

ชุดควบคุมอากาศยานไร้คนขับอัจฉริยะด้วยลีปโมชั่น

นางสาวพรกมล ชูชุมพร

นายรอยบุญ ลือพัศตรา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2558

Intelligent Drone Flight Controller by Leap Motion

Ms. Pornkamol Choochumporn

Mr. Royboon Luepaktra

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF BACHELOR OF COMPUTER ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY NORTH BANGKOK
ACADEMIC YEAR 2015

ปริญญานิพนธ์เรื่อง : ชุดควบคุมอากาศยานไร้คนขับอัจฉริยะด้วยลีปโมชั่น

ชื่อ : นางสาวพรกมล ชูชุมพร

นายรอยบุญ ลือพัศตรา

สาขาวิชา : วิศวกรรมคอมพิวเตอร์

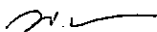
ภาควิชา : วิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะ : วิศวกรรมศาสตร์

อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.พงษ์ศักดิ์ กীরติวินทกร

ปีการศึกษา : 2558

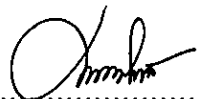
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ อนุมัติให้
ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์



(ผู้ช่วยศาสตราจารย์ ดร.นภค วิวัชรโกเศศ)

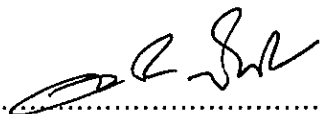
หัวหน้าภาควิชาวิศวกรรมไฟฟ้า

และคอมพิวเตอร์



(ผู้ช่วยศาสตราจารย์ ดร.พงษ์ศักดิ์ กীরติวินทกร)

ประธานกรรมการ



(ดร.ตनुชา ประเสริฐสม)

กรรมการ



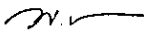
(อาจารย์โสภณ อภิรมย์วารการ)

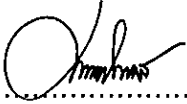
กรรมการ

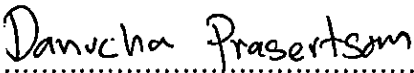
ลิขสิทธิ์ของภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ


Project Report Title : Intelligent Drone Flight Controller by Leap Motion
Name : Ms. Pornkamol Choochumporn
Mr. Royboon Luepaktra
Major Field : Computer Engineering
Department : Electrical and Computer Engineering
Faculty : Engineering
Project Advisor(s) : Asst. Prof. Dr. Phongsak Keeratiwintakorn
Academic Year : 2015

Accepted by the Faculty of Engineering, King Mongkut's University of Technology North Bangkok in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Engineering


..... Chairperson of Department of Electrical
(Asst. Prof. Dr. Nophadon Wiwatcharagoses) and Computer Engineering


..... Chairperson
(Asst. Prof. Dr. Phongsak Keeratiwintakorn)


..... Member
(Dr. Danucha Prasertsom)


..... Member
(Mr. Sophon Apiromworrakarn)

Copyright of the Department of Electrical and Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology North Bangkok

บทคัดย่อ

เนื่องจากในปัจจุบันมัลติโรเตอร์ (โดรน) ได้เป็นที่นิยมเพิ่มขึ้นอย่างมาก โดยได้มีการนำไปใช้งานด้านต่าง ๆ เช่น การถ่ายภาพทางอากาศและการสำรวจพื้นที่ ซึ่งการควบคุมการบินของมัลติโรเตอร์ (โดรน) แต่ละรุ่น ส่วนใหญ่เป็นการใช้วิทยุบังคับในการควบคุม ซึ่งวิทยุบังคับมีจำนวนช่องสัญญาณหลายช่องสำหรับควบคุมการบิน และการควบคุมการบินดังกล่าว ผู้ใช้งานหรือผู้ควบคุมการบินจำเป็นต้องมีทักษะในการบังคับมัลติโรเตอร์ (โดรน) ให้เคลื่อนที่ไปในลักษณะทิศทางที่ต้องการ ซึ่งผู้ควบคุมการบินต้องมีการฝึกฝนเป็นระยะเวลานาน เพื่อให้เกิดทักษะและความชำนาญในการควบคุมการบินของมัลติโรเตอร์ (โดรน)

ทางผู้จัดทำจึงได้พัฒนาอุปกรณ์ที่ใช้สำหรับควบคุมการบินของมัลติโรเตอร์ (โดรน) แทนวิทยุบังคับ ซึ่งผู้ใช้งานสามารถเรียนรู้การใช้งานได้อย่างรวดเร็วและไม่จำเป็นต้องฝึกฝนเป็นระยะเวลานาน โดยใช้การควบคุมการบินของมัลติโรเตอร์ (โดรน) ด้วยการใช้ท่าทางและการเคลื่อนไหวของมือผู้ใช้งานด้วยอุปกรณ์ Leap Motion ซึ่งเป็นอุปกรณ์ตรวจจับท่าทางและการเคลื่อนไหวของมือและนิ้วมือ จากนั้นจึงนำไปประมวลผลด้วยโปรแกรมและไมโครคอนโทรลเลอร์เพื่อเปลี่ยนท่าทางและการเคลื่อนไหวของมือและนิ้วมือเป็นการควบคุมการบินแทน

ทำให้ผู้ควบคุมการบินสามารถเข้าใจวิธีการสำหรับควบคุมการบินของมัลติโรเตอร์ (โดรน) ได้ง่ายและรวดเร็ว ด้วยการใช้ท่าทาง การเคลื่อนไหวของมือและนิ้วมือ ซึ่งเป็นการเคลื่อนไหวโดยธรรมชาติของมนุษย์อยู่แล้ว ผู้ควบคุมการบินที่ใช้ชุดอุปกรณ์ที่พัฒนาขึ้น จึงไม่จำเป็นต้องใช้เวลาในการฝึกฝนนาน เพื่อให้เกิดทักษะการควบคุมการบินแบบที่ใช้วิทยุบังคับ

Abstract

As Multi-rotor is getting more popular in worldwide. Moreover, there are requirement to use it in many tasks such as photography, area exploration or hobbies activity especially any high risk tasks. The accuracy of Aviation is very important it could damage the equipment or accidents to user.

Therefore, Provider has realized that the ability to control the flight is important. Because each model has a different flight control which most of them need to used adequate radio channels to make it work properly. So, the user who is going to start using the multi-rotor should be well trained long enough. It may require a long period of training to be able to control multi-rotor aircraft and precisely in the desired direction. Therefore, The devices that can help to control a multi-rotor more easier was created. This device does not require training for a long time as it is equipped with a command from the user by hand which the movement of the hand is a gesture of human nature that everyone can do it. Also, it is easy to carry and convenient for using.

กิตติกรรมประกาศ

ขอกราบขอบพระคุณอาจารย์ที่ปรึกษาโครงการ ผู้ช่วยศาสตราจารย์ ดร.พงษ์ศักดิ์ กิริติวินกร ที่ได้ให้คำปรึกษาและคำแนะนำเกี่ยวกับการดำเนินงานของโครงการตลอดจนการดำเนินชีวิต จนโครงการนี้ได้ประสบความสำเร็จ เสร็จสมบูรณ์ได้

รวมถึงคณาจารย์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ที่ได้ให้ความรู้ในวิชาต่าง ๆ จนสามารถนำมาใช้จนโครงการนี้เสร็จสมบูรณ์ รวมทั้งสถาบันการศึกษามหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ที่ได้เอื้ออุปการะ สถานที่ ให้ได้ทำการเรียนรู้ ตลอดจนปฏิบัติงานจนสำเร็จลุล่วงไปด้วยดี

สุดท้ายนี้ผู้จัดทำกราบขอบพระคุณบิดามารดาและครอบครัว ตลอดจนเพื่อน ๆ ผู้เป็นแรงผลักดันและกำลังใจแก่ผู้จัดทำ

พรกมล ชูชุมพร

รอยบุญ ลือพัศตรา

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	จ
บทคัดย่อภาษาอังกฤษ	ฉ
กิตติกรรมประกาศ	ช
สารบัญตาราง	ญ
สารบัญภาพ	ฎ
บทที่ 1. บทนำ	1
1.1 ความเป็นมาและความสำคัญ	1
1.2 วัตถุประสงค์	1
1.3 ขอบเขตของโครงการ	1
1.4 ประโยชน์ที่คาดว่าจะได้รับ	2
1.5 แผนภาพรวมของระบบ	2
บทที่ 2. เอกสารและทฤษฎีที่เกี่ยวข้อง	4
2.1 การควบคุมทิศทางการบิน	4
2.2 ส่วนควบคุมภาคพื้นดิน	8
2.3 ส่วนควบคุมภาคอากาศ (อุปกรณ์บนลำสถิติโรเตอร์)	14
บทที่ 3. วิธีการดำเนินงาน	25
3.1 ระบบของส่วนควบคุมภาคพื้นดิน	25
3.2 ระบบของส่วนควบคุมภาคอากาศ	33
บทที่ 4. ผลการดำเนินงาน	53
4.1 ผลการดำเนินงานของส่วนควบคุมภาคพื้นดิน	53
4.2 ผลการดำเนินงานของส่วนควบคุมภาคอากาศ	64
บทที่ 5. สรุปผล ข้อเสนอแนะ และปัญหา	77
5.1 สรุปผล	77
5.2 ปัญหาที่พบ	77
5.3 ข้อเสนอแนะ	78
เอกสารอ้างอิง	79

สารบัญ (ต่อ)

ประวัติผู้แต่ง

หน้า
80

สารบัญตาราง

ตารางที่		หน้า
2-1	เปรียบเทียบรายละเอียดของ Arduino UNO กับ Arduino Due	18
3-1	คำสั่งที่ใช้งานสำหรับตรวจจับท่าทางและมุมการเคลื่อนที่ของมือผู้ใช้งาน	28
3-2	ค่าที่นำมาใช้งานโดยพิจารณาจากท่าทางที่แตกต่างกันของมือ	29
3-3	ค่าที่ใช้งานสำหรับตัวแปร X1 ถึง X7 ที่ใช้ในการส่งข้อมูลจากส่วนควบคุมภาคพื้นดิน	32
3-4	การเชื่อมต่อระหว่าง Wireless Datalink กับ Arduino	34
3-5	การเชื่อมต่อระหว่าง Arduino กับ Flight Controller	35
3-6	หน้าที่ของแต่ละ Input Channel สำหรับ Flight Controller	35
3-7	ความหมายของประเภทข้อมูลสำหรับตัวแปร X1 ถึง X7 ที่ใช้ในการส่งข้อมูล	40
3-8	ขั้นตอนการตรวจสอบข้อมูลขั้นที่ 3 (ตรวจสอบหาค่าที่เหมือนกันมากที่สุดในปีเฟอร์)	41
3-9	แสดงการเปลี่ยนแปลงค่าองศาของมือไปเป็นค่าสำหรับกำหนดให้กับสัญญาณ PWM ของมุม Roll	42
3-10	แสดงการเปลี่ยนแปลงค่าองศาของมือไปเป็นค่าสำหรับกำหนดให้กับสัญญาณ PWM ของมุม Pitch	43
3-11	แสดงการเปลี่ยนแปลงค่าองศาของมือไปเป็นค่าสำหรับกำหนดให้กับสัญญาณ PWM ของมุม Yaw	43
3-12	ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้สำหรับฟังก์ชัน failSafeStateFlightControl_0()	50
3-13	ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้สำหรับฟังก์ชัน failSafeStateFlightControl_1()	51
3-14	ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้สำหรับฟังก์ชัน failSafeStateFlightControl_2()	51
3-15	ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้สำหรับฟังก์ชัน failSafeStateFlightControl_3()	51

สารบัญภาพ

ภาพที่	หน้า
1-1 ภาพรวมของระบบสำหรับชุดควบคุมมัลติโรเตอร์ด้วยอุปกรณ์ Leap Motion	2
2-1 แกนการหมุนของมัลติโรเตอร์	5
2-2 การหมุนตามแกน Longitudinal Axis หรือการเคลื่อนที่รอบแกน Roll	6
2-3 การหมุนตามแกน Lateral Axis หรือการเคลื่อนที่รอบแกน Pitch	6
2-4 การหมุนตามแกน Vertical Axis หรือการเคลื่อนที่รอบแกน Yaw	7
2-5 ลักษณะของอุปกรณ์ Leap Motion	10
2-6 ลักษณะของอุปกรณ์ Wireless Datalink (Telemetry 915 MHz)	13
2-7 หน้าต่างโปรแกรม Mission Planner	14
2-8 ลักษณะของอุปกรณ์ Arduino UNO	16
2-9 แสดงตำแหน่ง External Power Supply ของอุปกรณ์ Arduino UNO	17
2-10 แสดงลักษณะสายไฟชนิด XT-60 to DC Jack	18
2-11 ลักษณะของอุปกรณ์ Flight Controller (ArduPilot Mega 2.6)	19
2-12 ลักษณะของอุปกรณ์ Frame S500	20
2-13 ลักษณะของอุปกรณ์ ใบพัด 12*4.5 Niron	21
2-14 ลักษณะของอุปกรณ์ Electronic Speed Controller ขนาด 30A	21
2-15 ลักษณะของอุปกรณ์ Brushless Motor รุ่น 4010 ขนาด 850 KV	22
2-16 ลักษณะของอุปกรณ์ Battery รุ่น Thunder Power 3 Cell 11.1V ขนาด 4300 mAh	23
2-17 ลักษณะของอุปกรณ์ GPS รุ่น Ublox NEO-7N	23
2-18 ลักษณะของอุปกรณ์ Power Modul รุ่น RCTimer for APM Max ขนาด 30V, 90A	24
3-1 ภาพรวมของฮาร์ดแวร์สำหรับส่วนควบคุมภาคพื้นดิน	26
3-2 ภาพรวมของระบบสำหรับส่วนควบคุมภาคพื้นดิน	27
3-3 รายชื่ออุปกรณ์ที่ใช้งานบนส่วนควบคุมภาคอากาศ	34
3-4 ขั้นตอนการทำงานทั้งหมดของระบบสำหรับส่วนควบคุมภาคอากาศ	37
3-5 ขั้นตอนการทำงานสำหรับรับข้อมูลจากส่วนควบคุมภาคพื้นดิน และตรวจสอบขั้นที่ 1 (ตรวจสอบความถูกต้องของข้อมูลในแต่ละตำแหน่ง)	39
3-6 ขั้นตอนการทำงานสำหรับควบคุมการบินของมัลติโรเตอร์ ในกรณีที่มีมือควบคุม ส่วนที่ 1	46

สารบัญภาพ (ต่อ)

ภาพที่	หน้า
3-7 ขั้นตอนการทำงานสำหรับควบคุมการบินของมัลติโรเตอร์ ในกรณีที่มีมือควบคุม ส่วนที่ 2	47
3-8 ขั้นตอนการทำงานของระบบสำหรับควบคุมการสร้างสัญญาณ PWM ในกรณีที่มีมือควบคุม	49
4-1 การเชื่อมต่อระหว่างอุปกรณ์ Leap Motion กับคอมพิวเตอร์	54
4-2 การเชื่อมต่อระหว่างอุปกรณ์ Wireless Datalink กับคอมพิวเตอร์	54
4-3 ทำท่าทางการแบมือโดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	55
4-4 ทำท่าทางการกำมือโดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	55
4-5 ทำท่าทางการกระดิกนิ้วชี้โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	56
4-6 ทำท่าทางการกระดิกนิ้วกลางโดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	56
4-7 ทำท่าทางการแบมือและเอียงมือทางด้านซ้าย โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	57
4-8 ทำท่าทางการแบมือและเอียงมือทางด้านขวา โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	57
4-9 ทำท่าทางการแบมือและเอียงมือทางด้านหน้า โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	58
4-10 ภาพแสดงท่าทางการแบมือและเอียงทางด้านหลัง โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	58
4-11 ทำท่าทางการแบมือและหมุนมือตามเข็มนาฬิกา โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	59
4-12 ทำท่าทางการแบมือและหมุนมือทวนเข็มนาฬิกา โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion	59
4-13 จอแสดงผลของ Desktop Application โดยแสดงข้อมูลที่ใช้งานจากท่าทางต่าง ๆ ของมือ ในกรณีที่มีมือผู้ใช้งานควบคุม	61

สารบัญภาพ (ต่อ)

ภาพที่	หน้า
4-14 จอแสดงผลของ Desktop Application โดยแสดงข้อมูลที่ใช้งานจากท่าทางต่าง ๆ ของมือ ในกรณีที่ไม่มีมือผู้ใช้งานควบคุม	61
4-15 โปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application โดยแสดงข้อมูลใน กรณีที่เริ่มต้นส่งข้อมูลและมีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion	63
4-16 โปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application โดยแสดงข้อมูลใน กรณีที่ยังไม่เริ่มต้นส่งข้อมูลและไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion	63
4-17 ตัวอย่างของชุดข้อมูลที่ส่งออกไปให้กับส่วนควบคุมภาคอากาศ โดยแสดงผล ผ่านส่วนOutput ของโปรแกรม Microsoft Visual Studio	64
4-18 การเชื่อมต่อระหว่าง Arduino UNO กับ Wireless Datalink (915 MHz)	65
4-19 การเชื่อมต่ออุปกรณ์ที่ใช้งานบนส่วนของมัลติโรเตอร์	66
4-20 การรับข้อมูลจากส่วนควบคุมภาคพื้นดินและตรวจสอบความถูกต้องใน ขั้นตอนที่ 1 (ตรวจสอบตำแหน่งข้อมูล) บน Serial Monitor ของ โปรแกรม Arduino Sketch	67
4-21 การตรวจสอบในขั้นตอนที่ 2 (ตรวจสอบความยาวของชุดข้อมูล) และ 3 (ตรวจสอบค่าที่เหมือนกันมากที่สุดในปีเฟอร์) พร้อมแสดง ผลลัพธ์สุดท้ายบน Serial Monitor ของโปรแกรม Arduino Sketch	68
4-22 การหาผลลัพธ์สุดท้ายขณะมีการเปลี่ยนแปลงระหว่างมีมือเป็นขณะไม่มีมือ บน Serial Monitor ของโปรแกรม Arduino Sketch	69
4-23 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Stand by บน Serial Monitor ของโปรแกรม Arduino Sketch	70
4-24 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Arm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch	70
4-25 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Control Flight บน Serial Monitor ของโปรแกรม Arduino Sketch	71
4-26 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Disarm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch	71

สารบัญภาพ (ต่อ)

ภาพที่	หน้า
4-27 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Stand by บน Serial Monitor ของโปรแกรม Arduino Sketch	72
4-28 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Arm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch	73
4-29 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Control Flight บน Serial Monitor ของโปรแกรม Arduino Sketch	73
4-30 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Disarm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch	74
4-31 ค่าของสัญญาณ PWM ที่สร้างขึ้น บน Serial Monitor ของโปรแกรม Arduino Sketch และตรวจสอบค่าของสัญญาณ PWM ผ่านโปรแกรม Mission Planner	75
4-32 ค่าของสัญญาณ PWM ทั้ง 5 Chanel ที่สร้างขึ้นพร้อมระบุขั้นตอนการทำงาน ขณะนั้นบน Serial Monitor ของโปรแกรม Arduino Sketch	76
4-33 โปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application ที่รับค่ามาจาก ส่วนควบคุมภาคอากาศ	76

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากในปัจจุบัน มัลติโรเตอร์ได้เป็นที่นิยมเพิ่มขึ้นอย่างรวดเร็วหลาย รวมถึงความต้องการนำมัลติโรเตอร์ไปใช้งานในด้านต่าง ๆ โดยเฉพาะงานที่มีความเสี่ยงสูงหรือใช้ในพื้นที่ที่ไม่เอื้ออำนวยต่อมนุษย์ที่จะเข้าไปทำการสำรวจ ซึ่งความแม่นยำในการบินถือเป็นเรื่องที่สำคัญมาก และการควบคุมการบินในแต่ละรุ่นย่อมมีการควบคุมการบินที่แตกต่างกันออกไป โดยส่วนใหญ่แล้วต้องใช้วิทยุบังคับที่มีจำนวนช่องสัญญาณเป็นจำนวนมาก เพื่อให้สามารถควบคุมการบินของมัลติโรเตอร์ได้ ซึ่งทำให้ผู้ใช้งานที่จะเริ่มนำมัลติโรเตอร์มาใช้ต้องมีการฝึกบินมาก่อนพอสมควร โดยอาจต้องใช้ระยะเวลาในการฝึกฝนนานกว่าที่จะสามารถควบคุมมัลติโรเตอร์ให้บินได้อย่างแม่นยำและไปในทิศทางที่ต้องการได้

1.2 วัตถุประสงค์

เพื่อออกแบบและพัฒนาระบบการควบคุมอากาศยานไร้คนขับให้สามารถทำการควบคุมการบินโดยสั่งการใช้งานด้วยมือ แทนการถืออุปกรณ์รีโมทควบคุมตลอดเวลา ซึ่งสะดวกสบายต่อการใช้งานในด้านต่าง ๆ และทำให้การเริ่มต้นเพื่อฝึกควบคุมการบินของมัลติโรเตอร์นั้นง่ายยิ่งขึ้นและใช้เวลาสั้นลง

1.3 ขอบเขตของโครงการ

สร้างระบบการควบคุมอากาศยานไร้คนขับด้วยอุปกรณ์ Leap Motion เพื่อให้สามารถควบคุมการบินได้นิ่ง แม่นยำและสะดวกสบายต่อการใช้งานมากยิ่งขึ้น โดยจัดทำอุปกรณ์ที่มีความสามารถดังต่อไปนี้

1.3.1 อุปกรณ์สามารถส่งคำสั่งเพื่อใช้งานแทนวิทยุบังคับโดยทั่วไปได้

1.3.2 อุปกรณ์สามารถสื่อสารกับส่วนควบคุมการบินผ่านทางไวเลส ดาต้าลิงค์ได้

1.3.3 สามารถควบคุมการบินในท่าทางพื้นฐาน ได้แก่ การบินขึ้น-ลง การเดินทาง-ถอยหลัง การเลี้ยวซ้าย-เลี้ยวขวา และหมุนตามเข็มนาฬิกา-หมุนทวนเข็มนาฬิกาได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้เรียนรู้เกี่ยวกับส่วนประกอบต่าง ๆ ของมัลติโรเตอร์ ตั้งแต่อุปกรณ์แต่ละส่วน การควบคุมด้วยวิทยุบังคับ และการติดตั้งการใช้งานมัลติโรเตอร์

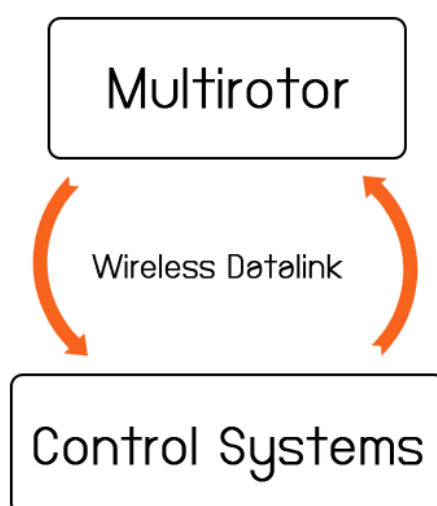
1.4.2 เข้าใจถึงหลักการทำงานของ Leap motion และสามารถใช้งาน Leap motion ได้อย่างถูกต้อง

1.4.3 ได้เรียนรู้การเขียน โปรแกรมสำหรับควบคุมไมโครคอนโทรลเลอร์ซึ่งเป็นส่วนสำหรับควบคุมการบิน

1.4.4 ได้เรียนรู้รูปแบบการสื่อสารของส่วนควบคุมการบินกับอุปกรณ์ควบคุมการบินที่ภาคพื้นดิน

