้ได้ในช่วงความถี่ระหว่าง 100 – 400 เฮิร์ต เช่นกัน

Tout: เป็นขาบัฟเฟอร์เอาท์พุทสำหรับการวัดค่าอุณหภูมิ ค่าแรงดันที่ได้จากขา Tout นี้จะเป็นตัวบ่งชี้ถึงอุณหภูมิที่มีการเปลี่ยนแปลง ซึ่งค่าแรงดันที่แสดงจากขานี้จะเป็นค่าอุณหภูมิที่เปลี่ยนแปลงจากอุณหภูมิแวดล้อมรอบๆ ไม่ใช่ค่าอุณหภูมิที่วัดได้จริง

SCK: เป็นขาสัญญาณนาฬิกามาตรฐานที่กำหนดมาจากโรงงานอยู่ภายในไอซีมีความถี่ 800 กิโลเฮิร์ต

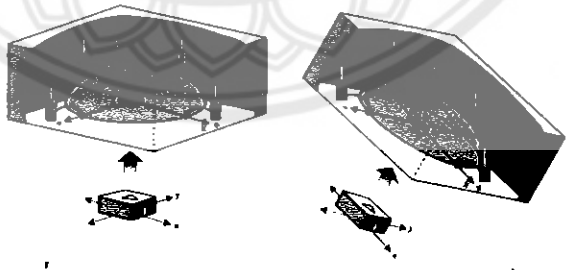
Vref: เป็นขาสัญญาณเปรียบเทียบหรือขาแรงดันอ้างอิงที่อยู่ระหว่าง 2.50 โวลต์ และใช้กระแส100 ไมโครแอมป์

2.5.3 หลักการทำงาน

ในตัว MEMSIC2125 จะมีลักษณะคล้ายๆห้องที่เก็บแก๊สอยู่โดยที่จะมีแหล่งกำเนิดความร้อนอยู่ตรงกลางระหว่างแกน X และ แกน Y ที่ตัดกัน และมีเซนเซอร์สำหรับตรวจจับอุณหภูมิอยู่ทั้ง 4 มุม

ลักษณะการทำงานคือหากความเร่งหรือแรงโน้มถ่วงไม่มีการเปลี่ยนแปลง (มีค่าเป็น 0) แก๊สจะแพร่อยู่ตรงกลางของห้องทำให้อุณหภูมิของเซนเซอร์ทั้ง 4 มีขนาดเท่ากัน แต่เมื่อมีแรงภายนอกมากระทำให้เกิดแรงโน้มถ่วงเช่นเกิดการเอียงหรือเคลื่อนไหวจะทำให้แก๊สนั้นแพร่ไปใกล้เซนเซอร์สำหรับวัดอุณหภูมิตัวใดตัวหนึ่งทำให้ค่าที่วัดได้จากเซนเซอร์นั้นแตกต่างกันไป ดังแสดงในรูปที่ 2.10

MEMSIC2125 จะแปลงค่าจากอุณหภูมิเป็นสัญญาณพัลส์ที่มีค่าดิวตี้ไซเคิลที่แตกต่างกันไปซึ่งจะง่ายต่อการวัดและการแปลงค่าสำหรับการนำไปใช้โดยบอร์ดไมโครคอนโทลเลอร์

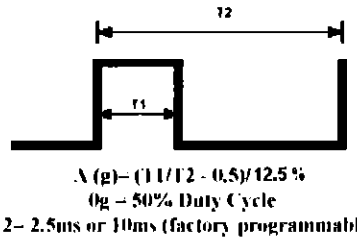


รูปที่ 2. 10 แสดงหลักการทำงานของเซนเซอร์ [12]

2.5.4 Duty Cycle

โดยปกติสัญญาณพัลส์ที่เกิดขึ้นจะเกิดขึ้น 2 ค่า คือค่า X และค่า Y ในภาวะปกตินั้นคือกรณีที่แรงโน้มถ่วงปกติหรือค่า g เป็นศูนย์ค่าเอาท์พุตที่ได้จะมีค่าดิวตี้ไซเคิล 50% หรืออัตราส่วน

ระหว่างT1(เวลาเมื่อลอจิกเป็น1) กับ T2 (ความกว้างพัลส์ซึ่งในเซนเซอร์ตัวนี้มีค่า 10 ms) มีค่าเท่ากับ 50 % ดังแสดงในรูปที่ 2.11



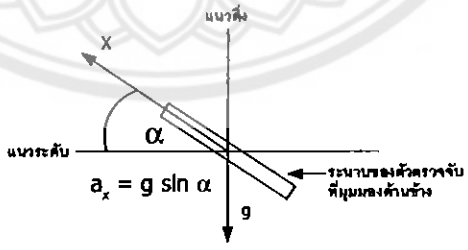
รูปที่ 2. 11 แสดงเอาต์พุทดิวตี้ไซเคิล

ค่าดิวตี้ไซเคิ้ลจะเปลี่ยนแปลงไปตามสเกลเท่ากับ 12.5% ต่อค่า g หนึ่งค่า ดังนั้นเราสามารถหาค่าความเร่งได้โดยเข้าสมการต่อไปนี้

$$A(g) = \frac{a_x}{g} = \frac{\left(\frac{T1}{T2} - 0.5\right)}{0.125}$$ **สมการ 2. 1**

2.5.5 การประยุกต์ใช้งานโมดูล MEMSIC2125 ตรวจจับความลาดเอียง

เมื่อเซนเซอร์ทำมุมเอียงในแนวใดๆกับแกนแนวดิ่งของโลกจะส่งผลให้อนุภาคของแก๊สภายในตัวเซนเซอร์นั้นเคลื่อนที่ทำให้อุณหภูมิที่ตรวจจับได้ไม่สมดุลส่งผลให้มีการเปลี่ยนแปลงค่าดิวตี้ไซเคิลของสัญญาณพัลส์ที่ขาเอาท์พุต โดยค่าดิวตี้ไซเคิลที่เปลี่ยนแปลงไปจากสภาวะปกตินั้นจะมีความสัมพันธ์กับมุมที่เอียงไปในแต่ละแกน ดังแสดงในรูปที่ 2.12



รูปที่ 2. 12 การประยุกต์ใช้งานโมดูล MEMSIC2125 กับความลาดเอียง[12]

การคำนวณหามุมเอียงนั้นสามารถทำได้โดยหาค่า A(g) จากสัญญาณพัลส์ที่อ่านมาได้จากสมการที่2.1 หลังจากนั้นนำค่า A(g)ที่ได้มาเข้าสมการ

จาก $$a(g) = a_x/g = g \sin \propto$$

จะได้ $$\propto = \arcsin(a(g))$$ สมการ 2. 2

จากสมการที่ 2.2 นั้นสามารถสร้างตารางสรุปการมุมเอียงจากความเร่งที่ขา X_{out} หรือ Y_{out} ที่ความเร่งค่าต่างๆดังแสดงในตารางที่ 2.1

ตารางที่ 2. 1 ตารางแสดงค่าความเร่งที่ขา X_{out} หรือ Y_{out} ที่มุมเอียงค่าต่างๆ[12]

มุมเอียง (องศา)	เอาต์พุตที่ขา Xout หรือ Yout (g)
90	1.000
85	0.996
80	0.985
70	0.940
60	0.866
45	0.707
30	0.500
20	0.342
10	0.174
5	0.087
0	0.000

2.6 อุปกรณ์สำหรับตรวจจับมุมเอียงในแนวแกน Z

ตัวเซนเซอร์สำหรับตรวจจับมุมในแนวแกน Z ที่นำมาใช้งานมีชื่อว่า CMPS03 Digital Compass[14] เป็นโมดูลเข็มทิศดิจิตอล หลักการทำงานโดยอาศัยการตรวจจับสนามแม่เหล็กโลก

2.6.1 รายละเอียดทั่วไป

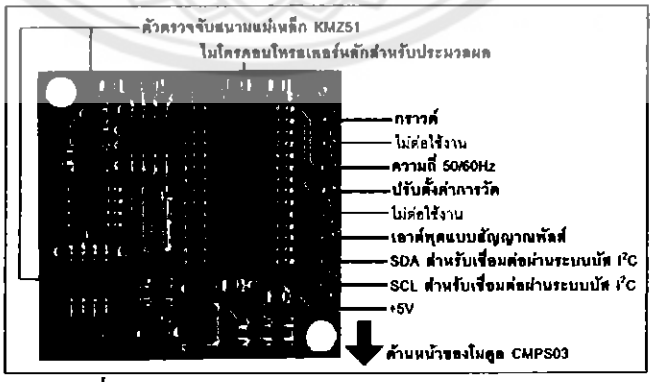
CMPS03 เป็นโมดูลเข็มทิศดิจิตอล เป็นผลงานของ Devanteech ออกแบบมาเพื่อช่วยในการกำหนดทิศทางการเคลื่อนที่ของหุ่นยนต์อัตโนมัติ และนำมาใช้ในการสร้างเครื่องมือวัดและตรวจสอบทิศทางระบบอิเล็กทรอนิกส์ โดยหัวใจหลักของโมดูล CMPS03 คือ ตรวจจับสนามแม่เหล็กโลก [14]

CMPS03 ต้องการไฟเลี้ยง +5V และกระแสไฟฟ้า 20mA ใช้ตัวจับสนามแม่เหล็กเบอร์ KMZ51 ของ Philips จำนวน 2 ตัว เพื่อช่วยตรวจจับตำแหน่งของสนามแม่เหล็กโลก ซึ่งมีความละเอียดของมุม 0.1 องศา ค่าความผิดพลาดที่ได้หลังจากการปรับประมาณ 3-4 องศา โดยเอาต์พุตที่ได้จะเป็นสัญญาณพัลส์และข้อมูลดิจิตอล สัญญาณพัลล์จะมีความกว้าง 1 ถึง 37 มิลลิวินาที มีอัตราการเพิ่มครั้งละ 0.1 มิลลิวินาที ส่วนข้อมูลดิจิตอลจะติดต่อผ่านบัส I^2C[15] รองรับสัญญาณนาฬิกาความถี่สูงถึง 1MHz โดยให้ข้อมูล 2 รูปแบบคือ 0-255 และ 0-3599 CMPS03 มีอุปกรณ์เสริม คือ บอร์ด ADX-CMPS03 เป็นบอร์ดอะแดปเตอร์ และสาย PCB34 สำหรับเชื่อมต่อควบคุมกับบอร์ดควบคุมหุ่นยนต์ของ i-nex[16]

2.6.2 ตำแหน่งขาและการใช้งาน

โมดูลเซนเซอร์ CMPS03 มีขาทั้งหมดอยู่ 9 ขา ดังรูปที่ 2.13 โดยที่แต่ละขานั้นมีรายละเอียดดังนี้

- ขา 1 ต่อสำหรับต่อกับไฟ 5V
- ขา 2 , ขา 3 ใช้สำหรับอ่านค่ามุมจากเซ็นเซอร์ผ่าน I^2C[15]
- ขา 4 ให้ค่าออกมาเป็น PWM [17]
- ขา 5 , ขา 8 ไม่ได้ใช้งาน
- ขา 6 ใช้สำหรับปรับแต่งค่าทิศทางอ้างอิง
- ขา 7 ใช้สำหรับเลือกย่านความถี่ 50 หรือ 60 Hz
- ขา 9 ใช้สำหรับต่อกับกราว



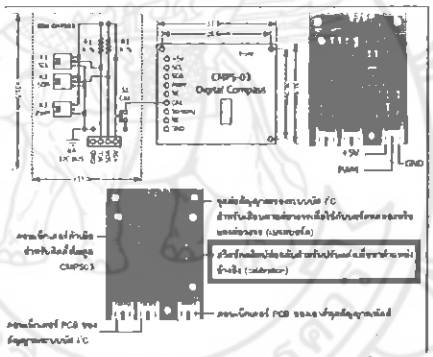
รูปที่ 2. 13 แสดงรูปร่างและขาการต่อใช้งาน[14]

2.6.3 การปรับแต่งค่าทิศทางอ้างอิง

เพื่อให้การวัดทิศทางของโมดูล CMPS03 มีความแม่นยำมากที่สุด จึงมีอินพุตสำหรับปรับแต่งค่าทิศทางอ้างอิง โดยจะป้อนสัญญาณลอจิก 0 เข้าที่ขาอินพุต ซึ่งบอร์ด ADX-CMPS03 ที่แยกมานั้นมีสวิตซ์เพื่อกดสำหรับปรับแต่งค่า ดังรูปที่ 2.14 บริเวณที่มีกรอบสีแดงล้อมรอบ โดยมีขั้นตอนการปรับแต่งดังนี้

1. วางโมดูล CMPS03 ขนานกับพื้น หันด้านหน้าของโมดูลไปทางทิศเหนือ กดสวิตช์ 1 ครั้ง
2. วางโมดูล CMPS03 ขนานกับพื้น หันหน้าของโมดูลไปทางทิศตะวันออก กดสวิตช์ 1 ครั้ง
3. วางโมดูล CMPS03 ขนานกับพื้น หันหน้าของโมดูลไปทางทิศใต้ กดสวิตช์ 1 ครั้ง
4. วางโมดูล CMPS03 ขนานกับพื้น หันหน้าของโมดูลไปทางทิศตะวันตก กดสวิตช์ 1 ครั้ง

เมื่อปรับแต่งเสร็จ ไม่จำเป็นต้องปรับค่าใหม่อีกครั้งเมื่อจ่ายไฟเลี้ยงครั้งใหม่ เพราะโมดูลจะเก็บค่าอ้างอิงที่ได้ไว้ในหน่วยความจำอีอีพรอม[18]



รูปที่ 2. 14 แสดงวงจรบอร์ด ADX-CMPS03 และเชื่อมต่อกับ CMPS03[14]

2.6.4 การอ่านค่าทิศทางเอาต์พุตสัญญาณพัลส์

การอ่านค่าทิศทางจากเอาต์พุตสัญญาณพัลส์ เป็นการนำค่าความกว้างพัลส์ที่ได้มาระบุตำแหน่งองศา จาก 0 ถึง 359.9 องศา โดยมีย่านความกว้างของสัญญาณพัลส์ที่เป็นลอจิก “1” จาก 1 มิลลิวินาทีไปจนถึง 36.99 มิลลิวินาที มีความละเอียด 0.1 มิลลิวินาทีต่อองศา โดยที่ระหว่างพัลส์นั้นจะมีช่วงลอจิก “0” กว้าง 65 มิลลิวินาที[14]

2.7 ไลบรารีสำหรับสร้างภาพ 3 มิติในการแสดงผล

ทางผู้ดำเนินโครงงานได้ใช้ ไลบรารี ที่ชื่อว่า OpenGL มาใช้ในสำหรับการสร้างภาพ 3 มิติในการแสดงผล โดยจะขอกล่าวถึงรายละเอียดจำแนกออกเป็นส่วนๆตามลำดับดังนี้

1. OpenGL คืออะไรพอสังเขป

2.ไลบรารีที่เกี่ยวข้องกับ OpenGL โดยรวม

3. ฟังก์ชันการมองภาพ 3 มิติของ OpenGL

4. การวาดรูปสี่เหลี่ยมโดยใช้คำสั่งของ OpenGL

5. หลักการหมุนภาพ (Rotation)

2.7.1 OpenGL คืออะไร

OpenGL (Open Graphics Library) เป็นซอฟต์แวร์ไลบรารี (Software Library) ที่ใช้ติดต่อกับฮาร์ดแวร์เพื่อการแสดงภาพกราฟฟิก โดยจะมีคำสั่งพื้นฐานสำหรับการวาดภาพ คือ จุด เส้นและรูปเหลี่ยมต่างๆ การแสดงภาพแรนเดอร์[19]และยังสามารถควบคุมการทำงานของแอปพลิเคชัน 3 มิติได้ (คือแอปพลิเคชันที่แสดงผลได้ 3 มิติคือ กว้าง, สูง, ลึก ในหน้าจอแสดงผล 2 มิติ)

ภาษาที่สามารถใช้กับ OpenGL ได้แก่ C/C++ (VC++,Borland C++,C++ Builder,C compiler,on UNIX),Delphi,Visual Basic,Java Perl,Python,Fortran และ Ada เป็นต้น[20]

โครงสร้างของ OpenGL ถูกออกแบบมาให้ทำงานได้บนทุกๆ แพล็ตฟอร์ม (Independent Platform) ทำให้สามารถเคลื่อนย้ายโปรแกรมที่สร้างเรียบร้อยแล้วไปใช้งานบนแพล็ตฟอร์มอื่นได้อย่างสะดวกและไม่ต้องเปลี่ยนแปลงโปรแกรมเลย จึงทำให้ไม่มีคำสั่งที่จัดการกับระบบปฏิบัติการเลย อีกทั้งยังไม่มีคำสั่งเพื่อรับอินพุตจากผู้ใช้อีกด้วย หน้าที่ทั้งสองอย่างนี้หากทำการพัฒนาโปรแกรมบน Windows จะมียูทิลิตี้ที่ช่วยจัดการคือ GLUT (OpenGL Utility Toolkit) นอกจากนี้ยังไม่มีคำสั่งระดับสูงที่จะใช้วาดวัตถุ 3 มิติแบบซับซ้อน เช่น รถยนต์ อวัยวะหรือโมเลกุลต่างๆ สิ่งที่ OpenGL มีให้คือการวาดรูปทรงเลขาคณิตอย่างง่ายได้แก่ จุด เส้น และรูปหลายเหลี่ยม ซึ่งผู้ใช้งานจะต้องนำรูปเหล่านี้มาประกอบกันเพื่อให้เกิดรูปทรง 3 มิติ ที่ซับซ้อน โดยมีไลบรารีที่เป็นตัวช่วยจัดการทำงานในหน้าที่นี้คือ Open Inventor[21]

2.7.2 ไลบรารีที่เกี่ยวข้อง

2.7.2.1 OpenGL Utilities (GLU)

OpenGL Utility (GLU) เป็นไลบรารีที่ประกอบด้วยรูทีนมากมายในการจัดการมุมมองเพื่อแสดงรูปพื้นฐานและออปเจ็กต์ที่ซับซ้อนที่ประกอบขึ้นจากเส้นและรูปหลายเหลี่ยม, แสดงรูปลูกบาศก์ ,การเรนเดอร์พื้นผิว(surface-rendering)[19] และงานที่ซับซ้อน ซึ่งฟังก์ชันของ GLU จะขึ้นต้นด้วยคำว่า glu เสมอ การเรียกใช้จะต้อง include ไฟล์ header ที่ชื่อ glu.h ในตอนต้นของโค้ดโปรแกรม นอกจากนี้ยังมีชุดเครื่องมือสำหรับพัฒนาเชิงวัตถุ (Object-Oriented) ที่อ้างอิงกับ OpenGL อีก เรียกว่า Open Invector ซึ่งมีรูทีนและรูปทรงออปเจ็กต์สำหรับแอปพลิเคชัน 3 มิติจำนวนมาก โดยชุดเครื่องมือนี้เขียนด้วยภาษา C++[20]

2.7.2.2 OpenGL Utility Toolkit(GLUT)

OpenGL Utility Toolkit(GLUT) คือ ไลบลารีของระบบกราฟฟิกที่ช่วยในการติดต่อกับการแสดงผลทางจอภาพ ซึ่ง GLUT เป็นชุดเครื่องมือที่มีไลบลารีของฟังก์ชันสำหรับการใช้งานกับระบบวินโดว์ของจอภาพทั่วๆไป คำสั่งของ GLUT จะขึ้นต้นด้วยคำว่า glut เสมอ การเรียกใช้ต้อง include ไฟล์ header ที่มีชื่อ glut.h ในตอนต้นของโค้ดโปรแกรมเช่นกัน โดยในไลบลารี จะประกอบด้วยเมธอดสำหรับการสร้างและเรนเดอร์ quadric curves และพื้นผิว

2.7.3 ฟังก์ชันการมองภาพ 3 มิติของ OpenGL

OpenGL Utility Library (GLU) มีฟังก์ชันสำหรับการกำหนดพารามิเตอร์การมองภาพ 3 มิติ และฟังก์ชันอื่นในการกำหนดค่าการแปลง Projection แบบ Perspective แบบสมมาตร ฟังก์ชันอื่นๆ เช่น ฟังก์ชันที่ใช้ใน Projection เชิงตั้งฉาก Projection แบบPerspective เฉียง และการแปลง Viewport ก็มีอยู่แล้วในไลบรารีพื้นฐานของ OpenGL นอกจากนี้ ฟังก์ชันของ GLUT ก็ได้รวบรวมฟังก์ชันพื้นฐานในการแสดงและจัดการวินโดว์แสดงผลไว้ให้แล้วด้วย[20]

2.7.3.1 ฟังก์ชันการมองเห็นภาพ

หากต้องการแสดงภาพใน OpenGL จะมีการสร้างเมทริกซ์และรวมเข้ากับเมทริกซ์แสดงผลในปัจจุบัน หลังจากนั้นเมทริกซ์ที่ใช้ในการมองภาพจะรวมเข้ากับการแปลงทางเรขาคณิตที่เรากำหนด หลังจากนั้นเมทริกซ์รวมจะแปลงจากโคออร์ดิเนตทางกายภาพไปเป็นโคออร์ดิเนตสำหรับการมอง เรากำหนดโหมดการมองด้วยคำสั่ง

```
glMatrixMode (GL_MODELVIEW);
```

คำสั่งของ OpenGL ที่ใช้สำหรับกำหนดขนาดของหน้าจอทั้งหมดที่ OpenGL จะสามารถวาดลงไปได้คือคำสั่ง

```
glViewport(x, y, width, height);
```

สำหรับรายละเอียดของพารามิเตอร์เป็นดังนี้ x,y เป็นการกำหนดจุดโคออดิเนตมุมซ้ายล่างของหน้าจอ ค่า default จะกำหนดมาเป็น 0 0 ค่า width นั้นจะเป็นพารามิเตอร์สำหรับกำหนดความกว้างของหน้าจอ และค่า height จะเป็นพารามิเตอร์สำหรับกำหนดความสูงของหน้าจอ

คำสั่งสำหรับการกำหนดจุดสำหรับการกำหนดการมองคือคำสั่ง

gluLookAt (x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);

ค่าพารามิเตอร์ทั้งหมดในฟังก์ชันนี้กำหนดเป็นเลขทศนิยม ฟังก์ชันนี้กำหนดเฟรมอ้างอิงการมองไว้ที่จุดกำเนิดที่โคออร์ดิเนตตำแหน่ง P_0 = (x0, y0, z0) (จุดที่ตาเรามอง) ตำแหน่งอ้างอิงเป็น P_{ref} = (xref, yref, zref) (จุดที่วางวัตถุ) และ view-up vector V = (Vx, Vy, Vz) (คือการวางกล้องโดยที่จะกำหนดว่าจะวางกล้องในแนวทางไหน เช่น V = (0,1,0) เส้น imaginary line จะเริ่มตั้งแต่จุดกลางของภาพไปยังด้านบนของกล้อง (แนวแกน Y) หรือ V = (1,0,0) เส้น imaginary line