1.4.5 ได้เรียนรู้การส่งข้อมูลด้วยรูปแบบไวลเลส คาดาลิงค์

1.5 แผนภาพรวมของระบบ



ภาพที่ 1-1 ภาพรวมของระบบ

จากภาพที่ 1-1 จะเห็นได้ว่าระบบถูกแบ่งออกเป็น 2 ส่วนหลักได้แก่

1. Multicopter หรือเรียกอีกอย่างว่า Aircraft Control Systems (ACS) ซึ่งมีชื่อเรียกเป็นภาษาไทยคือ ส่วนควบคุมภาคอากาศ โดยมีองค์ประกอบหลักด้วยกัน 2 ส่วนได้แก่ 1. ชุดควบคุมการบินของมอเตอร์โดยมีอุปกรณ์ที่สำคัญได้แก่ Flight Controller, ESC, Motor, Battery, Propeller เป็นต้น ในส่วนนี้จะมีหน้าที่หลักในการควบคุมการบินของมอเตอร์ในส่วนความเร็วรอบของมอเตอร์ทั้ง 4 ตัวเพื่อให้สามารถเคลื่อนที่ไปในทิศทางที่ผู้ใช้งานต้องการได้ 2. ชุดประมวลผลข้อมูลเพื่อสร้างสัญญาณ PWM โดยมีอุปกรณ์ที่สำคัญได้แก่ Arduino UNO, 915MHz Telemetry (Tx/Rx) ในส่วนนี้จะมีหน้าที่หลักในการประมวลผลค่าของมุม Roll, Pitch, Yaw ที่ส่วนควบคุมภาคพื้นดินส่งขึ้นมา และนำไปสร้างเป็นสัญญาณ PWM เพื่อป้อนเป็นสัญญาณ Input ให้กับ Flight Controller ในการจัดการกับความเร็วรอบของมอเตอร์ทั้ง 4 ตัวเพื่อให้เกิดการเคลื่อนที่ตามที่ต้องการ

2. Control Systems หรือเรียกอีกอย่างว่า Ground Control Systems (GCS) ซึ่งมีชื่อเรียกเป็นภาษาไทยคือ ส่วนควบคุมภาคพื้นดิน โดยมีองค์ประกอบหลักได้แก่ 1. Leap Motion ทำหน้าที่ในการตรวจจับลักษณะท่าทางของมือผู้ใช้งาน พร้อมทั้งวัดค่าของมุม Roll, Pitch, Yaw ของมือผู้ใช้งานขณะที่มือผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion 2. Notebook ทำหน้าที่ประมวลผลท่าทางของมือและค่าของมุม Roll, Pitch, Yaw ให้ออกมาเป็นตัวเลขในกลุ่มที่ต้องการและส่งออกให้กับส่วนควบคุมภาคอากาศ 3. 915MHz Telemetry เป็นอุปกรณ์สำหรับรับข้อมูลบางส่วนจากส่วนควบคุมภาคอากาศมาแสดงบน Desktop Application และส่งข้อมูลที่ Notebook ประมวลผลเสร็จแล้วไปให้กับส่วนควบคุมภาคอากาศ

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

การควบคุมการบินของมัลติโรเตอร์สามารถแบ่งส่วนสำคัญออกเป็น 3 ส่วนหลักได้แก่

1. การควบคุมทิศทางการบิน การเคลื่อนที่ในทิศทางต่าง ๆ ของมัลติโรเตอร์นั้น มีองค์ประกอบที่สำคัญด้วยกัน 3 ค่า ได้แก่ค่ามุมของ Roll, Pitch และ Yaw จะส่งผลต่อการเคลื่อนที่ไปทางด้านซ้าย-ขวา เดินหน้า-ถอยหลัง หมุนตัวลำทวนเข็ม-ตามเข็ม นอกจากนั้นยังมีท่าที่สำคัญได้แก่ การกำมือสำหรับสั่งเริ่มหรือหยุดการทำงานของมอเตอร์ และท่ากระดิกนิ้วชี้เพื่อความสูง กระดิกนิ้วกลางเพื่อลดความสูง

2. ส่วนควบคุมภาคพื้นดิน มีหน้าที่หลักในการนำค่ามุมของ Roll, Pitch และ Yaw รวมไปถึงท่าทางต่าง ๆ มาประมวลผลเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ แล้วจึงส่งไปให้กับส่วนควบคุมภาคอากาศ

3. ส่วนควบคุมภาคอากาศ รับผลลัพธ์จากส่วนควบคุมภาคพื้นดิน และนำมาประมวลผลอีกรอบเพื่อสร้างสัญญาณ PWM ให้กับ Flight Controller สำหรับนำไปจัดการควบคุมความเร็วรอบของมอเตอร์ทั้ง 4 ตัว เพื่อให้เกิดการเคลื่อนที่ตามที่ต้องการของผู้ใช้งาน

2.1 การควบคุมทิศทางการบิน

มีองค์ประกอบหลักที่สำคัญ 3 อย่างด้วยกันได้แก่ เซ็นเซอร์ตรวจจับการเคลื่อนไหวของมือ การเคลื่อนที่ของมัลติโรเตอร์ และท่าทางสำหรับการเคลื่อนที่ของมัลติโรเตอร์

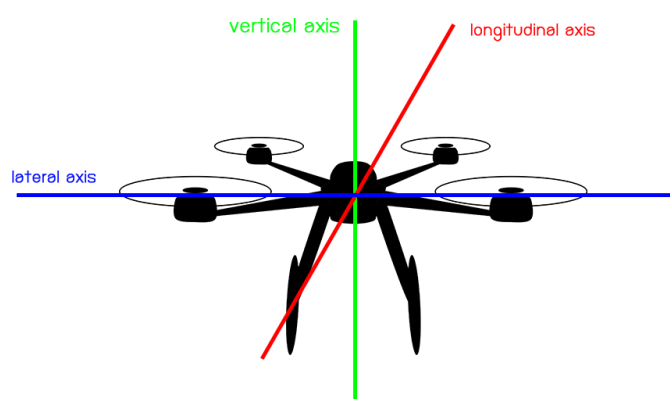
2.1.1 เซ็นเซอร์ตรวจจับการเคลื่อนไหวของมือ

เป็นเซ็นเซอร์อินฟราเรดมีลักษณะสีแดงอยู่ในอุปกรณ์ Leap Motion โดยจะมีเซ็นเซอร์อยู่ทั้งหมด 3 ตัว ซึ่งมีหน้าที่หลักในการตรวจจับการเคลื่อนไหวของมือ รวมไปถึงท่าทางต่าง ๆ ของมือผู้ใช้งาน โดยเราสามารถเขียนโปรแกรมเชื่อมต่อกับอุปกรณ์ Leap Motion เพื่อดึงค่าที่ต้องการได้ โดยในโครงงานของเราได้ทำการเขียนโปรแกรมเพื่อตรวจสอบ 1. ค่าของมุม Roll, Pitch, Yaw เพื่อใช้สำหรับวิเคราะห์การเคลื่อนที่ของมือเพื่อนำไปควบคุมการเคลื่อนที่ของมัลติโรเตอร์ให้สามารถเคลื่อนที่ได้ตามที่ต้องการ 2. ท่าทางการกำมือ เพื่อใช้สำหรับสั่ง Arm Motor (มอเตอร์เริ่มทำงาน),

Disarm Motor (มอเตอร์หยุดทำงาน) ทำทางการกระดิกนิ้วชี้ นิ้วกลาง เพื่อใช้สำหรับเพิ่มความสูง ลดความสูงตามลำดับ เซ็นเซอร์สำหรับตรวจจับนี้จะไม่สามารถทำงานได้ หรือทำงานได้ไม่ดีเมื่อเจอกับแสงที่มีรังสีอินฟราเรด เช่น แสงอาทิตย์

2.1.2 การเคลื่อนที่ของมัลติโรเตอร์

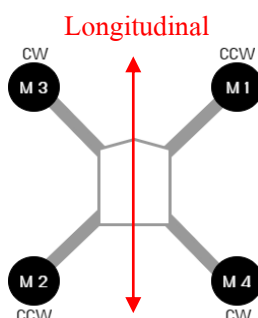
แบ่งออกเป็น 3 แกนหลักได้แก่ Longitudinal Axis, Vertical Axis และ Lateral Axis



ภาพที่ 2-1 แกนการหมุนของมัลติโรเตอร์

1. แกน Longitudinal Axis

เป็นแกนตามยาวตั้งแต่หัวไปจรดหางของตัวลำมัลติโรเตอร์ และการหมุนหรือเคลื่อนที่ของลำตัวรอบแกนนี้เราเรียกว่าการเคลื่อนที่รอบแกน “Roll” หรือแบงก์ (Bank) การทำให้เกิดการหมุนรอบแกนนี้ทำได้โดย Ailerons เป็นตัวสร้างความแตกต่างระหว่างแรงยก (Lift) ของปีก-มัลติโรเตอร์ทั้งสองข้าง โดยเกิด Lift ที่ไม่เท่ากัน จึงเป็นสาเหตุให้เกิดการหมุนของแกนนี้ และทำให้มัลติโรเตอร์เอียงไปทางใดทางหนึ่งตามที่เรบังคับ Ailerons ให้เกิด Lift ที่แตกต่างกัน อาการนี้เรียกว่า Roll หรือ Bank เช่น เราต้องการเอียงเครื่องไปทางด้านซ้าย สามารถทำได้โดยการสร้าง Lift ที่ปีกด้านขวาให้มีมากกว่าทางด้านซ้าย (มอเตอร์ M2 และ M3 จะหมุนช้าลง) ก็จะเกิดการหมุนของแกนนี้ทำให้มัลติโรเตอร์เอียงไปทางซ้าย เป็นต้น

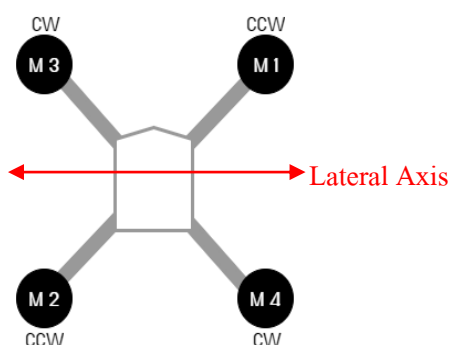


ภาพที่ 2-2 การหมุนตามแกน Longitudinal Axis หรือการเคลื่อนที่รอบแกน Roll

[http://antodominic.com/beagle-copter/beagle_files/YQDJVGs.png]

2. แกน Lateral Axis

เป็นแกนขวางจากปีกข้างหนึ่งไปยังปีกอีกข้างหนึ่งของตัวลำมัลติโรเตอร์ และการหมุนหรือเคลื่อนลำตัวรอบจุดนี้เราเรียกว่าการเคลื่อนที่รอบแกน “Pitch” ซึ่งเกิดจากผลของการกระทำที่อีเลเวเตอร์ (Elevator) เมื่อเกิดการหมุนของแกนนี้จะทำให้หัวของมัลติโรเตอร์ยกขึ้น (Pitch Up) หรือกดลง (Pitch Down) เป็นผลให้มัลติโรเตอร์เดินไปด้านหน้าหรือด้านหลัง เช่น เราต้องการนำเครื่องไปด้านหน้า สามารถทำได้โดยลดความเร็วมอเตอร์ M1 และ M3 ลง

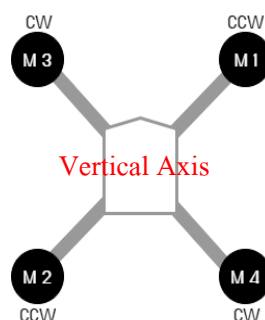


ภาพที่ 2-3 การหมุนตามแกน Lateral Axis หรือการเคลื่อนที่รอบแกน Pitch

[http://antodominic.com/beagle-copter/beagle_files/YQDJVGs.png]

3. แกน Vertical Axis

เป็นแกนในแนวตั้งที่ผ่านจุดที่เรียกว่า center of gravity และการหมุนหรือเคลื่อนที่ของลำตัวรอบจุดนี้ เราเรียกว่าการเคลื่อนที่รอบแกน “Yaw” ซึ่งเป็นผลมาจากรัดเดอร์ (Rudder) เมื่อมีการหมุนของแกนนี้จะทำให้หัวของมัลติโรเตอร์หมุนไปทางซ้ายหรือขวา การควบคุมจะมาจากการใช้ Pedal Rudder โดยใช้เท้าถีบซ้ายหรือขวาเพื่อบังคับ Rudder ของเครื่องบินอีกทีหนึ่ง เช่น เราต้องการนำเครื่องหมุนไปทางซ้าย สามารถทำได้โดยลดความเร็วมอเตอร์ M3 และ M4 ลง



ภาพที่ 2-4 การหมุนตามแกน Vertical Axis หรือการเคลื่อนที่รอบแกน Yaw

[http://antodominic.com/beagle-copter/beagle_files/YQDJVGs.png]

2.1.3 ท่าทางการเคลื่อนที่ของมัลติโรเตอร์

มีท่าทางที่สำคัญสำหรับควบคุมการเคลื่อนที่ของมัลติโรเตอร์ทั้งหมด 9 ท่าทางด้วยกัน ประกอบด้วย ท่าทางการกำมือ ท่าทางการกระดิกนิ้วชี้ ท่าทางการกระดิกนิ้วกลาง ท่าทางการแบมือ แล้วเอียงไปทางด้านซ้าย ท่าทางการแบมือแล้วเอียงไปทางด้านขวา ท่าทางการแบมือแล้วเอียงไปทางด้านหน้า ท่าทางการแบมือแล้วเอียงไปทางด้านหลัง ท่าทางการแบมือแล้วหมุนไปในทิศวนเข็มนาฬิกา และท่าทางการแบมือแล้วหมุนไปในทิศตามเข็มนาฬิกา

1. ท่าทางการกำมือ

เป็นท่าทางที่ใช้สำหรับควบคุมการสั่งให้มอเตอร์เริ่มทำงาน (Arm Motor) หรือเป็นการสั่งให้มอเตอร์หยุดทำงาน (Disarm Motor)

2. ท่าทางการกระดิกนิ้วชี้

เป็นท่าทางที่ใช้สำหรับควบคุมการเคลื่อนที่ขึ้นไปด้านบนเพื่อเพิ่มความสูงของมัลติโรเตอร์ ขณะที่กำลังทำการบินอยู่ในอากาศ

3. ทำทางการกระดิกนี้่กลาง

เป็นท่าทางที่ใช้สำหรับควบคุมการเคลื่อนที่ลงด้านล่างเพื่อลดความสูงของมัลติโรเตอร์ ขณะที่กำลังทำการบินอยู่ในอากาศ

4. ทำทางการแบมือแล้วเอียงไปทางด้านซ้าย

เป็นท่าทางที่ใช้สำหรับควบคุมการเคลื่อนที่ของมัลติโรเตอร์ให้มีการเคลื่อนที่ไปทางด้านซ้ายของตัวลำ ขณะที่กำลังทำการบินอยู่ในอากาศ

5. ทำทางการแบมือแล้วเอียงไปทางด้านขวา

เป็นท่าทางที่ใช้สำหรับควบคุมการเคลื่อนที่ของมัลติโรเตอร์ให้มีการเคลื่อนที่ไปทางด้านขวาของตัวลำ ขณะที่กำลังทำการบินอยู่ในอากาศ

6. ทำทางการแบมือแล้วเอียงไปทางด้านหน้า

เป็นท่าทางที่ใช้สำหรับควบคุมการเคลื่อนที่ของมัลติโรเตอร์ให้มีการเคลื่อนที่ไปทางด้านหน้าของตัวลำ ขณะที่กำลังทำการบินอยู่ในอากาศ

7. ทำทางการแบมือแล้วเอียงไปทางด้านหลัง

เป็นท่าทางที่ใช้สำหรับควบคุมการเคลื่อนที่ของมัลติโรเตอร์ให้มีการเคลื่อนที่ไปทางด้านหลังของตัวลำ ขณะที่กำลังทำการบินอยู่ในอากาศ

8. ทำทางการแบมือแล้วหมุนไปในทิศทวนเข็มนาฬิกา

เป็นท่าทางที่ใช้สำหรับควบคุมการหมุนของตัวลำมัลติโรเตอร์ให้มีการหมุนไปในทิศทวนเข็มนาฬิกา หรือเป็นการหมุนตัวลำไปทางด้านซ้ายขณะที่กำลังทำการบินอยู่ในอากาศ

9. ทำทางการแบมือแล้วหมุนไปในทิศตามเข็มนาฬิกา

เป็นท่าทางที่ใช้สำหรับควบคุมการหมุนของตัวลำมัลติโรเตอร์ให้มีการหมุนไปในทิศตามเข็มนาฬิกา หรือเป็นการหมุนตัวลำไปทางด้านขวาขณะที่กำลังทำการบินอยู่ในอากาศ

2.2 ส่วนควบคุมภาคพื้นดิน

หน้าที่สำหรับส่วนควบคุมภาคพื้นดินคือ รับข้อมูลได้แก่ค่าของมุม Roll, Pitch, Yaw ซึ่งจะพิจารณาในหน่วยขององศา ที่มีค่าตั้งแต่ -180 ถึง 180 องศา และตรวจจับท่าทางต่าง ๆ จากผู้ควบคุมการบินด้วยอุปกรณ์ Leap Motion ซึ่งค่าเหล่านี้จะมีลักษณะเป็นตัวเลขตามที่กำหนดไว้เพื่อนำไปทำการประมวลผลแล้วจึงส่งข้อมูลไปที่ไมโครคอนโทรลเลอร์ซึ่งอยู่ในส่วนควบคุมภาคอากาศ โดยผ่านอุปกรณ์ Wireless Datalink และสำหรับโปรแกรม Mission Planner ใช้สำหรับตั้งค่าเบื้องต้น

ให้แก่อุปกรณ์ Flight Controller ก่อนขึ้นบิน โดยวิธีการตั้งค่าสามารถศึกษาได้จากเว็บไซต์ของ Mission Planner โดยตรง นอกจากนั้นยังใช้สำหรับสังเกตค่าของสัญญาณ PWM ที่ไมโครคอนโทรลเลอร์ป้อนให้กับ Flight Controller ว่าแต่ละ Input Channel สามารถใช้งานได้หรือไม่

ลักษณะการเชื่อมต่อเริ่มจากนำอุปกรณ์ Leap Motion เชื่อมต่อเข้ากับคอมพิวเตอร์ผ่านทาง USB Portable และเชื่อมต่ออุปกรณ์ Wireless Datalink ซึ่งในที่นี้คือ 915MHz Telemetry เข้ากับคอมพิวเตอร์ผ่านทาง USB Portable และสำหรับคอมพิวเตอร์ต้องมีการลง SDK Leap Motion เพื่อให้สามารถใช้งานอุปกรณ์ Leap Motion ในการตรวจจับท่าทางต่าง ๆ ของมือผู้ใช้งานได้

ส่วนควบคุมภาคพื้นดินมีองค์ประกอบที่สำคัญได้แก่ 1. Leap Motion 2. โปรแกรมสำหรับประมวลผลข้อมูลจาก Leap Motion 3. Wireless Datalink และ 4. โปรแกรม Mission Planner

2.2.1 Leap Motion

Leap Motion มีลักษณะเป็นแท่งสี่เหลี่ยมผืนผ้าดังแสดงในภาพที่ 2-5 และรองรับการใช้งานทั้งบนระบบ Mac และ Windows โดยในส่วนการทำงานของ Leap Motion นั้น บริเวณด้านหน้า (สีดำ) จะมีเซ็นเซอร์อินฟราเรดสีแดง 3 ตำแหน่งยื่นขึ้นมาแบบ 3 มิติ ครอบคลุมพื้นที่ 150 องศา และสามารถตรวจจับการเคลื่อนไหวของวัตถุได้ในระยะประมาณ 1 นิ้วถึง 2 ฟุต

Leap Motion เป็นอุปกรณ์ที่เหมาะสมสำหรับใช้ในการควบคุมการเคลื่อนที่ของ- มัลติโรเตอร์ด้วยมือแทนการใช้งานวิทยุบังคับ เนื่องจากอุปกรณ์ Leap Motion ถูกพัฒนาขึ้นมาเพื่อตรวจจับการเคลื่อนไหวของมือโดยเฉพาะ และยังสามารถตรวจจับท่าทางต่าง ๆ ของมือได้จึงมีความเหมาะสมอย่างมากกับการนำมาใช้งานในโครงการนี้

การนำไปใช้ประยุกต์ใช้งานของอุปกรณ์ Leap Motion สามารถใช้งานแทนอุปกรณ์ต่าง ๆ ได้ เช่น การใช้มือควบคุมลูกศรบนระบบ Window หรือ Mac แทนการใช้เมาส์หรือมัลติทัชบนเครื่อง Notebook เป็นต้น

ลักษณะการตรวจจับตำแหน่งแบบสามมิติ คือ การอ้างอิงตามแกนแนวนอน (X) แกนความสูง (Y) และแกนแนวขวาง (Z) โดย Leap Motion จะมีจุด origin อยู่ตรงกลางของตัวอุปกรณ์ ตรวจจับท่าทางได้หลายแบบหลายทิศทาง เช่น วาดนิ้วเป็นวงกลมตามหรือทวนเข็มนาฬิกา (Circle) การแตะ (Tap) การเลื่อนหรือปัด (Scroll, Swipe) แปะหรือกางมือ (Open, Close Hand) เชิดมือขึ้นหรือคว่ำกดมือลง (Inline Upward/Downward) เป็นต้น

สถานะการทำงานของอุปกรณ์ Leap Motion มี 4 สถานะ ดังนี้

1. สีเขียว : ทำงานได้ตามปกติ

2. สีเหลือง : บริเวณรอบ ๆ controller มีแสงจ้า ซึ่งรบกวนการทำงานของ Leap Motion โดยจะลดความแม่นยำในการตรวจสอบลงไปบ้าง ซึ่งจะเปลี่ยนเป็นการทำงานแบบ Robust Mode

3. สีดำ : controller ถูกปิดการทำงานอยู่หรือ detect device ไม่พบ

4. สีแดง : มีฝุ่นหรือสิ่งสกปรกเกาะอยู่บนหน้ากระจก โดย Leap Motion จะเตือนให้เราเช็ดออก หรือมีแสงจ้ามาก ๆ ส่องเข้าด้านหน้าของเซ็นเซอร์โดยตรง ซึ่งทำให้ใช้งานไม่ได้ตามปกติ

ข้อดีของอุปกรณ์ Leap motion เนื่องจากเป็นเทคโนโลยีใหม่ที่มีข้อดีหลาย ๆ อย่าง เช่น ราคาไม่แพงมาก มีขนาดเล็ก พกพาสะดวก สามารถใช้กับ PC หรือ Notebook ในระบบ Windows และ Mac ได้ สามารถนำไปใช้งานได้หลายรูปแบบ ทั้งในด้านวิชาการและบันเทิง และยังไม่ต้องใช้ระยะเวลาในการฝึกฝนนานสำหรับการใช้งานประเภทต่าง ๆ

ข้อเสียของอุปกรณ์ Leap motion นอกจากจะมีข้อดีแล้ว ก็ยังมีข้อเสียคือ จากการใช้ Infrared ในการตรวจจับ ระยะจึงค่อนข้างจำกัดอยู่ที่ประมาณ 1 ฟุตรอบ ๆ อุปกรณ์ รวมถึงการแพ้แสงของ Leap motion ที่ต้องวางในที่ที่ไม่มีแสงมาสะท้อนมากเกินไป และข้อเสียจากการตรวจจับคือ หากเอานิ้วชิดกันจะเห็นเป็นนิ้วเดียวหรือหันหน้ามือไม่ตรงกับ sensor จะทำให้หานิ้วมือเราไม่เจอ



ภาพที่ 2-5 ลักษณะของอุปกรณ์ Leap Motion

[http://regmedia.co.uk/2013/07/22/leap_1.jpg]

2.2.2 โปรแกรมสำหรับประมวลผลข้อมูลจาก Leap Motion

โปรแกรมประมวลผลถูกพัฒนาด้วยภาษา C# โดยมีหน้าที่หลักได้แก่ 1. ประมวลผลค่าของมือผู้ใช้งาน 2. ประมวลผลค่าของมุม Roll, Pitch, Yaw 3. ประมวลผลท่าทางการกำมือ และ 4. ประมวลผลท่าทางการกระดิกนิ้วชี้และนิ้วกลาง

1. ประมวลผลค่าของมือผู้ใช้งาน

สำหรับการประมวลผลค่าของมือผู้ใช้งาน เป็นการเรียกใช้คำสั่งที่ทางผู้พัฒนาอุปกรณ์ Leap Motion ได้สร้างไว้เรียบร้อยแล้ว โดยสามารถศึกษาได้จากเว็บไซต์ของ Leap Motion ในส่วนของ C# Documentary โดยตรงได้ ซึ่งผลลัพธ์ที่ได้จากการใช้คำสั่งทางกลุ่มของเราได้เลือกให้มีค่าเป็นตัวเลขจำนวนเต็มระหว่าง 0 กับ 1 โดยแบ่งออกเป็นถ้าตรวจจับได้ว่าไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion จะให้ค่าเป็นตัวเลขจำนวนเต็มเท่ากับ 1 แต่ถ้าตรวจจับได้ว่าไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion จะให้ค่าเป็นตัวเลขจำนวนเต็มเท่ากับ 0 หลังจากได้ค่าที่ต้องการเรียบร้อยแล้วจะนำส่งสู่ส่วนควบคุมภาคอากาศ เพื่อนำไปพิจารณาว่าควรจะควบคุมการบินของมอเตอร์ด้วยโหมดการควบคุมแบบปกติ คือควบคุมการเคลื่อนที่ของมอเตอร์ตามการเคลื่อนไหวของมือผู้ใช้งานหรือโหมดการควบคุมแบบ FailSafe คือควบคุมการเคลื่อนที่ของมอเตอร์ให้บินอยู่นิ่งเฉยไว้เพื่อป้องกันการเสียหายถ้ากำลังทำการบินอยู่ หรือถ้าลงจอดแล้วหรือยังไม่ขึ้นบินให้ทำการหยุดการทำงานของมอเตอร์เพื่อป้องกันอันตรายต่าง ๆ

2. ประมวลผลค่าของมุม Roll, Pitch, Yaw

ขั้นตอนที่หนึ่ง ใช้งานคำสั่งที่ทางผู้พัฒนาอุปกรณ์ Leap Motion ได้สร้างไว้เรียบร้อยแล้ว โดยสามารถศึกษาได้จากเว็บไซต์ของ Leap Motion ในส่วนของ C# Documentary โดยตรงได้ ซึ่งผลลัพธ์ที่ได้ทั้งค่าของ Roll, Pitch, Yaw ในตอนแรกจะอยู่ในหน่วยของเรเดียน (rad.) และทางกลุ่มของเราได้ทำการแปลงให้อยู่ในหน่วยขององศา (degree) เพื่อให้สามารถเข้าใจมุมการเคลื่อนที่ของมอเตอร์ได้ง่ายขึ้น โดยค่ามุมทั้ง 3 ค่าดังกล่าวจะอยู่ในช่วงของ -180 ถึง 180 องศา แต่จะพิจารณาเฉพาะช่วงของมุม -100 ถึง 100 องศาเท่านั้น เนื่องจากเป็นองศาที่มนุษย์ทุกคนสามารถหมุนมือไปได้โดยไม่ฝืนธรรมชาติมากเกินไป ซึ่งค่าของมุมทั้ง 3 ที่อ่านได้ออกมานั้นจะทำการแยกเก็บไว้คนละตัวแปรกัน

ขั้นตอนที่สอง นำค่าของตัวแปรที่เก็บค่ามุมทั้ง 3 ค่ามาแยกพิจารณาทีละช่วงโดยแบ่งเป็น 9 ช่วงได้แก่

ช่วงที่ 1 พิจารณาตั้งแต่มุม -100 ถึง -81 องศา

ช่วงที่ 2 พิจารณาตั้งแต่มุม -80 ถึง -61 องศา

ช่วงที่ 3 พิจารณาตั้งแต่มุม -60 ถึง -41 องศา

ช่วงที่ 4 พิจารณาตั้งแต่มุม -40 ถึง -21 องศา

ช่วงที่ 5 พิจารณาตั้งแต่มุม -20 ถึง 20 องศา

ช่วงที่ 6 พิจารณาตั้งแต่มุม 21 ถึง 40 องศา

ช่วงที่ 7 พิจารณาตั้งแต่มุม 41 ถึง 60 องศา

ช่วงที่ 8 พิจารณาตั้งแต่มุม 61 ถึง 80 องศา

ช่วงที่ 9 พิจารณาตั้งแต่มุม 81 ถึง 100 องศา

หลังจากนั้นทำการใช้ตัวแปรอีก 3 ตัวสำหรับเก็บค่าของมุม Roll, Pitch, Yaw ตามลำดับที่ได้แปลงจากองศาเป็นช่วง 1 ถึง 9 เรียบร้อยแล้วซึ่งค่าเหล่านี้จะใช้สำหรับส่งออกไปให้กับส่วนควบคุมภาคอากาศเพื่อนำไปพิจารณาการเคลื่อนที่ทางด้านซ้าย-ขวา เดินหน้า-ถอยหลัง หมุนทวนเข็ม-ตามเข็ม ของมอเตอร์โรเตอร์อีกที

3. ประมวลผลท่าทางการกำมือ

สำหรับการประมวลผลท่าทางการกำมือ เป็นการเรียกใช้คำสั่งที่ทางผู้พัฒนาอุปกรณ์ Leap Motion ได้สร้างไว้เรียบร้อยแล้ว โดยสามารถศึกษาได้จากเว็บไซต์ของ Leap Motion ในส่วนของ C# Documentary โดยตรงได้ ซึ่งผลลัพธ์ที่ได้จากการใช้คำสั่งทางกลุ่มของเราได้เลือกให้มีค่าเป็นตัวเลขจำนวนเต็มระหว่าง 0 กับ 1 โดยแบ่งเป็นถ้าตรวจจับได้ว่าการกำมือจะได้ค่าเท่ากับ 1 แต่ถ้าตรวจจับได้ว่าไม่ได้กำมือจะได้ค่าเท่ากับ 0 หลังจากได้ค่าที่ต้องการเรียบร้อยแล้วจึงทำการจัดส่งให้กับส่วนควบคุมภาคอากาศเพื่อนำไปประมวลผลสำหรับการสั่งเริ่มต้น หรือหยุดการทำงานของมอเตอร์ทั้ง 4 ของมอเตอร์โรเตอร์

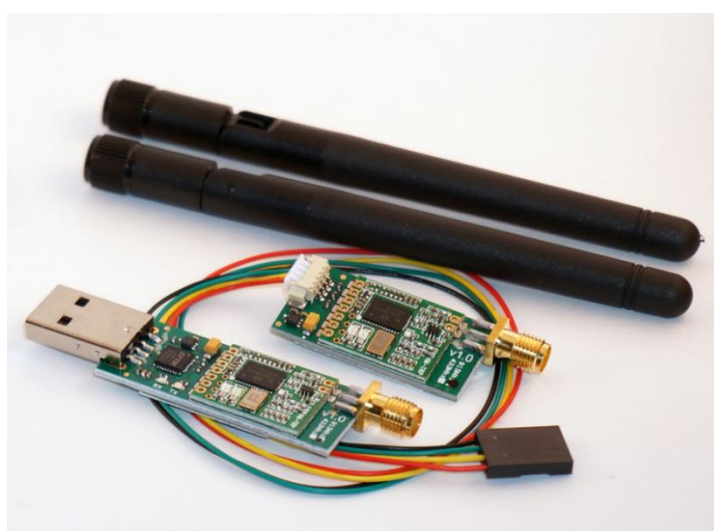
4. ประมวลผลท่าทางการกระดิกนิ้วชี้และนิ้วกลาง

สำหรับการประมวลผลท่าทางการกระดิกนิ้วชี้และนิ้วกลาง เป็นการเรียกใช้คำสั่งที่ทางผู้พัฒนาอุปกรณ์ Leap Motion ได้สร้างไว้เรียบร้อยแล้ว โดยสามารถศึกษาได้จากเว็บไซต์ของ Leap Motion ในส่วนของ C# Documentary โดยตรงได้ ซึ่งผลลัพธ์ที่ได้จากการใช้คำสั่งทางกลุ่มของเราได้เลือกให้มีค่าสลับกันไปมาระหว่างค่าของตัวเลขจำนวนเต็ม 0 กับ 1 โดยถ้าตรวจจับได้ว่ามีท่าทางการกระดิกนิ้วชี้หรือนิ้วกลาง แล้วค่าปัจจุบันเท่ากับ 0 จะกลายเป็น 1 แต่ถ้าค่าปัจจุบันเท่ากับ 1 จะกลายเป็น 0 โดยค่าของท่าทางการกระดิกนิ้วชี้และนิ้วกลางนั้นจะทำการแยกกันเก็บไว้คนละตัวแปรกัน หลังจากที่ได้ค่าตามต้องการเรียบร้อยแล้วจึงส่งไปยังส่วนควบคุมภาคอากาศเพื่อนำไปประมวลผลต่อการเพิ่มระดับหรือลดระดับความสูงระหว่างที่ทำการบินของมอเตอร์

2.2.3 Wireless Datalink (Telemetry 915 MHz)

Wireless Datalink คือ ช่องทางการสื่อสารระหว่างส่วนควบคุมการบินและผู้ควบคุมการบิน สามารถใช้ได้ทั้งสัญญาณวิทยุที่ส่งเฉพาะการควบคุมเบื้องต้น จนกระทั่งใช้ในการรับ-ส่งสถานะของตัวกลับมายังภาคพื้นดิน ซึ่ง Wireless Datalink ที่เลือกใช้งานเป็น Telemetry ที่มีความถี่ 915 MHz ทำหน้าที่ในการส่งข้อมูลระหว่างส่วนควบคุมภาคพื้นดินกับส่วนควบคุมภาคอากาศที่อยู่บนมอเตอร์โรเตอร์ ลักษณะการใช้งานจะทำหน้าที่เป็นทั้งตัวส่ง (Sender) และตัวรับ (Receiver) ทั้งในส่วนของส่วนควบคุมภาคพื้นดินและส่วนควบคุมภาคอากาศ โดยที่ส่วนควบคุม

ภาคพื้นดินจะทำการเชื่อมต่ออยู่กับคอมพิวเตอร์ผ่านทาง USB Portable และที่ส่วนควบคุมภาคอากาศจะเชื่อมต่ออยู่กับ Controller บนตัวลำโพงมอเตอร์ Wireless Datalink มีคุณสมบัติรองรับการเชื่อมต่อที่ระยะมากกว่า 30 m ถึง 1 km ในพื้นที่โล่งแจ้ง รองรับอัตราความเร็วในการส่งข้อมูลอยู่ที่ 57,600 บิตต่อวินาทีและที่สำคัญมีราคาถูกเมื่อเทียบกับอุปกรณ์ที่มีคุณสมบัติใกล้เคียงกันอย่างเช่น Xbee จึงได้เลือกนำมาเป็นอุปกรณ์สำหรับสื่อสารข้อมูลระหว่างส่วนควบคุมภาคพื้นดินกับส่วนควบคุมภาคอากาศ และสามารถดูลักษณะของอุปกรณ์ Wireless Datalink ได้ดังภาพที่ 2-6

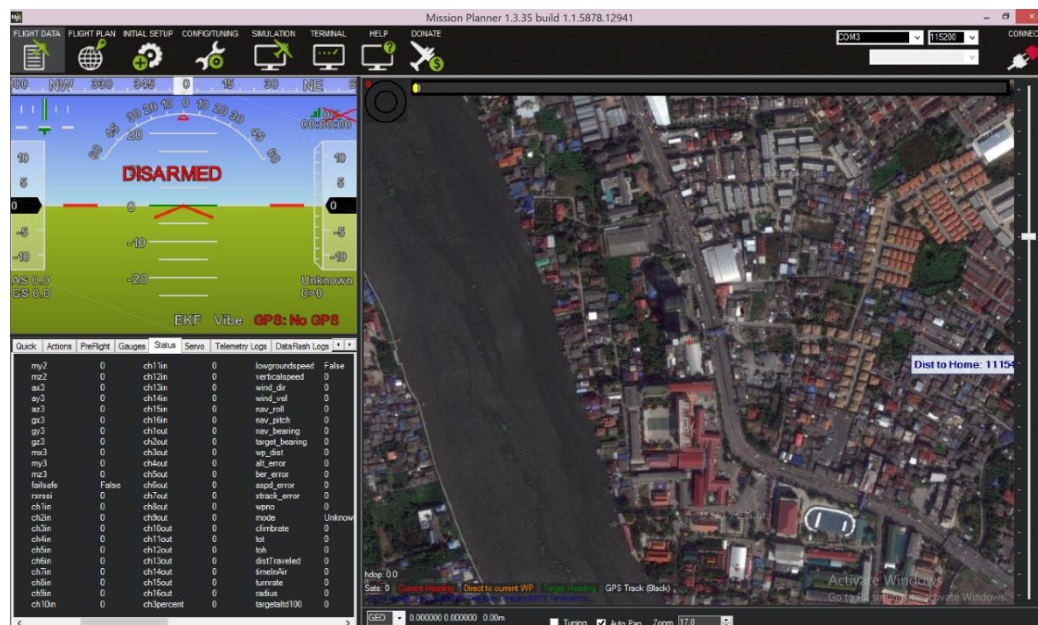


ภาพที่ 2-6 ลักษณะของอุปกรณ์ Wireless Datalink (Telemetry 915 MHz)

[<http://quadcoptergarage.com/product/rctimer-arduiflyer-v2-5-2-kit-with-915mhz-telemetry-and-gps/>]

2.2.4 Mission Planner

เป็นซอฟต์แวร์สำหรับส่วนควบคุมการบิน (Flight Controller) รองรับการทำงานทั้งหมดที่ Ardupilot Mega 2.6 รองรับ เช่น ปรับค่าพารามิเตอร์ก่อนขึ้นบิน การตั้งจุด Waypoint สำหรับการบินตามตำแหน่งที่กำหนด วางแผนการบินอัตโนมัติ (Flight Plan) ดูการบินบนแผนที่แบบ Real Time รวมไปถึงสั่งการมอเตอร์ที่กำลังบินให้ทำท่าต่าง ๆ ได้ทันที โดยซอฟต์แวร์ตัวนี้ทางกลุ่มของเราได้เลือกมาทำหน้าที่ในการสังเกตผลของสัญญาณ PWM ที่สร้างให้กับส่วนควบคุมการบินว่าตอบสนองหรือไม่ และใช้ในการตั้งค่าพารามิเตอร์ของส่วนควบคุมการบินก่อนขึ้นบิน โดยโปรแกรม Mission Planner มีลักษณะหน้าต่างของโปรแกรกดังภาพที่ 2-7



ภาพที่ 2-7 หน้าต่างโปรแกรม Mission Planner

2.3 ส่วนควบคุมภาคอากาศ (อุปกรณ์บนลำโม่ดีโรเตอร์)

ส่วนควบคุมภาคอากาศมีหน้าที่หลัก คือรับข้อมูลจากส่วนควบคุมภาคพื้นดินที่ผ่านการประมวลผลเรียบร้อยแล้ว มาสร้างเป็นสัญญาณ PWM จำนวน 5 ช่อง ได้แก่ Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel และ Flight Mode Channel โดยสัญญาณ PWM ที่สร้างขึ้นนั้นจะมีความถี่อยู่ที่ 50 Hz ซึ่งในการสร้างสัญญาณ PWM นั้น ทางกลุ่มของเราได้เลือกใช้คำสั่ง digitalWrite() ควบคู่กับคำสั่ง delayMicrosecond() โดยเป็นการกำหนดระยะเวลาของสัญญาณช่วงที่เป็น High Logic ให้มีค่าอยู่ระหว่าง 1,000 ถึง 2,000 ซึ่งเป็นค่าตัวเลขที่สามารถติดต่อกับ Flight Controller ทางด้าน Input Channel ได้ ซึ่งค่าของตัวเลขดังกล่าวสำหรับ Roll Channel, Pitch Channel, Yaw Channel จะพิจารณาจากผลลัพธ์ที่ได้จากส่วนควบคุมภาคพื้นดินที่เป็นชุดตัวเลขระหว่าง 1 ถึง 9 แต่ค่าของ Throttle Channel จะพิจารณาจากผลลัพธ์ที่ได้จากส่วนควบคุมภาคพื้นดินจากท่าทางการกระดิกนิ้วชี้หรือนิ้วกลาง และสุดท้ายค่าของ Flight Mode Channel จะพิจารณาจากผลลัพธ์ที่ได้จากส่วนควบคุมภาคพื้นดินจากท่าทางการกำมือและแบมือ นอกจากกำหนดเวลาให้กับสัญญาณ High Logic แล้วยังต้องกำหนดเวลาให้กับสัญญาณช่วงที่เป็น Low Logic ด้วย โดยใช้สูตรว่า Low Logic เท่ากับ 20,000 ลบด้วยช่วงเวลาที่เป็นสัญญาณ High Logic ซึ่งการทำเช่นนี้จะทำให้สัญญาณ PWM ที่สร้างขึ้นมีความถี่ของสัญญาณเท่ากับ 50 Hz พอดี สำหรับการสร้างสัญญาณ

PWM นั้นสามารถเลือกใช้งานคำสั่งอย่างเช่น `analogWrite()` จะได้ผลลัพธ์เช่นเดียวกันแต่มีข้อจำกัดทางด้านจำนวน Pin ที่สามารถสร้างสัญญาณ PWM และมีความถี่เท่ากับ 50 Hz ได้ ซึ่งมีไม่เพียงพอต่อการใช้งานทั้ง 5 ช่องสัญญาณ รวมไปถึงการใช้ Library Servo แล้วใช้คำสั่ง `WriteMicrosecond()` จะได้ผลลัพธ์เช่นเดียวกันกับแบบที่มีการใช้งานคำสั่ง `analogWrite()` แต่ส่งผลกระทบต่อเรื่องของการสื่อสารของอุปกรณ์ Wireless Datalink เนื่องจากการสื่อสารของอุปกรณ์ Wireless Datalink มีการเรียกใช้งาน Library SoftSerial ซึ่งมีการใช้งาน interrupt ของไมโครคอนโทรลเลอร์อยู่แล้ว จึงทำให้เกิดการทำงานทับซ้อนกัน ซึ่งทำให้ข้อมูล que ที่สื่อสารระหว่างส่วนควบคุมภาคพื้นดินกับส่วนควบคุมภาคอากาศมีข้อผิดพลาดระหว่างการสื่อสารอย่างมาก จึงไม่สามารถใช้งานได้

ลักษณะการเชื่อมต่อของส่วนควบคุมภาคอากาศ จะเริ่มจากรับข้อมูลที่ได้จากส่วนควบคุมภาคพื้นดินด้วยอุปกรณ์ Wireless Datalink และทำการป้อนเข้ากับบอร์ด Arduino UNO ซึ่งกำหนดให้ทั้ง 2 อุปกรณ์นี้ทำหน้าที่แทน Radio Receiver หลังจากนั้นทำการประมวลผลจะได้สัญญาณ PWM แล้วจึงนำไปป้อนเข้ากับ Flight Controller ทางด้าน Input Channel เพื่อควบคุม Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel ซึ่งการควบคุม Input Channel เหล่านี้ Flight Controller จะไปควบคุมการทำงานของ ESC โดยการใช้ Output Channel ของ Flight Controller เชื่อมต่อเข้ากับ ESC ด้วยสายไฟซึ่งต้องใช้ทั้งหมด 4 Output Channel ด้วยกัน เนื่องจากการควบคุมมอเตอร์ 4 ตัว ซึ่งการที่ ESC ถูกควบคุมนั้นจะมีผลต่อความเร็วรอบของมอเตอร์ทั้ง 4 ตัว จึงทำให้ผู้ใช้งานสามารถควบคุมการเคลื่อนที่ของมัลติโรเตอร์ได้

ส่วนควบคุมภาคอากาศมีองค์ประกอบที่สำคัญ 2 ส่วนได้แก่ 1. ส่วนประมวลผลข้อมูลจากภาคพื้นดิน ซึ่งประกอบด้วย Arduino UNO และ Wireless Datalink และ 2. ส่วนประกอบของตัวลำมัลติโรเตอร์ ซึ่งประกอบด้วย Frame S500, Flight Controller (APM2.6), ESC, Brushless Motor, Battery, GPS และ Power Module

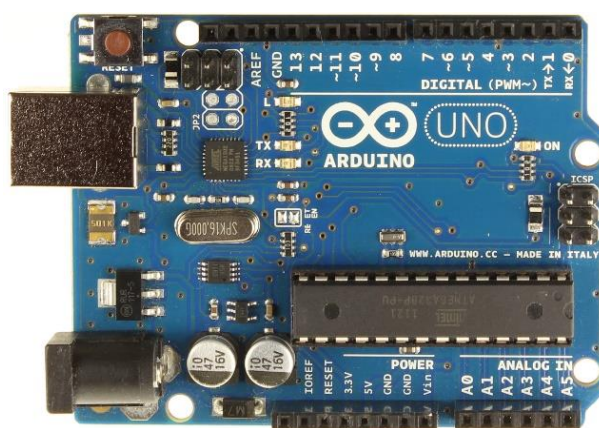
2.3.1 ส่วนประมวลผลข้อมูลจากภาคพื้นดิน

เป็นส่วนที่ทำหน้าที่รับชุดข้อมูลจากส่วนควบคุมภาคพื้นดินผ่านอุปกรณ์ Wireless Datalink แล้วนำข้อมูลเหล่านั้นมาประมวลผล และนำผลลัพธ์ที่ได้ไปกำหนดเป็นค่าให้กับสัญญาณ PWM ที่สร้างขึ้น เพื่อป้อนให้กับ Flight Controller ทางด้าน Input Channel ด้วยอุปกรณ์ Arduino UNO

1. Arduino UNO

เป็นบอร์ดไมโครคอนโทรลเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source เป็นบอร์ดที่ถูกออกแบบมาให้ใช้งานได้ง่ายและสามารถดัดแปลงเพิ่มเติมร่วมกับอุปกรณ์ต่าง ๆ ได้ รวมถึง

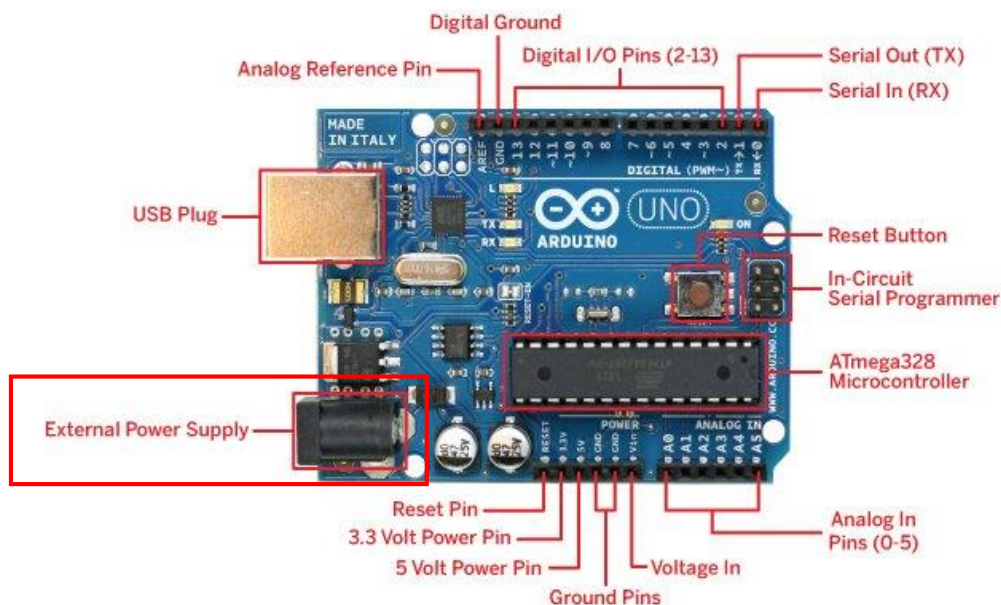
สามารถนำมาใช้งานแทน Flight Controller ได้ด้วย แต่ต้องทำการติดตั้งเซนเซอร์ในการใช้วัดค่าต่างๆ เพิ่มเติมเช่น เซ็นเซอร์วัดความสูง, GPS Module เป็นต้น แต่ทางกลุ่มเราได้ใช้ Arduino UNO ร่วมกับอุปกรณ์ Wireless Datalink (อธิบายในหัวข้อ 2.2.3) ให้ทำหน้าที่แทน Radio Receiver เท่านั้น โดยมีหน้าที่หลักเพื่อแปลงข้อมูลที่รับจากส่วนควบคุมภาคพื้นดินให้ได้ผลลัพธ์ออกมาเป็นสัญญาณ PWM แล้วนำผลลัพธ์ดังกล่าวป้อนให้กับ Flight Controller ทางด้าน Input Channel เพื่อให้ Flight Controller ไปจัดการกับปริมาณกระแสไฟฟ้าที่จ่ายให้กับ ESC ซึ่งการทำเช่นนี้จะมีผลโดยตรงต่อความเร็วรอบของมอเตอร์ทั้ง 4 ตัว จึงทำให้มอเตอร์เกิดการเคลื่อนที่ตามมือของผู้ใช้งานได้ สามารถดูลักษณะของอุปกรณ์ Arduino UNO ได้ดังภาพที่ 2-8