จะเริ่มตั้งแต่จุดกลางของภาพไปยังซ้ายของกล้อง (แนวแกน X) คือเหมือนเอียงกล้องไปตามแนวแกน X แล้วถ่ายเป็นต้น) ถ้าไม่ได้กำหนดค่าพารามิเตอร์ในฟังก์ชัน ค่า Default ใน OpenGL จะกำหนดเป็น

- P_0 = (0, 0, 0)
- P_{ref} = (0, 0, -1)
- V = (0, 1, 0)

ค่า Default เหล่านี้ เฟรมอ้างอิงการมองจะเหมือนกับเฟรมทางกายภาพที่มีทิศทางกับแกน -z (แกน z คือแกนในแนวลึกของหน้าจอ -z คือลึกและเวกเตอร์ชี้เข้าไปในหน้าจอ) ในหลายๆ แอปพลิเคชั่น เพื่อความสะดวกเราสามารถใช้ค่า Default สำหรับพารามิเตอร์การมองได้

2.7.3.2 ฟังก์ชัน Projection เชิงตั้งฉาก

เนื่องจากเมทริกซ์ใน Projection ถูกเก็บอยู่ในโหมด Projection ของ OpenGL ดังนั้นจะต้องมีการกำหนดโหมดในการแสดงเมทริกซ์เพื่อ Projection ขึ้นมาก่อน โดยใช้คำสั่ง

glMatrixMode (GL_PROJECTION);

หลังจากนั้น เมื่อเรามีการใช้คำสั่งเพื่อการแปลงเมทริกซ์ที่เป็นผลลัพธ์จะรวมกับเมทริกซ์ใน Projection ปัจจุบัน พารามิเตอร์ใน Projection เชิงตั้งฉากจะใช้ฟังชัน

glOrtho (xwmin, xwmax, ywmin, ymax, dnear, dfar);

ค่าพารามิเตอร์ทุกค่าในฟังก์ชันนี้กำหนดเป็นเลขทศนิยม เราใช้ฟังก์ชัน glOrtho เพื่อเลือกโคออร์ดิเนตของวินโดว์ขริบภาพและระยะใกล้-ไกลของระนาบการขริบจากจุดกำเนิดการมอง

ฟังก์ชัน glOrtho สร้าง Projection แบบขนานที่ตั้งฉากกับระนาบการมอง (ระนาบการขริบด้านใกล้) ดังนั้นฟังก์ชันนี้จะสร้างปริมาตรการมองระยะอนันต์สำหรับ Projection เชิงตั้งฉากสำหรับระนาบการขริบและวินโดว์ขริบภาพที่กำหนด ใน OpenGL จะต้องกำหนดระนาบการขริบระยะใกล้และไกลสำหรับ Projection ด้วย

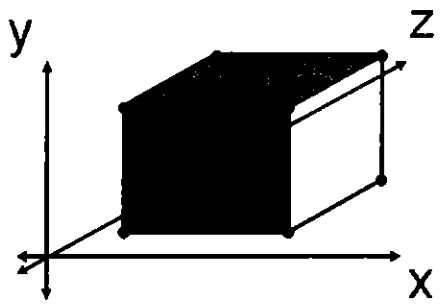
ค่าพารามิเตอร์ dnear และ dfar เป็นระยะทางที่มีทิศทางกับแกน -zview (การมองในแนวลึกเข้าไปในหน้าจอ) จากจุดกำเนิดการมอง ยกยกอย่างเช่น ถ้า dfar = 55.0 ดังนั้นระนาบการขริบระยะไกลจะอยู่ที่ตำแหน่งโคออร์ดิเนต Z_{far} = -55.0 ค่าลบของพารามิเตอร์แสดงถึงระยะทางที่อยู่ข้างหลังจุดกำเนิดการมองเมื่อเทียบกับแกน Z_{view} โดยสามารถกำหนดค่าต่างๆ (บวก, ลบ หรือศูนย์) ให้กับพารามิเตอร์เหล่านี้เมื่อ dnear < dfar

ผลลัพธ์ของปริมาตรการมองสำหรับ Projection นี้จะเป็นทรงสี่เหลี่ยมด้านขนาน ตำแหน่งโคออร์ดิเนตภายในปริมาตรการมองจะถูกแปลงไปยังตำแหน่งภายในลูกบาศก์เชิงปริมาตรในเฟรมอ้างอิงแบบมือซ้าย ด้วยค่า Z_{near} = -dnear และ Z_{far} = -dfar

ค่า Default ของพารามิเตอร์ในฟังก์ชันใน Projection เชิงตั้งฉากคือ +-1 ซึ่งจะมีการสร้างปริมาตรการมองที่เป็นลูกบาศก์เชิงสมมาตรในระบบการมองแบบมือขวา ค่า Default ของฟังก์ชันจะเป็น

```
glOrtho (-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
```

ดังนั้นค่า Default ของวินโดว์ขริบภาพจะเป็นสี่เหลี่ยมเชิงสมมาตร และค่า Default ของปริมาตรการมองจะเป็นลูกบาศก์เชิงสมมาตรที่มีค่า Z_{near} = 1.0 (หลังตำแหน่งการมอง) และ Z_{far} = -1.0 รูป 2.15 แสดงภาพและตำแหน่งของปริมาตรการมองของ Projection เชิงตั้งฉากที่เป็นค่าDefault



รูปที่ 2. 15 ภาพและตำแหน่งของปริมาตรการมองของ Projection เชิงตั้งฉากที่เป็นค่า Default[22]

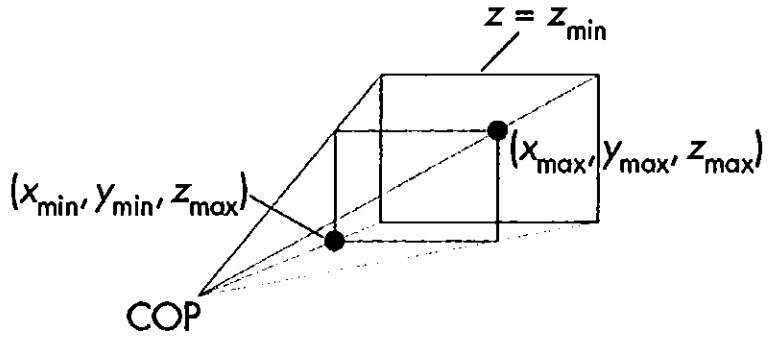
2.7.3.3 ฟังก์ชัน Projection แบบ Perspective เชิงสมมาตร

มีฟังก์ชัน 2 ฟังก์ชันที่ใช้ในการสร้างจอภาพการมองภาพแบบ Perspective ฟังก์ชันหนึ่งสร้างปริมาตรการมองกรวยรูปตัดเชิงสมมาตรที่เกี่ยวกับทิศทางการมอง (แกน $-Z_{vciw}$) ฟังก์ชันอื่นสามารถใช้ได้ทั้ง Projection แบบ Perspective แบบสมมาตร หรือ Projection แบบ Perspective เฉียง สำหรับทั้งสองฟังก์ชันจุดอ้างอิง Projection อยู่ที่จุดกำเนิดโดยอร์ดิเนตการมองและระนาบการขริบระยะใกล้หรือระนาบการมอง

ปริมาตรการมองกรวยรูปตัด Projection แบบ Perspective เชิงสมมาตรกำหนดได้ด้วยฟังก์ชัน GLU ดังนี้

```
gluPerspective (theta, aspect, dnear, dfar);
```

เมื่อค่าพารามิเตอร์ทั้ง 4 ค่ากำหนดเป็นเลขทศนิยม พารามิเตอร์ 2 ตัวแรกกำหนดเป็นขนาดและตำแหน่งของวินโดว์ขริบภาพบนระนาบระยะใกล้ ส่วนพารามิเตอร์อีก 2 ตัวกำหนดระยะทางจากจุดมอง (จุดกำเนิด) ดังแสดงในรูป 2.16 ไปยังระนาบการขริบระยะใกล้และระยะไกล พารามิเตอร์ theta แสดงถึงมุมการมองซึ่งเป็นมุมระหว่างระนาบการขริบด้านบนและด้านล่าง มุมนี้สามารถกำหนดได้ตั้งแต่ 0 องศา ถึง 180 องศา พารามิเตอร์ aspect เป็นค่าของอัตราส่วนแอสเป็กต์ (aspect ratio) ซึ่งเป็นค่า ความกว้าง/ความสูง ของวินโดว์ขริบภาพ



รูปที่ 2. 16 หลักการ Perspective เชิงสมมาตร [23]

สำหรับ Projection แบบ Perspective ใน OpenGL ระนาบการขริบระยะใกล้และระยะไกลต้องอยู่ที่ใดที่หนึ่งบนแกน $-Z_{view}$ เสมอและไม่สามารถอยู่หลังจุดการมองได้ ข้อจำกัดนี้ไม่ได้จำกัดเฉพาะโปรเจ็กต์ชันเชิงตั้งฉากเท่านั้น แต่ใช้ไม่ได้เมื่อเป็นโปรเจ็กต์ชันแบบเพอร์สเปกทีพของออปเจ็กต์ที่ระนาบการมองอยู่หลังจุดการมอง ดังนั้นทั้งค่า dnear และ dfar ต้องกำหนดเป็นค่าบวก และตำแหน่งของระนาบใกล้และไกลถูกคำนวณเป็น Z_{near} = -dnear และ Z_{far} = -dfar

หากไม่กำหนดฟังก์ชัน Projection จอภาพจะแสดงด้วยค่า Default ของ Projection เชิงตั้งฉาก ในกรณีนี้ ปริมาตรการมองก็คือรูปลูกบาศก์เชิงสมมาตร

2.7.4 การวาดรูปสี่เหลี่ยมโดยใช้คำสั่ง OpenGL

การวาดรูปสี่เหลี่ยมใน OpenGL โดยการใช้การเชื่อมต่อจุด 4 จุดเป็นรูปสี่เหลี่ยมปิดมีรูปแบบการวาดรูปสี่เหลี่ยม 2 แบบ คือ แบบรูปสี่เหลี่ยมแยกแต่ละรูป (GL_QUADS) และรูปสี่เหลี่ยมปิด (GL_QUAD_STRIP) ดังนี้

- GL_QAUADS สำหรับวาดรูปสี่เหลี่ยมโดยใช้จุด 4 จุด ถ้ามีจุดไม่ครบ 4 จุด จุดที่เหลือเศษจะถูกละทิ้งไป เช่น ถ้ามีจุด v0-v7 ฟังก์ชันนี้จะสร้างรูปสี่เหลี่ยมจาก v0-v1-v2-v3 และ v4-v5-v6-v7 เป็นต้น
- GL_QUAD_STRIP สำหรับการวาดรูปสี่เหลี่ยมเช่นเดียวกับฟังก์ชัน GL_QUADS แต่ฟังก์ชันนี้จะวาดรูปสี่เหลี่ยมต่อเนื่องโดยใช้ชุดของจุดที่เปลี่ยนไป เช่น ถ้ามีจุด v0-v7 จะวาดรูปสี่เหลี่ยม คือ v0-v1-v3-v2, v2-v3-v5-v4, v4-v5-v7-v6 เป็นต้น

2.7.5 การหมุนภาพ (Rotation)

การหมุน(Rotation) คือ การเปลี่ยนตำแหน่งของออปเจ็กต์ไปในแนวเส้นโค้งของวงกลมในระนาบ xy (ขนานกับโคออร์ดิเนตของแกน z) จากตำแหน่งหนึ่งไปอีกตำแหน่ง โดยไม่เปลี่ยนขนาดของออปเจ็กต์นั้น

สำหรับการหมุนออปเจ็กต์นั้น จะต้องกำหนดตัวแปร 2 ตัว คือ มุมที่ต้องการหมุน (θ) ซึ่งถ้าค่าเป็น + หมายถึงการหมุนทวนเข็มนาฬิกา และถ้าค่าเป็น – หมายถึงการหมุนตามเข็มนาฬิกา ส่วนตัวแปรที่สองคือ จุดศูนย์กลางการหมุน ตำแหน่ง (x_r, y_r) หลังจากที่หมุนออปเจ็กต์ไปแล้วระยะห่างระหว่างจุดหมุนกับออปเจ็กต์ยังคงมีค่าเท่าเดิม ลักษณะของออปเจ็กต์ยังคงเหมือนเดิม แต่จะวางในตำแหน่งที่ตำแหน่งที่ต่างไปจากเดิมอันเนื่องมาจากการหมุนนั่นเอง การหมุนสามารถหมุนได้ครั้งละหลายๆ ออปเจ็กต์ก็ได้ จุดหมุนจะอยู่ภายในหรือภายนอกออปเจ็กต์ก็ได้ และตำแหน่งของออปเจ็กต์ที่เกิดจากการหมุนอาจจะอยู่ในจอภาพ นอกจอภาพ หรืออยู่ในจอภาพบางส่วนก็ได้ ขึ้นอยู่กับขนาดของจอภาพที่ทำให้เกิดการขริบภาพ (clipping) แสดงถึงการหมุนออปเจ็กต์ที่จุดหมุน (x_r, y_r) ด้วยมุม θ

คำสั่ง OpenGL ที่ใช้ในการหมุนออปเจ็กต์ คือ

```
glRotatef (GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
```

โดยที่ angle คือ มุมที่ต้องการให้หมุน (เป็นองศา)

- x คือ กำหนดให้หมุนตามแนวแกน x (ต้องการให้หมุนใส่เป็นเลข 1)
- y คือ กำหนดให้หมุนตามแนวแกน y (ต้องการให้หมุนใส่เป็นเลข 1)
- z คือ กำหนดให้หมุนตามแนวแกน z (ต้องการให้หมุนใส่เป็นเลข 1)

2.8 บทสรุป

ในบทที่ 2 นี้จะกล่าวถึง หลักการและทฤษฎีทั้งหมดที่เกี่ยวข้องที่นำเอามาใช้สำหรับสร้างระบบ เริ่มตั้งแต่ กลศาสตร์ของอากาศยานปีกหมุน , แกนอ้างอิงสำหรับวัดมุมเอียง, ความหมายและความสำคัญของ GUI, บอร์ดไมโครคอนโทลเลอร์ที่เป็นส่วนหนึ่งของระบบ, โมดูลเซนเซอร์ที่นำเอามาตรวจจับวัดมุมเอียงในแนวแกนต่างๆ (x, y, z) และไลบรารีที่ OpenGL ซึ่งใช้ในการสร้างหน้าจอ GUI เพื่อแสดงผล ตามลำดับ

บทที่ 3
วิธีการดำเนินการ

ในบทนี้ผู้ดำเนินโครงงานจะกล่าวถึงรายละเอียดการดำเนินโครงงานโดยบ่งเป็น 3 ส่วน ดังนี้

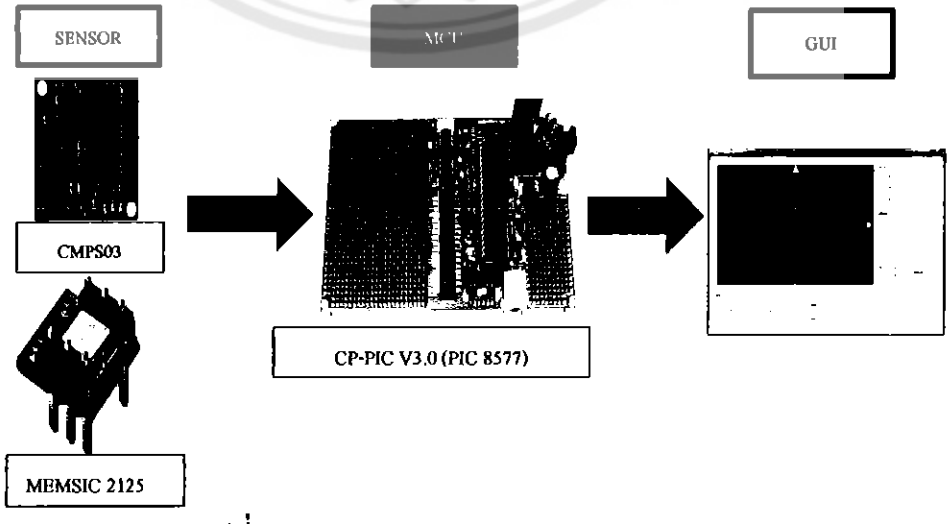
1. ภาพรวมของระบบ จะบอกถึงภาพรวมของระบบว่าประกอบด้วยอะไรบ้าง ใช้โมดูลอะไรในการสร้างระบบขึ้นมา แต่ละส่วนสัมพันธ์กันอย่างไร และระบบมีการทำงานอย่างไร

2. การเขียนโปรแกรมลงบนบอร์ดไมโครโปรเซสเซอร์เพื่อใช้ในการอ่านเซนเซอร์ที่ใช้สำหรับวัดมุมเอียง

3. การเขียนโปรแกรมสร้างหน้าจอ GUI สำหรับแสดงผล

3.1 ภาพรวมของระบบ (System Overview)

ในการวัดมุมเอียงใช้เซนเซอร์ 2 ตัว ได้แก่ CMPS03 เป็นเข็มทิศสำหรับวัดมุมเอียงในแนวแกน Z (Yaw) และ MEMSIC 2125 เป็นโมดูลวัดความเร่งแบบ 2 แกน ใช้สำหรับวัดมุมเอียงในแนวแกน X (Row) และ Y (Pitch) หลังจากนั้นจะส่งสัญญาณที่ได้ไปที่ CP-PIC V3.0 (PIC 16F877) ซึ่งเป็นไมโครโปรเซสเซอร์ทำหน้าที่ในการประมวลผลของสัญญาณที่ได้มาจากเซนเซอร์ทั้ง 2 ตัว จะได้ผลลัพธ์ออกมาเป็นมุมเอียงของทั้ง 3 แกน (X, Y และ Z) แล้วส่งผลลัพธ์ที่ได้นั้นไปแสดงออกทางหน้าจอคอมพิวเตอร์ส่วนบุคคล (PC) เป็นรูปกล่องสี่เหลี่ยมสามมิติโดยแสดงตามมุมเอียงที่วัดค่าได้จากเซนเซอร์ ดังแสดงในรูปที่ 3.1



รูปที่ 3. 1 ภาพรวมของระบบ (System Overview)

3.2 การเขียนโปรแกรมบนบอร์ดไมโครโปรเซสเซอร์เพื่อใช้ในการอ่านค่าจากเซนเซอร์ที่ใช้สำหรับวัดมุมเอียงทั้ง 2 ตัว

3.2.1 การใช้โมดูล Timer ของ PIC 16F877 ในการจับเวลา

หลักการในการประมวลผลสัญญาณจากเซนเซอร์ทั้ง 2 ตัว นั้นมีหลักการเดียวกัน คือ ต้องใช้ Timer เป็นตัวจับเวลาว่าคาบเวลาของสัญญาณพัลส์ที่มี logic เป็น "1" ผู้ดำเนินโครงงานขอกล่าวถึงวิธีการใช้ Timer และการเอาค่าที่ได้จาก Timer มาแทนค่าลงในสมการเพื่อคำนวณค่ามุมของเซนเซอร์แต่ละตัวตามลำดับ

Timer เป็นวงจรนับที่มีอยู่ภายในชิพ PIC จะทำการนับสัญญาณพัลส์ ซึ่งเป็นสัญญาณที่ได้จากสัญญาณนาฬิกาจากเซนเซอร์ทั้ง 2 ตัว คือ Memsic2125 และ CMPS03 ที่ป้อนให้กับ PIC โดยผู้ดำเนินโครงงานเลือกใช้ Timer 1 เป็นตัววัดคาบซึ่ง Timer1 เป็น Timer ขนาด 16 บิต ประกอบด้วยรีจิสเตอร์ TMR1L ขนาด 8 บิต และ TMR1H ขนาด 8 บิต ซึ่งรูปที่ 3.2 แสดงถึงรายละเอียดของรีจิสเตอร์ T1CON และตารางที่ 3.1 แสดงรายละเอียดของรีจิสเตอร์ T1CON

T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
bit 7							bit 0

รูปที่ 3. 2 รายละเอียดของรีจิสเตอร์ T1CON[11]

ตารางที่ 3. 1 รายละเอียดของรีจิสเตอร์ T1CON[11]

บิตที่	ชื่อ	คำอธิบาย
7	-	ให้เป็นค่า '0'
6	-	ให้เป็นค่า '0'
5	T1CKPS1	ใช้เลือกอัตราส่วนในการหารความถี่ของ Prescaler ตามตารางที่ 3.2
4	T1CKPS0	ใช้เลือกอัตราส่วนในการหารความถี่ของ Prescaler ตามตารางที่ 3.2
3	T1OSCEN	ใช้เปิด/ปิดการทำงานของออสซิลเลเตอร์ของ Timer1 - 1 = ปิดการใช้ Oscillator ของ Timer 1 - 0 = เปิดการใช้ Oscillator ของ Timer 1

บิตที่	ชื่อ	คำอธิบาย
2	T1SYNC	ใช้เปิด/ปิดการ Synchronize ของสัญญาณพัลส์อินพุตจากภายนอก - 1 = ปิดการ Synchronize - 0 = เปิดการ Synchronize
1	TMR1CS	ใช้เลือกแหล่งจ่ายสัญญาณพัลส์อินพุตให้กับ Timer 1 - 1=ใช้เลือกแหล่งจ่ายสัญญาณพัลส์อินพุตขอบขาขึ้นจากภายนอกผ่าขา T1CKI/T1OSO - 0 = ใช้เลือกแหล่งจ่ายสัญญาณพัลส์พุตจากคริสตอล
0	TMR1ON	ใช้เปิด/ปิดการทำงานของ Timer 1 - 1 = เปิดการทำงานของ Timer1 ให้ Timer1 เริ่มนับ - 0 = ปิดการทำงานของ Timer1 ให้ Timer1 หยุดนับ

ตารางที่ 3. 2 รายละเอียดอัตราส่วนในการหารความถี่ของ Prescaler[11]

T1CKPS1	T1CKPS1	อัตราส่วนการหารความถี่
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

การคำนวณค่าในการนับขึ้นทุก 1 ครั้งของ Timer สามารถคำนวนได้จากสมการที่ 3.1 นี้

$$\text{ความถี่ในการนับขึ้น 1 ครั้ง} = \left(\frac{OSC}{4}\right) * \text{อัตราส่วนหารความถี่ } Prescale \qquad \textbf{สมการที่ 3. 1}$$

เมื่อ OSC คือความถี่ของสัญญาณนาฬิกาคริสตอล และ 1 Machine Cycle ของ PIC นั้นมีค่าเท่ากับ 4

หากต้องการคำนวณหาคาบเวลาในการนับขึ้น 1 ครั้งของ Timer สามารถหาได้จากสมการที่ 3.2

$$F = \frac{1}{T} \qquad \textbf{สมการที่ 3. 2}$$

ดังนั้นคาบเวลาในการนับขึ้นทุกๆ 1 ครั้งของ Timer มีค่าเท่า

$$\text{คาบในการที่ใช้ในการนับขึ้น 1 ครั้ง} = \frac{1}{\text{ความถี่ที่ใช้ในการนับขึ้น 1 ครั้ง}} \qquad \text{สมการที่ 3. 3}$$

แทนค่าสมการที่ 3.1 ลงในสมการที่ 3.3 จะได้

$$\text{คาบในการที่ใช้ในการนับขึ้น 1 ครั้ง} = \frac{4}{OCS * \text{อัตราส่วนหารความถี่ } Prescale} \qquad \text{สมการที่ 3. 4}$$

3.2.2 หลักการประมวลผลสัญญาณที่ได้จากโมดูล วัดความเร่ง 2 แกน MEMSIC 2125

จากสมการที่ 2.1 สมการหาค่าความเร่งของโมดูลวัดความเร่ง 2 แกน คือ

$$a(g) = \frac{(\frac{T1}{T2} - 0.5)}{0.125}$$

โดยที่ค่า T1 เราสามารถคำนวณได้จากสมการที่ 3.4 โดยที่ความถี่ของสัญญาณนาฬิกาที่ใช้นั้นมีค่าเท่ากับ 10 MHz ดังนั้นค่า T1 จะเท่ากับ

$$T1 = T1count * \left(\frac{4}{10M * Prescale}\right) = \frac{T1Count * 0.4\ \mu s}{Prescale} \qquad \text{สมการที่ 3. 5}$$

โดยที่ค่า Prescale ที่เอามาหารนั้นต้องเป็นอัตราส่วน เช่น Prescale = 1: 4 ต้องหารด้วย 1/4หรือคูณด้วย 4 และ กำหนดให้ T1Count คือจำนวนครั้งที่นับได้ทั้งหมดของ Timer1 ในช่วงที่สัญญาณพัลส์มีโลจิก "1"

ดังนั้นสรุปสมการที่ใช้สำหรับคำนวณค่า a_x และ a_y สำหรับโมดูลสำหรับวัดความเร่ง 2 แกนได้คือ

$$a(g) = \frac{\left(\frac{T1}{10} - 500\right) * 8}{1000} \qquad \text{สมการที่ 3. 6}$$

โดยที่ค่า T1 ต้องอยู่ในหน่วยไมโครวินาที (μs)

ดังนั้นจะสามารถหามุมเอียงตามแนวแกน x และ y ได้จาก

$$\text{angleX or Y(degree)} = \arcsin(a) \qquad \text{สมการที่ 3. 7}$$

สำหรับค่าที่ออกมานั้นจะได้มุมในหน่วยขององศาเรเดียน (radian) ถ้าต้องการให้ค่ามุมอยู่ในหน่วยองศาดีกรี (degree) สามารถทำได้โดย

$$\text{angleX(degree)} = \frac{(\arcsin(a) * 180)}{\text{pi}} \qquad \textbf{สมการที่ 3. 8}$$

โดยที่ค่า pi = 3.14159265358979323846264338327950288419716939937 51[24]

3.2.3 หลักการประมวลผลสัญญาณที่ได้จากโมดูลเข็มทิศดิจิตอล CMPS03

จากหัวข้อที่ได้กล่าวไว้แล้วใน 2.6.4 การอ่านค่าทิศทางจากเอาต์พุตสัญญาณพัลส์ว่าการอ่านค่าทิศทางจากเอาต์พุตสัญญาณพัลส์ เป็นการนำค่าความกว้างพัลส์ที่ได้มาระบุตำแหน่งองศา จาก 0 ถึง 359.9 องศา โดยมีย่านความกว้างของสัญญาณพัลส์ที่เป็นลอจิก "1" จาก 1 มิลลิวินาทีไปจนถึง 36.99 มิลลิวินาที มีความละเอียด 0.1 มิลลิวินาทีต่อองศา โดยที่ระหว่างพัลส์นั้นจะมีช่วงลอจิก "0" กว้าง 65 มิลลิวินาที เราสามารถแปลงมาเป็นสมการได้คือ

$$\text{angle\_Z} = \frac{(T - 1000)}{100} \qquad \textbf{สมการที่ 3. 9}$$

โดยที่ค่า T ต้องอยู่ในหน่วยไมโครวินาที (µs)
ซึ่งค่าที่ได้ออกมาจะมีค่าตั้งแต่ 0-359.9 องศา

3.2.4 Pseudo code ของการควบคุมไมโครคอนโทลเลอร์

ในส่วนของหัวข้อนี้จะขอแยกกล่าวถึงเป็น 3 ส่วนคือ

3.2.4.1 Pseudo code ในการอ่านค่าสัญญาณพัลส์ที่มี logic = 1 จากโมดูล MEMSIC 2125

3.2.4.2 Pseudo code ในการอ่านคาบสัญญาณพัลส์ที่มี logic = 1 จากโมดูล CMPS03

3.2.4.3 Pseudo code ของการควบคุมไมโครคอนโทลเลอร์โดยรวม

3.2.4.1 Pseudo code ในการอ่านค่าสัญญาณพัลส์ที่มี logic = 1 จากโมดูล MEMSIC 2125

รับค่าแกน x เข้า PORTB1 และรับค่าแกน Y เข้า PORTB2 โดยที่ฟังก์ชัน get_pulse_x() นั้นจะ return คาบเวลาที่เป็นโลจิก 1 ของแกน x และ get_pulse_y() นั้นจะ return คาบเวลาที่เป็นโลจิก 1 ของแกน y ตามลำดับ

แสดง Pseudo code ในการอ่านค่าคาบสัญญาณพัลส์ที่มี logic = 1 ของแกน X

```
01: get_pulse_x()
02:     SET TMR1H = 0x00 and TMR1L = 0x00 // กำหนดค่าเริ่มต้นให้เป็น 0
03:     SET T1CON = 0x00  //ยังไม่ให้ Timer เริ่มนับ กำหนด Prescale เป็น 1:1
04:     if  PORTB1 ==1
05:             while(PORTB1 == 1)  // วนลูปรอจนกว่า PORTB1 จะเป็น 0
06:     Start Timer 1 // เมื่อ PORTB1 เป็น 0 สั่งให้ Timer 1 เริ่มนับ
07:     while(PORTB1 ==0)  //วนลูปรอจนกว่า PORTB1 จะเป็น 1
08:     tOld =(256*TMR1H)+TMR1L // ให้ tOld เก็บค่าแรกเมื่อพัลส์มี logic = 1 และแปลงค่าเป็นฐาน10
09:     while(PORTB1 == 1) // วนลูปรอจนกว่า PORTB1 จะเป็น 0
10:     tNew =(256*TMR1H)+TMR1L // ให้ tNew เก็บค่าสุดท้ายก่อนที่พัลส์จะมี logic = 0 และแปลงค่าเป็นฐาน
11:     return tNew-tOld  // return ค่า tNew - tOld ซึ่งค่าคาบสัญญาณที่เป็น logic 1 ของสัญญาณพัลส์
```

แสดง Pseudo code ในการอ่านค่าคาบสัญญาณพัลส์ที่มี logic = 1 ของแกน Y

```
01: get_pulse_y()
02:     SET TMR1H = 0x00 and TMR1L = 0x00 // กำหนดค่าเริ่มต้นให้เป็น 0
03:     SET T1CON = 0x00  //ยังไม่ให้ Timer เริ่มนับ กำหนด Prescale เป็น 1:1
04:     if  PORTB2 ==1
05:             while(PORTB2 == 1)  // วนลูปรอจนกว่า PORTB2 จะเป็น 0
06:     Start Timer 1 // เมื่อ PORTB1 เป็น 0 สั่งให้ Timer 1 เริ่มนับ
07:     while(PORTB2 ==0)  //วนลูปรอจนกว่า PORTB2 จะเป็น 1
08:     tOld =(256*TMR1H)+TMR1L // ให้ tOld เก็บค่าแรกเมื่อพัลส์มี logic = 1 และแปลงค่าเป็นฐาน10
09:     while(PORTB2 == 1) // วนลูปรอจนกว่า PORTB2 จะเป็น 0
10:     tNew =(256*TMR1H)+TMR1L // ให้ tNew เก็บค่าสุดท้ายก่อนที่พัลส์จะมี logic = 0 และแปลงค่าเป็นฐาน
11:     return tNew-tOld  // return ค่า tNew - tOld ซึ่งค่าคาบสัญญาณที่เป็น logic 1 ของสัญญาณพัลส์
```

3.2.4.2 Pseudo code ในการอ่านคาบสัญญาณพัลส์ที่มี logic = 1 จากโมดูล CMPS03

ในส่วนของการอ่านค่าจากเซนเซอร์โมดูล CMPS03 จะใช้หลักการ Capture และ Interrupt

ซึ่งใน Mikro C compiler การโปรแกรมโดยวิธี Interrupt นั้นต้องโปรแกรมอยู่ในฟังก์ชัน interrupt() และการ Capture ต้องต่อสัญญาณเข้าขา CCP1 ซึ่งตรงกับ PORTC2

แสดง Pseudo code ในการอ่านค่าคาบสัญญาณพัลส์ที่มี logic = 1 ของแกน Z

```
01: interrupt()
02:     SET TMR1H = 0x00 and TMR1L = 0x00 // กำหนดค่าเริ่มต้นให้เป็น 0
03:     SET T1CON = 0x20  //ยังไม่ให้ Timer เริ่มนับ กำหนด Prescale เป็น 1:4
04:     SET tz = 0 // set tz ให้มีค่าเริ่มต้นเป็น 0
05:         if(CCP1F == 1) //เมื่อเกิดการ interrupt ขึ้น บิต CCP1F จะ set เป็น 1
06:             SET CAPTURE = 1 // ตัวแปรเพื่อบอกว่ากำลัง capture อยู่
07:             Start Timer 1 // สั่งให้ timer 1 เริ่มนับ
08:             while(PORTC2 == 1)  //วนลูปรอจนกว่า PORTC2 จะเป็น 0
09:             tz =(256*TMR1H)+TMR1L // เก็บค่า tz เมื่อ PORTC2 เป็น 0
10:     CLEAR CCP1F = 0   // clear bit flag
11:     Stop Timer 1 // ให้ timer หยุดนับ
12:     Return tz //return ค่า tz ซึ่งเป็นคาบของสัญญาณพัลส์ที่มี logic = 1
13: END
```

3.2.4.3 Pseudo code ของการควบคุมไมโครคอนโทลเลอร์ทั้งหมด

```
01: Main()
02:     SET PORTB and PORTC are INPUT //กำหนดให้ PORTB และ PORTC เป็น input
03:     SET CCP1 = 0x05 //กำหนดให้ทำการ capture ทุกขอบขาขึ้น
04:     SET CCP1IE =1  // เปิดการ interrupt สำหรับการ capture
05:     CLEAT CCP1F =0 //clear bit flag
06:     while(true) // while loop forever
07:         if(CAPTURE == 1) // เมื่อเกิดการ capture ขึ้น
08:             CLEAR CAPTURE = 0  // เพื่อพร้อมใช้งานในรอบถัดไปหลังจากประมวลผลเสร็จ
09:             CLEAR CCP1IE //ปิดการ interrupt ในขณะที่กำลังประมวลผลอยู่
10:             tz = tz * 1.6 // ใช้สมการที่ 3.5 ในการคำนวณ
11:             angleZ = (tz-1000)/100 //ใช้สมการที่ 3.8 ในการคำนวณ
12:             tx = get_pulse_x() // อ่านค่าจาก MEMSIC2125 จากหัวข้อ 3.2.3.1
```

13: ty = get_pulse_y() // อ่านค่าจาก MEMSIC2125 จากหัวข้อ 3.2.3.2

14: SENT angleZ, tx, tz TO PC VIA RS232 // ส่งทั้ง 3 ค่าไปยัง PC ผ่าน RS232

15: CLEAR CCP1F

16: SET CCP1E // เปิดการ interrupt

17: END

3.3 การเขียนโปรแกรมสร้างหน้าจอ GUI สำหรับแสดงผล

ในขั้นตอนการดำเนินงานส่วนนี้ผู้ดำเนินโครงงานจะกล่าวถึงไลบรารีที่นำมาใช้สร้างโปรแกรม สำหรับการแสดงผล

3.3.1 ไลบรารีที่ใช้ในการโปรแกรม

เนื่องจาก OpenGL นั้นไม่สนับสนุนภาษา C# ซึ่งเป็นภาษาที่ผู้ดำเนินโครงงานเลือกมาเขียนโปรแกรมสำหรับการแสดงผล แต่มีกลุ่มๆหนึ่งชื่อว่า TaoMONO[25] ได้สร้างส่วนขยายขึ้นมาเป็นอีกไลบรารีหนึ่งเพื่อที่สามารถนำ OpenGL มาใช้บนภาษา C# ได้ ไลบรารีนั้นมีชื่อมา TaoOpenGL[25] ซึ่งเป็นไลบรารีที่ทางผู้ดำเนินโครงงานเลือกเอามาใช้งานในการเขียนโปรแกรมสำหรับการแสดงผล

3.3.2 การกำหนดค่าการมองภาพของโปรแกรม

ใน OpenGL นั้นจะมีคำสั่ง gluLookAt (x0, y0, z0, xref, yref, zref, Vx, Vy, Vz) เพื่อใช้ในการกำหนดมุมมองการมองภาพ 3 มิติที่เราสร้างขึ้น ซึ่งทางผู้ดำเนินโครงงานได้กำหนดค่าดังต่อไปนี้

```
Glu.gluLookAt(0,1,0, 0, 0, -5, 0, 1, 0);
```

สำหรับรายละเอียดของพารามิเตอร์แต่ละตัวเป็นดังนี้คือ

- พารามิเตอร์ 3 ตัวแรก คือ 0,1,0 คือการกำหนดกล้องว่าจะวางอยู่ตำแหน่งไหน ซึ่งทางผู้ดำเนินโครงงานจะให้กล้องวางที่ตำแหน่ง x =0 , y =1, z=0 โดยที่การที่เรากำหนดแกน y = 1 นั้น เพื่อที่จะให้ได้ภาพมุมสูงจะได้มองเห็นวัตถุในมุมมองที่ดีขึ้น ดังแสดงในรูปที่ 3.3 ซึ่งเป็นรูปเปรียบเทียบค่าเมื่อตั้งค่า y = 0 กับ ตั้งค่า y = 1

- พารามิเตอร์ 3 ตัวถัดมาคือ 0, 0, -5 คือตำแหน่งของวัตถุที่ทำการมอง เหตุผลที่กำหนด z = -5 เนื่องจากทางผู้ดำเนินโครงงานได้ย้ายวัตถุให้ไปอยู่ในตำแหน่ง -5 หลังจากที่สร้างภาพเสร็จแล้ว
- พารามิเตอร์ 3 ตัวสุดท้าย คือ 0,1,0 คือค่า view up vector ซึ่งทางผู้ดำเนินโครงงานเองต้องการให้การวางกล้องสำหรับถ่ายภาพนั้นอยู่ในมุมมองที่ปกติ คือตามแกน Y ทางผู้ดำเนินโครงงานจึงได้ตั้งค่าพารามิเตอร์เป็นค่าดังกล่าว

รูปที่ 3. 3 เปรียบเทียบการตั้งค่า y = 0 (ภาพซ้ายมือ) และ y = 1 (ภาพขวามือ)

3.3.3 การกำหนดค่าฟังก์ชัน Projection แบบ Perspective เชิงสมมาตร

ใน OpenGL จะใช้คำสั่ง gluPerspective (theta, aspect, dnear, dfar) เพื่อใช้ในการกำหนดค่าต่างๆในการ Projection แบบ Perspective ซึ่งผู้จัดทำได้กำหนดค่าดังนี้

```
Glu.gluPerspective(45.0f, (double)width / (double)height, 0.01f, 5000.0f);
```

ซึ่งมีรายละเอียดของพารามิเตอร์ต่างๆเป็นดังนี้

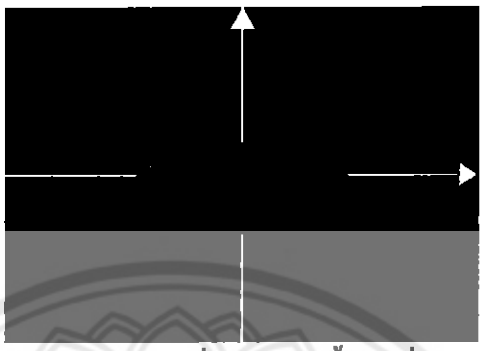
- 45.0f คือมุมกว้างระระหว่างระนาบการขริบด้านบนและด้านล่างซึ่งเราจะกำหนดค่าไว้ที่ 45 องศา
- ค่า ratio ผู้จัดทำจะกำหนดเป็นอัตราส่วนระหว่างความกว้างและความยาวของจอแสดงผล
- 0.01f คือระยะของระนาบใกล้
- 5000.f คือระยะของระนาบไกล ดังนั้นวัตถุที่เราต้องการ Projection แบบ Perspective ต้องอยู่ในช่วงนี้เท่านั้นถึงจะสามารถเห็นภาพได้

3.3.4 การวาดรูปภาพลูกบาศก์แบบสี่เหลี่ยมผืนผ้า

เนื่องจากทางผู้ดำเนินโครงงานต้องการวัตถุที่เป็นรูปกล่องเป็นรูปในลักษณะลูกบาศก์แบบสี่เหลี่ยมผืนผ้าแต่ใน OpenGL นั้นไม่มีคำสั่งดังกล่าว แต่มีเพียงคำสั่งสำหรับวาดรูปสี่เหลี่ยมเท่านั้นทางผู้จัดทำจึงสร้างรูปภาพวัตถุดังกล่าวโดยการสร้างจากการสร้างรูปสี่เหลี่ยมผืนผ้า 6 รูปมาต่อกัน โดยใช้คำสั่งต่อไปนี้

```
Gl.glBegin(Gl.GL_QUADS);

        //Poster
        Gl.glColor3ub(255, 0, 255);
        Gl.glVertex3d(1, 0.25, -1.5);
        Gl.glVertex3d(1, -0.25, -1.5);
        Gl.glVertex3d(-1, -0.25, -1.5);
        Gl.glVertex3d(-1, 0.25, -1.5);

        //Bottom
        Gl.glColor3ub(0, 255, 255);
        Gl.glVertex3d(-1, -0.25, -1.5);
        Gl.glVertex3d(1, -0.25, -1.5);
        Gl.glVertex3d(1, -0.25, 1.5);
        Gl.glVertex3d(-1, -0.25, 1.5);

        //Left Side
        Gl.glColor3ub(255, 255, 0);
        Gl.glVertex3d(-1, 0.25, -1.5);
        Gl.glVertex3d(-1, -0.25, -1.5);
        Gl.glVertex3d(-1, -0.25, 1.5);
        Gl.glVertex3d(-1, 0.25, 1.5);

        //Right Side
        Gl.glColor3ub(0, 0, 255);
        Gl.glVertex3d(1, 0.25, 1.5);
        Gl.glVertex3d(1, -0.25, 1.5);
        Gl.glVertex3d(1, -0.25, -1.5);
        Gl.glVertex3d(1, 0.25, -1.5);

        ///Top
        Gl.glColor3ub(0, 255, 0);
        Gl.glVertex3d(-1, 0.25, -1.5);
        Gl.glVertex3d(-1, 0.25, 1.5);
        Gl.glVertex3d(1, 0.25, 1.5);
        Gl.glVertex3d(1, 0.25, -1.5);

        // Anterior
        Gl.glColor3ub(255, 0, 0);
        Gl.glVertex3d(-1, 0.25, 1.5);
        Gl.glVertex3d(-1, -0.25, 1.5);
        Gl.glVertex3d(1, -0.25, 1.5);
        Gl.glVertex3d(1, 0.25, 1.5);

        Gl.glEnd();
```

ซึ่งก่อนที่จะสร้างรูป สี่เหลี่ยมแต่ละรูปนั้นต้องมีคำสั่ง Gl.glColor3ub(...,...,...) เพื่อที่จะให้รูปสี่เหลี่ยมที่ทางผู้ดำเนินโครงงานวาดแต่ละด้านนั้นมีสีคนละสีกัน เพื่อความชัดเจนในการแสดงผลดังแสดงในรูปที่ 3.4



รูปที่ 3. 4 รูปลูกบาศก์ที่แต่ละด้านนั้นมีสีที่แตกต่างกัน

3.3.5 การหมุนรูปลูกบาศก์แบบสี่เหลี่ยมผืนผ้าที่สร้างขึ้นตามค่าที่ได้จากเซนเซอร์

หลังจากที่ทางผู้ดำเนินโครงงานสร้างภาพเสร็จแล้วนั้น รูปภาพดังกล่าวต้องหมุนตามค่ามุมตามที่อ่านได้จากเซนเซอร์ทั้ง 2 ตัว (MEMSIC 2125 และ CMPS03) เพื่อที่จะแสดงภาพจำลองว่ามุมในแต่ละแกนของเซนเซอร์ทั้ง 2 ตัวนั้นทำมุมกี่องศา กล่าวคือรูปลูกบาศก์ที่สร้างขึ้นก็คือรูปภาพจำลองของอุปกรณ์ที่พัฒนาขึ้นนั่นเอง เมื่อระบบเอียงทำมุมในแกนใด ภาพจำลองจะสามารถแสดงมุมเอียงตามการเอียงของระบบได้อย่างถูกต้อง

ซึ่งใน OpenGL จะมีคำสั่งต่อไปนี้ในการหมุนภาพ

glRotatef (GLfloat angle, GLfloat x, GLfloat y, GLfloat z);

โดยที่รายละเอียดของพารามิเตอร์ต่างๆ ได้กล่าวไว้ในหัวข้อ 2.7.5

โดยในโปรแกรมนั้นใช้คำสั่งต่อไปนี้

```
Gl.glRotated(angle_y, 1, 0, 0);
Gl.glRotated(angle_z, 0, 1, 0);
Gl.glRotated(angle_x, 0, 0, 1);
```