ภาพที่ 2-8 ลักษณะของอุปกรณ์ Arduino UNO

[http://www.jameco.com/webapp/wcs/stores/servlet/Product_10001_10001_2163840_-1]

ปัจจุบัน Arduino มีการทำออกมาในหลายชนิดด้วยกัน ซึ่งจะแตกต่างกันในส่วนของคุณสมบัติ ความเร็วสัญญาณนาฬิกา จำนวนขาดีจิทัลและแอนะล็อก จำนวนยูอาร์ที และขนาดของ SRAM เป็นต้น แต่ในโครงการนี้สามารถเลือกใช้งานบอร์ด Arduino ได้ 2 ชนิดเท่านั้น ได้แก่ Arduino UNO และ Arduino Due และทั้ง 2 รุ่นดังกล่าวมีคุณสมบัติต่างกันดังแสดงในตารางที่ 2-1 โดยเหตุผลที่เลือกใช้งานบอร์ด Arduino ได้เพียง 2 รุ่นนี้เท่านั้น เนื่องจากอันดับแรกทางกลุ่มของเราได้พิจารณาจากส่วนของ External Power Supply ของอุปกรณ์เป็นหลัก สามารถดูตำแหน่งของ External Power Supply ได้จากภาพที่ 2-9



ภาพที่ 2-9 แสดงตำแหน่ง External Power Supply ของอุปกรณ์ Arduino UNO

[<https://www.robomart.com/arduino-uno-online-india>]

และเนื่องจากเวลานำมัลติโรเตอร์ขึ้นบินนั้นอุปกรณ์ Arduino UNO จะต้องทำงานตลอดเวลา ซึ่งจะต้องรับไฟเลี้ยงภายนอกมาจาก Battery ที่ใช้สำหรับป้อนกับให้ Flight Controller และ ESC อยู่แล้ว ซึ่งเหตุผลที่เลือกรับไฟเลี้ยงภายนอกจาก Battery นั้น เพื่อเป็นการไม่เพิ่มน้ำหนักให้กับตัวลำของมัลติโรเตอร์ และการที่จะรับไฟจาก Battery ได้นั้นจะต้องใช้สายไฟที่ด้านหนึ่งเป็นหัวเชื่อมต่อชนิด XT-60 เพื่อเชื่อมต่อด้านที่ออกมาจาก Battery และหัวเชื่อมต่ออีกด้านหนึ่งเป็น DC Jack เพื่อเชื่อมต่อเข้ากับบอร์ด Arduino UNO เท่านั้น ซึ่งเหตุผลที่ต้องใช้สายไฟชนิดนี้เนื่องจากเป็นสายไฟที่สามารถสั่งทำพิเศษได้ตามกลุ่มซื้อขายอุปกรณ์มัลติโรเตอร์อยู่แล้ว สามารถดูลักษณะสายไฟชนิด XT-60 to DC Jack ดังกล่าว ได้จากภาพที่ 2-10 จากเหตุผลดังกล่าวจึงทำให้บอร์ด Arduino ที่สามารถเลือกใช้งานได้นั้น เหลือแค่ Arduino UNO กับ Arduino Due เท่านั้น และถ้าพิจารณาเหตุผลถัดมาคือ พิจารณาจากขนาดของบอร์ดที่จะต้องนำมาวางบนตัวลำของมัลติโรเตอร์ ซึ่งมีพื้นที่ที่จำกัดและไม่สามารถวางบอร์ด Arduino รุ่น Arduino Due ได้ จึงทำให้ทางกลุ่มของเราสรุปได้ว่าเลือกใช้งานบอร์ด Arduino เป็นรุ่น Arduino UNO



ภาพที่ 2-10 แสดงลักษณะสายไฟชนิด XT-60 to DC Jack

[http://www.fpvmodel.com/power-cord-with-2-1mmx5-5mm-barrel-plug-to-xt-60-male-plug_g397.html]

ตารางที่ 2-1 เปรียบเทียบรายละเอียดของ Arduino UNO กับ Arduino Due

คุณสมบัติ	Arduino UNO	Arduino Due
หน่วยประมวลผล	ATmega328P	AT91SAM3X8E
ระดับแรงดัน	5V	3.3V
ความเร็วสัญญาณนาฬิกา	16 MHz	84 MHz
จำนวนขาแอนะล็อก	6	12
จำนวนขาคิจิทัล	14	54
จำนวนยูอาร์ที	1	4
จำนวนเอสพีไอ	1	1
วงจรมินิเวลจิงภายใน	ไม่มี	มี
ขนาดเอสแรม	2 KB	96 KB
ขนาดแฟลช	32 KB	512 KB

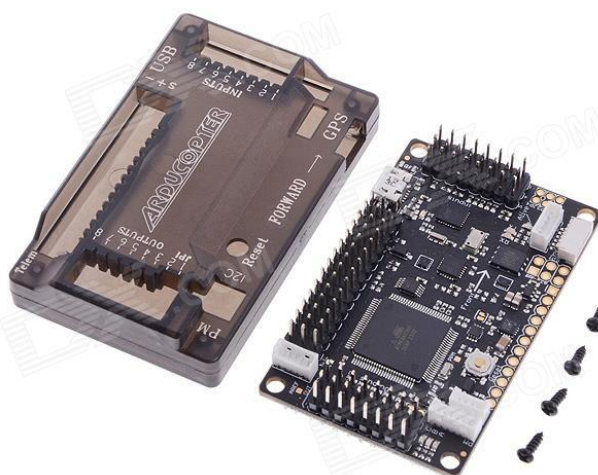
2.3.2 ส่วนประกอบของตัวลำโพงมอเตอร์

เป็นส่วนที่มีหน้าที่หลักในการควบคุมความเร็วของมอเตอร์ทั้ง 4 ตัว ด้วยอุปกรณ์ Flight Controller เพื่อให้เกิดการเคลื่อนที่ในทิศทางต่าง ๆ ตามที่ผู้ใช้งานต้องการ โดยการควบคุมของ Flight Controller นั้น จะควบคุมตามสัญญาณ Input ที่รับเข้ามาทางด้าน Input Channel จากส่วนประมวลผลข้อมูลจากภาคพื้นดิน โดยสำหรับส่วนประกอบของตัวลำโพงมอเตอร์มีส่วนประกอบที่สำคัญดังนี้

1. Flight Controller (ArduPilot Mega 2.6)

ทำหน้าที่จัดการกับความเร็วรอบของมอเตอร์ทั้ง 4 ตัวโดยการควบคุมปริมาณของกระแสไฟฟ้าที่ป้อนให้กับ ESC แต่การจัดการดังกล่าวนี้จะขึ้นอยู่กับสัญญาณ PWM ที่ Arduino UNO ป้อนให้กับ Flight Controller ทางด้าน Input Channel ทั้ง 5 ช่องทาง โดยลักษณะการจัดการต่าง ๆ จะมีอธิบายไว้ในบทที่ 3 และ 4

สำหรับการเลือกใช้งาน Flight Controller เป็นชนิด ArduPilot Mega 2.6b (APM2.6) นั้น เนื่องจาก APM2.6 รองรับการพัฒนาร่วมกับอุปกรณ์อื่น อาทิเช่น รองรับการพัฒนาร่วมกับบอร์ด Arduino UNO ในการควบคุมสัญญาณ PWM ทางด้าน Input Channel ทั้ง 5 ช่องทาง และที่สำคัญ APM2.6 มีราคาที่ไม่สูงมากเกินไป เมื่อเทียบกับ Flight Controller ชนิด Pixhawk ที่รองรับการพัฒนาร่วมกับอุปกรณ์อื่นเช่นเดียวกัน สามารถดูลักษณะของอุปกรณ์ Flight Controller (ArduPilot Mega 2.6) ได้ดังภาพที่ 2-11



ภาพที่ 2-11 ลักษณะของอุปกรณ์ Flight Controller (ArduPilot Mega 2.6)

[http://img.dxcn.com/productimages/sku_266684_1.jpg]

2. Frame S500

มีหน้าที่เป็นฐานสำหรับติดตั้งอุปกรณ์ต่าง ๆ ของตัวลำโผลติโรเตอร์ เช่น Flight Controller, Arduino UNO, GPS, ESC, Brushless Motor, Battery เป็นต้น โดยมีขนาดของเส้นทแยงมุมเท่ากับ 500 mm ดังภาพที่ 2-12 และหากเลือก Frame ที่มีขนาดใหญ่กว่านี้จะทำให้รับน้ำหนักได้มากขึ้น แต่ต้องใช่มอเตอร์ที่มีกำลังสูงขึ้นด้วย หากมีขนาดเล็กก็จะสามารถรองรับน้ำหนักได้น้อย แต่ก็ทำให้บินได้ฉวัดเฉยีนมากขึ้น ซึ่งวัสดุที่ต่างกันจะมีผลในเรื่องของน้ำหนักและการสั่น



ภาพที่ 2-12 ลักษณะของอุปกรณ์ Frame S500

[<https://www.unmannedtechshop.co.uk/s500-quadcopter-frame/>]

3. ใบพัด

ใบพัดที่เลือกใช้เป็นใบพัดชนิด Niron มีขนาด 12*4.5 นิ้ว ดังภาพที่ 2-13 สำหรับการเลือกใช้งานใบพัดนั้นขึ้นอยู่กับชนิดของมอเตอร์ที่ใช้งาน รวมไปถึงกำลังของมอเตอร์ที่ใช้งานด้วย ซึ่งจะส่งผลโดยตรงต่อขนาดของใบพัดที่สามารถใช้งานได้



ภาพที่ 2-13 ลักษณะของอุปกรณ์ ใบพัด 12*4.5 Niron

[<http://srv-live.lazada.co.th/p/image-9518434-1-product.jpg>]

4. Electronic Speed Controller (ESC)

อุปกรณ์ตัวนี้จะใช้สำหรับควบคุมความเร็วรอบของมอเตอร์แต่ละตัว ซึ่ง ESC จะรับคำสั่งมาจาก Flight Controller ทางด้าน Output Channel แล้วจัดการจ่ายไฟเลี้ยงจาก Battery ไปให้กับมอเตอร์ตามคำสั่งที่ได้รับ ก็จะมีการจ่ายกระแสช้าหรือมากขึ้นอยู่กับคำสั่งที่รับมาจาก Flight Controller อีกที สรุปคือ ESC ทำหน้าที่ในการขยายสัญญาณระหว่างตัว Flight Controller ไปยังมอเตอร์ ซึ่งสามารถป้องกันสัญญาณขาดระหว่างการส่งได้ เพื่อใช้ควบคุมความเร็วรอบของมอเตอร์ ทำให้การบินที่มีเสถียรภาพ โดยจะใช้ขนาด 30 A ขึ้นไป เพื่อให้เพียงพอกับขนาดของมอเตอร์ที่นำมาใช้งาน สามารถดูลักษณะของอุปกรณ์ Electronic Speed Controller ขนาด 30 A ได้ดังภาพที่ 2-14



ภาพที่ 2-14 ลักษณะของอุปกรณ์ Electronic Speed Controller ขนาด 30A

[http://img.dxcn.com/productimages/sku_347982_2.jpg]

5. Brushless Motor รุ่น 4010 ขนาด 850 KV

สำหรับมอเตอร์ที่ใช้กับมัลติโรเตอร์นั้น จะเป็นมอเตอร์ชนิด Brushless Motor คือ มอเตอร์ชนิดไม่มีแปรงถ่าน มอเตอร์ชนิดนี้ขดลวดจะอยู่นิ่ง แต่ส่วนที่เป็นแม่เหล็กจะเป็นตัวหมุนแทน ทำหน้าที่ขับใบพัดเพื่อให้เกิดแรงยกแต่ละด้านของตัวลำ สาเหตุที่ต้องใช้ Brushless Motor เป็นเรื่องของประสิทธิภาพและกินพลังงานน้อย ทำให้โดรนบินได้สูงและนานขึ้น ซึ่งความเร็วรอบของมอเตอร์จะถูกกำหนดด้วย Electronic Speed Controller (ESC) สามารถดูลักษณะของอุปกรณ์ Brushless Motor รุ่น 4010 ขนาด 850 KV ได้ดังภาพที่ 2-15



ภาพที่ 2-15 ลักษณะของอุปกรณ์ Brushless Motor รุ่น 4010 ขนาด 850 KV

[<http://www.queenhobby.com/product/upload/photos/3661/201506300229582.jpg>]

6. Battery

แบตเตอรี่ที่ใช้กับมัลติโรเตอร์ ก็คือแบตเตอรี่ชนิด Lithium Polymer (LiPo) หรือเรียกว่า แบตลิโพอ ดังภาพที่ 2-16 สาเหตุที่ใช้แบตเตอรี่ชนิดนี้ในการสร้างมัลติโรเตอร์เพราะมีน้ำหนักเบา และให้ประสิทธิภาพที่สูง โดยแบตเตอรี่จะทำหน้าที่เป็นแหล่งจ่ายไฟให้กับ Flight Controller



ภาพที่ 2-16 ลักษณะของอุปกรณ์ Battery รุ่น Thunder Power 3 Cell 11.1V ขนาด 4300 mAh

[<http://images.rcuniverse.com/market/itemimages/Dec/Ig-1191198-0-2610.jpg>]

7. GPS

อุปกรณ์รับสัญญาณจีพีเอสหรือ Global Positioning System ทำหน้าที่สร้างระบบ Waypoint นั่นคือการสร้างตำแหน่งการบินของมอเตอร์ในขณะที่ไม่มีการบังคับ และใช้แก้ปัญหาในกรณีที่สัญญาณการควบคุมขาดหายไป ซึ่ง GPS จะถูกเชื่อมต่ออยู่กับ Flight Controller ซึ่งที่ทางกลุ่มของเราใช้คือ GPS รุ่น Ublox NEO-7N สามารถดูลักษณะได้ดังภาพที่ 2-17

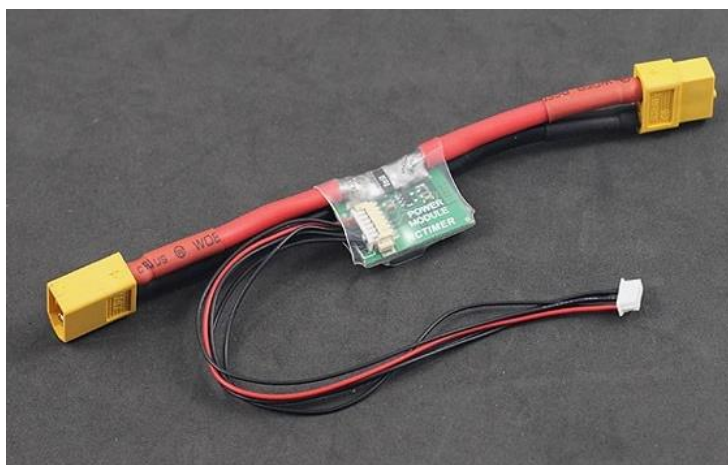


ภาพที่ 2-17 ลักษณะของอุปกรณ์ GPS รุ่น Ublox NEO-7N

[<http://g04.a.alicdn.com/kf/HTB1PGwcJFXXXXabXXXXq6xXFXXXw/High-Precision-GPS-Module-with-Compass-for-NEO-7N-Pixhawk-APM-Multicopter.jpg>]

8. Power Module

อุปกรณ์สำหรับอ่านค่าแรงดันและกระแสของมอเตอร์ โดยจะทำการแปลงไฟเลี้ยงที่มาจากแบตเตอรี่ให้พอดีกับ Flight Controller เพื่อป้องกันอุปกรณ์เสียหาย สามารถดูลักษณะของอุปกรณ์ Power Modul รุ่น RCtimer for APM Max ขนาด 30 V, 90 A ได้ดังภาพที่ 2-18



ภาพที่ 2-18 ลักษณะของอุปกรณ์ Power Modul รุ่น RCtimer for APM Max ขนาด 30V, 90A

[<http://media.kooltoyz.co.uk/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/a/r/arduapm.jpg>]

บทที่ 3

วิธีดำเนินงาน

การพัฒนาระบบสำหรับควบคุมการบินของมัลติโรเตอร์สามารถแบ่งระบบออกเป็น 2 ส่วนหลัก ๆ ได้แก่ 1. ระบบของส่วนควบคุมภาคพื้นดิน 2. ระบบของส่วนควบคุมภาคอากาศ

3.1 ระบบของส่วนควบคุมภาคพื้นดิน

เป็นส่วนสำหรับติดต่อกับผู้ใช้งาน โดยการรับข้อมูลผ่านอุปกรณ์ Leap Motion ที่เชื่อมต่ออยู่กับคอมพิวเตอร์และมัลติคอนโทรลเลอร์สำหรับประมวลผลค่าที่ Leap Motion อ่านได้ โดยใช้ภาษา C# ในการพัฒนาอัลกอริทึมดังกล่าว เมื่อได้ผลลัพธ์ที่ต้องการแล้วจึงทำการส่งผลลัพธ์ดังกล่าวไปยังส่วนควบคุมภาคอากาศผ่านอุปกรณ์ Wireless Datalink เพื่อให้ผู้ใช้งานสามารถควบคุมการบินของมัลติโรเตอร์ได้ นอกจากนี้ในส่วนควบคุมภาคพื้นดิน ยังได้มีการแสดงผลลัพธ์ที่ใช้สำหรับส่งออกไปยังส่วนควบคุมภาคอากาศในรูปแบบของ Desktop Application เพื่อให้ผู้ใช้งานสามารถสังเกตผลลัพธ์ที่ถูกส่งออกไปและผลตอบสนองที่ได้กลับมาจากส่วนควบคุมภาคอากาศได้

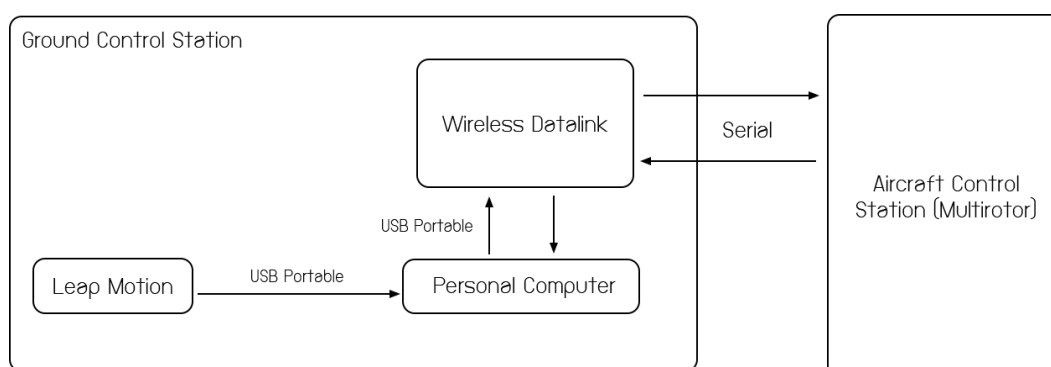
สำหรับขั้นตอนการพัฒนาระบบในส่วนของส่วนควบคุมภาคพื้นดินนั้นมีขั้นตอนการพัฒนา ดังนี้

1. ออกแบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคพื้นดิน
 2. ออกแบบภาพรวมของโปรแกรมส่วนควบคุมภาคพื้นดิน
 3. พัฒนาโปรแกรมตรวจจับจำนวนมือ ทำทางของมือ และมุม Roll, Pitch, Yaw ของมือผู้ใช้งาน
 4. พัฒนาโปรแกรมประมวลผลทำทาง และมุม Roll, Pitch, Yaw ของมือผู้ใช้งานที่ตรวจจับได้
 5. พัฒนาโปรแกรมแสดงผลการทำงานของระบบในรูปแบบ Desktop Application
 6. พัฒนาโปรแกรมการส่งข้อมูลของส่วนควบคุมภาคพื้นดิน
- 3.1.1 ออกแบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคพื้นดิน

มีลักษณะการเชื่อมต่อของอุปกรณ์ฮาร์ดแวร์ตามภาพที่ 3-1 โดยมีการเชื่อมต่อคือนำอุปกรณ์ Leap Motion และ Wireless Datalink เชื่อมต่อเข้ากับ Notebook ผ่านทาง USB Portable โดยแต่ละอุปกรณ์จะมีหน้าที่ดังนี้

- Leap Motion เป็นอุปกรณ์สำหรับตรวจจับการเคลื่อนไหวท่าทางต่าง ๆ ของมือผู้ใช้งาน เพื่อนำไปประมวลผลในอัลกอริทึมที่ออกแบบ จนกระทั่งได้เป็นคำสั่งต่าง ๆ ออกมา และนำคำสั่งดังกล่าวส่งออกไปให้กับส่วนควบคุมภาคพื้นอากาศเพื่อให้ผู้ใช้งานสามารถควบคุมการบินของมัลติโรเตอร์ในลักษณะท่าทางต่าง ๆ ได้ เช่น การเดินหน้า-ถอยหลัง บินไปทางซ้าย-ขวา บินหมุนวน เข้ม-ตามเข็ม เป็นต้น โดยท่าทางที่ใช้ในการควบคุมลักษณะการบินของมัลติโรเตอร์ ได้แก่ 1. การแบมือบนอุปกรณ์ Leap Motion เพื่อตรวจจับการเปลี่ยนของค่า Roll, Pitch, Yaw 2. การกำมือและแบมือ 3. การกระดิกนิ้วชี้และนิ้วกลาง สำหรับลักษณะการเชื่อมต่อระหว่าง Leap Motion กับคอมพิวเตอร์ ใช้การเชื่อมต่อผ่านทาง USB Portable

- Wireless Datalink มีหน้าที่หลักในการส่งและรับข้อมูลระหว่างส่วนควบคุมภาคพื้นดิน กับส่วนควบคุมภาคอากาศ โดย Wireless Datalink มีรูปแบบการสื่อสารแบบ Serial ในย่านความถี่ 915 MHz มีอัตราความเร็วในการส่งข้อมูลอยู่ที่ 57,600 บิตต่อวินาที และรองรับการเชื่อมต่อที่ระยะไม่เกิน 1 กิโลเมตรในพื้นที่โล่ง สำหรับลักษณะการเชื่อมต่อระหว่าง Wireless Datalink กับคอมพิวเตอร์ ใช้การเชื่อมต่อผ่านทาง USB Portable

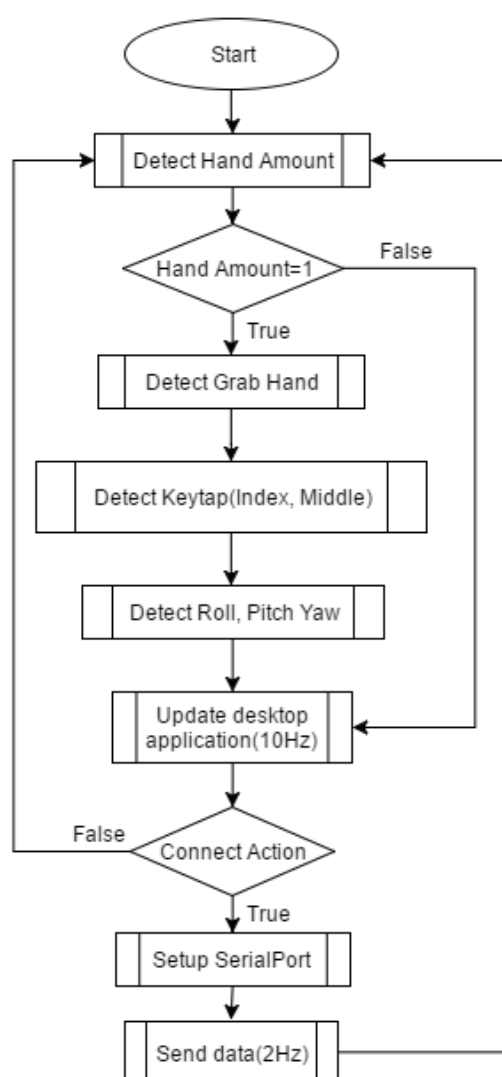


ภาพที่ 3-1 ภาพรวมของฮาร์ดแวร์สำหรับส่วนควบคุมภาคพื้นดิน

3.1.2 ออกแบบภาพรวมของโปรแกรมส่วนควบคุมภาคพื้นดิน

ระบบทั้งหมดสำหรับส่วนควบคุมภาคพื้นดิน ทางกลุ่มของเราได้พัฒนาโดยใช้โปรแกรม Microsoft Visual Studio และเลือกใช้ภาษา C# ในการพัฒนาส่วนต่าง ๆ โดยมีภาพรวมการทำงาน เริ่มจากตรวจสอบมือของผู้ใช้งานว่าอยู่บนอุปกรณ์ Leap Motion หรือไม่ ถ้ามีมือของผู้ใช้งานจะทำการตรวจสอบท่าทางการกำมือ การกระดิกนิ้วชี้ นิ้วกลาง และตรวจสอบมุมการเคลื่อนไหวของมือผู้ใช้งาน หลังจากนั้นจะทำการอัปเดตข้อมูลบางส่วนไปแสดงที่หน้าจอแสดงผลบนโปรแกรม Desktop Application โดยมีความถี่ในการอัปเดตข้อมูลที่ 10 Hz ต่อมาจะทำการตรวจสอบว่า