สำหรับรายละเอียดพารามิเตอร์ เป็นดังนี้

angle_x = ขนาดของมุมเอียงตามแนวแกน X (Roll)

angle_y = ขนาดของมุมเอียงตามแนวแกน Y (Pitch)

angle_z = ขนาดมุมเอียงตามแนวแกน Z (Yaw)

การแมพค่านั้นเราต้องแมพค่ากับแกนอ้างอิงเดียวกัน ยกตัวอย่างเช่น แกน X ใน OpenGL คือแกนที่เป็นแกนแนว horizontal (Pitch) ของหน้าจอโดยจะตรงกับแกน Y ของระบบซึ่งคือแกนที่เป็นแกนแนว horizontal (Pitch) ของระบบ หรือ แกน Y ของ OpenGL คือแกนในแนว vertical (Yaw) ของหน้าจอโดยจะตรงกับแกน Z ของระบบซึ่งคือแกนที่เป็นแกนแนว vertical (Yaw) ของระบบ ดังที่กล่าวไว้แล้วในหัวข้อที่ 2.2

3.3.6 โปรโตคอลในการรับส่งข้อมูลระหว่างบอร์ดไมโคลคอนไมโครคอนโทลเลอร์กับ PC

โปรโตคอลในการรับส่งค่าระหว่างบอร์ดไมโครคอนโทลเลอร์กับ PC จะเป็นดังนี้

'S' (1byte)	X_ON (6bytes)	'N' (1 byte)	Angle_Z (6 bytes)	'N' (1 byte)	Y_ON (6 bytes)	'\n'

สำหรับรายละเอียดของโปรโตคอลเป็นดังนี้

- 'S' มีขนาด 1 byte ใช้แสดงถึงการเริ่มต้นส่งชุดของข้อมูล
- X_ON มีขนาด 6 bytes ส่งค่าคาบของพัลส์ที่มีโลจิกเป็น 1 ในหน่วยไมโครวินาที ของแกน X เพื่อให้ PC ประมวลผลหาค่าออกมาเป็นมุมต่อไป
- Angle_Z มีขนาด 6 bytes คือค่ามุมที่อ่านได้จากเซนเซอร์ CMPS03 ในแนวแกน Z
- Y_ON มีขนาด 6 bytes ส่งค่าคาบของพัลส์ที่มีโลจิกเป็น 1 ในหน่วยไมโครวินาที ของแกน Y เพื่อให้ PC ประมวลผลหาค่าออกมาเป็นมุมต่อไป
- 'N' มีขนาด 1 byte ใช้สำหรับแยกระหว่างค่าในแต่ละแกน
- '\n' ขึ้นบรรทัดใหม่ในข้อมูลแต่ละชุด

ตัวอย่างชุดของการส่งข้อมูลจากบอร์ดไมโครคอนโทลเลอร์ไปยัง PC ดังแสดงในรูปที่ 3.5

```
S012470N000036N012550<LF>
S012458N000036N012546<LF>
S012474N000036N012554<LF>
S012466N000036N012538<LF>
S012462N000036N012550<LF>
S012466N000036N012530<LF>
S012454N000036N012566<LF>
S012466N000036N012526<LF>
S012454N000036N012578<LF>
```

รูปที่ 3. 5 โปรโตคอลสำหรับส่งข้อมูลจากบอร์ดไมโครคอนโทลเลอร์ไปยัง PC

หมายเหตุ สาเหตุที่ส่งเป็น bytes เนื่องจากส่งข้อมูลเป็นชุดของ String เพื่อความง่ายในการโปรแกรม

3.3.7 Pseudo code ของการเขียนโปรแกรมหน้าจอ GUI สำหรับแสดงผล

ในหัวข้อนี้จะกล่าวแยกออกเป็น 3 ส่วนคือ

3.3.7.1 Pseudo code สำหรับการรับค่าจากบอร์ดไมโครคอนโทลเลอร์มาประมวลผล

3.3.7.2 Pseudo code สำหรับการตั้งค่าเริ่มต้นให้กับ OpenGL

3.3.7.3 Pseudo code สำหรับนำผลลัพธ์ที่ได้จากข้อที่ 3.3.7.1 มาแสดงผลในหน้าจอ GUI

3.3.7.7 Pseudo code สำหรับการรับค่าจากบอร์ดไมโครคอนโทลเลอร์มาประมวลผล

การรับค่าจาก serial port ในภาษา C# นั้นจะต้องอยู่ในรูปแบบของ object และ event กล่าวคือโปรแกรมจะทำงานเองอัตโนมัติเมื่อมีค่าเข้ามายัง serial port

```
01: serialPort1_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
// การรับค่าในรูปแบบ object และ event
02:     x = serialPort.readLine() // ให้ x คือชุดของข้อมูลที่อ่านได้จาก Serial Port
03:     if(x.Length == 21 and x[0] == 'S') //เช็คว่าชุดข้อมูลครบหรือไม่ และ byte เริ่มต้นต้องเป็น 'S'
04:         SET ON_X = x.substring(1,6) //ให้ค่า ON_X คือ byte ที่ 1 ถึง byte ที่ 6
05:         SET angle_Z = x.substring(8,6) //ให้ค่า ON_X คือ byte ที่8 ถึง byte ที่ 13
06:         SET ON_Y = x.substring(15,6) //ให้ค่า ON_X คือ byte ที่ 15 ถึง byte ที่ 20
07:         SET accX = (((ON_X/10)-500)*8)/1000 // หาความแร่งในแนวแกน X ดังสมการที่3.6
08:         SET accY =(((ON_Y/10)-500)*8)/1000 // หาความแร่งในแนวแกน Y ดังสมการที่3.6
09:         SET angle_X = (Arcsin(accX)*180)/pi // หาค่ามุมเอียงในแนวแกน X ดังสมการที่ 3.8
10:         SET angle_Y = (Arcsin(accY)*180)/pi // หาค่ามุมเอียงในแนวแกน Y ดังสมการที่ 3.8
11:     END
```

3.3.7.2 Pseudo code สำหรับการตั้งค่าเริ่มต้นให้กับ OpenGL

ใช้สำหรับกำค่าเริ่มต้นต่างๆให้กับ OpenGL ตามที่ผู้ดำเนินโครงงานต้องการก่อนการแสดงผล

```
01: init()
02:     SET width = width of frame // ให้ width เท่ากับความกว้างของหน้าจอแสดงผล
03:     SET height = height of frame //ให้ height เท่ากับความสูงของหน้าจอแสดงผล
```

```
04:     Gl.glViewport(0,0,width,heigth) //ให้หน้าจอสำหรับการวาดมีขนาดกว้าง = width และสูงเท่ากับ height
05:     Gl.glMatrixMode(GL_PROJECTION) //กำหนดโหมดเป็นโหมด projection
06:     Gl.glLoadIdentity() //โหลดค่า identity matrix เริ่มต้น
07:     Glu.gluPerspective(45.0f, width / height, 0.01f, 5000.0f) // กำหนด parameter ดังที่กล่าวไว้ในหัวข้อ
3.3.3
08:     Glu.gluLookAt(0,1,0,0,-5,0,0,1,0) // กำหนด parameter ต่างๆดังที่กล่าวไว้ในหัวข้อ 3.3.2
09:     Gl.glClearColor(0,0,0,0) //กำหนดให้พื้นหลังของหน้าจอเป็นสีดำ
10:     END
```

3.3.7.3 Pseudo code สำหรับนำผลลัพธ์ที่ได้จากข้อที่ 3.3.7.1 มาแสดงผลในหน้าจอ GUI

ในการแสดงผลนั้นทางผู้จัดทำจะให้การวาดรูปนั้นอยู่ในฟังก์ชัน Paint ซึ่งจะอยู่ในรูปของ object และ event คือเมื่อมีเหตุการณ์ให้เกิดการวาดรูป จะเกิดการวาดรูปลงบนหน้าจอแสดงผลอัตโนมัติ

```
01: Paint(object sender, PaintEventArgs e)
02:     Gl.glTranslated(0, 0, -5) //ย้ายตำแหน่งรูปภาพไปยังตำแหน่งแกน z(ความลึก)ที่ค่า -5
03:     // วาดเส้นตรงสีขาวเพื่อเป็นแกนอ้างอิง x,y
04:     Gl.glColor3ub(255, 255, 255) //กำหนดสีเส้นให้เป็นสีขาว
05:     Gl.glLineWidth(3) //กำหนดขนาดของเส้นเท่ากับ 3
06:     Gl.glBegin(Gl.GL_LINES) //วาดเส้นตรงอ้างอิงสำหรับแกน x
07:             Gl.Vertex3d(-3, 0, 0)
08:             Gl.Vertex3d(2.75, 0, 0)
09:     Gl.glEnd()
10:     Gl.glBegin(Gl.GL_LINES) //วาดเส้นตรงอ้างอิงสำหรับแกน Y
11:             Gl.Vertex3d(0, -3, 0)
12:             Gl.Vertex3d(0, 2.75, 0);
13:     GL.End();
14: //วาดหัวสามเหลี่ยมสีขาวสำหรับแกนอ้างอิง x,y
15:     GL.Begin(GL_TRIANGLES);
16:             Gl.Vertex2d(-0.15, 1.75);
```

```
17:             Gl.Vertex2d(0.15, 1.75);
18:             Gl.Vertex2d(0, 2);
19:     Gl.End();
20:     Gl.Begin(GL_TRIANGLES);
21:             Gl.Vertex2d(2.75, 0.15);
22:             Gl.Vertex2d(2.77, -0.15);
23:             Gl.Vertex2d(3, 0);
24:     Gl.End();
25: //หมุนรูปตามค่าในหัวข้อ 3.3.7.1
26:     Gl.glRotated(angle_Y, 1, 0, 0);
27:     Gl.glRotated(angle_Z, 0, 1, 0);
28:     Gl.glRotated(angle_X, 0, 0, 1);
29: //วาดรูปลูกบาศก์สี่เหลี่ยมผืนผ้าสำหรับแสดงผล
30:     Gl.glBegin(Gl.GL_QUADS);
31: //Posterior
32:     Gl.glColor3ub(255, 0, 255);
33:         Gl.glVertex3d(1, 0.25, -1.5);
34:         Gl.glVertex3d(1, -0.25, -1.5);
35:         Gl.glVertex3d(-1, -0.25, -1.5);
36:         Gl.glVertex3d(-1, 0.25, -1.5);
37: //Bottom
38:     Gl.glColor3ub(0, 255, 255);
39:         Gl.glVertex3d(-1, -0.25, -1.5);
40:         Gl.glVertex3d(1, -0.25, -1.5);
41:         Gl.glVertex3d(1, -0.25, 1.5);
42:         Gl.glVertex3d(-1, -0.25, 1.5);
43: //Left Side
44:     Gl.glColor3ub(255, 255, 0);
45:         Gl.glVertex3d(-1, 0.25, -1.5);
46:         Gl.glVertex3d(-1, -0.25, -1.5);
47:         Gl.glVertex3d(-1, -0.25, 1.5);
```

```
48:                 Gl.glVertex3d(-1, 0.25, 1.5);
49: //Right Side
50:       Gl.glColor3ub(0, 0, 255);
51:                 Gl.glVertex3d(1, 0.25, 1.5);
52:                 Gl.glVertex3d(1, -0.25, 1.5);
53:                 Gl.glVertex3d(1, -0.25, -1.5);
54:                 Gl.glVertex3d(1, 0.25, -1.5);
55: //Top
56:           Gl.glColor3ub(0, 255, 0);
57:                  Gl.glVertex3d(-1, 0.25, -1.5);
58:                 Gl.glVertex3d(-1, 0.25, 1.5);
59:                 Gl.glVertex3d(1, 0.25, 1.5);
60:                 Gl.glVertex3d(1, 0.25, -1.5);
61: //Anterior
62:           Gl.glColor3ub(255, 0, 0);
63:                 Gl.glVertex3d(-1, 0.25, 1.5);
64:                   Gl.glVertex3d(-1, -0.25, 1.5);
65:                 Gl.glVertex3d(1, -0.25, 1.5);
66:                 Gl.glVertex3d(1, 0.25, 1.5);
68:       Gl.glEnd();
69: END
```

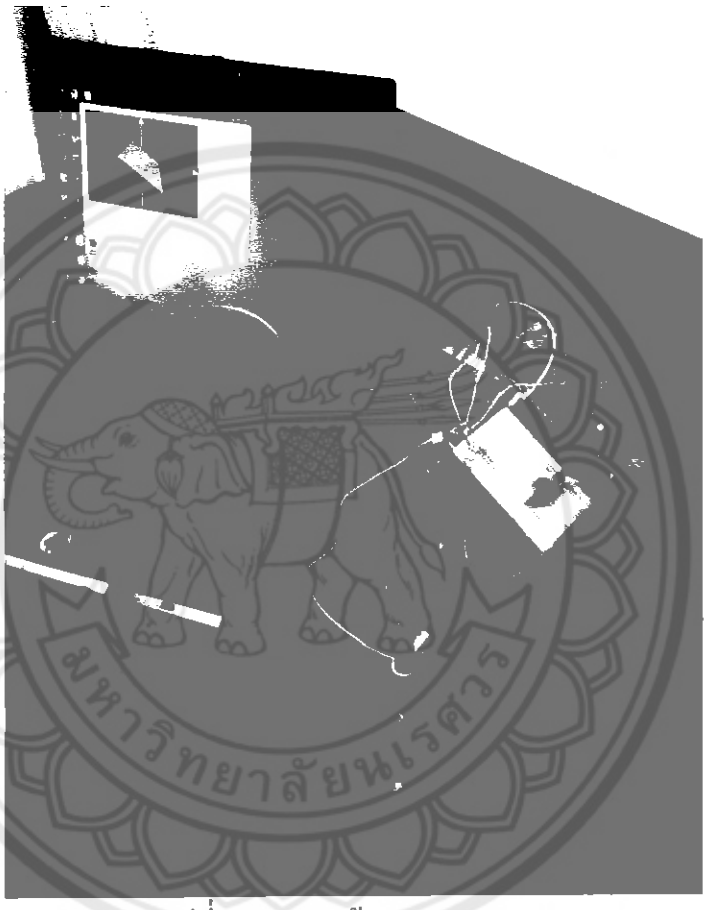
3.4 บทสรุป

ในบทที่ 3 นี้จะกล่าวถึงการดำเนินงานทั้งหมด ซึ่งอันดับแรกจะกล่าวถึงภาพรวมของระบบ (System Overview) ว่าระบบนั้นมีองค์ประกอบอะไรและมีหลักการทำงานอย่างไร จากนั้นจะกล่าวถึงการเขียนโปรแกรมลงบนบอร์ดไมโครโปรเซสเซอร์ในการอ่านค่าเซนเซอร์เพื่อนำมาประมวลผลและส่งไปให้กับ PC ว่ามีวิธีการอย่างไร เริ่มจาก การใช้โมดูล Timer1 ในการจับเวลา หลักการประมวลผลสัญญาณที่ได้จากเซนเซอร์ MEMSIC2125 และ CMPS03 พร้อมทั้ง Pseudo Code ที่สำคัญของวิธีการต่างๆที่กล่าวไว้ จากนั้นจะกล่าวถึงการเขียนโปรแกรมของหน้าจอแสดงผลหรือ GUI โดยจะกล่าวถึงตั้งแต่ไลบรารีที่นำเอามาใช้สำหรับการเขียน การกำหนดค่าการมองเห็นภาพ การกำหนดค่าสำหรับการ Projection การวาดรูปลูกบาศก์สำหรับการแสดงผล การหมุนภาพ

โปรโตคอลสำหรับการส่งผ่านชุดข้อมูลระหว่างบอร์ดไมโครคอนโทลเลอร์กับ PC และสุดท้าย Pseudo Code ที่สำคัญในการโปรแกรมเพื่อสร้างหน้าจอแสดงผล

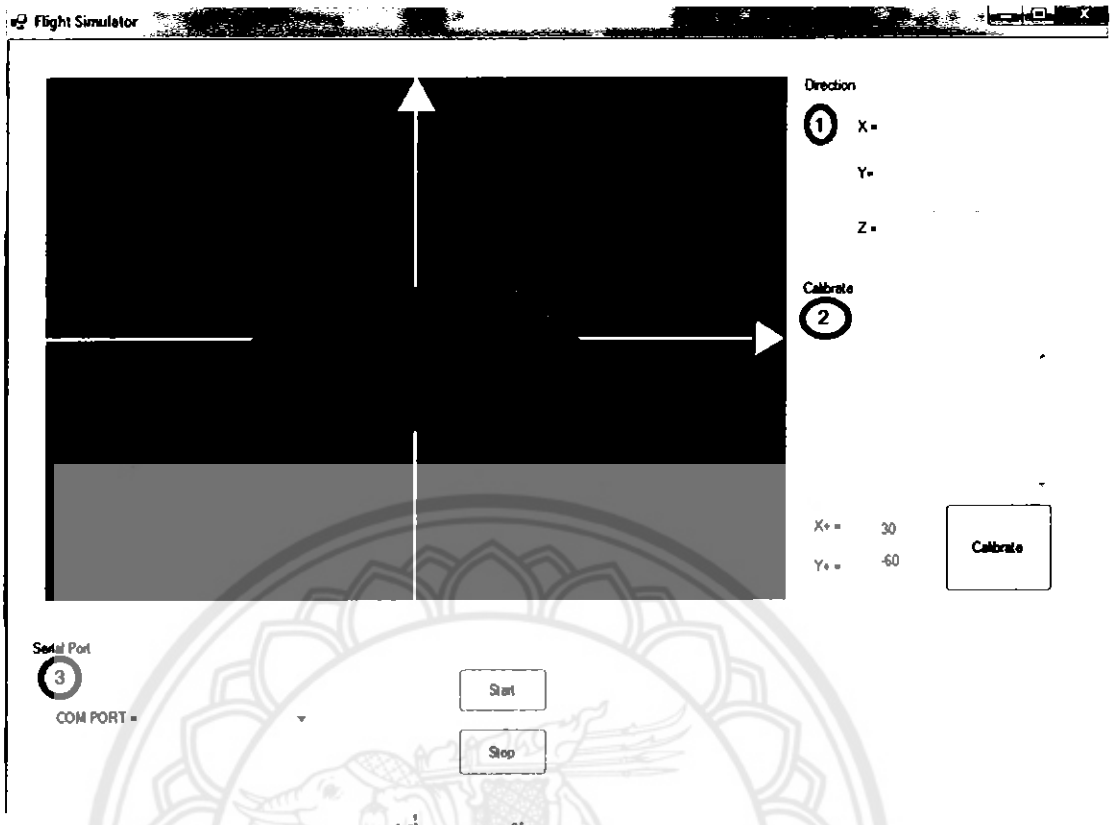
บทที่ 4
การทดลองและผลการทดลอง

4.1 ต้นแบบของระบบ



รูปที่ 4. 1 แสดงต้นแบบของระบบ

ระบบที่พัฒนาขึ้น(บอร์ดไมโครคอนโทลเลอร์ฝั่งขวามือในรูปที่ 4.1) คือระบบวัดมุมเอียง ซึ่งจะทำหน้าที่วัดมุมเอียงและประมวลผล จากนั้นจะส่งค่ามุมเอียงที่วัดได้ไปยัง PC (Notebook ฝั่งซ้ายมือในรูปที่ 4.1) ผ่าน Serial Port เพื่อไปแสดงผลจำลองการบินบนหน้าจอ GUI โดยที่จะแสดงเป็นรูปกล่องลูกบาศก์จำลอง ซึ่งจะแสดงทิศทางเดียวกันกับระบบที่พัฒนาขึ้น ดังแสดงในรูปที่ 4.1



รูปที่ 4. 2 หน้าจอ GUI ของระบบ

หน้าจอ GUI ของระบบจะประกอบไปด้วย 4 องค์ประกอบ ดังแสดงในรูปที่ 4.2 ดังนี้

1. **Direction** คือส่วนที่จะแสดงมุมเอียงที่วัดได้ในแต่ละแนวแกนเป็นตัวเลข มีหน่วยเป็นองศาดีกรี

2. **Calibrate** คือส่วนใช้ปรับเทียบค่าในการวัดมุมเอียงในแนวแกน X และ Y เพื่อให้ค่าที่ถูกต้องที่สุด สามารถปรับได้โดยเมื่อระบบวางที่มุมเอียง X และ Y ที่ 0 องศาดีกรี ทำการปรับช่อง X+ = และ Y+= จนกว่าค่า X และ Y ที่แสดงในข้อที่ 1 จะแสดงค่าเป็น 0 องศาดีกรี

3. **Serial Port** คือส่วนที่ใช้สำหรับเลือก port ที่ต่อกับระบบ

4. คือส่วนที่ใช้สำหรับการแสดงผล ซึ่งคือภาพจำลองของระบบโดยที่จะแสดงรูปตามมุมเอียงที่วัดได้จากเซนเซอร์ทั้ง 2 ตัว (CPMS03 และ MEMSIC 2125) ใน 3 แกน (Roll, Pitch, Yaw)

4.2 ผลการทดลองการวัดมุมเอียงของเซนเซอร์ MEMSIC2125

4.2.1 วิธีการการทดลอง

การทดลองนั้นจะใช้ครึ่งวงกลมสำหรับวัดมุมมาใช้วัดมุมเอียงของระบบในแนวแกน X และแกน Y เพื่อใช้เป็นจุดเทียบกับที่ระบบวัดได้ ดังแสดงในรูปที่ 4.3 ส่วนที่ลูกศรสีแดงชี้

สำหรับการวัดค่ามุมเอียงนั้น จะวัดค่าตั้งแต่ -80 องศา ถึง 80 องศา โดยจะเพิ่มขึ้นทุกๆ 10 องศา โดยที่ในแต่ละครั้งที่วัดนั้น จะวัดทั้งหมด 10 ค่า โดยแต่ละค่าที่วัดนั้นจะมีเวลาห่างกัน 1 วินาที

รูปที่ 4. 3 วิธีการทดลองวัดมุมเอียงในแนวแกน X และ Y

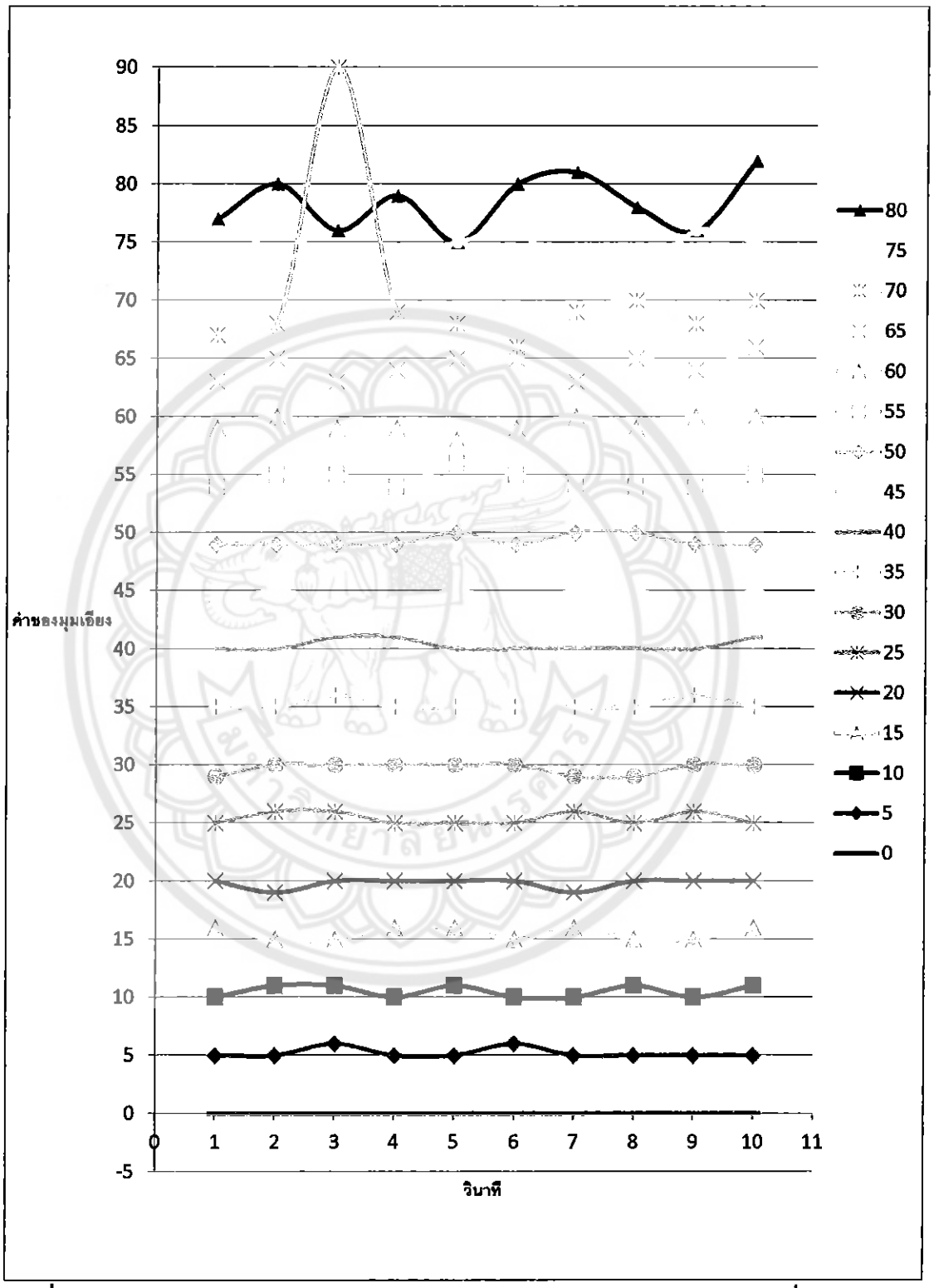
4.2.2 ผลการทดลอง

4.2.2.1 ตารางแสดงผลการทดลองวัดมุมเอียงในแนวแกน X

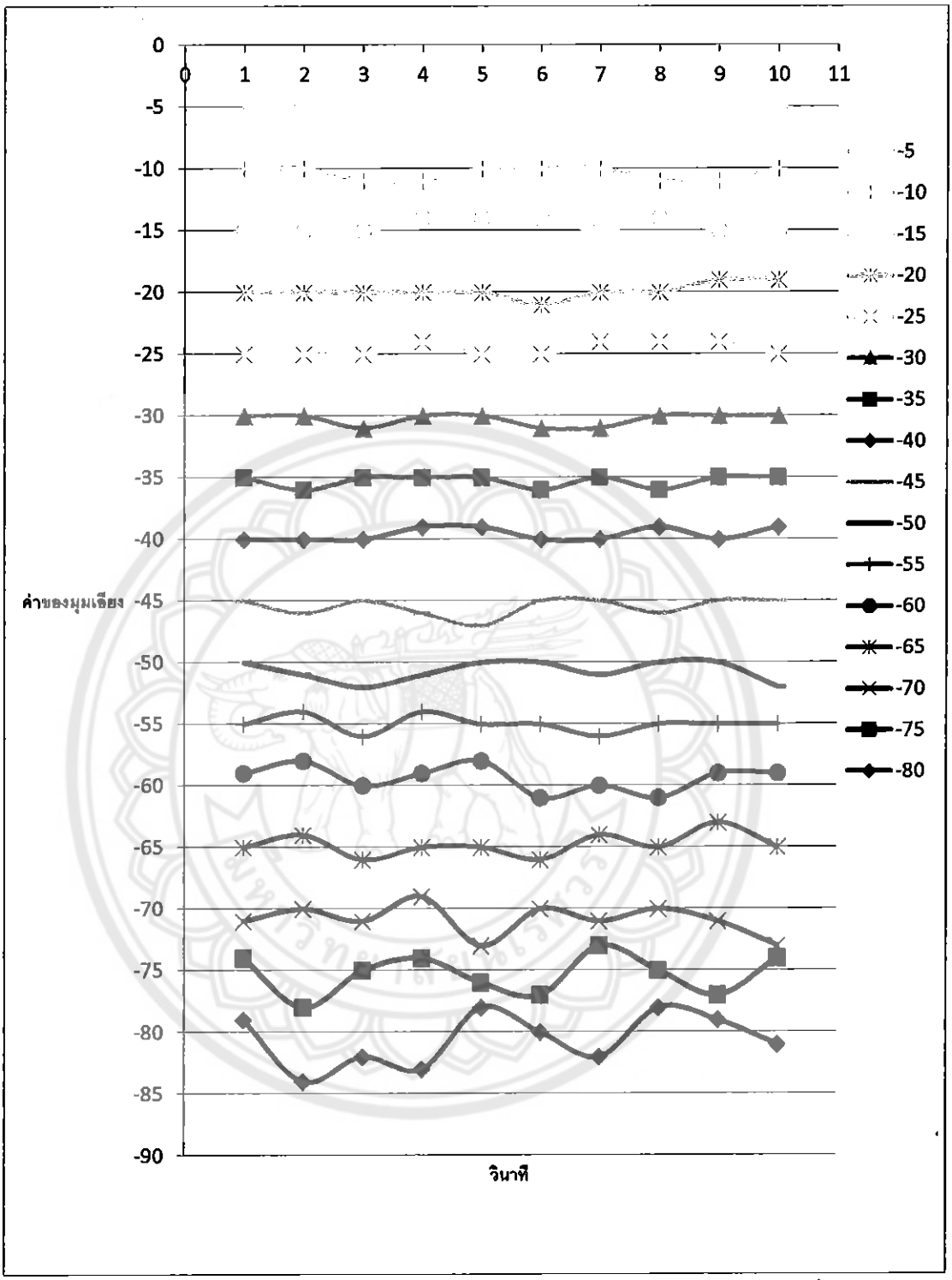
ตารางที่ 4. 1 แสดงผลการทดลอง ค่าเฉลี่ย และค่าความคลาดเคลื่อนของการทดลอง ของการวัดมุมเอียงในแนวแกน X

มุมเอียง (องศา)	ค่ามุมที่วัดได้ (องศา)										ค่าเฉลี่ย	ค่าความคลาดเคลื่อน (องศา)
	วินาทีที่ 1	วินาทีที่ 2	วินาทีที่ 3	วินาทีที่ 4	วินาทีที่ 5	วินาทีที่ 6	วินาทีที่ 7	วินาทีที่ 8	วินาทีที่ 9	วินาทีที่ 10		
-80	-79	-84	-82	-83	-78	-80	-82	-78	-79	-81	-80.6	0.6
-75	-74	-78	-75	-74	-76	-77	-73	-75	-77	-74	-75.3	0.3
-70	-71	-70	-71	-69	-73	-70	-71	-70	-71	-73	-70.9	0.9
-65	-65	-64	-66	-65	-65	-66	-64	-65	-63	-65	-64.8	0.2
-60	-59	-58	-60	-59	-58	-61	-60	-61	-59	-59	-59.4	0.6
-55	-55	-54	-56	-54	-55	-55	-56	-55	-55	-55	-55	0
-50	-50	-51	-52	-51	-50	-50	-51	-50	-50	-52	-50.7	0.7
-45	-45	-46	-45	-46	-47	-45	-45	-46	-45	-45	-45.5	0.5
-40	-40	-40	-40	-39	-39	-40	-40	-39	-40	-39	-39.6	0.4
-35	-35	-36	-35	-35	-35	-36	-35	-36	-35	-35	-35.3	0.3
-30	-30	-30	-31	-30	-30	-31	-31	-30	-30	-30	-30.3	0.3
-25	-25	-25	-25	-24	-25	-25	-24	-24	-24	-25	-24.6	0.4
-20	-20	-20	-20	-20	-20	-21	-20	-20	-19	-19	-19.9	0.1
-15	-15	-15	-15	-14	-14	-14	-15	-14	-15	-15	-14.6	0.4
-10	-10	-10	-11	-11	-10	-10	-10	-11	-11	-10	-10.4	0.4
-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	0
0	0	0	0	0	0	0	0	0	0	0	0	0
5	5	5	6	5	5	6	5	5	5	5	5.2	0.2
10	10	11	11	10	11	10	10	11	10	11	10.5	0.5
15	16	15	15	16	16	15	16	15	15	16	15.5	0.5
20	20	19	20	20	20	20	19	20	20	20	19.8	0.2
25	25	26	26	25	25	25	26	25	26	25	25.4	0.4
30	29	30	30	30	30	30	29	29	30	30	29.7	0.3
35	35	35	36	35	35	35	35	35	36	35	35.2	0.2
40	40	40	41	41	40	40	40	40	40	41	40.3	0.3
45	44	45	44	44	45	44	45	45	44	45	44.5	0.5
50	49	49	49	49	50	49	50	50	49	49	49.3	0.7
55	54	55	55	54	56	55	54	54	54	55	54.6	0.4
60	59	60	59	59	58	59	60	59	60	60	59.3	0.7
65	63	65	63	64	65	65	63	65	64	66	64.3	0.7
70	67	68	90	69	68	66	69	70	68	70	70.5	0.5
75	72	76	73	74	75	76	71	72	76	75	74	1
80	77	80	76	79	75	80	81	78	76	82	78.4	1.6
ค่าความคลาดเคลื่อนเฉลี่ย (องศา)												0.448

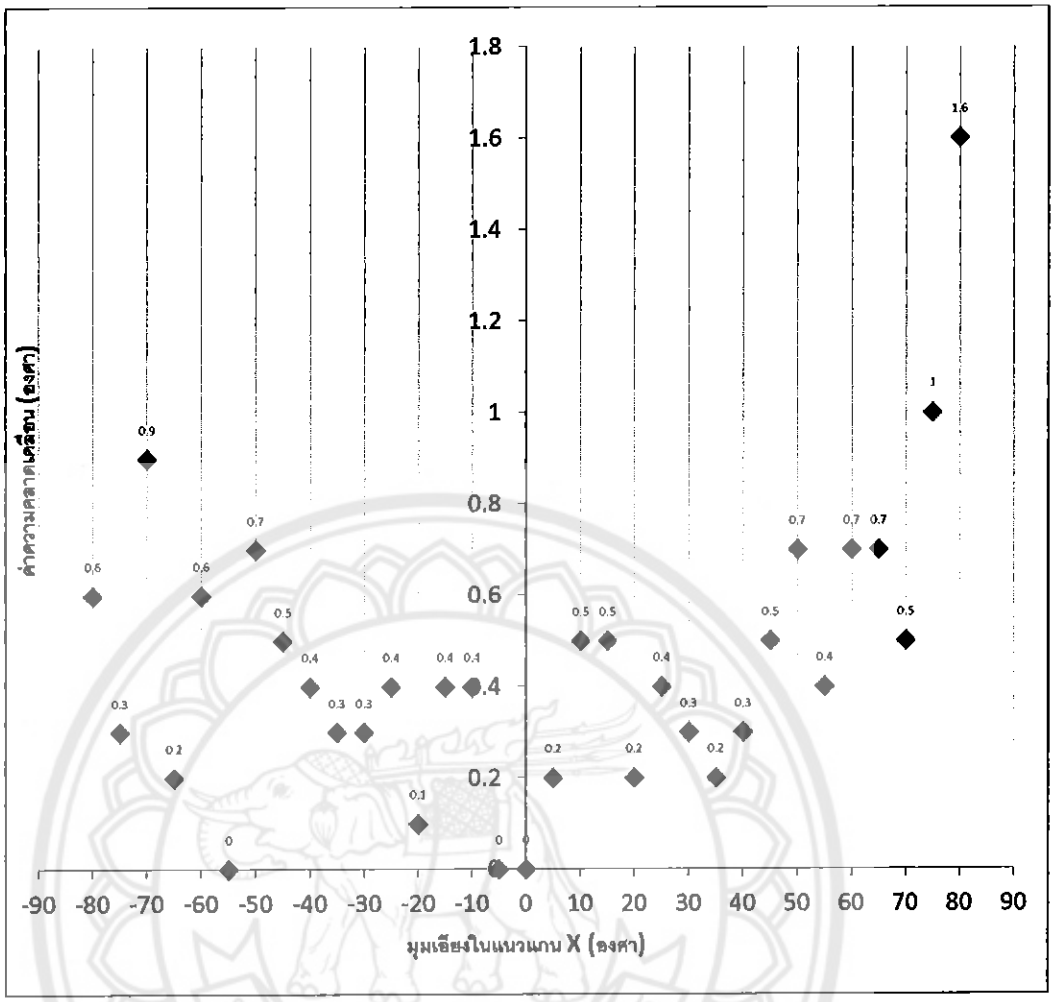
4.2.2.2 กราฟแสดงผลการทดลองของการวัดมุมเอียงในแนวแกน X



รูปที่ 4. 4 กราฟแสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน X (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 0 ถึง 80 องศา



รูปที่ 4. 5 กราฟแสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน X (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง -5 ถึง - 80 องศา



รูปที่ 4. 6 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน (แกน Y) กับค่ามุมเอียงในแนวแกน X (แกน X)

4.2.2.3 วิเคราะห์ผลการทดลองของการวัดมุมเอียงโดยเซนเซอร์ MEMSIC 2125 ในแนวแกน X

จากตารางที่ 4.1 รูปที่ 4.4 และรูปที่ 4.5 นั้น ค่าที่วัดได้จากเซนเซอร์นั้นจะไม่นิ่ง เมื่อทำการวัดมุมจากวินาทีที่ 1 ถึง วินาทีที่ 10 นั้น ค่าจะแกว่งขึ้นหรือแกว่งลง 1 องศา ทุกๆ 2-3 วินาที

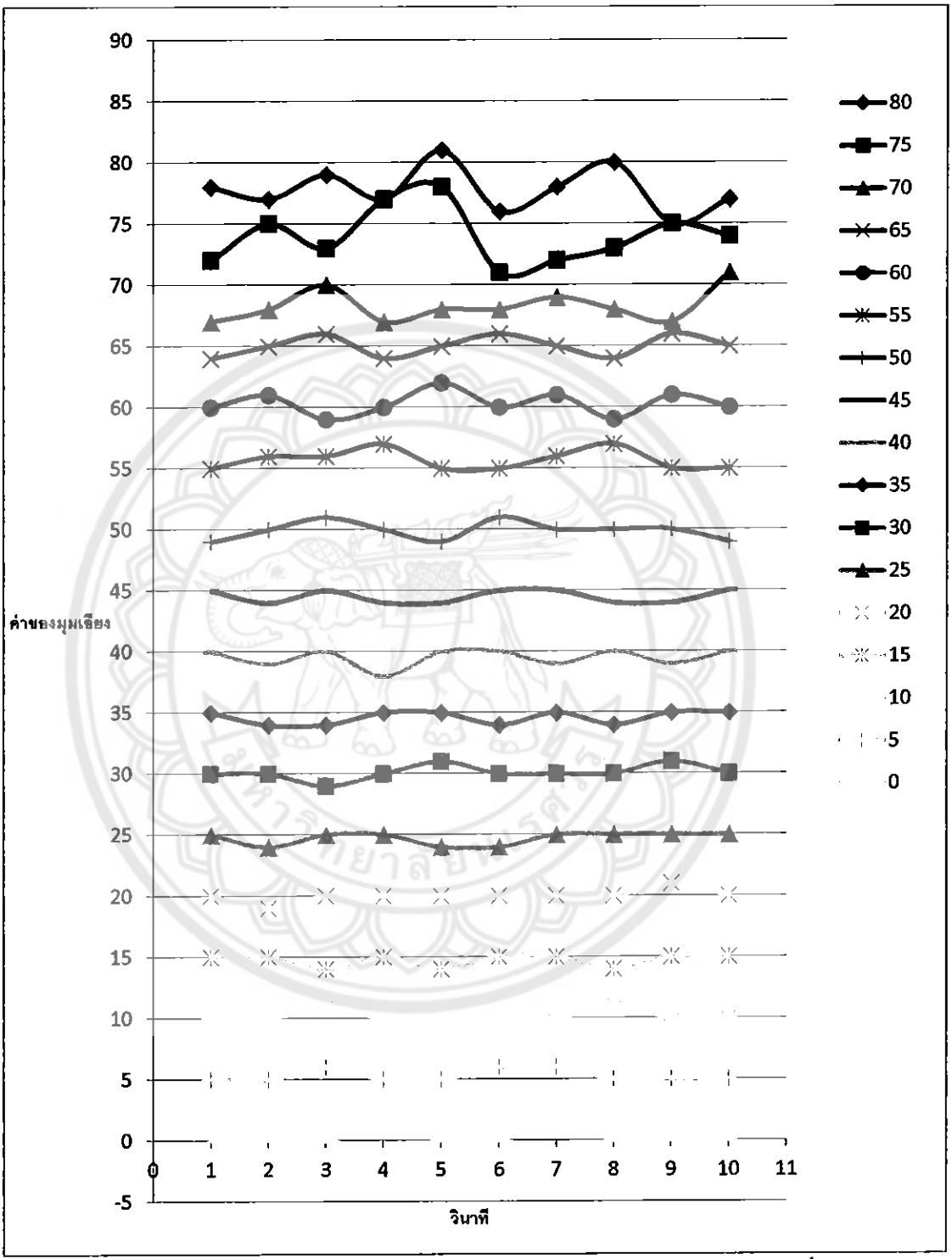
จากรูปที่ 4.6 นั้น เซนเซอร์นั้นจะมีความแม่นยำในการวัดอยู่ในช่วงประมาณ -50 ถึง 50 องศา โดยที่ค่าความผิดพลาดในการวัดจะไม่เกิน 0.5องศา ยิ่งค่าที่วัดยิ่งเข้าใกล้ค่า -90 องศาหรือ 90 องศามากเท่าไร ค่ามุมเอียงจะมีค่าความผิดพลาดของการวัดมากขึ้น เช่น ที่ มุมเอียง 80 องศาให้ค่าที่ผิดพลาดเฉลี่ยใน 10 วินาที ถึง 1 องศา แต่ประสิทธิภาพโดยรวมของเซนเซอร์ถือว่าพอใช้ได้เนื่องจากค่าความผิดพลาดเฉลี่ยมีค่าเพียง 0.448 องศา

4.2.2.4 ตารางแสดงผลการทดลองวัดมุมเอียงในแนวแกน Y

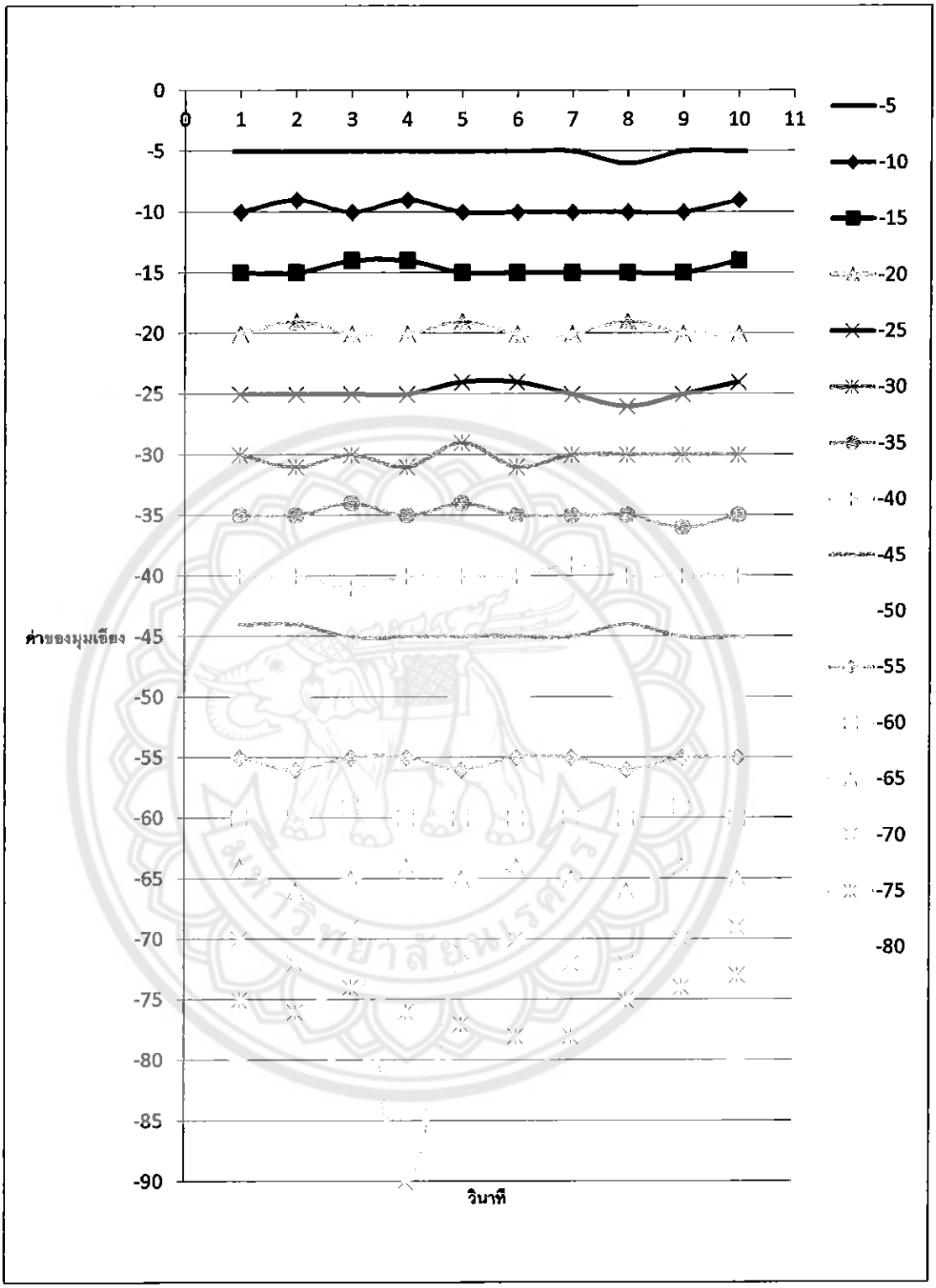
ตารางที่ 4. 2 แสดงผลการทดลอง ค่าเฉลี่ย และค่าความคลาดเคลื่อนของการทดลอง ของการวัดมุมเอียงในแนวแกน Y