ผู้ใช้งานได้มีการกดปุ่มเพื่อเชื่อมต่อระหว่างส่วนควบคุมภาคพื้นดินกับส่วนควบคุมภาคอากาศหรือไม่ โดยผู้ใช้งานสามารถกดได้จากหน้าจอแสดงผลบนโปรแกรม Desktop Application และถ้าตรวจสอบได้ว่าการเชื่อมต่อเรียบร้อยแล้วจะทำการส่งข้อมูลออกไปให้กับส่วนควบคุมภาคอากาศโดยมีความเร็วในการส่งข้อมูลที่ความถี่ 2 Hz แต่ถ้ายังไม่ได้กดปุ่มเชื่อมต่อ ข้อมูลที่ตรวจจับจากมือของผู้ใช้งานจะไม่ถูกส่งออกไปแต่จะแสดงบนหน้าจอแสดงผลของโปรแกรม Desktop Application เท่านั้น และสำหรับกรณีที่ตรวจสอบได้ว่าไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion ระบบจะทำการกำหนดค่าให้กับท่าทางการกำมือ การกระดิกนิ้วชี้ นิ้วกลาง และมุม Roll, Pitch, Yaw โดยอัตโนมัติ และมีภาพรวมของการทำงานตามภาพที่ 3-2



ภาพที่ 3-2 ภาพรวมของระบบสำหรับส่วนควบคุมภาคพื้นดิน

3.1.3 พัฒนาโปรแกรมตรวจจับจำนวนมือ ทำทางของมือ และมุม Roll, Pitch, Yaw ของมือผู้ใช้งาน

ถ้าพิจารณาจากภาพที่ 3-2 คือส่วนของ Detect Hand Amount, Detect Grab Hand, Detect Keytap (Index, Middle) และ Detect Roll Pitch Yaw โดยขั้นตอนการพัฒนาจะเริ่มจากการติดตั้ง Leap Motion SDK ลงบนคอมพิวเตอร์ และทำการศึกษา C# SDK Documentation จากเว็บไซต์ในหัวข้อ Hand, Finger, Frame เพื่อให้สามารถเรียกใช้คำสั่งในการตรวจจับลักษณะท่าทางของมือผู้ใช้งานได้ โดยลักษณะของคำสั่งต่าง ๆ ที่มีไว้สำหรับตรวจจับมือ ทำทางการกำมือ การกระดิกนิ้วชี้และนิ้วกลาง รวมถึงมุม Roll, Pitch, Yaw ที่ได้จากการเคลื่อนที่ของมือผู้ใช้งาน สามารถดูจากตารางที่ 3-1

ตารางที่ 3-1 คำสั่งที่ใช้งานสำหรับตรวจจับท่าทางและมุมการเคลื่อนที่ของมือผู้ใช้งาน

รูปแบบคำสั่ง	ท่าทางที่ตรวจจับ
Hands.Count	ตรวจนับจำนวนมือของผู้ใช้งาน
Hands.GrabStrength	ตรวจจับท่าทางการกำมือ
FingerType.TYPE_INDEX	ตรวจจับท่าทางการกระดิกนิ้วชี้
FingerType.TYPE_MIDDLE	ตรวจจับท่าทางการกระดิกนิ้วกลาง
Hands.PalmNormal.Roll	ตรวจจับการเอียงมือไปทางด้านซ้ายหรือขวา
Hands.Direction.Pitch	ตรวจจับการเอียงมือไปทางด้านหน้าหรือหลัง
Hands.Direction.Yaw	ตรวจจับการหมุนมือไปทางทิศทวนเข็มนาฬิกาหรือตามเข็มนาฬิกา

3.1.4 พัฒนาโปรแกรมประมวลผลท่าทางและมุม Roll, Pitch, Yaw ของมือผู้ใช้งานที่ตรวจจับได้

ในส่วนนี้จะเริ่มประมวลผลการกำมือ การกระดิกนิ้วชี้และนิ้วกลาง และค่าของมุม Roll Pitch Yaw เฉพาะกรณีที่ตรวจจับได้ว่ามีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion แล้วเท่านั้น โดยแบ่งลักษณะท่าทางที่ตรวจจับออกเป็น

1. ท่าทางการแบมือบนอุปกรณ์ Leap Motion เพื่อรับค่าของมุม Roll, Pitch, Yaw จะใช้คำสั่ง Hands.PalmNormal, Hands.Direction.Pitch, และ Hands.Direction.Yaw ในการตรวจจับ ซึ่งจะทำให้ได้ค่าของมุมทั้ง 3 มุมดังกล่าวในช่วง -180 ถึง 180 องศา ซึ่งจะอยู่ในรูปแบบของตัวเลข

จำนวนเต็ม แต่เราจะพิจารณาเฉพาะค่าในช่วง -100 ถึง 100 องศา เท่านั้นเนื่องจากเป็นองศาที่อยู่ในช่วงที่มือของผู้ใช้งานสามารถเคลื่อนที่ไปให้ถึงได้อย่างสะดวก

2. ทำทางการกำมือและแบมือบนอุปกรณ์ Leap Motion จะใช้คำสั่ง Hands.GrabStrength ในการตรวจจับมือของผู้ใช้งานว่ามีการกำมือหรือแบมืออยู่ โดยในการพัฒนาได้กำหนดไว้ว่าถ้าตรวจจับได้ว่ามีการกำมืออยู่จะให้ค่าเท่ากับ 1 และถ้าตรวจจับได้ว่าแบมืออยู่จะให้ค่าเท่ากับ 0 ซึ่งผลลัพธ์ดังกล่าวนี้มีคุณสมบัติเป็นตัวเลขจำนวนเต็ม

3. การกระดิกนิ้วชี้และนิ้วกลาง จะใช้คำสั่ง FingerType.TYPE_INDEX และ FingerType.TYPE_MIDDLE ในการตรวจจับ และในการพัฒนาจะกำหนดให้ผลลัพธ์ที่ได้มีลักษณะการเก็บค่าแบบ Toggle คือการสลับค่าไปมาระหว่างค่า 0 และ 1 ซึ่งผลลัพธ์ดังกล่าวมีคุณสมบัติเป็นตัวเลขจำนวนเต็ม และการเก็บผลลัพธ์จะทำการแยกตัวแปรสำหรับค่าของนิ้วชี้และนิ้วกลางออกจากกัน

สำหรับผลลัพธ์ที่กำหนดให้กับท่าทางต่าง ๆ ที่ตรวจจับได้ สามารถตรวจสอบได้จากตารางที่ 3-2 และในกรณีที่มือของผู้ใช้งานไม่ได้อยู่บนอุปกรณ์ Leap Motion ค่าของการกำมือ การกระดิกนิ้วชี้ การกระดิกนิ้วกลาง และค่าของมุม Roll Pitch Yaw จะถูกกำหนดให้เท่ากับ 0 ไว้เสมอ

ตารางที่ 3-2 ค่าที่นำมาใช้งานโดยพิจารณาจากท่าทางที่แตกต่างกันของมือ

ลักษณะของมือ	ค่าที่ใช้งาน
1. มีมือ / ไม่มีมือ อยู่บนอุปกรณ์ Leap Motion	มีมือจะได้ค่าเท่ากับ 1 / ไม่มีมือจะได้ค่าเท่ากับ 0
2. กำมือ / ไม่กำมือ อยู่บนอุปกรณ์ Leap Motion	กำมือจะได้ค่าเท่ากับ 1 / แบมือจะได้ค่าเท่ากับ 0
3. กระดิกนิ้วชี้	สลับค่าไปมาระหว่าง 0 และ 1 เมื่อกระดิกนิ้วชี้
4. กระดิกนิ้วกลาง	สลับค่าไปมาระหว่าง 0 และ 1 เมื่อกระดิกนิ้วกลาง
5. แบมือบนอุปกรณ์ Leap Motion และมีการเอียงซ้าย หรือ ขวา เป็นการตรวจจับในค่า Roll	- เอียงไปทางซ้ายจะได้ค่าในช่วง 0 ถึง 100 - เอียงไปทางขวาจะได้ค่าในช่วง 0 ถึง (-100)

ตารางที่ 3-2 (ต่อ) ค่าที่นำมาใช้งานโดยพิจารณาจากท่าทางที่แตกต่างกันของมือ

6. แบบมือบนอุปกรณ์ Leap Motion และมีการเอียงไปด้านหน้า หรือ หลัง เป็นการตรวจจับในค่า Pitch	- เอียงไปด้านหน้าจะได้ค่าในช่วง 0 ถึง (-100) - เอียงไปด้านหลังจะได้ค่าในช่วง 0 ถึง 100
7. แบบมือบนอุปกรณ์ Leap Motion และมีการหมุนมือตามเข็มนาฬิกา หรือ ทวนเข็มนาฬิกา เป็นการตรวจจับในค่า Yaw	- หมุนตามเข็มนาฬิกาจะได้ค่าในช่วง 0 ถึง (-100) - หมุนทวนเข็มนาฬิกาจะได้ค่าในช่วง 0 ถึง 100

3.1.5 พัฒนาโปรแกรมแสดงผลการทำงานของระบบในรูปแบบ Desktop Application

ถ้าพิจารณาจากภาพที่ 3-2 คือส่วนของ Update desktop application (10 Hz) โดยในส่วนนี้จะพัฒนาให้มีการแสดงผลทุก ๆ 0.1 วินาทีเท่านั้น เนื่องจากการทดสอบพบว่าถ้าใช้เวลาสำหรับแสดงผลเร็วกว่า 0.1 วินาที จะทำให้หน้าแสดงผลมีอาการกระตุก ซึ่งเกิดจากการสลับรูปภาพอย่างรวดเร็วในหลาย ๆ ตำแหน่งพร้อมกัน และถ้าใช้เวลาสำหรับการแสดงผลมากกว่า 0.1 วินาที พบว่าการแสดงผลจะมีความล่าช้ามากจนเกินไป ซึ่งอาจทำให้ผู้ควบคุมการบินของมัลติโรเตอร์เกิดความผิดพลาดได้ เช่น ผู้ควบคุมหยุดการเอียงมือไปด้านซ้ายเรียบร้อยแล้วแต่ถ้าหน้าจอยังแสดงผลอยู่ อาจทำให้ผู้ควบคุมเข้าใจผิดและพยายามเอียงมือไปทางด้านขวามากขึ้นเพื่อให้มือตั้งตรง ซึ่งจะส่งผลให้มัลติโรเตอร์เกิดการเคลื่อนที่ไปด้านขวาแทน และสำหรับการพัฒนาโปรแกรมแสดงผลการทำงานนี้ มีการทำงานแบ่งออกเป็น 5 ส่วนหลักได้แก่

1. ส่วนสำหรับทำหน้าที่ในการกำหนดการเริ่มต้นส่งข้อมูลหรือการหยุดการส่งข้อมูลสำหรับการพัฒนาได้ทำเป็นรูปภาพ 2 รูปซ้อนทับกันอยู่ โดยเริ่มต้นจะให้ เป็นรูปภาพของปุ่มที่มีลักษณะสีแดง และเมื่อมีการคลิกในบริเวณของรูปภาพดังกล่าวจะทำการแสดงรูปภาพอีกรูปขึ้นมา ซึ่งมีลักษณะเป็นปุ่มกดสีเขียว พร้อมทั้งส่งคำสั่งสำหรับเชื่อมต่อ Port ระหว่างส่วนควบคุมภาคพื้นดิน กับ ส่วนควบคุมภาคอากาศ เพื่อเป็นการเริ่มต้นส่งข้อมูล แต่ถ้ามีการคลิกในบริเวณของรูปภาพดังกล่าวอีกครั้งจะทำให้แสดงรูปภาพของปุ่มกดสีแดงขึ้นมาแทน พร้อมทั้งส่งคำสั่งปิดการเชื่อมต่อ Port ระหว่างส่วนควบคุมภาคพื้นดินกับส่วนควบคุมภาคอากาศ

2. ส่วนสำหรับแสดงค่า Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel ซึ่งเป็นค่าที่ใช้ในการควบคุมการบินของมัลติโรเตอร์ โดยส่วนนี้จะเป็นการรับค่ากลับมาจากส่วนควบคุมภาคอากาศ เนื่องจากทางส่วนควบคุมภาคพื้นดินจะส่งค่าออกไปในรูปแบบ 1 ถึง 9 และให้ส่วนควบคุมภาคอากาศไปแปลงให้อยู่ในช่วง 1250 ถึง 1750 แล้วจึงส่งค่ากลับมายังส่วนควบคุมภาคพื้นดินเพื่อใช้สำหรับแสดงผล

3. ส่วนสำหรับแสดงมุมของ Roll, Pitch, Yaw ที่ตรวจจับได้จากการเคลื่อนไหวของมือผู้ใช้งาน ในการพัฒนาจะทำการแสดงในลักษณะของตัวเลขจำนวนเต็ม ซึ่งค่าของมุมทั้ง 3 ดังกล่าวจะถูกแสดงอยู่ในช่วงของ -100 ถึง 100 เท่านั้น

4. ส่วนแสดงรูปภาพแนวการเคลื่อนที่ของมัลติโรเตอร์ทั้ง 6 แนวการเคลื่อนที่ โดยการเคลื่อนที่ไปด้านซ้ายหรือขวาจะพิจารณาจาก Roll Channel การเคลื่อนที่ไปด้านหน้าหรือด้านหลังจะพิจารณาจาก Pitch Channel และการเคลื่อนที่หมุนตัวลำทวนเข็มนาฬิกาหรือตามเข็มนาฬิกาจะพิจารณาจาก Yaw Channel ซึ่งค่าของ Roll Channel, Pitch Channel, Yaw Channel จะรับมาจากส่วนควบคุมภาคอากาศ และสำหรับการพัฒนาการแสดงรูปภาพของแต่ละ Channel ทั้ง 3 นี้ จะใช้หลักการเดียวกันคือมีรูปภาพซ้อนกันอยู่ 3 รูปภาพด้วยกัน โดยมีการแสดงรูปภาพที่แตกต่างกันดังนี้

การพัฒนาสำหรับส่วนของ Roll Channel จะแสดงรูปมัลติโรเตอร์เอียงไปทางด้านซ้ายก็ต่อเมื่อค่าของ Roll Channel น้อยกว่า 1500, แสดงรูปของมัลติโรเตอร์ที่เอียงไปทางด้านขวาต่อเมื่อค่าของ Roll Channel มากกว่า 1500, และแสดงรูปของมัลติโรเตอร์ตั้งตรงในมุมมองจากด้านหน้าของตัวลำก็ต่อเมื่อค่าของ Roll Channel เท่ากับ 1500

การพัฒนาสำหรับส่วนของ Pitch Channel จะแสดงรูปมัลติโรเตอร์เอียงไปทางด้านหน้าก็ต่อเมื่อค่าของ Pitch Channel น้อยกว่า 1500, แสดงรูปของมัลติโรเตอร์ที่เอียงไปทางด้านหลังต่อเมื่อค่าของ Pitch Channel มากกว่า 1500, และแสดงรูปของมัลติโรเตอร์ตั้งตรงในมุมมองจากด้านข้างของตัวลำก็ต่อเมื่อค่าของ Pitch Channel เท่ากับ 1500

การพัฒนาสำหรับส่วนของ Yaw Channel จะแสดงรูปมัลติโรเตอร์พร้อมกับลูกศรที่หมุนในทิศทวนเข็มนาฬิกาก็ต่อเมื่อค่าของ Yaw Channel น้อยกว่า 1500, แสดงรูปของมัลติโรเตอร์พร้อมกับลูกศรที่หมุนในทิศตามเข็มนาฬิกาต่อเมื่อค่าของ Yaw Channel มากกว่า 1500, และแสดงรูปของมัลติโรเตอร์ตั้งตรงในมุมมองจากด้านบนของตัวลำก็ต่อเมื่อค่าของ Yaw Channel เท่ากับ 1500

5. ส่วนแสดงสถานะของมือขณะมีมือหรือไม่มีมืออยู่บนอุปกรณ์ Leap Motion และขณะกำมือหรือแบมืออยู่บนอุปกรณ์ Leap Motion ในการพัฒนาได้ใช้วิธีการซ้อนรูปอยู่ในตำแหน่งเดียวกันทั้งหมด 3 รูปได้แก่ รูปแบมือสีเนื้อ รูปแบมือสีเทา และรูปกำมือสีเนื้อ โดยใน 2 รูปแรกจะพิจารณาผลลัพธ์ที่ได้จากคำสั่ง Hands.Count โดยถ้าผลลัพธ์เท่ากับ 0 จะแสดงรูปแบมือสีเทา แต่ถ้าผลลัพธ์เท่ากับ 1 จะแสดงรูปแบมือสีเนื้อ และรูปที่ 3 จะพิจารณาผลลัพธ์ที่ได้จากคำสั่ง Hands.Count ร่วมกับผลลัพธ์จากคำสั่ง Hands.GrabStrength ซึ่งถ้ามีค่าเท่ากับ 1 ทั้งคู่จะแสดงรูปภาพกำมือสีเนื้อขึ้นมา แต่ถ้าผลลัพธ์ที่ได้จากคำสั่ง Hands.Count เท่ากับ 1 แต่ผลลัพธ์ที่ได้จากคำสั่ง Hands.GrabStrength เท่ากับ 0 จะแสดงรูปแบมือสีเนื้อขึ้นมาแทน

3.1.6 พัฒนาโปรแกรมการส่งข้อมูลของส่วนควบคุมภาคพื้นดิน

ในการพัฒนาจะกำหนดความถี่ในการส่งข้อมูลอยู่ที่ 0.5 วินาทีต่อ 1 ชุดข้อความ และใช้รูปแบบการส่งข้อมูล 1 ชุดในลักษณะที่เริ่มต้นด้วยเครื่องหมาย “ < ” ซึ่งกำหนดให้เป็น Start Message และปิดท้ายข้อมูลด้วยเครื่องหมาย “ > ” ซึ่งกำหนดให้เป็น End Message และภายในข้อมูล 1 ชุดนี้จะประกอบด้วย X1, X2, X3, X4, X5, X6, X7 โดยค่าของ X1 ถึง X7 สามารถตรวจสอบได้จากตารางที่ 3-3 และการส่งข้อมูลนี้ ก่อนจะเริ่มทำการส่งข้อมูลออกไปให้กับส่วนควบคุมภาคอากาศ ได้พัฒนาให้มีการตรวจสอบก่อนว่ามีการกดปุ่มเพื่อเชื่อมต่อ Port ระหว่างส่วนควบคุมภาคพื้นดินกับ ส่วนควบคุมภาคอากาศแล้วหรือไม่ ถ้ามีเรียบร้อยแล้วจึงทำการส่งชุดข้อมูลดังกล่าวออกไป

ตารางที่ 3-3 ค่าที่ใช้งานสำหรับตัวแปร X1 ถึง X7 ที่ใช้ในการส่งข้อมูลจากส่วนควบคุมภาคพื้นดิน

ตัวแปร	ค่าที่ใช้งาน
X1	มีมือเท่ากับ 1, ไม่มีมือเท่ากับ 0
X2	กำมือเท่ากับ 1, ไม่กำมือเท่ากับ 0
X3	กระดิกนิ้วชี้จะทำการสลับค่าระหว่าง 0 กับ 1
X4	กระดิกนิ้วกลางจะทำการสลับค่าระหว่าง 0 กับ 1
X5	- แบนมือบนอุปกรณ์ Leap Motion และเอียงมือทางซ้าย จะได้ค่าในช่วง 6 ถึง 9 - แบนมือบนอุปกรณ์ Leap Motion และเอียงมือทางขวา จะได้ค่าในช่วง 1 ถึง 4 - แบนมือตรงบนอุปกรณ์ Leap Motion จะได้ค่าเท่ากับ 5
X6	- แบนมือบนอุปกรณ์ Leap Motion และเอียงมือไปด้านหน้า จะได้ค่าในช่วง 1 ถึง 4 - แบนมือบนอุปกรณ์ Leap Motion และเอียงมือไปด้านหลัง จะได้ค่าในช่วง 6 ถึง 9 - แบนมือตรงบนอุปกรณ์ Leap Motion จะได้ค่าเท่ากับ 5
X7	- แบนมือบนอุปกรณ์ Leap Motion และหมุนมือตามเข็มนาฬิกา จะได้ค่าในช่วง 1 ถึง 4 - แบนมือบนอุปกรณ์ Leap Motion และหมุนมือทวนเข็มนาฬิกา จะได้ค่าในช่วง 6 ถึง 9 - แบนมือตรงบนอุปกรณ์ Leap Motion จะได้ค่าเท่ากับ 5

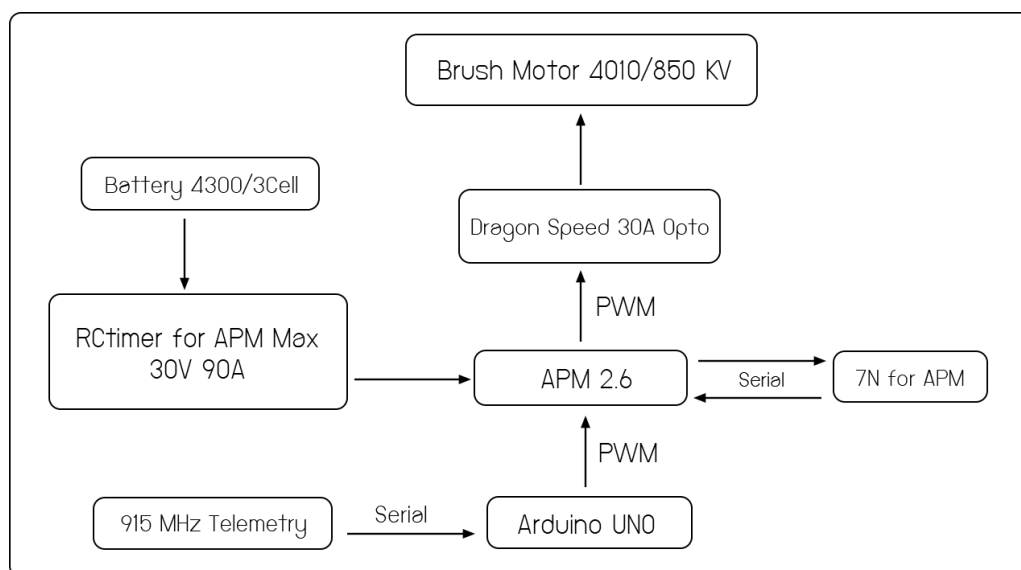
3.2 ระบบของส่วนควบคุมภาคอากาศ

จากส่วนควบคุมภาคพื้นดินจะส่งข้อมูลที่ผ่านการประมวลผลแล้วมาให้กับส่วนควบคุมภาคอากาศผ่านอุปกรณ์ Wireless Datalink และส่วนควบคุมภาคอากาศจะนำข้อมูลเหล่านั้นมาทำการประมวลผลอีกครั้งเพื่อนำไปสร้างเป็นสัญญาณ PWM จำนวน 5 ช่อง และป้อนให้กับ Flight Controller ทางด้านอินพุต เพื่อให้ Flight Controller พิจารณาค่าของอินพุตที่แตกต่างกันทั้ง 5 ช่อง และนำไปจัดการกับแอคทูเอเตอร์ที่ใช้สำหรับควบคุมความเร็วมอเตอร์ทั้ง 4 ตัว เพื่อให้ผู้ใช้งานสามารถควบคุมการบินในลักษณะต่าง ๆ ของมัลติโรเตอร์ได้

สำหรับขั้นตอนการพัฒนาระบบในส่วนของส่วนควบคุมภาคอากาศนั้นมีขั้นตอนการพัฒนาดังนี้

1. ออกแบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคอากาศ
 2. ออกแบบภาพรวมของโปรแกรมส่วนควบคุมภาคอากาศ
 3. พัฒนาโปรแกรมสำหรับตั้งค่าเริ่มต้นให้กับ Wireless Datalink และ Servo Motor ทั้ง 4 ตัว
 4. พัฒนาโปรแกรมสำหรับรับส่งข้อมูลกับส่วนควบคุมภาคพื้นดินและตรวจสอบตำแหน่งข้อมูล
 5. พัฒนาโปรแกรมสำหรับตรวจสอบความยาวข้อมูล
 6. พัฒนาโปรแกรมสำหรับหาค่าของข้อมูลที่เหมือนกันมากที่สุดในปีเฟอร์
 7. พัฒนาโปรแกรมสำหรับแปลงค่าของ Roll, Pitch, Yaw
 8. พัฒนาโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่มีมือควบคุม
 9. พัฒนาโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่ไม่มีมือควบคุม และสัญญาณการเชื่อมต่อขาดหาย
 10. พัฒนาโปรแกรมสำหรับสร้างสัญญาณ PWM ตามค่าที่ถูกกำหนด
- #### 3.2.1 ออกแบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคอากาศ

มีลักษณะการเชื่อมต่อของอุปกรณ์ฮาร์ดแวร์ตามภาพที่ 3-3 โดยมีการเชื่อมต่อคือนำอุปกรณ์ Wireless Datalink เชื่อมต่อเข้ากับ Arduino UNO ผ่านทางสายไฟสำหรับเชื่อมต่อ หลังจากนั้นใช้สายไฟเส้นเดิมสำหรับต่อจาก Arduino UNO ไปเข้าที่ Flight Controller ทางด้าน Input Channel และในส่วนที่เหลือทำการเชื่อมต่ออุปกรณ์ต่าง ๆ ที่จำเป็นสำหรับการบินของมัลติโรเตอร์ ได้แก่ GPS, ESC, Motor, ใบพัด, Battery, Power Module ตามระบบปกติ



ภาพที่ 3-3 รายชื่ออุปกรณ์ที่ใช้งานบนส่วนควบคุมภาคอากาศ

จากภาพที่ 3-3 อุปกรณ์ที่สำคัญในการพัฒนาระบบของโครงการเรามีด้วยกันทั้งหมด 3 อุปกรณ์ ได้แก่ Wireless Datalink, Arduino UNO และ Flight Controller (APM 2.6) ซึ่งแต่ละอุปกรณ์จะมีหน้าที่ดังนี้

- Wireless Datalink มีหน้าที่สำหรับรับส่งข้อมูลกับส่วนควบคุมภาคพื้นดิน โดยมีลักษณะการเชื่อมต่อระหว่าง Wireless Datalink กับ Arduino ตามตารางที่ 3-4

ตารางที่ 3-4 การเชื่อมต่อระหว่าง Wireless Datalink กับ Arduino

Wireless Datalink	Arduino
GND	GND
VCC	5V
RX	Analog Pin 1 (A1)
TX	Analog Pin 0 (A0)

- Arduino UNO มีหน้าที่หลักคือ ใช้สำหรับประมวลผลค่าที่ได้รับจากส่วนควบคุมภาคพื้นดินผ่านทางอุปกรณ์ Wireless Datalink และสร้างสัญญาณ PWM จำนวน 5 ช่องเพื่อป้อนให้กับ Flight Controller ทางด้านอินพุต โดยมีลักษณะการเชื่อมต่อระหว่าง Arduino กับ Flight Controller ตามตารางที่ 3-5

ตารางที่ 3-5 การเชื่อมต่อระหว่าง Arduino กับ Flight Controller

Arduino	Flight Controller
Digital Pin 2	Input Pin 1 (Roll Channel)
Digital Pin 3	Input Pin 2 (Pitch Channel)
Digital Pin 4	Input Pin 3 (Throttle Channel)
Digital Pin 5	Input Pin 4 (Yaw Channel)
Digital Pin 6	Input Pin 5 (Flight Mode Channel)
GND	GND

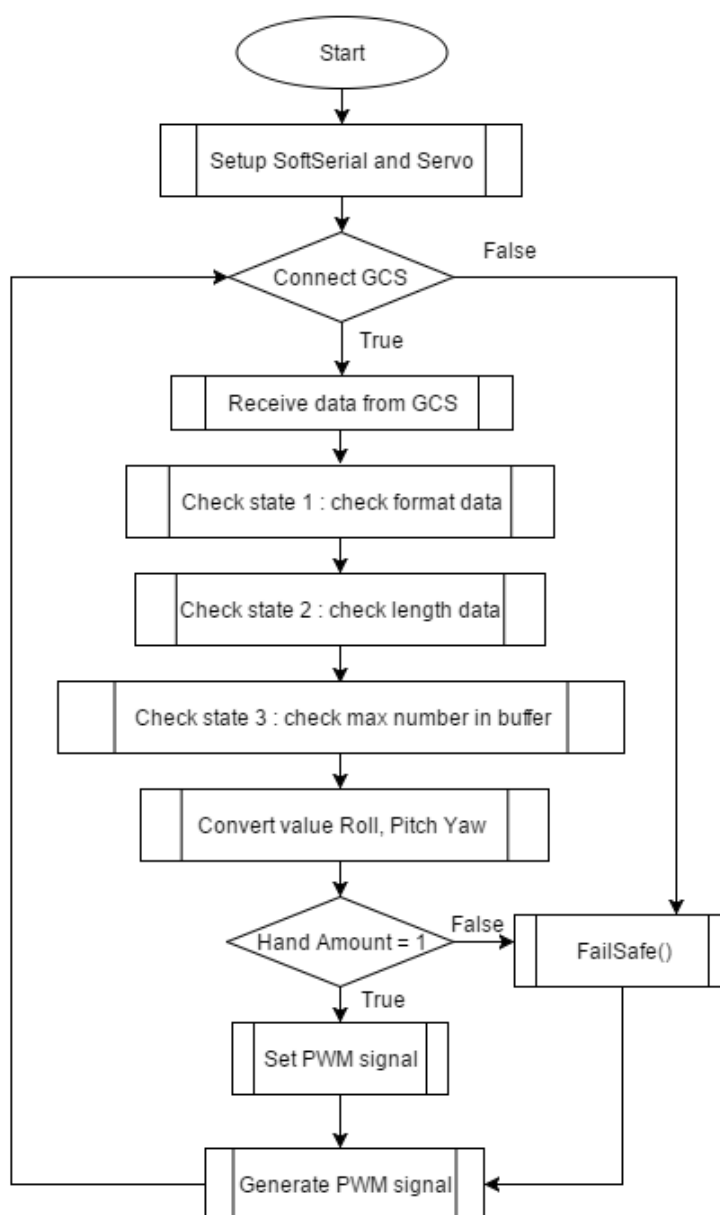
- Flight Controller (APM 2.6) มีหน้าที่สำคัญในการควบคุมการบินของมอเตอร์ในท่าทางต่าง ๆ โดยผู้ใช้งานสามารถควบคุมการทำงานของ Flight Controller ตัวนี้ได้จากการปรับเปลี่ยนค่าของฟังก์ชันพุด โดยมีลักษณะการควบคุมที่แตกต่างกันออกไปทั้ง 5 ช่องอินพุต ตามตารางที่ 3-6

ตารางที่ 3-6 หน้าที่ของแต่ละ Input Channel สำหรับ Flight Controller

Channel	หน้าที่
Input Channel 1	ควบคุมการเคลื่อนที่ในแนวแกน Roll
Input Channel 2	ควบคุมการเคลื่อนที่ในแนวแกน Pitch
Input Channel 3	ควบคุมความเร็วของ Throttle
Input Channel 4	ควบคุมการเคลื่อนที่ในแนวแกน Yaw
Input Channel 5	ควบคุม Flight Mode

3.2.2 ออกแบบภาพรวมของโปรแกรมส่วนควบคุมภาคอากาศ

ระบบทั้งหมดสำหรับส่วนควบคุมภาคอากาศ ทางกลุ่มของเราได้พัฒนาโดยใช้โปรแกรม Arduino Sketch ซึ่งใช้ภาษา C ในการพัฒนาส่วนต่าง ๆ โดยมีภาพรวมการทำงานเริ่มจาก ตั้งค่าเริ่มต้นของระบบให้กับอุปกรณ์ Wireless Datalink และ Servo Motor ทั้ง 4 ตัว หลังจากนั้นทำการตรวจสอบว่ามีข้อมูลเข้ามาจากส่วนควบคุมภาคพื้นดินหรือไม่ โดยถ้ามี จะทำการรับข้อมูลดังกล่าว และทำการตรวจสอบในขั้นตอนที่ 1 ถึง 3 เพื่อให้ข้อมูลสุดท้ายที่ได้มีความถูกต้องมากที่สุด ซึ่งข้อมูลที่ได้แก่ 1.ข้อมูลว่ามีมือหรือไม่ 2.ข้อมูลว่ามีการกำมือหรือไม่ 3.ข้อมูลว่ามีการกระดิกนิ้วชี้หรือไม่ 4.ข้อมูลว่ามีการกระดิกนิ้วกลางหรือไม่ 5.ข้อมูลของมุม Roll, Pitch, Yaw หลังจากนั้นนำค่าผลลัพธ์สุดท้ายที่ได้มาในของส่วน Roll, Pitch, Yaw มาทำการแปลงจากช่วงของตัวเลขจำนวนเต็ม 1 ถึง 9 ไปเป็นช่วง 1000 ถึง 2000 หลังจากนั้นจะทำการตรวจสอบว่ามีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion หรือไม่ ซึ่งถ้ามีมืออยู่บนอุปกรณ์ Leap Motion จะใช้ค่าของ Roll, Pitch, Yaw ที่ได้ทำการแปลงแล้ว มาเป็นตัวกำหนดเพื่อใช้สำหรับสร้างเป็นค่าของสัญญาณ PWM แต่ถ้าตรวจสอบแล้วพบว่าไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion และจากตอนแรกถ้าตรวจสอบแล้วพบว่าไม่มีข้อมูลเข้ามาจากส่วนควบคุมภาคพื้นดิน จะทำการกำหนดค่าของสัญญาณ PWM โดยอัตโนมัติ โดยไม่สนใจค่าของ Roll, Pitch, Yaw ที่รับเข้ามาจากส่วนควบคุมภาคพื้นดิน สามารถดูขั้นตอนการทำงานของระบบโดยรวมได้จากภาพที่ 3-4



ภาพที่ 3-4 ขั้นตอนการทำงานทั้งหมดของระบบสำหรับส่วนควบคุมภาคอากาศ

3.2.3 พัฒนาโปรแกรมสำหรับตั้งค่าเริ่มต้นให้กับ Wireless Datalink และ Servo Motor ทั้ง 4 ตัว

ส่วนตั้งค่าเริ่มต้นของระบบหรือจากภาพที่ 3-4 คือส่วนของ Setup SoftSerial and Servo ส่วนของ Setup SoftSerial คือการเรียกใช้งาน SoftwareSerial Library เพื่อให้สามารถส่ง-รับข้อมูลระหว่าง Wireless Datalink ได้ โดยไม่จำเป็นต้องใช้ Digital Pin 0, 1 ของบอร์ด Arduino UNO โดยในการทดลองนี้ได้กำหนดให้ใช้ Analog Pin 0(rx) และ 1(tx) ของบอร์ด Arduino UNO และตั้งค่า Baudrate เท่ากับ 57,600 บิตต่อวินาที เนื่องจากเป็นข้อกำหนดของอุปกรณ์ที่ใช้งานนั้นคือ 915 MHz

telemetry และสำหรับส่วนของ Setup Servo เป็นการสร้างสัญญาณ PWM โดยใช้การกำหนดลอจิก High และ Low ด้วยคำสั่ง `digitalWrite()` และใช้คำสั่ง `delayMicrosecond()` เพื่อให้สัญญาณมีลอจิกเป็น High หรือ Low ตามเวลาที่ต้องการ ซึ่งในการทดลองนี้ได้กำหนดให้ใช้ Digital Pin 2, 3, 4, 5, 6 ในการสร้างสัญญาณ PWM โดยข้อดีของการใช้งานคำสั่ง คือสามารถควบคุม Servo Motor หรือ Brushless Motor ได้หลายตัวพร้อมกัน หรืออีกทางเลือกในกรณีที่ไม่ต้องการใช้งาน Servo Library คือการใช้งานคำสั่ง `analogWrite()` ในการสร้างสัญญาณ PWM โดยการใช้งานกับ Brushless Motor ได้นั้น จำเป็นต้องใช้งานเฉพาะ Pin ที่สร้างสัญญาณ PWM ได้และต้องมีความถี่ของสัญญาณ PWM อยู่ที่ 400-500 Hz หรือ 50 Hz เท่านั้น ซึ่งจะมี Pin ที่ได้ใช้งานได้เพียง Digital Pin 3, 9, 10, 11 เท่านั้น ซึ่งจะพบว่าไม่เพียงพอต่อการใช้งานในการทดลองครั้งนี้ที่ต้องการใช้งาน Pin ทั้งหมด 5 Pin ด้วยกัน

3.2.4 พัฒนาโปรแกรมสำหรับรับส่งข้อมูลกับส่วนควบคุมภาคพื้นดินและตรวจสอบตำแหน่งข้อมูล

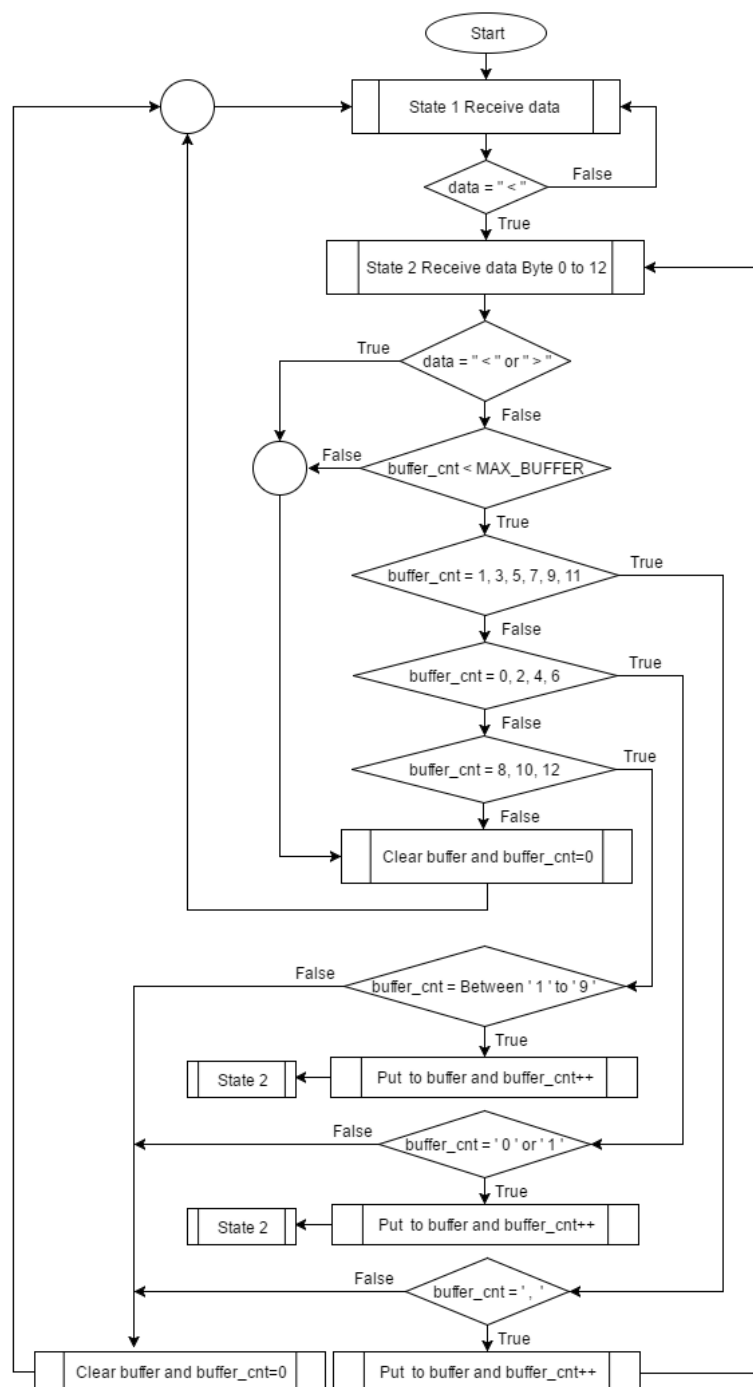
สำหรับส่วนรับข้อมูลจากส่วนควบคุมภาคพื้นดินและตรวจสอบตำแหน่งข้อมูล เป็นการตรวจสอบในขั้นตอนที่ 1 หรือจากภาพที่ 3-4 คือส่วนของ Receive data from GCS และส่วนของ Check State 1 โดยทั้งสองส่วนนี้จะทำงานควบคู่กันและสามารถเขียนขั้นตอนการทำงานได้ดังนี้

1. ทำการรับข้อมูลจากส่วนควบคุมภาคพื้นดินเข้ามาทีละ 1 Byte และทำการตรวจสอบในทุก ๆ Byte ที่เข้ามา ซึ่งถ้าไม่ตรงตามข้อกำหนดที่ตกลงกันไว้ จะทำการเคลียร์ฟลอร์และไปรอรับข้อมูลชุดใหม่ทันที โดยเริ่มต้นจาก State ที่ 1 คือการตรวจสอบข้อมูลแต่ละ Byte ที่เข้ามาจะต้องเป็นเครื่องหมาย "<" เท่านั้น ซึ่งทางผู้จัดทำได้กำหนดไว้ให้เป็น Start Message และเมื่อเจอเครื่องหมาย "<" ตามที่ต้องการแล้ว จึงจะส่งไปยัง State ที่ 2 แต่ถ้ายังไม่เจอเครื่องหมาย "<" จะทำการรออยู่ใน State ที่ 1 จนกว่าจะเจอเครื่องหมาย "<"

2. เมื่อเข้ามาใน State ที่ 2 จะทำการตรวจสอบจากตัวแปร `*buffer_cnt` ตามตำแหน่งของข้อมูลที่ได้รับเข้ามาโดยกำหนดไว้ว่าในตำแหน่งที่ 0, 2, 4, 6 ข้อมูลจะเป็นได้แค่ค่าระหว่าง 0 และ 1 เท่านั้น และข้อมูลในตำแหน่งที่ 1, 3, 5, 7, 9, 11 จะต้องเป็นเครื่องหมาย ";" เท่านั้น และข้อมูลในตำแหน่งที่ 8, 10, 12 จะต้องเป็นได้แค่ค่าระหว่าง 1 ถึง 9 เท่านั้น ซึ่งถ้าไม่ตรงตามเงื่อนไขดังกล่าวจะทำการเคลียร์ฟลอร์และส่งกลับไปยัง State ที่ 1 ทันทีเพื่อรอรับข้อมูลชุดใหม่

3. ถ้าอยู่ใน State ที่ 2 แล้วเจอเครื่องหมาย ">" ซึ่งทางผู้จัดทำได้กำหนดไว้ให้เป็น End Message จะทำการเคลียร์ฟลอร์และส่งกลับไปยัง State ที่ 1 เพื่อรอรับข้อมูลชุดใหม่ทันที

จากขั้นตอนทั้ง 3 ขั้นตอนสามารถเขียนอธิบายเป็น Flow-Chart ได้ตามภาพที่ 3-5



ภาพที่ 3-5 ขั้นตอนการทำงานสำหรับรับข้อมูลจากส่วนควบคุมภาคพื้นดินและตรวจสอบขั้นที่ 1
(ตรวจสอบความถูกต้องของข้อมูลในแต่ละตำแหน่ง)

สำหรับส่วนส่งข้อมูลมีหน้าที่ส่งค่าของ Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel กลับไปส่วนควบคุมภาคพื้นดินผ่านอุปกรณ์ Wireless Datalink เพื่อให้ส่วนควบคุมภาคพื้นดินสามารถแสดงค่าต่าง ๆ เหล่านี้บน Desktop Application เพื่อให้ผู้ใช้งานสามารถตรวจสอบค่าต่าง ๆ ได้ ซึ่งค่าเหล่านี้เป็นค่าที่สำคัญในการควบคุมการบินของมัลติโรเตอร์

3.2.5 พัฒนาโปรแกรมสำหรับตรวจสอบความยาวข้อมูล

ส่วนตรวจสอบความยาวข้อมูล เป็นการตรวจสอบในขั้นตอนที่ 2 หรือจากภาพที่ 3-4 คือส่วนของ Check State 2 เป็นส่วนสำหรับตรวจสอบความยาวของข้อมูลใน 1 ชุดที่ผ่านการตรวจสอบจากขั้นที่ 1 มาแล้ว โดยทำการตรวจสอบจากค่าของตัวแปร buffer_cnt ว่ามีค่าเท่ากับ 13 หรือไม่ ถ้าเป็นจริงจึงทำการแยกข้อมูลใน 1 ชุดออก และทำการเก็บเข้าไปยังตัวแปรของแต่ละชนิด โดยข้อมูลใน 1 ชุดที่รับเข้ามาจะประกอบไปด้วย X1, X2, X3, X4, X5, X6, X7 ซึ่งมีความยาวของข้อมูลเท่ากับ 13 โดยความหมายของข้อมูลแต่ละตัวสามารถดูได้จากตารางที่ 3-7 และสำหรับกรณีที่ความยาวของข้อมูลไม่เท่ากับ 13 จะทำการไปรอรับข้อมูลชุดใหม่เข้ามาทันที

ตารางที่ 3-7 ความหมายของประเภทข้อมูลสำหรับตัวแปร X1 ถึง X7 ที่ใช้ในการส่งข้อมูล

ประเภทของข้อมูล	ความหมาย
X1	เก็บค่าของจำนวนมือที่ตรวจจับได้ โดยมีค่าระหว่าง 0, 1
X2	เก็บค่าของท่าทางการกำมือที่ตรวจจับได้ โดยมีค่าระหว่าง 0, 1
X3	เก็บค่าของท่าทางการกระดิกนิ้วชี้ โดยมีค่าระหว่าง 0, 1
X4	เก็บค่าของท่าทางการกระดิกนิ้วกลาง โดยมีค่าระหว่าง 0, 1
X5	เก็บค่าของมือในมุม Roll โดยมีค่าระหว่าง 1-9
X6	เก็บค่าของมือในมุม Pitch โดยมีค่าระหว่าง 1-9
X7	เก็บค่าของมือในมุม Yaw โดยมีค่าระหว่าง 1-9

3.2.6 พัฒนาโปรแกรมสำหรับหาค่าของข้อมูลที่เหมือนกันมากที่สุดในปีเฟอร์

ส่วนตรวจสอบเพื่อหาค่าของข้อมูลที่เหมือนกันมากที่สุดในปีเฟอร์ เป็นการตรวจสอบในขั้นตอนที่ 3 หรือจากภาพที่ 3-4 คือส่วนของ Check State 3 จะทำหน้าที่ตรวจสอบข้อมูลที่ถูกรับอยู่ในตัวแปร ซึ่งเป็นแบบ Array ของแต่ละชนิด โดยมีทั้งหมด 7 ชนิดข้อมูล โดยกำหนดให้ Array ที่ใช้มีขนาดเท่ากับ 3 และทำการตรวจสอบเพื่อหาค่าที่เหมือนกันมากที่สุดที่อยู่ใน Array ณ ขณะนั้น โดยทำการตรวจสอบในทุก ๆ ครั้งที่มีข้อมูลผ่านการตรวจสอบจากขั้นตอนที่ 1 และ 2 เข้ามาได้ ซึ่ง