มุมเอียง (องศา)	ค่ามุมที่วัดได้ (องศา)										ค่าเฉลี่ย (องศา)	ค่าความคลาดเคลื่อน (องศา)
	วินาทีที่ 1	วินาทีที่ 2	วินาทีที่ 3	วินาทีที่ 4	วินาทีที่ 5	วินาทีที่ 6	วินาทีที่ 7	วินาทีที่ 8	วินาทีที่ 9	วินาทีที่ 10		
-80	-80	-83	-79	-83	-79	-76	-83	-84	-81	-80	-80.8	0.8
-75	-75	-76	-74	-76	-77	-78	-78	-75	-74	-73	-75.6	0.6
-70	-70	-72	-69	-90	-71	-70	-72	-72	-70	-69	-72.5	2.5
-65	-64	-66	-65	-64	-65	-64	-65	-66	-64	-65	-64.8	0.2
-60	-60	-60	-59	-60	-60	-60	-60	-60	-59	-60	-59.8	0.2
-55	-55	-56	-55	-55	-56	-55	-55	-56	-55	-55	-55.3	0.3
-50	-50	-50	-49	-49	-50	-50	-49	-50	-50	-50	-49.7	0.3
-45	-44	-44	-45	-45	-45	-45	-45	-44	-45	-45	-44.7	0.3
-40	-40	-40	-41	-40	-40	-40	-39	-40	-40	-40	-40	0
-35	-35	-35	-34	-35	-34	-35	-35	-35	-36	-35	-34.9	0.1
-30	-30	-31	-30	-31	-29	-31	-30	-30	-30	-30	-30.2	0.2
-25	-25	-25	-25	-25	-24	-24	-25	-26	-25	-24	-24.8	0.2
-20	-20	-19	-20	-20	-19	-20	-20	-19	-20	-20	-19.7	0.3
-15	-15	-15	-14	-14	-15	-15	-15	-15	-15	-14	-14.7	0.3
-10	-10	-9	-10	-9	-10	-10	-10	-10	-10	-9	-9.7	0.3
-5	-5	-5	-5	-5	-5	-5	-5	-6	-5	-5	-5.1	0.1
0	0	0	0	-1	0	0	-1	0	0	0	-0.2	0.2
5	5	5	6	5	5	6	6	5	5	5	5.3	0.3
10	10	10	11	10	10	10	10	11	10	10	10.2	0.2
15	15	15	14	15	14	15	15	14	15	15	14.7	0.3
20	20	19	20	20	20	20	20	20	21	20	20	0
25	25	24	25	25	24	24	25	25	25	25	24.7	0.3
30	30	30	29	30	31	30	30	30	31	30	30.1	0.1
35	35	34	34	35	35	34	35	34	35	35	34.6	0.4
40	40	39	40	38	40	40	39	40	39	40	39.5	0.5
45	45	44	45	44	44	45	45	44	44	45	44.5	0.5
50	49	50	51	50	49	51	50	50	50	49	49.9	0.1
55	55	56	56	57	55	55	56	57	55	55	55.7	0.7
60	60	61	59	60	62	60	61	59	61	60	60.3	0.3
65	64	65	66	64	65	66	65	64	66	65	65	0
70	67	68	70	67	68	68	69	68	67	71	68.3	1.7
75	72	75	73	77	78	71	72	73	75	74	74	1
80	78	77	79	77	81	76	78	80	75	77	77.8	2.2
ค่าความคลาดเคลื่อนเฉลี่ย (องศา)												0.469

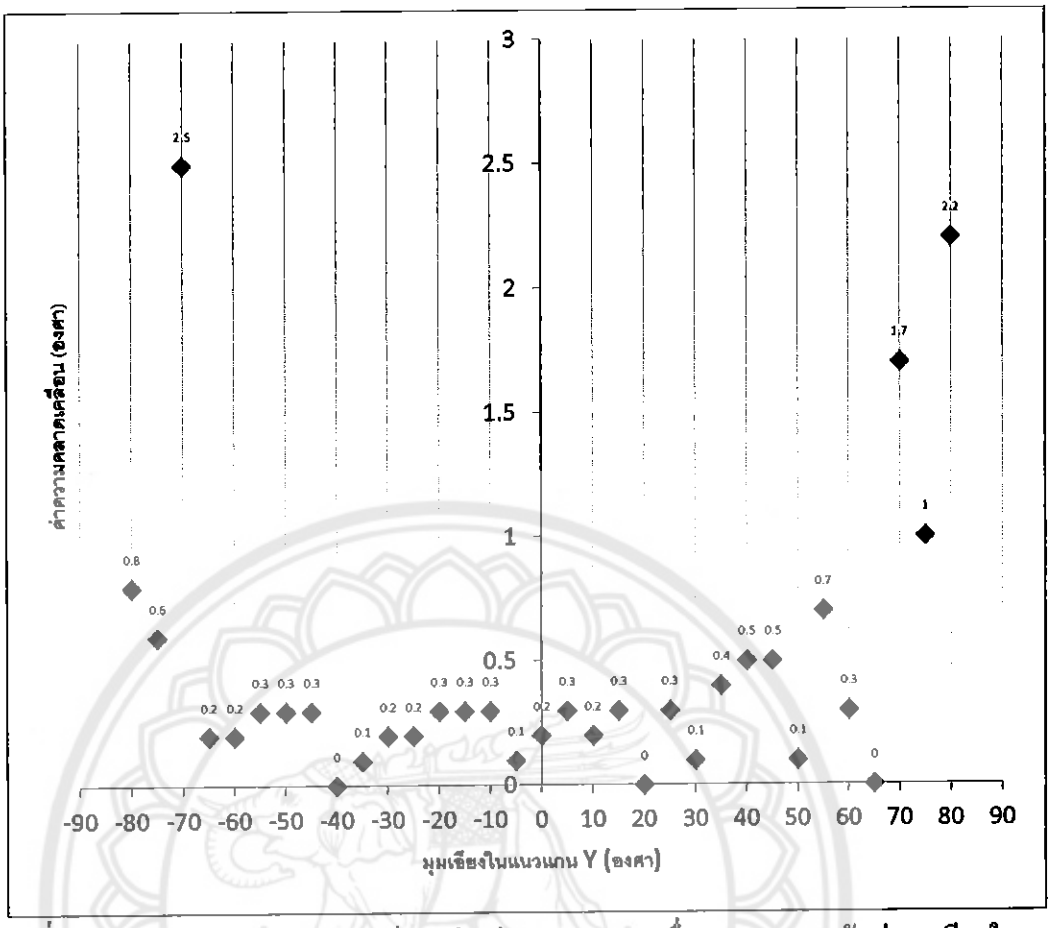
4.2.2.5 กราฟแสดงผลการทดลองของการวัดมุมเอียงในแนวแกน Y



รูปที่ 4. 7 กราฟแสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Y (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 0 องศา ถึง 80 องศา



รูปที่ 4. 8 แสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Y (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง -5 องศา ถึง -80 องศา



รูปที่ 4. 9 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน (แกน Y) กับค่ามุมเอียงในแนวแกน Y (แกน X)

4.2.3.6 วิเคราะห์ผลการทดลองของการวัดมุมเอียงโดยเซนเซอร์ MEMSIC 2125 ในแนวแกน Y

จากตารางที่ 4.2 รูปที่ 4.7 และรูปที่ 4.8 นั้น ค่าที่วัดได้จากเซนเซอร์นั้นจะไม่นิ่ง เมื่อทำการวัดมุมจากวินาทีที่ 1 ถึง วินาทีที่ 10 นั้น ค่าจะแกว่งขึ้นหรือแกว่งลง 1 องศา ทุกๆ 2-3 วินาที

จากรูปที่ 4.9 นั้น เซนเซอร์นั้นจะมีความแม่นยำในการวัดอยู่ในช่วงประมาณ -65 ถึง 65 องศา โดยที่ค่าความผิดพลาดในการวัดจะไม่เกิน 0.5องศา ยิ่งค่าที่วัดยิ่งเข้าใกล้ค่า -90 องศาหรือ 90 องศามากเท่าไร ค่ามุมเอียงจะมีค่าความผิดพลาดของการวัดมากขึ้น เช่น ที่ มุมเอียง 80 องศา ให้ค่าที่ผิดพลาดเฉลี่ยใน 10 วินาที ถึง 2.2 องศา หรือที่ค่ามุมเอียง -70 องศา ให้ค่าที่ผิดพลาดเฉลี่ยใน 10 วินาที ถึง 2.5 องศา แต่ประสิทธิภาพโดยรวมของเซนเซอร์ถือว่าพอใช้ได้เนื่องจากค่าความผิดพลาดเฉลี่ยมีค่าเพียง 0.469 องศา

4.3 ผลการทดลองการวัดมุมเอียงของเซนเซอร์ CMPS03

4.3.1 วิธีการทดลอง

การทดลองนั้นจะใช้ฐานวงกลมที่มีเสกลสำหรับการวัดทั้งหมด 360 องศา โดยที่เพิ่มขึ้นทีละ 10 องศา ใช้สำหรับการวัดมุมในแนวแกน Z เพื่อเป็นจุดเทียบกับที่ระบบวัดได้ ดังแสดงในรูปที่ 4.3 ส่วนที่ลูกศรสีแดงชี้

สำหรับการวัดค่ามุมเอียงนั้น จะวัดค่าตั้งแต่ 0 องศา ถึง 360 องศา โดยจะเพิ่มขึ้นทุกๆ 10 องศา โดยที่ในแต่ละครั้งที่วัดนั้น จะวัดทั้งหมด 10 ค่า โดยแต่ละค่าที่วัดนั้นจะมีเวลาห่างกัน 1 วินาที

รูปที่ 4. 10 วิธีการทดลองวัดมุมเอียงในแนวแกน Z

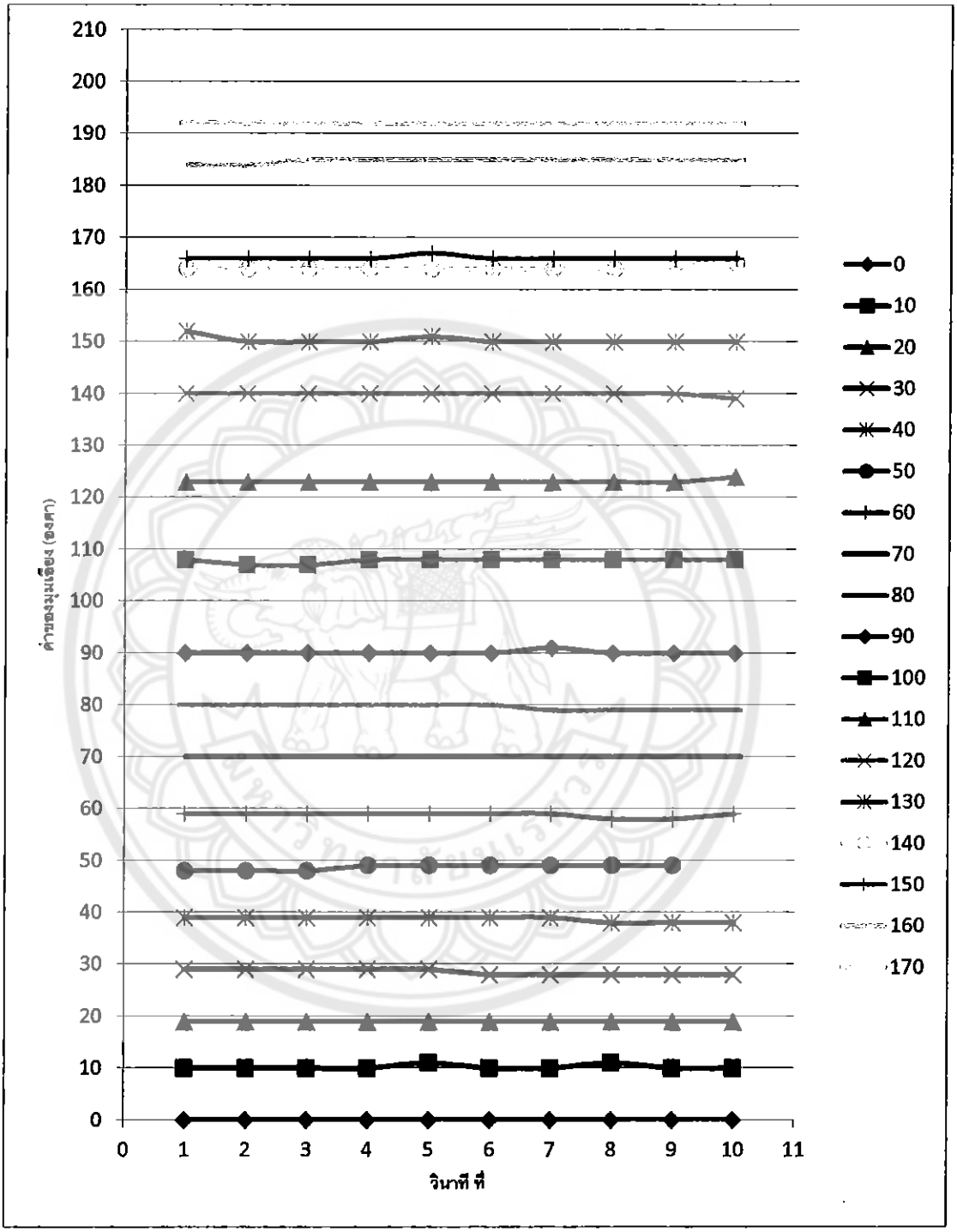
4.3.2 ผลการทดลอง

4.3.2.1 ตารางแสดงผลการทดลองวัดมุมเอียงในแนวแกน Z

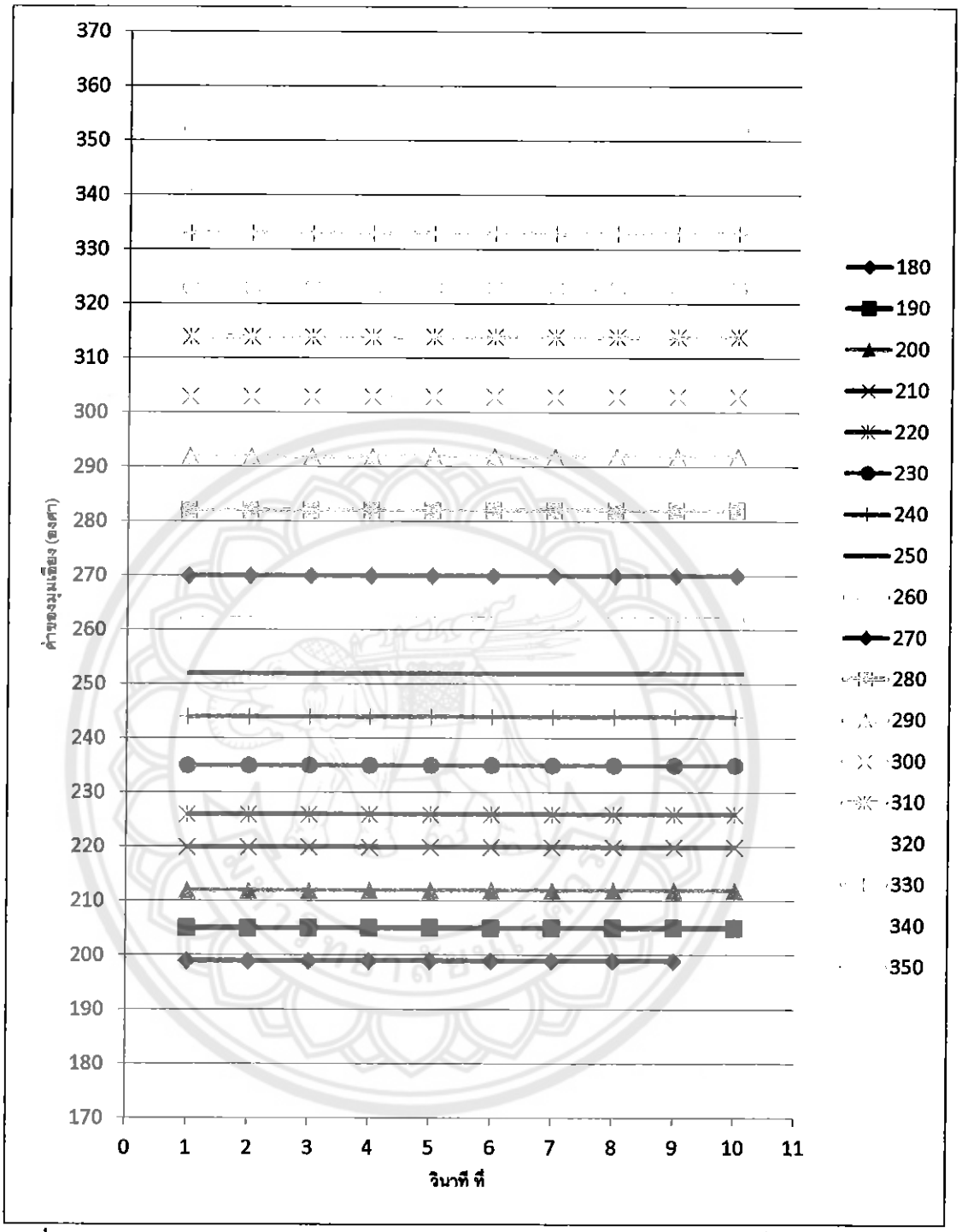
ตารางที่ 4. 3 แสดงผลการทดลอง ค่าเฉลี่ย และค่าความคลาดเคลื่อนของการทดลอง ของการวัดมุมเอียงในแนวแกน Z

มุมเอียง (องศา)	ค่ามุมที่วัดได้ (องศา)										ค่าเฉลี่ย (องศา)	ค่าความคลาดเคลื่อน (องศา)
	วิ่งที่ 1	วิ่งที่ 2	วิ่งที่ 3	วิ่งที่ 4	วิ่งที่ 5	วิ่งที่ 6	วิ่งที่ 7	วิ่งที่ 8	วิ่งที่ 9	วิ่งที่ 10		
0	0	0	0	0	0	0	0	0	0	0	0	0
10	10	10	10	10	11	10	10	11	10	10	10.2	0.2
20	19	19	19	19	19	19	19	19	19	19	19	1
30	29	29	29	29	29	28	28	28	28	28	28.5	1.5
40	39	39	39	39	39	39	39	38	38	38	38.7	1.3
50	48	48	48	49	49	49	49	49	49	49	48.7	1.3
60	59	59	59	59	59	59	59	58	58	59	58.8	1.2
70	70	70	70	70	70	70	70	70	70	70	70	0
80	80	80	80	80	80	80	79	79	79	79	79.6	0.4
90	90	90	90	90	90	90	91	90	90	90	90.1	0.1
100	108	107	107	108	108	108	108	108	108	108	107.8	7.8
110	123	123	123	123	123	123	123	123	123	124	123.1	13.1
120	140	140	140	140	140	140	140	140	140	139	139.9	19.9
130	152	150	150	150	151	150	150	150	150	150	150.3	20.3
140	164	164	164	164	164	164	164	164	165	165	164.2	24.2
150	166	166	166	166	167	166	166	166	166	166	166.1	16.1
160	184	184	185	185	185	185	185	185	185	185	184.8	24.8
170	192	192	192	192	192	192	192	192	192	192	192	22
180	199	199	199	199	199	199	199	199	199	199	199	19
190	205	205	205	205	205	205	205	205	205	205	205	15
200	212	212	212	212	212	212	212	212	212	212	212	12
210	220	220	220	220	220	220	220	220	220	220	220	10
220	226	226	226	226	226	226	226	226	226	226	226	6
230	235	235	235	235	235	235	235	235	235	235	235	5
240	244	244	244	244	244	244	244	244	244	244	244	4
250	252	252	252	252	252	252	252	252	252	252	252	2
260	262	262	262	262	262	262	262	262	262	262	262	2
270	270	270	270	270	270	270	270	270	270	270	270	0
280	282	282	282	282	282	282	282	282	282	282	282	2
290	292	292	292	292	292	292	292	292	292	292	292	2
300	303	303	303	303	303	303	303	303	303	303	303	3
310	314	314	314	314	314	314	314	314	314	314	314	4
320	323	323	323	323	323	323	323	323	323	323	323	3
330	333	333	333	333	333	333	333	333	333	333	333	3
340	341	341	341	341	341	341	341	341	341	341	341	1
350	352	352	352	352	352	352	352	352	352	352	352	2
360	359	359	359	359	359	359	359	359	359	359	359	1
ค่าความคลาดเคลื่อนเฉลี่ย												6.789