การทำเช่นนี้เพื่อป้องกันข้อมูลบางตัวที่สามารถผ่านการตรวจในขั้นที่ 1 และ 2 มาได้แต่ไม่ใช่ข้อมูล ที่ถูกต้อง ยกตัวอย่างเช่น ขณะมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion เพราะฉะนั้นค่าของ X1 ควรจะเป็น 1 ตลอด แต่ในระหว่างการส่งข้อมูลอาจเกิดข้อผิดพลาดทำให้ค่าของ X1 ที่ถูกส่งออกมา มีค่าเท่ากับ 0 ชั่วขณะ ซึ่งการใช้วิธีนี้จะสามารถป้องกันปัญหาเช่นนี้ได้ ซึ่งขั้นตอนการทำการดังกล่าว สามารถอธิบายออกมาในรูปแบบตารางได้ตามตารางที่ 3-8

ตารางที่ 3-8 ขั้นตอนการตรวจสอบข้อมูลขั้นที่ 3 (ตรวจสอบค่าที่เหมือนกันมากที่สุดในปีเฟอร์)

ข้อมูลที่ส่งเข้ามาจาก GCS	ข้อมูลภายใน Array	ผลลัพธ์ที่ได้
X1=1	[0]=1	1
X1=1	[0]=1, [1]=1	1
X1=1	[0]=1, [1]=1, [2]=1	1
X1=0	[0]=0, [1]=1, [2]=1	1
X1=1	[0]=0, [1]=1, [2]=1	1
X1=0	[0]=0, [1]=1, [2]=0	0

3.2.7 พัฒนาโปรแกรมสำหรับแปลงค่าของ Roll, Pitch, Yaw

ส่วนสำหรับแปลงค่าของ Roll, Pitch, Yaw หรือจากภาพที่ 3-4 คือส่วนของ Convert Roll, Pitch, Yaw เป็นอีกส่วนที่สำคัญเนื่องจากความจริงแล้วค่าของ Roll, Pitch, Yaw จะสามารถนำไปใช้ได้ควรมีค่าอยู่ในช่วงระหว่าง 1000-2000 แต่การที่จะส่งข้อมูลของค่า X5, X6, X7 ซึ่งก็คือค่าของ Roll, Pitch, Yaw ตามลำดับ ออกมาให้อยู่ในช่วง 1000-2000 นั้นมีอัตราข้อผิดพลาดเกิดขึ้นค่อนข้างสูง ยกตัวอย่างเช่น ค่าที่ถูกต้องคือ 1500, 1500, 1500 แต่ค่าที่ฝั่ง Arduino UNO รับเข้ามาได้นั้น กลายเป็น 1560, 1850, 1500 ซึ่งจะเห็นว่าเป็นค่าที่ผิดพลาดแต่สามารถใช้งานได้ แต่จะส่งผลเสียต่อการสร้างสัญญาณ PWM อย่างมาก โดยสังเกตจากค่าที่ผิดพลาดของ X6 ซึ่งก็คือค่าของ Pitch ถ้าไม่มีการแก้ปัญหาก็จะส่งผลให้การควบคุมมอเตอร์ไม่เนิ่งแบบที่ผู้ใช้งานต้องการ แต่จะส่งผลให้มอเตอร์นั้นมีการบินไปด้านหน้าหรือด้านหลังแทน

ทางผู้จัดทำจึงได้แก้ปัญหาด้วยการกำหนดให้ค่าของ X5, X6, X7 ถูกส่งออกมาจากส่วนควบคุมภาคพื้นดินแบบเป็นตัวเลขเพียงหลักเดียว ซึ่งค่าที่ใช้ได้แก่ 1 ถึง 9 เพื่อลดอัตราความผิดพลาดที่เกิดขึ้น และใช้วิธีการตรวจสอบทั้ง 3 ขั้นตอนเช่นเดียวกับค่าของ X1, X2, X3, X4

เมื่อได้ค่าที่ถูกต้องสำหรับ X5, X6, X7 เรียบร้อยแล้ว จะทำการใช้คำสั่ง map() ในโปรแกรม Arduino Sketch เพื่อทำการแปลงค่าจากช่วง 1 ถึง 9 ให้อยู่ในช่วงของ 1250 ถึง 1750 ซึ่ง

เหตุผลที่ผู้จัดทำแปลงค่าออกมาให้อยู่เฉพาะช่วงดังกล่าว เนื่องจากการทดสอบบินจริงด้วยอุปกรณ์วิทยุคอนโทรลพบว่า ถ้าดันสติกไปจนสุดที่ 1000 หรือ 2000 จะทำให้มอเตอร์เคลื่อนที่ไปในทิศทางนั้นได้เร็วขึ้น แต่ทำให้ความสูงของมอเตอร์นั้นต่ำลงเรื่อย ๆ จึงป้องกันเหตุการณ์นี้ด้วยการกำหนดค่าไว้ให้อยู่เฉพาะช่วง 1250 ถึง 1750 เท่านั้น ซึ่งเพียงพอต่อความเร็วในการเคลื่อนที่ในทิศทางต่าง ๆ โดยค่าของการแปลงจากช่วง 1 ถึง 9 ให้อยู่ช่วงของ 1250 ถึง 1750 สามารถดูได้จากตารางที่ 3-9, 3-10, 3-11 ซึ่งแสดงในส่วนของมุม Roll, Pitch, Yaw ตามลำดับ พร้อมทั้งอธิบายช่วงขององศาในมุมของ Roll, Pitch, Yaw ที่ตรวจจับได้จากมือของผู้ใช้งาน ในการเปลี่ยนไปยังค่าในช่วงของ 1 ถึง 9

ตารางที่ 3-9 แสดงการเปลี่ยนแปลงค่าองศาของมือไปเป็นค่าสำหรับกำหนดให้กับสัญญาณ PWM
ของมุม Roll

องศาของมือ	ช่วงตัวเลขสำหรับส่งข้อมูล	ช่วงตัวเลขสำหรับกำหนดให้กับ สัญญาณ PWM
-100 ถึง -81	1	1750
-80 ถึง -61	2	1687
-60 ถึง -41	3	1625
-40 ถึง -21	4	1562
-20 ถึง 20	5	1500
21 ถึง 40	6	1437
41 ถึง 60	7	1375
61 ถึง 80	8	1312
81 ถึง 100	9	1250

ตารางที่ 3-10 แสดงการเปลี่ยนแปลงค่าองศาของมือไปเป็นค่าสำหรับกำหนดให้กับสัญญาณ PWM
ของมุม Pitch

องศาของมือ	ช่วงตัวเลขสำหรับส่งข้อมูล	ช่วงตัวเลขสำหรับกำหนดให้กับ สัญญาณ PWM
-100 ถึง -81	1	1250
-80 ถึง -61	2	1312
-60 ถึง -41	3	1375
-40 ถึง -21	4	1437
-20 ถึง 20	5	1500
21 ถึง 40	6	1562
41 ถึง 60	7	1625
61 ถึง 80	8	1687
81 ถึง 100	9	1750

ตารางที่ 3-11 แสดงการเปลี่ยนแปลงค่าองศาของมือไปเป็นค่าสำหรับกำหนดให้กับสัญญาณ PWM
ของมุม Yaw

องศาของมือ	ช่วงตัวเลขสำหรับส่งข้อมูล	ช่วงตัวเลขสำหรับกำหนดให้กับ สัญญาณ PWM
-100 ถึง -81	1	1750
-80 ถึง -61	2	1687
-60 ถึง -41	3	1625
-40 ถึง -21	4	1562
-20 ถึง 20	5	1500
21 ถึง 40	6	1437
41 ถึง 60	7	1375
61 ถึง 80	8	1312
81 ถึง 100	9	1250

3.2.8 พัฒนาโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่มีมือควบคุม

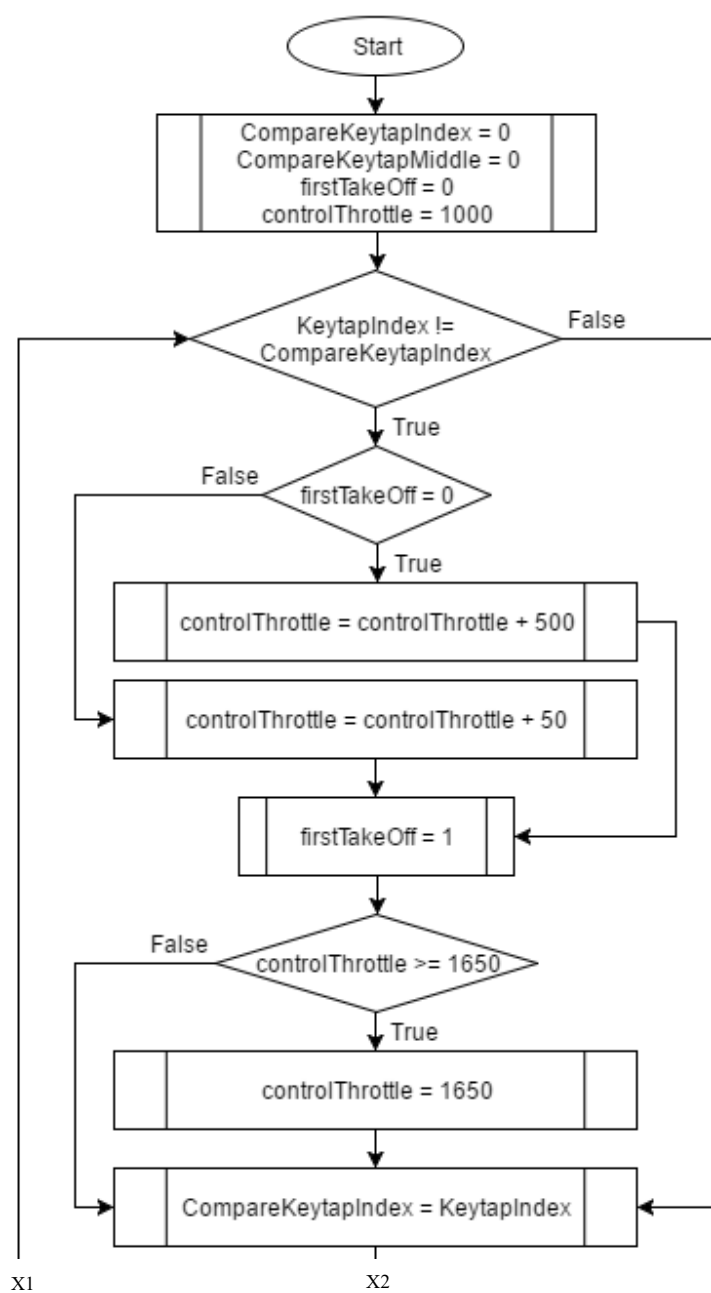
ส่วนควบคุมการกำหนดค่าสำหรับนำไปสร้างสัญญาณ PWM ในกรณีที่มีมือควบคุม หรือจากภาพที่ 3-4 คือส่วนของ Set PWM Signal ซึ่งจะแบ่งการทำงานหลักออกเป็น 5 การทำงานที่แตกต่างกัน ดังนี้

1. Stand by ในขั้นตอนนี้เป็นการวางมือบนอุปกรณ์ Leap Motion ในท่าทางการแบมือค้างไว้ ซึ่งจะเป็นการกำหนดค่าของสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel, Yaw Channel ให้มีการเปลี่ยนแปลงตามองศาของมือผู้ใช้งานที่เปลี่ยนแปลงไป โดยดูได้จากตารางที่ 3-8, 3-9, 3-10 ตามลำดับ และสำหรับค่าของ Throttle Channel กับ Flight Mode Channel จะถูกกำหนดไว้ที่ 1000 ซึ่งจะทำให้ Flight Mode สำหรับการบินอยู่ที่ Stabilize Mode แต่ในการบินจริงจะใช้ Flight Mode สำหรับการบินอยู่ที่ Loiter Mode แต่เมื่อมีการ Disarm Motor แล้วจำเป็นต้องเปลี่ยน Flight Mode การบินให้กลับมาอยู่ที่ Stabilize Mode ก่อน จนกระทั่งมีการ Arm Motor เสร็จเรียบร้อยแล้วจึงจะทำการเปลี่ยน Flight Mode สำหรับการบินไปที่ Loiter Mode เพื่อป้องกันความเร็วของมอเตอร์เพิ่มขึ้นเองขณะกำลังเพิ่ม Throttle Channel หลังจากที Arm Motor และเปลี่ยน Flight Mode สำหรับการบินไปที่ Loiter Mode เรียบร้อยแล้ว สำหรับขั้นตอน Stand by นี้ ถ้ามีการกำมือจะเป็นการสั่งให้ไปยังขั้นตอนการ Arm Motor หรือขั้นตอนที่ 2

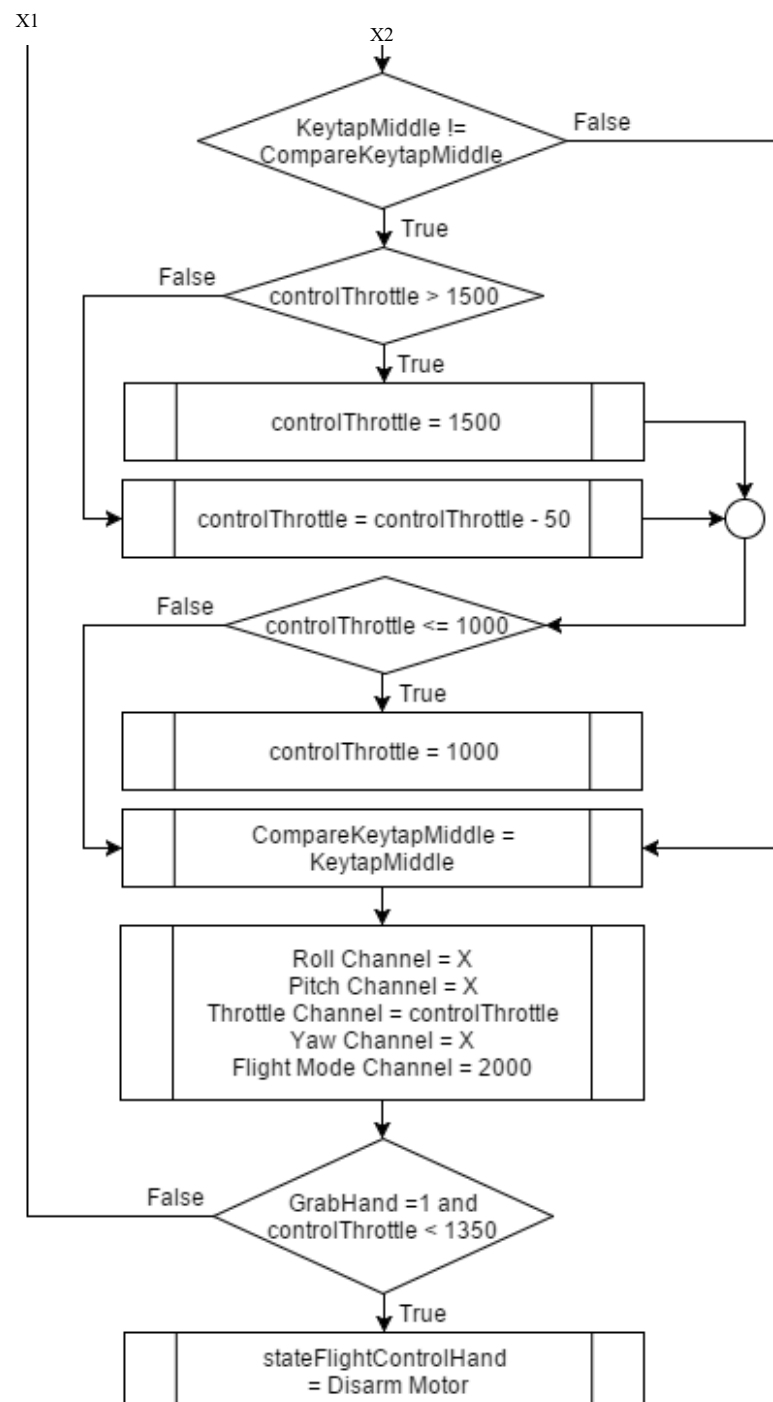
2. Arm Motor ในขั้นตอนนี้เป็นการกำมือค้างไว้บนอุปกรณ์ Leap Motion ซึ่งจะเป็นการกำหนดค่าของสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel ให้มีการเปลี่ยนแปลงตามองศาของมือผู้ใช้งานที่เปลี่ยนแปลงไป โดยดูได้จากตารางที่ 3-8, 3-9 ตามลำดับ และสำหรับค่าของ Throttle Channel, Flight Mode Channel จะถูกกำหนดไว้ที่ 1000 แต่สำหรับค่าของ Yaw Channel จะถูกกำหนดไว้ที่ 2000 ซึ่งการที่ Throttle Channel อยู่ที่ค่าต่ำสุดและ Yaw Channel อยู่ที่ค่าสูงสุดนั้นคือการสั่ง Arm Motor สำหรับขั้นตอน Arm Motor นี้ถ้ามีการแบมือจะเป็นการสั่งให้ไปยังขั้นตอนการ Control Flight หรือขั้นตอนที่ 3

3. Control Flight ในขั้นตอนนี้เป็นการแบมือค้างไว้บนอุปกรณ์ Leap Motion โดยสามารถเอียงมือซ้าย-ขวา เอียงมือไปด้านหน้า-หลัง และหมุนมือตามเข็มนาฬิกา-ทวนเข็มนาฬิกา เพื่อควบคุมการบินของมัลติโรเตอร์ โดยจะเป็นการกำหนดค่าของสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel, Yaw Channel ให้มีการเปลี่ยนแปลงตามองศาของมือผู้ใช้งานที่เปลี่ยนแปลงไป โดยดูได้จากตารางที่ 3-8, 3-9, 3-10 ตามลำดับ และสำหรับค่าของ Flight Mode Channel จะถูกกำหนดไว้ที่ 2000 เพื่อเปลี่ยน Flight Mode สำหรับการบินให้อยู่ใน Loiter Mode หรืออาจเรียกว่า GPS Mode ซึ่งจะเป็นโหมดสำหรับควบคุมการบินที่ง่ายที่สุด แต่สำหรับค่าของ Throttle Channel จะถูกกำหนดเริ่มต้นไว้ที่ 1000 และสามารถเพิ่มขึ้นได้ถ้ามีการกระดิกนิ้วชี้ โดยหลังจากผ่านขั้นตอนการ Arm

Motor และเข้ามายังขั้นตอน Control Flight ครั้งแรกถ้ากระดิกนิ้วชี้ จะทำการเพิ่ม Throttle Channel ให้เท่ากับ 1500 หลังจากนั้นถ้ามีการกระดิกนิ้วชี้ก็จะเพิ่มขึ้นครั้งละ 50 แต่จะเพิ่มมากที่สุดไม่เกิน 1650 ซึ่งจะได้ระดับความสูงโดยประมาณ 15-25 เมตร และถ้า Throttle Channel มีค่ามากกว่า 1500 เมื่อมีการกระดิกนิ้วกลางจะลดค่าของ Throttle Channel ให้เหลือเท่ากับ 1500 เพื่อรักษาระดับความสูง และหลังจากนั้นถ้ามีการกระดิกนิ้วกลางอีกจะลดลงทีละ 50 แต่จะลดลงไม่เกิน 1000 และสำหรับขั้นตอน Control Flight นี้ ถ้ามีการกำมือและค่าของ Throttle Channel เหลือน้อยกว่า 1350 จะเป็นการสั่งให้ไปยังขั้นตอนการ Disarm Motor หรือขั้นตอนที่ 4 โดยสำหรับในขั้นตอนการ Control Flight เนื่องจากเป็นขั้นตอนที่มีการทำงานค่อนข้างซับซ้อนจึงสามารถเขียนอธิบายในรูปของ Flow-Chart ได้ตามภาพที่ 3-6 และภาพที่ 3-7 เพื่อให้ผู้ใช้สามารถเข้าใจระบบการทำงานในส่วน of ขั้นตอนนี้ได้ง่ายขึ้น



ภาพที่ 3-6 ขั้นตอนการทำงานสำหรับควบคุมการบินของมัลติโรเตอร์ ในกรณีที่มีมือควบคุม
ส่วนที่ 1

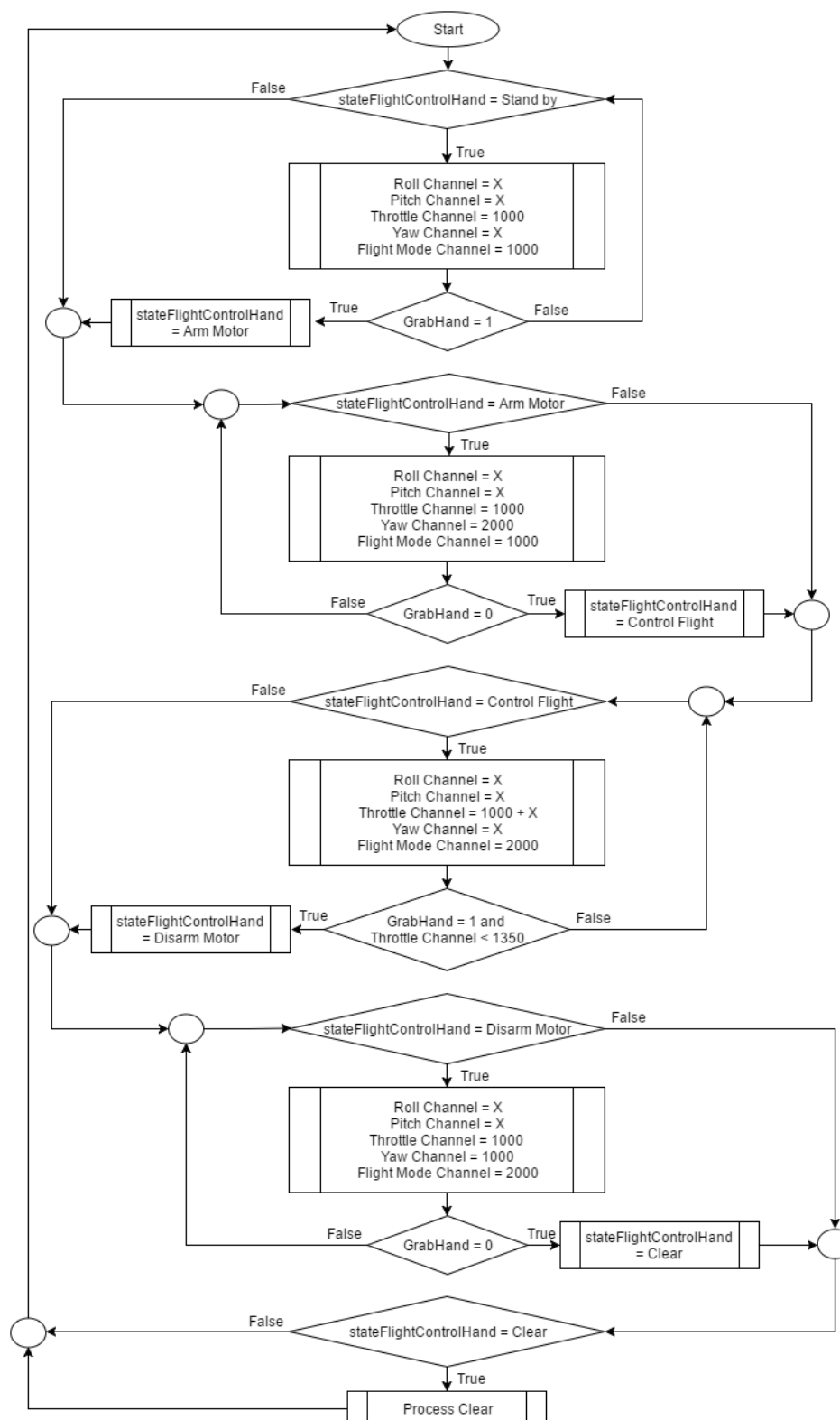


ภาพที่ 3-7 ขั้นตอนการทำงานสำหรับควบคุมการบินของมัลติโรเตอร์ ในกรณีที่มีมือควบคุม
ส่วนที่ 2

4. Disarm Motor ในขั้นตอนนี้เป็นการกำมือค้ำไว้บนอุปกรณ์ Leap Motion ซึ่งจะเป็นการกำหนดค่าของสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel ให้มีการเปลี่ยนแปลงตามองศาของมือผู้ใช้งานที่เปลี่ยนแปลงไป โดยดูได้จากตารางที่ 3-8, 3-9 ตามลำดับ และสำหรับค่าของ Throttle Channel, Yaw Channel จะถูกกำหนดไว้ที่ 1000 แต่สำหรับค่าของ Flight Mode Channel จะถูกกำหนดไว้ที่ 2000 ซึ่งการที่ Throttle Channel และ Yaw Channel อยู่ที่ค่าต่ำสุด นั้นคือการสั่ง Disarm Motor สำหรับขั้นตอน Disarm Motor นี้ถ้ามีการแบมือจะเป็นการสั่งให้ไปยังขั้นตอนที่ 5