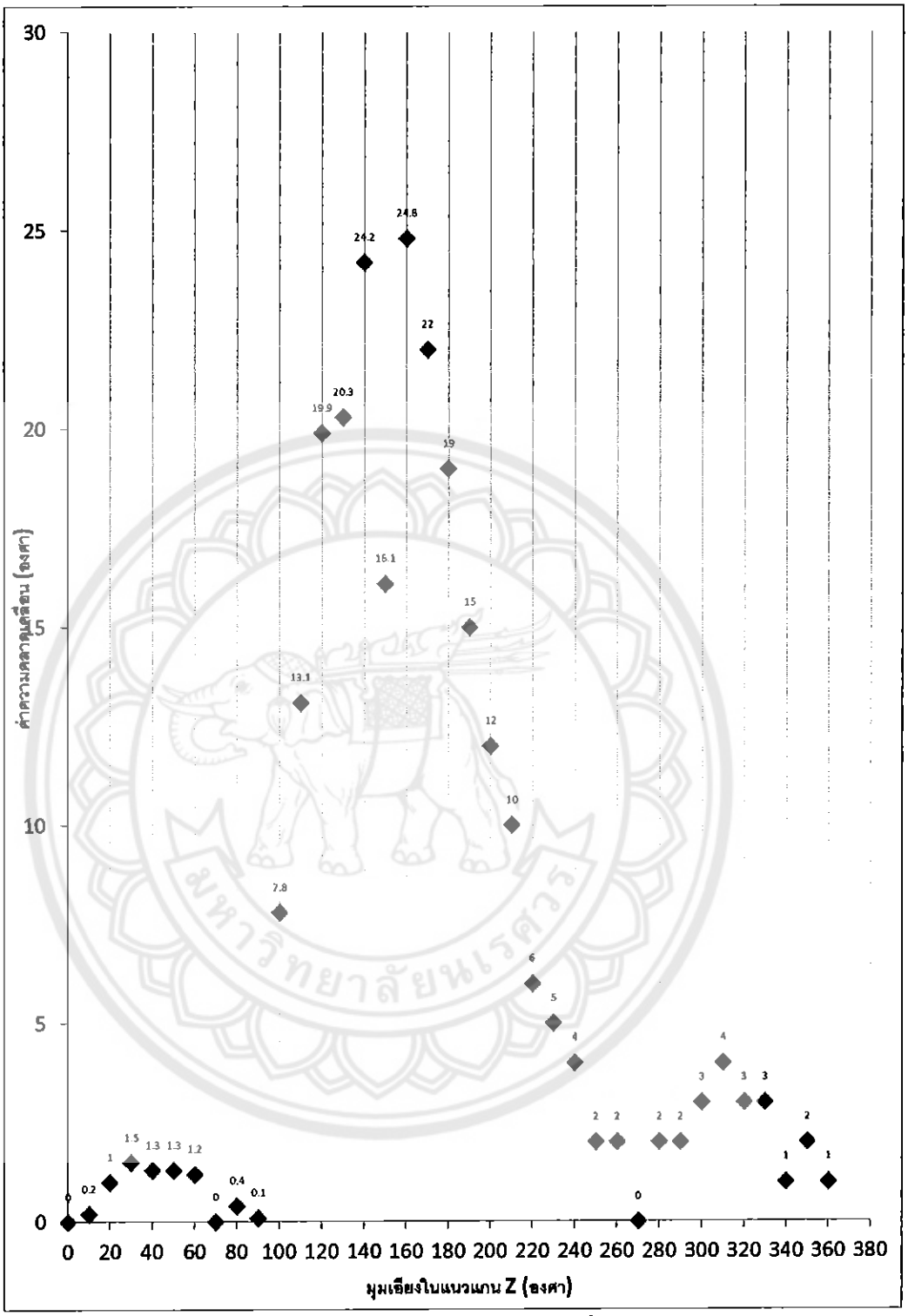
4.3.2.2 กราฟแสดงผลการทดลองของการวัดมุมเอียงในแนวแกน Z



รูปที่ 4. 11 แสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Z (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 0 - 170 องศา



รูปที่ 4. 12 แสดงความสัมพันธ์ของค่ามุมเอียงในแนวแกน Z (แกน Y) ณ วินาทีที่ 1 -10 (แกน X) ที่มุมเอียง 180 - 350 องศา



รูปที่ 4. 13 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน (แกน Y) กับค่ามุมเอียงในแนวแกน Z (แกน X)

4.3.2.3 วิเคราะห์ผลการทดลองการวัดมุมเอียงโดยเซนเซอร์ CMPS03

จากตารางที่ 4.3 รูปที่ 4.11 และรูปที่ 4.12 นั้น ค่าที่อ่านได้จาก CMPS 03 นั้นค่อนข้างนิ่งคือค่าไม่แกว่งขึ้นหรือแกว่งลงในช่วงวินาทีที่ 1 ถึง 10 ขณะที่ทำการวัด

แต่จากรูปที่ 4.13 นั้น ค่าความผิดพลาดในการวัดค่อนข้างที่จะสูง ยิ่งค่าที่วัดอยู่ในช่วง 100 ถึง 240 ค่ามุมเอียงที่วัดออกมาได้จะมีค่าที่ผิดพลาดที่สูงมาก เช่น ที่มุมเอียง 160 องศา วัดค่าออกมาได้ 184.8 องศา ซึ่งมีความผิดพลาดสูงถึง 24.8 องศาเป็นต้น

ค่าเฉลี่ยของความผิดพลาดในการวัดทั้งหมด มี ค่าสูงถึง 6.789 องศา กล่าวโดยสรุปคือ เซนเซอร์ CMPS03 นั้นให้ค่าในการวัดที่ค่อนข้างนิ่งแต่ค่าที่อ่านออกมานั้นมีความผิดพลาดค่อนข้างสูง

4.4 บทสรุป

ในบทนี้จะกล่าวถึงวิธีการทดลองและผลการทดลองของเซนเซอร์ 2 ตัว คือ MEMSIC 2125 และ CMPS03 โดยที่เซนเซอร์ MEMSIC 2125 จะให้ค่าในการวัดที่ไม่นิ่ง มีอัตราการแกว่งขึ้นลงของค่าอยู่ที่ประมาณ 1-3 วินาทีแต่ค่าในการวัดนั้นมีความแม่นยำสูง โดยที่ค่าความผิดพลาดของการวัดในแนวแกน X เฉลี่ยมีค่าเท่ากับ 0.448 และค่าความผิดพลาดของการวัดในแนวแกน Y เฉลี่ยมีค่าเท่ากับ 0.469 ซึ่งมีค่าไม่ถึง 1 องศา ส่วนเซนเซอร์ CMPS03 นั้นจะให้ค่าในการวัดที่ค่อนข้างนิ่งคือค่าจะไม่แกว่งขึ้นลง แต่ค่าในการวัดนั้นมีความแม่นยำต่ำ บางมุมเอียงที่วัดนั้นมีค่าความผิดพลาดมากกว่า 20 องศา ซึ่งถือว่าเป็นความผิดพลาดที่สูงมาก โดยที่ค่าความผิดพลาดของการวัดมุมเอียงในแนวแกน Z เฉลี่ยนั้นมีค่าเท่ากับ 6.789

บทที่ 5
สรุปผลการทดลอง

ระบบแสดงสถานะการบินสำหรับอากาศยานปีกหมุน (Flight Simulation Design for a Rotary-Wing Flying Robot) นี้เป็นระบบที่มีวัตถุประสงค์เพื่อพัฒนาต้นแบบสำหรับระบบแสดงสถานะการบินของเฮลิคอปเตอร์ในรูปแบบของข้อมูลทิศทางและระดับความเอียงของเฮลิคอปเตอร์ในขณะปฏิบัติงานอยู่ โดยข้อมูลที่วัดได้จะถูกส่งไปที่เครื่องคอมพิวเตอร์ส่วนบุคคล(PC) เพื่อแสดงผลออกทางหน้าจอ GUI (Graphic User Interface) ที่พัฒนาขึ้น

ผู้ดำเนินโครงงานใช้โปรแกรม Visual Studio 2010 ในการพัฒนาโปรแกรมโดยเลือกใช้ภาษา C# เนื่องจากเป็นภาษาที่ง่ายต่อการศึกษาและมีประสิทธิภาพประกอบกับทางผู้จัดทำมีความถนัดในภาษานี้เป็นพิเศษ นอกจากนั้นทางผู้ดำเนินโครงงานใช้ไลบรารี TaoOpenGl ซึ่งเป็นไลบรารีสำหรับสร้างภาพกราฟฟิกบนหน้าจอคอมพิวเตอร์และสามารถนำมาใช้กับภาษา C# ได้ ในการออกแบบภาพจำลองเพื่อแสดงผลของสถานะการบิน

5.1 สรุปผลการทดลอง

จากผลการทดลองในบทที่ 4 นั้น แยกสรุปได้เป็น 2 ส่วนดังนี้

1. เซนเซอร์ MEMSIC 2125 ให้ความแม่นยำในการวัดสูง กล่าวคือมีค่าความผิดพลาดในการวัดไม่เกิน 1 องศา ทั้ง 2 แนวแกน คือ แกน X และ แกน Y แต่ค่าที่วัดนั้นไม่นิ่งกล่าวคือ ค่าจะแกว่งขึ้นหรือแกว่งลงประมาณ 1 องศา ทุกๆ 1 ถึง 3 วินาที

2. เซนเซอร์ CMPS03 มีความแม่นยำในการวัดต่ำ กล่าวคือ ค่าความผิดพลาดในการวัดมีค่ามากสุดถึง 24.8 องศา และค่าความผิดพลาดในการวัดเฉลี่ยมีค่าเท่ากับ 6.789 องศา โดยที่ค่าความคลาดเคลื่อนดังกล่าวมีผลมาจาก เนื่องจาก CMPS03 นั้นเป็นเซนเซอร์ที่ใช้วิธีการตรวจจับสนามแม่เหล็กโลกในการวัด ซึ่งคลื่นสัญญาณของสนามแม่เหล็กโลกถูกรบกวนได้ง่าย เช่นถูกรบกวนจากคลื่นสัญญาณของอุปกรณ์สื่อสารหรือจากคอมพิวเตอร์ที่อยู่บริเวณภายในที่ทำการทดลอง ประกอบกับทดลองในอาคารซึ่งอาจทำให้เซนเซอร์ตรวจจับคลื่นจากสนามแม่เหล็กโลกได้ไม่ดีเท่าที่ควร แต่ค่าที่วัดออกมานั้นค่อนข้างนิ่งกล่าวคือ ค่าจะไม่แกว่งขึ้นหรือแกว่งลงในช่วง 1 ถึง 10 วินาทีที่ทำการวัด

นอกจากแล้วระบบนั้นมีการแสดงผลที่รวดเร็วแบบ real time กล่าวคือเมื่อเราทำการเอียงระบบไปในทิศทางใด รูปลูกบาศก์ในหน้าจอก็จะสามารถแสดงผลการเอียงตัวได้เหมือนกับระบบแบบทันทีทันใด

5.2 วิจารณ์ผลการทดลอง

1. ความถูกต้องที่ได้จากการวัดได้จากเซนเซอร์ MEMSIC 2125 ในการวุดมุมเอียงในแนวแกน X และ Y นั้นจะมีความถูกต้องเมื่อมุมเอียงทำมุมเอียงไม่เกิน -60 ถึง60 องศา เมื่อมุมเอียงที่วัดมีค่าเกินกว่านั้นจะทำให้ได้ค่าของการวัดมุมเอียงถูกต้องนั้นลดลง ซึ่งตรงกับ Spec ของเซนเซอร์ Memsic2125 ค่าที่วัดได้จากเซนเซอร์นั้นไม่นิ่งสามารถแก้ไขโดยใช้ทฤษฎีการ moving average[26] แต่ที่ผู้ดำเนินโครงงานไม่ใช้เพราะจะทำให้ระบบช้าและเกิดการแสดงผลไม่เป็นแบบ real time

2. ความคลาดเคลื่อนของเซนเซอร์ CMPS03 ที่ใช้สำหรับวัดมุมเอียงในแนวแกน Z นั้นจะมีค่าความคลาดเคลื่อนสูงเนื่องมาจากเซนเซอร์ที่ใช้ตรวจจับทิศทางนี้ เป็นการตรวจจับสนามแม่เหล็กโลก แล้วจึงรับค่าสัญญาณจากตัวตรวจจับมาประมวลผลเป็นข้อมูลดิจิตอลและสัญญาณพัลส์สำหรับการแสดงผลทิศทาง ซึ่งสนามแม่เหล็กโลกอาจถูกรบกวนได้ง่าย โดยคลื่นแม่เหล็กหรือคลื่นสัญญาณต่างๆ ซึ่งอาจจะเกิดจากสัญญาณรบกวนที่ส่งออกมาจากอุปกรณ์สื่อสารและคอมพิวเตอร์ที่อยู่ภายในบริเวณที่ทำการทดลอง นอกจากนั้นเรายังทดสอบในอาคารอาจทำให้เซนเซอร์ตรวจจับสนามแม่เหล็กโลกได้ไม่ดีทำให้ค่าที่ได้จากการวัดมุมเอียงนั้นคลาดเคลื่อนไปจากค่าจริง

5.3 การพัฒนาโครงงานต่อไปในอนาคต

1. ค่าที่ได้จากเซนเซอร์ MEMSIC 2125 ที่ใช้สำหรับวัดมุมเอียงในแนวแกน X และ แกน Y นั้นให้ค่าที่ไม่นิ่ง สามารถแก้ไขได้โดยใช้ moving average แต่จำทำให้ระบบช้า ไม่แสดงผลแบบ real time หรือแก้ได้โดยเปลี่ยนเซนเซอร์ที่มีคุณภาพดีกว่านี้ซึ่งจะส่งผลให้ระบบนี้ต้องการต้นทุนในการผลิตที่มากขึ้น

2. ค่าที่ได้จากเซนเซอร์ CMPS03 ที่ใช้สำหรับวัดมุมเอียงในแนวแกน Z นั้น มีค่าความถูกต้องในการวัดค่อนข้างน้อย สามารถแก้ไขได้โดยเปลี่ยนเซนเซอร์ที่มีคุณภาพดีกว่านี้ซึ่งจะส่งผลให้ระบบนี้ต้องการต้นทุนในการผลิตที่มากขึ้น

3. เซนเซอร์ CMPS03 ต้องวางแนวราบเท่านั้นถึงจะวัดค่าได้ถูกต้อง ถ้าเซนเซอร์นี้เอียงตัวค่าที่วัดได้จะผิดพลาดจากค่าจริงมาก ซึ่งทำให้ไม่สามารถใช้ได้จริงบนเฮลิคอปเตอร์เพราะเฮลิคอปเตอร์สามารถแก้ไขได้โดยเปลี่ยนเซนเซอร์ที่มีคุณภาพดีกว่านี้ซึ่งจะส่งผลให้ระบบนี้ต้องการต้นทุนในการผลิตที่มากขึ้น

5.4 ข้อเสนอแนะ

การพัฒนาโครงงานต่อในอนาคตนั้นควรมีความรู้พื้นฐานในเรื่องต่อไปนี้

1. การโปรแกรม Window Application ด้วยภาษา C#
2. ความรู้เกี่ยวกับ OpenGL Library
3. การโปรแกรมไมโครคอนโปรเซสเซอร์ตระกูล PIC ด้วยภาษา C
4. การต่อวงจรอิเล็กทรอนิกส์ขั้นพื้นฐาน

เอกสารอ้างอิง

[1] Eduzones. **กลศาสตร์การบินของเฮลิคอปเตอร์**.สืบค้นเมื่อ 25 กันยายน 2012, จาก http://blog.eduzones.com/wigi/82118.

[2] กองประชาสัมพันธ์สำนักงานเลขานุการกองทัพเรือ. **ซีฮอว์ค เฮลิคอปเตอร์ปราบเรือดำน้ำของกองทัพเรือไทย**.สืบค้นเมื่อ 25 กันยายน 2012, จาก http://www.navy.mi.th/sctr/news_release/2552/sep/seahawk.php.

[3] **KA-27 anti-submarine helicopter**.สืบค้นเมื่อ 25 กันยายน 2012, จาก http://www.guncopter.com/photos/ka-27-photo.php.

[4] defence.gov.au. **CH-47 Chinooks back from Afghanistan**. สืบค้นเมื่อ 25 กันยายน2012, จาก http://mymodelplanes.wordpress.com/2010/10/26/ch-47-chinooks-back-from-afghanistan/.

[5] ACME Worldwide Enterprises, I. **Rotary Aircraft True Q™ Dynamic Motion Seats for Simulators**.สืบค้นเมื่อ 25 กันยายน 2012, จาก http://www.acme-worldwide.com/dynamic_motion_seat_Rotary.htm.

[6] Wikipedia. **Douglas Engelbart**. สืบค้นเมื่อ 25 กันยายน 2012, จาก http://en.wikipedia.org/wiki/Douglas_Engelbart.

[7] Wikipedia. **Graphical user interface**. สืบค้นเมื่อ 25 กันยายน 2012, จาก http://en.wikipedia.org/wiki/Graphical_user_interface#PARC_user_interface.

[8] Wikipedia. **Microsoft Windows**. สืบค้นเมื่อ 25 กันยายน 2012, จาก http://en.wikipedia.org/wiki/Microsoft_Windows.

[9] Wikipedia. **Mac OS**. สืบค้นเมื่อ 25 กันยายน 2012, จาก http://en.wikipedia.org/wiki/Mac_OS.

[10] Wikipedia. **Linux**. สืบค้นเมื่อ 25 กันยายน 2012, จาก http://en.wikipedia.org/wiki/Linux.

[11] ETTTeam. **CP-PIC V3/485 (ICD2)**. สืบค้นเมื่อ 25 กันยายน 2012, จาก http://www.etteam.com/product/pic/cp-pic-v3-877-icd2.html.

[12] INEX. **MEMSIC2125 โมดูลวัดความเร่งแบบ *2* แกน**. สืบค้นเมื่อ 25 กันยายน 2012, จาก http://www.inex.co.th/inexstore/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=150&category_id=20&option=com_virtuemart&Itemid=1&vmcchk=1&Itemid=11#.US8S0jCnqtY.

[13] Wikipedia. **System on a chip.** สืบค้นเมื่อ 30 กันยายน 2012, จาก http://en.wikipedia.org/wiki/System_on_a_chip.

[14] INEX. **CMPS03 โมดูลเข็มทิศดิจิตอล.** สืบค้นเมื่อ 25 กันยายน 2012, จาก http://www.inex.co.th/inexstore/index.php?product_id=147&page=shop.product_details&category_id=20&flypage=flypage.tpl&option=com_virtuemart&Itemid=1#.US8TfDCnqtY.

[15] Wikipedia. **I²C.** สืบค้นเมื่อ 30 กันยายน 2012, จาก http://en.wikipedia.org/wiki/I%C2%B2C.

[16] INEX. **INEX COMPANY.** สืบค้นเมื่อ 30 กันยายน 2012, จาก http://www.inex.co.th/inexstore/.

[17] Wikipedia. **Pulse-width modulation.** สืบค้นเมื่อ 30 กันยายน 2012, จาก http://en.wikipedia.org/wiki/Pulse-width_modulation.

[18] Wikipedia. **EEPROM.** สืบค้นเมื่อ 30 กันยายน 2012, จาก http://en.wikipedia.org/wiki/EEPROM.

[19] Wikipedia. **การเรนเดอร์ภาพด้วยคอมพิวเตอร์.** สืบค้นเมื่อ 7 ตุลาคม 2012, จาก http://th.wikipedia.org/wiki/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%80%E0%B8%A3%E0%B8%99%E0%B9%80%E0%B8%94%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B8%A0%E0%B8%B2%E0%B8%9E%E0%B8%94%E0%B9%89%E0%B8%A7%E0%B8%A2%E0%B8%84%E0%B8%AD%E0%B8%A1%E0%B8%9E%E0%B8%B4%E0%B8%A7%E0%B9%80%E0%B8%95%E0%B8%AD%E0%B8%A3%E0%B9%8C.

[20] โมลิสกุลมงคง ไพศาล. **คอมพิวเตอร์กราฟฟิกส์2550,** ดวงกมลสมัย,บจก.

[21] Wikipedia. **Open Inventor.** สืบค้นเมื่อ 7 ตุลาคม 2012, จาก http://en.wikipedia.org/wiki/Open_Inventor.

[22] Rachanark, N. **วัตถุปิดที่เพิ่มจุดแบบทวีคูณเมื่อเพิ่มมิติ.** สืบค้นเมื่อ 7 ตุลาคม 2012, จาก http://www.bloggang.com/mainblog.php?id=navagan&month=07-05-2009&group=6&gblog=6.

[23] Loren K. Rhodes, P.D. **Projection Matrix.** สืบค้นเมื่อ 10 ตุลาคม 2012, จาก http://jcsites.juniata.edu/faculty/rhodes/graphics/projectionmat.htm.

[24] Wikipedia. **พาย (ค่าคงตัว).** สืบค้นเมื่อ 10 มกราคม 2013, จาก http://th.wikipedia.org/wiki/%E0%B8%9E%E0%B8%B2%E0%B8%A2_(%E0%B8%8

4%E0%B9%88%E0%B8%B2%E0%B8%84%E0%B8%87%E0%B8%95%E0%B8%B1%E0%B8%A7).

[25] Tao-Mono. **TaoOpenGL.** สืบค้นเมื่อ 10 ธันวาคม 2012, จาก http://www.mono-project.com/Tao.

[26] Wikipedia. **Moving Average.** สืบค้นเมื่อ 20 กุมภาพันธ์ 2013, จาก http://en.wikipedia.org/wiki/Moving_average.

ภาคผนวก

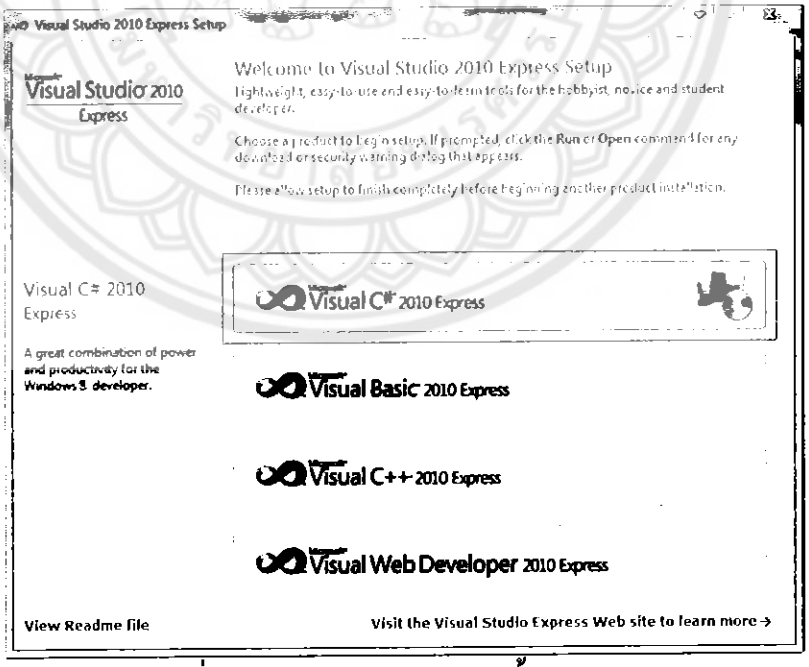
ก. การติดตั้งโปรแกรม Microsoft Visual Studio 2010 Express

1. การติดตั้งโปรแกรม Microsoft Visual Studio 2010 Express สามารถดาวน์โหลดโปรแกรมได้ที่ http://www.microsoft.com/visualstudio/eng/products/visual-studio-2010-express

2. เมื่อผู้ใช้งานทำการดาวน์โหลดโปรแกรม Microsoft Visual Studio 2010 Express ให้ผู้ใช้งานทำการติดตั้งโปรแกรม โดยการดับเบิ้ลคลิกที่ไฟล์ Setup ดังรูปที่ ก.1 จากนั้นจะเข้าสู่การติดตั้งโปรแกรม ดังรูปที่ ก.2

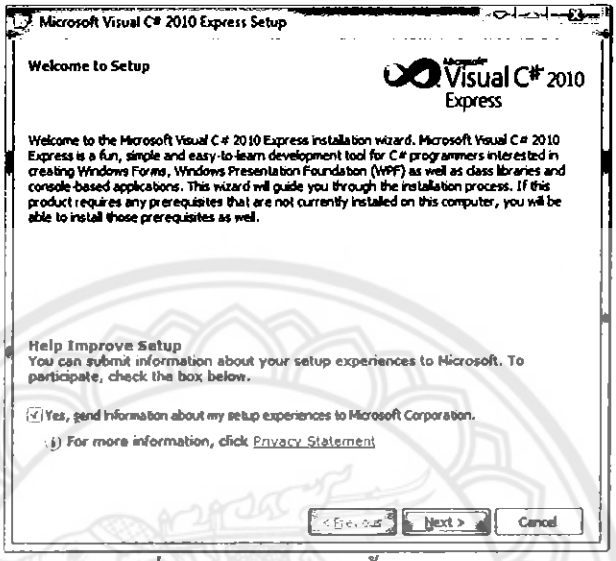
Name	Date modified	Type	Size
Include	3/21/2010 1:18 AM	File folder	
VBExpress	3/21/2010 1:18 AM	File folder	
VCExpress	3/21/2010 1:18 AM	File folder	
VCSExpress	3/21/2010 1:19 AM	File folder	
VWDExpress	3/21/2010 1:19 AM	File folder	
Autorun	9/25/2009 11:28 AM	Setup Information	1 KB
Setup	9/25/2009 11:28 AM	HTML Application	11 KB

รูปที่ ก. 1 การติดตั้งโปรแกรม Microsoft Visual Studio



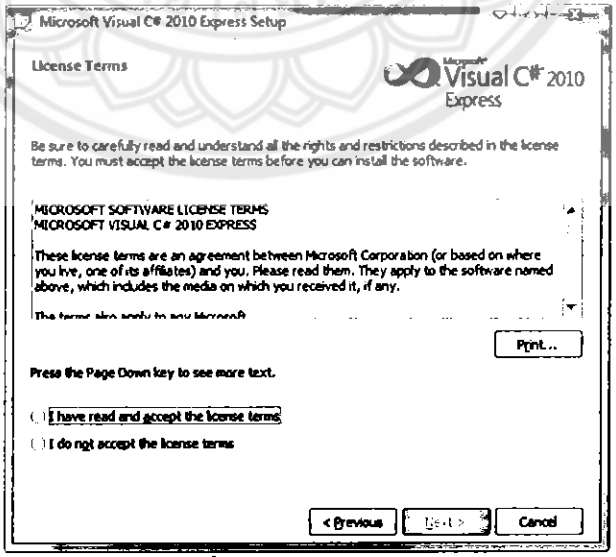
รูปที่ ก. 2 หน้าต่างแสดงการติดตั้งโปรแกรม

3. เมื่อเข้าสู่หน้าต่างการติดตั้งโปรแกรม ดังรูปที่ ก.2 แล้ว ให้ผู้ใช้งานเลือก Microsoft Visual C# 2010 Express จากนั้นรอสักครู่ โปรแกรมจะดำเนินการเข้าสู่ขั้นตอนการติดตั้ง ดังรูปที่ ก.3 ให้ผู้ใช้งานคลิก Next > เพื่อไปยังขั้นตอนถัดไป



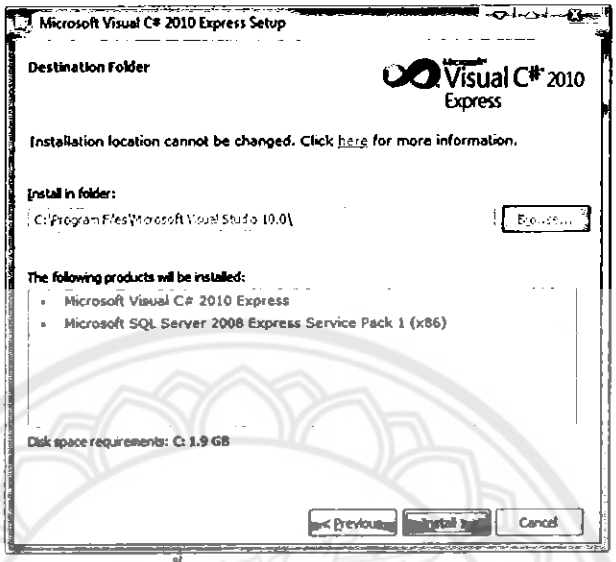
รูปที่ ก. 3 เข้าสู่การติดตั้งโปรแกรม

4. จากนั้นให้ผู้ใช้งานอ่านรายละเอียดของ License Terms เมื่อผู้ใช้งานอ่านเรียบร้อย ให้ผู้ใช้งานทำการคลิกที่ I have read and accept the license terms และคลิก Next > ต่อไปเรื่อยๆ เพื่อเข้าสู่ขั้นตอนตำแหน่งการติดตั้งโปรแกรม ดังรูปที่ ก.4



รูปที่ ก. 4 License Terms

5. เมื่อเข้าสู่ขั้นตอนการติดตั้งตำแหน่งของโปรแกรม ให้ผู้ใช้งานคลิก Install เพื่อติดตั้งโปรแกรม ให้ผู้ใช้งานรอสักครู่ จนกระทั่งโปรแกรมทำการติดตั้งจนเสร็จสมบูรณ์



รูปที่ ก. 5 ตำแหน่งการติดตั้งโปรแกรม Microsoft Visual Studio 2010 Express

6. เมื่อโปรแกรมทำการติดตั้งเสร็จสมบูรณ์แล้ว ผู้ใช้งานสามารถใช้ Microsoft Visual Studio 2010 Express ในการพัฒนาโปรแกรมต่อไปได้

ข. การติดตั้ง TaoOpenGL Library

1.การติดตั้ง TaoOpenGL Framework สามารถดาวน์โหลดโปรแกรมได้ที่ http://sourceforge.net/projects/taoframework/

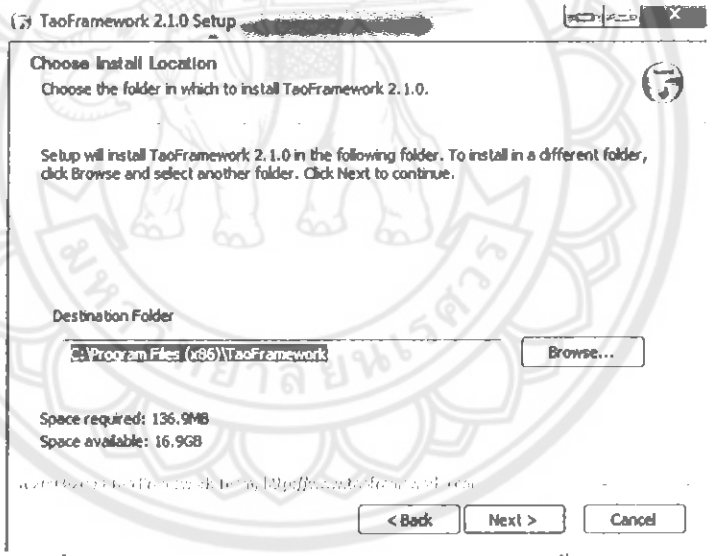
2. ดับเบิ้ลคลิก ไฟล์ที่ดาวโหลดมาดังรูปที่ ข.1

รูปที่ ข. 1 การติดตั้ง Library TaoOpenGL

3. หลังจากเสร็จข้อ 2 แล้ว จะได้หน้าต่างดังรูปที่ ข.2 ให้คลิก Next ไปเรื่อยๆ

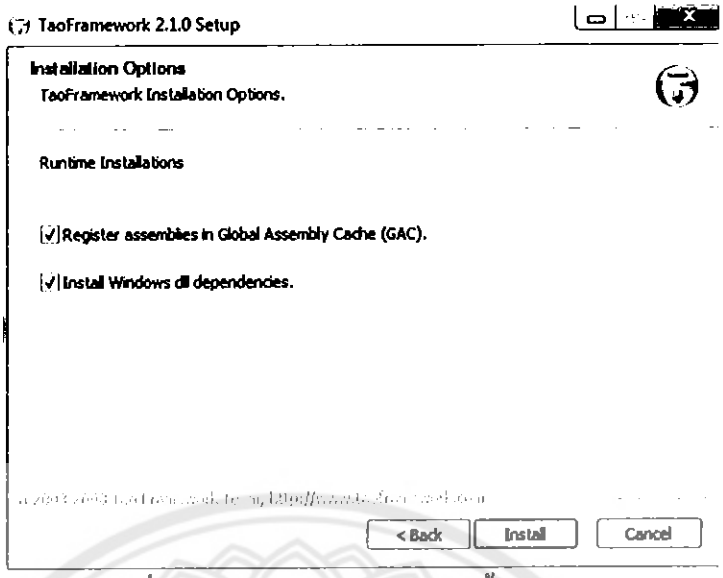
รูปที่ ข. 2 หน้าต่างแสดงการติดตั้งโปรแกรม

4. เมื่อถึงหน้าต่างดังรูปที่ ข.3 ให้เลือก Directory ที่ต้องการติดตั้งโปรแกรม **หลังจากนั้นคลิก** Next

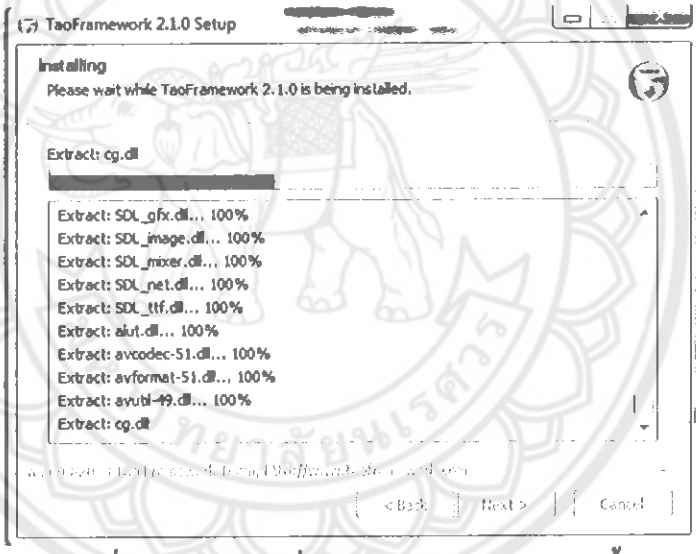


รูปที่ ข. 3 หน้าต่างเลือก Directory ที่ต้องการติดตั้งโปรแกรม

5. ให้กด Next มาเรื่อยๆจนถึงหน้าต่างสำหรับการติดตั้งโปรแกรม ดังรูปที่ ข.4 คลิก Install เพื่อทำการติดตั้ง หลักจากคลิก Install แล้วโปรแกรมจะทำการติดตั้งดังรูปที่ ข.5

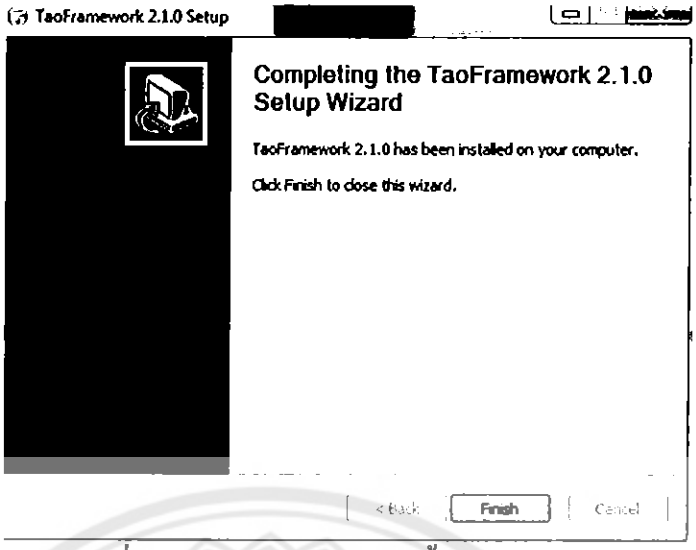


รูปที่ ข. 4 หน้าต่างสำหรับการติดตั้งโปรแกรม



รูปที่ ข. 5 หน้าต่างเมื่อโปรแกรมกำลังทำการติดตั้ง

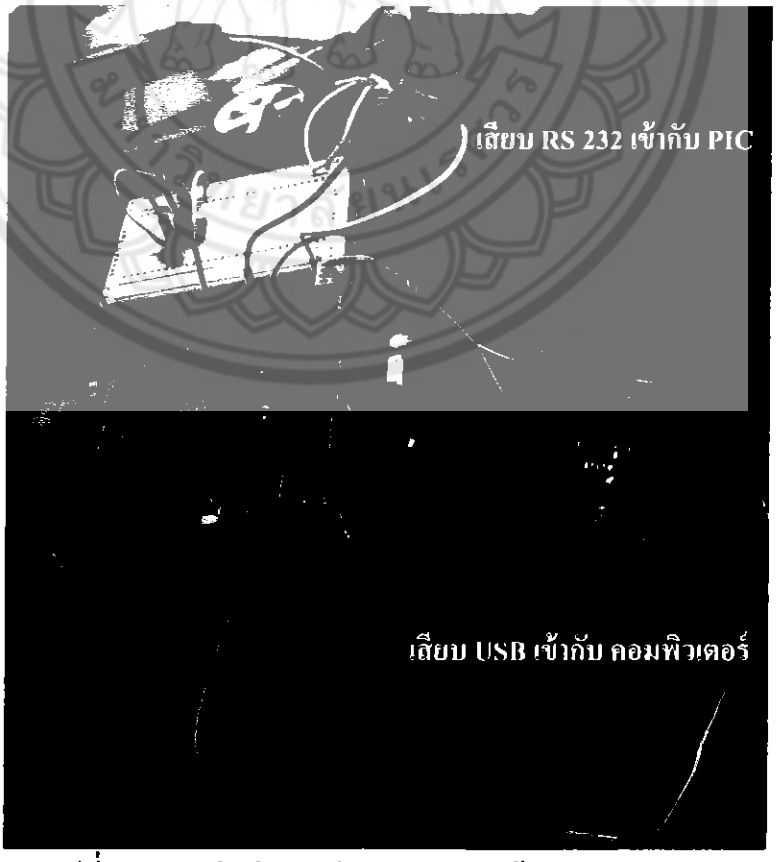
6. หลักจากติดตั้งเสร็จแล้วจะได้ดังรูป ข.6 ให้คลิก Finish เป็นการเสร็จสิ้นการติดตั้งโปรแกรม



รูปที่ ข. 6 หน้าต่างหลังจากติดตั้งโปรแกรมเสร็จ

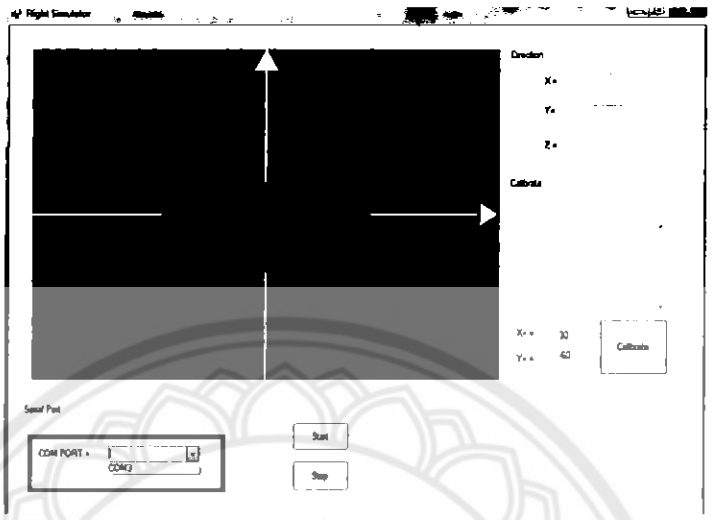
ค. คู่มือการใช้งานโปรแกรม Flight Simulator

1. เสียบ PORT RS232 เข้ากับบอร์ด PIC V3.0 16F877 และเสียบ PORT USB เข้ากับคอมพิวเตอร์ดังแสดงในรูปที่ ค.1



รูปที่ ค. 1 การติดต่อระหว่างระบบและหน้าจอแสดงผล GUI

2. เปิดโปรแกรม Flight Simulator คลิกที่ COM PORT เพื่อเลือก COM PORT สำหรับการติดต่อระบบกับหน้าจอแสดงผล GUI ดังรูปที่ ค.2 หลังจากนั้นคลิกที่ Start เพื่อเริ่มต้นการเชื่อมต่อ

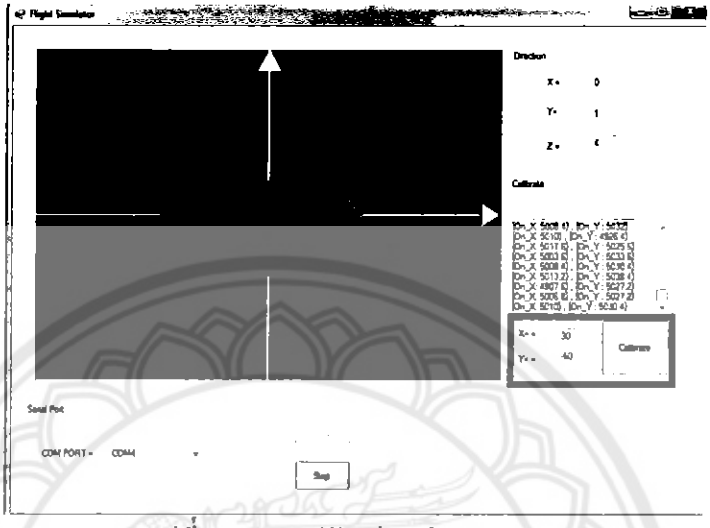


รูปที่ ค. 2 การเลือก COM PORT ที่ใช้สำหรับติดต่อระหว่าง GUI กับ ระบบ

3. เมื่อระบบกับ GUI ติดต่อกันแล้ว ภาพจำลอง 3 มิติในหน้าจอ GUI จะแสดงสถานะมุมเอียงเดียวกันกับระบบ ดังเสดงในรูปที่ ค.3

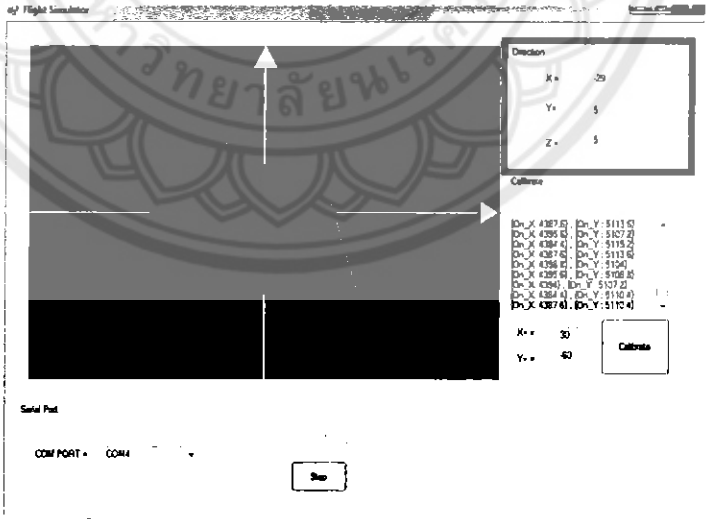
รูปที่ ค. 3 GUI กับระบบเมื่อติดต่อกันสำเร็จ

4. ให้ผู้ใช้วางระบบให้อยู่ในแนวระนาบ คือมีมุมเอียง 0 องศาทั้งในแนวแกน X และ Y เพื่อปรับเทียบค่า โดยพยายามใส่ค่าในช่องปรับเทียบค่าดังรูปที่ ค.4 พยาพยามใส่ให้ค่า On_X และ On_Y มีค่าเท่ากับ 5000



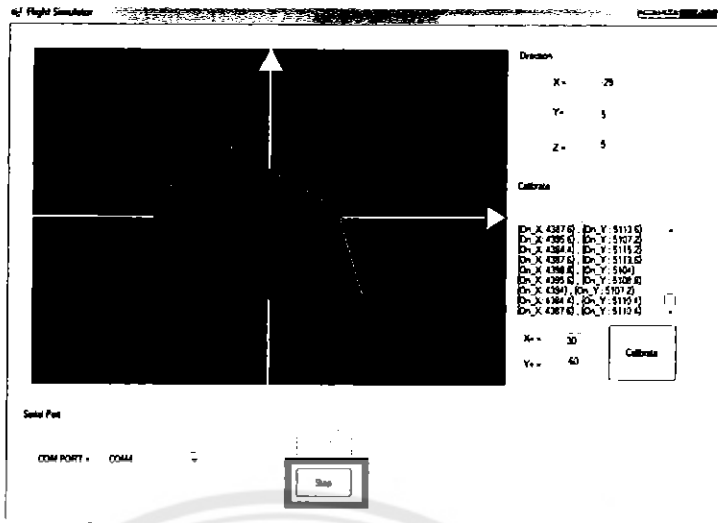
รูปที่ ค. 4 การปรับเทียบค่าของระบบ

5. ผู้ใช้สามารถอ่านค่ามุมเอียงต่างๆได้ในส่วนของ Direction ของโปรแกรม โดยค่ามุมเอียงมีหน่วยเป็นองศาดีกรี ดังแสดงในรูปที่ ค.5



รูปที่ ค. 5 การอ่านค่ามุมเอียงบนหน้าจอ GUI ของระบบ

6. เมื่อผู้ใช้ต้องการเลิกการใช้งานระบบ สามารถยกเลิกการติดต่อระหว่าง ระบบกับหน้าจอ GUI ได้โดยคลิกที่ปุ่ม Stop ดังแสดงในรูปที่ ค.6



รูปที่ ค. 6 ยกเลิกการติดต่อระหว่างระบบกับหน้าจอ GUI