5. ในขั้นตอนนี้จะเป็นการเคลียค่าของตัวแปรต่าง ๆ ที่จำเป็นต้องใช้ในแต่ละรอบของการควบคุมการบินของมัลติโรเตอร์ ตัวอย่างเช่น ค่าสำหรับตรวจสอบการกระดิกนิ้วชี้ นิ้วกลาง เป็นต้น และเมื่อเคลียเสร็จสิ้นจะกำหนดให้ไปเริ่มที่ขั้นตอนที่ 1 หรือขั้นตอนการ Stand by นั่นเอง

จาก 5 ขั้นตอนที่ได้อธิบายในข้างต้นสามารถเขียนออกให้อยู่ในรูปของ Flow-Chart ได้ตามภาพที่ 3-8



ภาพที่ 3-8 ขั้นตอนการทำงานของระบบสำหรับควบคุมการสร้างสัญญาณ PWM ในกรณีที่
มีมือควบคุม

3.2.9 พัฒนาโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่ไม่มีมือควบคุมและสัญญาณการเชื่อมต่อขาดหาย

ส่วนควบคุมการกำหนดค่าสำหรับนำไปสร้างสัญญาณ PWM ในกรณีที่ไม่มีมือควบคุมและสัญญาณการเชื่อมต่อขาดหาย หรือจากภาพที่ 3-4 คือส่วนของ FailSafe() ในส่วนนี้จะเป็นการตรวจสอบว่าในขั้นตอนของ 3.2.3.6 นั้นดำเนินการอยู่ในขั้นตอนใดแล้ว ถ้าดำเนินการอยู่ในขั้นตอนการ Stand by จะทำการเรียกใช้งาน FailSafe ที่ชื่อฟังก์ชัน failSafeStateFlightControl_0() แต่ถ้าดำเนินการอยู่ในขั้นตอนการ Arm Motor จะทำการเรียกใช้งาน FailSafe ที่ชื่อฟังก์ชัน failSafeStateFlightControl_1() ถ้าดำเนินการอยู่ในขั้นตอนการ Control Flight จะทำการเรียกใช้งาน FailSafe ที่ชื่อฟังก์ชัน failSafeStateFlightControl_2() โดยสำหรับฟังก์ชันนี้จะต้องมีการตรวจสอบค่าของตัวแปร controlThrottle ซึ่งเป็นค่าของ Throttle Channel ในขั้นตอนที่ 3.2.3.6 โดยแบ่งเป็น 2 กรณี นั่นคือ ถ้าตรวจสอบได้ว่า controlThrottle มีค่ามากกว่าเท่ากับ 1350 แสดงว่ามอเตอร์กำลังบินอยู่บนอากาศเพราะฉะนั้นถ้าสัญญาณขาดหายหรือไร้การควบคุมจะต้องกำหนดค่าของ Channel ต่าง ๆ ให้มอเตอร์สามารถกลับขึ้นไปบินหรือบินอยู่หนึ่ง ๆ ได้ แต่ถ้าตรวจสอบได้ว่าค่าของ controlThrottle น้อยกว่า 1350 แสดงว่ามอเตอร์ลงจอดเรียบร้อยแล้ว จะทำการกำหนดค่าของ Yaw Channel และ Throttle Channel ให้มีค่าต่ำสุด เพื่อทำการ Disarm Motor เพื่อป้องกันอันตรายที่อาจเกิดขึ้นได้ ถ้าดำเนินการอยู่ในขั้นตอนการ Disarm Motor จะทำการเรียกใช้งาน FailSafe ที่ชื่อฟังก์ชัน failSafeStateFlightControl_3() โดยการกำหนดค่าของสัญญาณ PWM ให้กับ Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel ในแต่ละฟังก์ชันนั้นสามารถดูได้จากตารางที่ 3-12 ถึง 3-15

ตารางที่ 3-12 ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้ สำหรับฟังก์ชัน

failSafeStateFlightControl_0()

Channel	PWM Value
Roll Channel	1500
Pitch Channel	1500
Throttle Channel	1000
Yaw Channel	1500
Flight Mode Channel	1000

ตารางที่ 3-13 ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้ สำหรับฟังก์ชัน

failSafeStateFlightControl_1()

Channel	PWM Value
Roll Channel	1500
Pitch Channel	1500
Throttle Channel	1000
Yaw Channel	1000
Flight Mode Channel	1000

ตารางที่ 3-14 ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้ สำหรับฟังก์ชัน

failSafeStateFlightControl_2()

Channel	PWM Value
Roll Channel	1500
Pitch Channel	1500
Throttle Channel	If ControlThrottle \geq 1350, Throttle Channel = 1500 If ControlThrottle $<$ 1350, Throttle Channel = 1000
Yaw Channel	If ControlThrottle \geq 1350, Yaw Channel = 1500 If ControlThrottle $<$ 1350, Yaw Channel = 1000
Flight Mode Channel	2000

ตารางที่ 3-15 ค่าของสัญญาณ PWM ในแต่ละ Channel ที่ถูกกำหนดไว้ สำหรับฟังก์ชัน

failSafeStateFlightControl_3()

Channel	PWM Value
Roll Channel	1500
Pitch Channel	1500
Throttle Channel	1000
Yaw Channel	1000
FlightMode Channel	2000

3.2.10 พัฒนาโปรแกรมสำหรับสร้างสัญญาณ PWM ตามค่าที่ถูกกำหนด

ส่วนสำหรับสร้างสัญญาณ PWM หรือจากภาพที่ 3-4 คือส่วนของ Generate PWM Signal ในส่วนนี้จะทำหน้าที่ในการนำค่าของ Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel ที่ถูกกำหนดในขั้นตอนที่ 3.2.8 หรือ 3.2.9 มาใส่เป็นพารามิเตอร์ให้กับคำสั่ง `delayMicrosecond()` หลังจากที่ใช้คำสั่ง `digitalWrite(1)` เรียบร้อยแล้ว เพื่อเป็นการกำหนดให้ Digital Pin 2, 3, 4, 5, 6 ของ Arduino UNO สร้างสัญญาณ PWM ที่มีสัญญาณเป็นสถานะลอจิก High ตามเวลาที่กำหนดไว้ และหลังจากนั้นตามด้วยการใช้คำสั่ง `digitalWrite(0)` ร่วมกับคำสั่ง `delayMicrosecond()` อีกครั้ง แต่ใส่พารามิเตอร์เป็น 20,000 ลบด้วยค่าที่ได้จากขั้นตอนที่ 3.2.8 หรือ 3.2.9 เพื่อเป็นการกำหนดให้ Digital Pin 2, 3, 4, 5, 6 ของ Arduino UNO สร้างสัญญาณ PWM ที่มีสัญญาณเป็นสถานะลอจิก Low ต่อจากตอนแรกที่มีสถานะเป็นลอจิก High การทำเช่นนี้จะเป็นการสร้างสัญญาณ PWM ให้มีความถี่ 50 Hz ซึ่งเป็นความถี่ที่ใช้สำหรับติดต่อกับ Flight Controller ทางด้าน Input Channel

บทที่ 4

ผลการดำเนินงาน

ผลการดำเนินงานแบ่งออกเป็น 2 ส่วนหลักคือ 1. ผลการดำเนินงานของส่วนควบคุมภาคพื้นดิน 2. ผลการดำเนินงานของส่วนควบคุมภาคอากาศ

4.1 ผลการดำเนินงานของส่วนควบคุมภาคพื้นดิน

สำหรับผลการดำเนินงานของส่วนควบคุมภาคพื้นดินนั้น สามารถแบ่งผลการดำเนินงานตามลำดับการพัฒนาโปรแกรมได้เป็น

1. การทดสอบระบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคพื้นดิน
 2. การทดสอบโปรแกรมตรวจจับลักษณะท่าทางของมือ
 3. การทดสอบโปรแกรมประมวลผลจากท่าทางของมือที่ตรวจจับได้
 4. การทดสอบโปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application
 5. การทดสอบโปรแกรมสำหรับส่งข้อมูลไปยังส่วนควบคุมภาคอากาศ
- 4.1.1 การทดสอบระบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคพื้นดิน

การทดสอบระบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคพื้นดินประกอบไปด้วย การเชื่อมต่อของ Leap Motion และ Wireless Datalink เข้ากับ Notebook ผ่านทาง USB Portable โดยมีลักษณะการเชื่อมต่อของอุปกรณ์ต่าง ๆ นี้

1. Leap Motion ทำหน้าที่ตรวจจับการเคลื่อนไหวในท่าทางต่าง ๆ จากมือของผู้ใช้งาน โดยจะมีการเชื่อมต่ออยู่กับคอมพิวเตอร์ผ่านทาง USB Portable ตามภาพที่ 4-1



ภาพที่ 4-1 การเชื่อมต่อระหว่างอุปกรณ์ Leap Motion กับคอมพิวเตอร์

2. Wireless Datalink (915 MHz Telemetry) ทำหน้าที่ส่งข้อมูลที่ผ่านการประมวลผลจากซอฟต์แวร์ของส่วนควบคุมภาคพื้นดินเรียบร้อยแล้ว ไปยังส่วนควบคุมภาคอากาศ โดยจะมีการเชื่อมต่ออยู่กับคอมพิวเตอร์ผ่านทาง USB Portable ตามภาพที่ 4-2

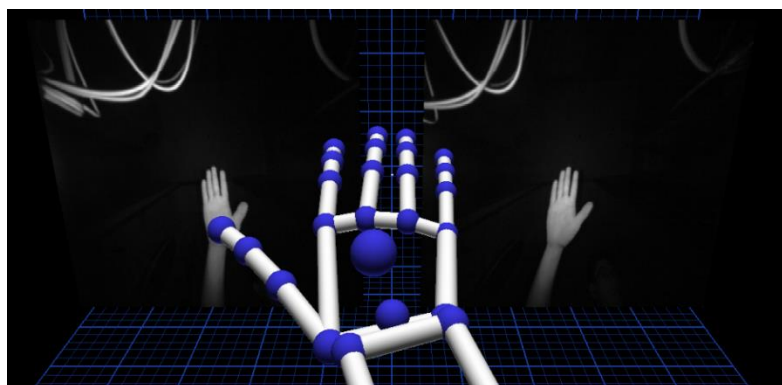


ภาพที่ 4-2 การเชื่อมต่อระหว่างอุปกรณ์ Wireless Datalink กับคอมพิวเตอร์

4.1.2 การทดสอบโปรแกรมตรวจจับลักษณะท่าทางของมือ

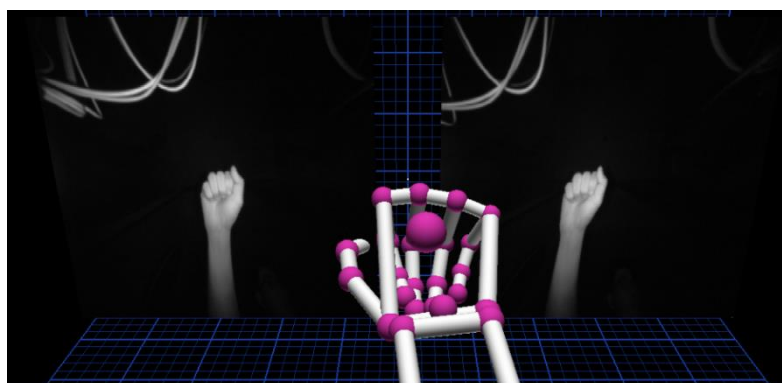
เป็นการนำมือของผู้ใช้งานหรือผู้ที่ต้องการควบคุมการบินของมัลติโรเตอร์มาอยู่บนอุปกรณ์ Leap Motion ซึ่งท่าทางของมือที่นำมาใช้ในการทดลองจะประกอบไปด้วยท่าทางต่าง ๆ 7 ท่าทาง ดังนี้

1. การแบมือ (Hand Amount) เป็นท่าทางการแบมือไว้เฉย ๆ เพื่อตรวจสอบว่า ณ ขณะนั้นมีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion หรือไม่ สามารถดูท่าทางของมือได้จากภาพที่ 4-3



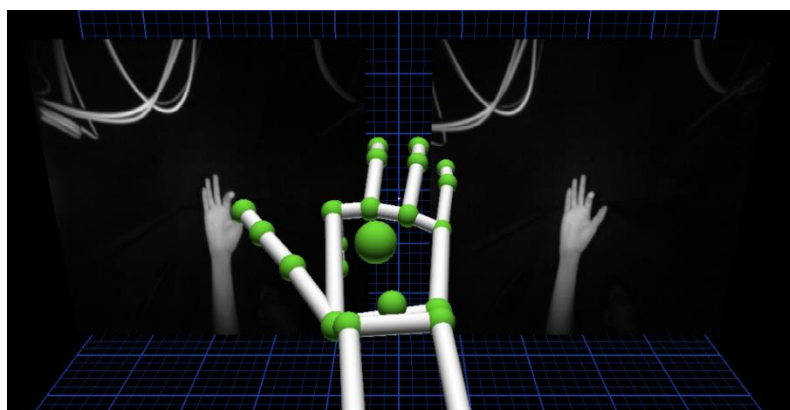
ภาพที่ 4-3 ท่าทางการแบมือ โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

2. การกำมือ (Grab Hand) เป็นการท่าทางการกำมือไว้ เพื่อตรวจสอบว่า ณ ขณะนั้นผู้ใช้งานได้มีการกำมือหรือแบมือบนอุปกรณ์ Leap Motion หรือไม่ สามารถดูท่าทางของมือได้จากภาพที่ 4-4



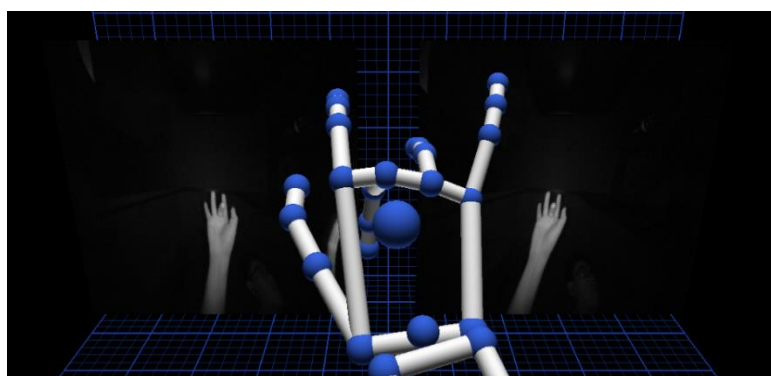
ภาพที่ 4-4 ท่าทางการกำมือ โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

3. การกระดิกนิ้วชี้ (Keytap Index) เป็นการทําท่าทางการกระดิกนิ้วชี้ สามารถดูท่าทางของมือได้จากภาพที่ 4-5



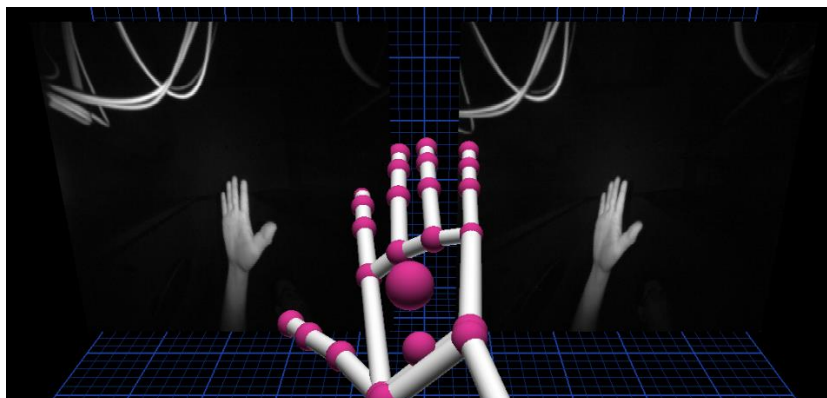
ภาพที่ 4-5 ท่าทางการกระดิกนิ้วชี้โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

4. การกระดิกนิ้วกลาง (Keytap Middle) เป็นการทําท่าทางการกระดิกนิ้วกลาง สามารถดูท่าทางของมือได้จากภาพที่ 4-6

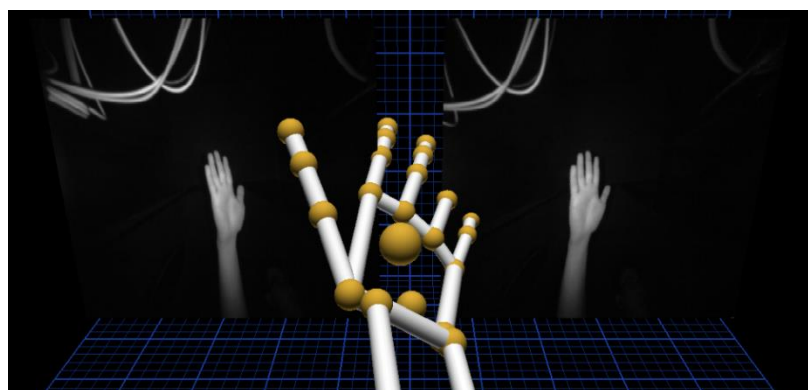


ภาพที่ 4-6 ท่าทางการกระดิกนิ้วกลางโดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

5. การแบมือและเอียงมือทางซ้ายหรือขวา (Roll) เป็นการทําท่าทางการแบมือบนอุปกรณ์ Leap Motion แต่มีการเอียงลงทางด้านซ้ายหรือขวา ซึ่งจะเป็นการวัดค่าของมุม Roll สามารถดูท่าทางของมือได้จากภาพที่ 4-7 และ 4-8

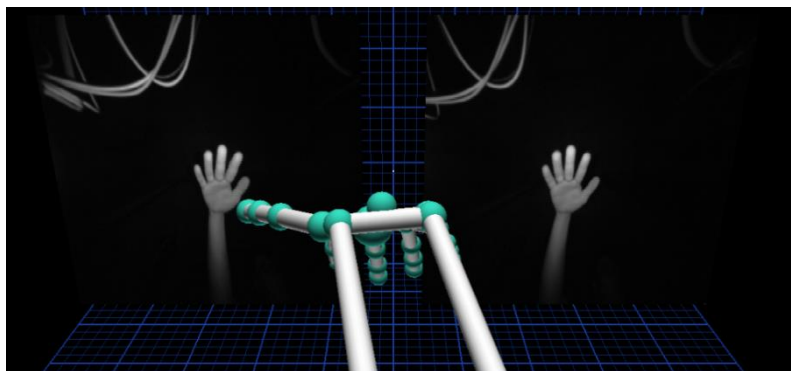


ภาพที่ 4-7 ท่าทางการแบมือและเอียงมือทางด้านซ้าย โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

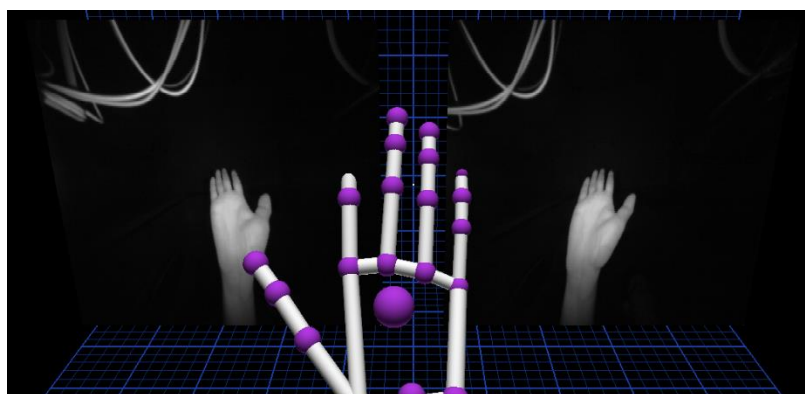


ภาพที่ 4-8 ท่าทางการแบมือและเอียงมือทางด้านขวา โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

6. การแบมือและเอียงมือไปด้านหน้าหรือหลัง (Pitch) เป็นการท่าทางการแบมือบนอุปกรณ์ Leap Motion แต่มีการเอียงลงทางด้านหน้าหรือด้านหลัง ซึ่งจะเป็นการวัดค่าของมุม Pitch สามารถดูท่าทางของมือได้จากภาพที่ 4-9 และ 4-10

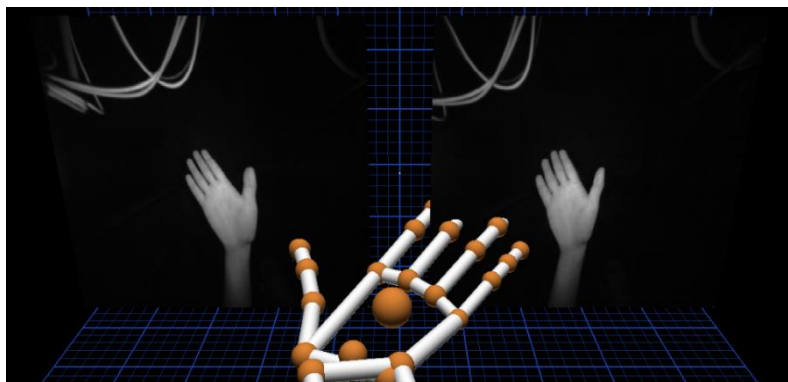


ภาพที่ 4-9 ทำท่าทางการแบมือและเอียงมือทางด้านหน้า โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

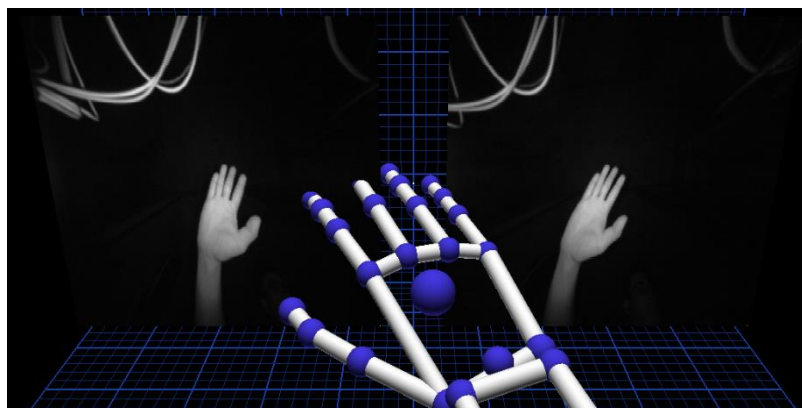


ภาพที่ 4-10 ภาพแสดงท่าทางการแบมือและเอียงทางด้านหลัง โดยแสดงผลด้วยโหมด Visualizer ของโปรแกรม Leap Motion

7. การแบมือและหมุนมือตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา (Yaw) เป็นการทำให้ท่าทางการแบมือบนอุปกรณ์ Leap Motion แต่มีการหมุนมือไปในทิศตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา ซึ่งจะเป็นการวัดค่าของมุม Yaw สามารถดูท่าทางของมือได้จากภาพที่ 4-11 และ 4-12



ภาพที่ 4-11 ทำทางการแบมือและหมุนมือตามเข็มนาฬิกา โดยแสดงผลด้วยโหนด Visualizer ของโปรแกรม Leap Motion



ภาพที่ 4-12 ทำทางการแบมือและหมุนมือทวนเข็มนาฬิกา โดยแสดงผลด้วยโหนด Visualizer ของโปรแกรม Leap Motion

4.1.3 การทดสอบโปรแกรมประมวลผลจากท่าทางของมือที่ตรวจจับได้

จากท่าทางทั้ง 7 รูปแบบในหัวข้อ 4.1.2 สามารถใช้ภาษา C# ในการพัฒนาและตรวจจับท่าทางเหล่านั้นได้ โดยเริ่มต้นจากการนำท่าทางต่าง ๆ มาแปลงเป็นค่าตามที่ต้องการ ดังที่เคยกล่าวไว้ในบทที่ 3 หลังจากนั้นทำการทดสอบแสดงค่าต่าง ๆ บน Form ในรูปแบบของตัวเลขตามที่กำหนดไว้ ซึ่งทางผู้จัดทำได้ใช้โปรแกรม Microsoft Visual Studio ในการช่วยสร้าง Form ขึ้นมา โดยค่าที่นำมาแสดงสามารถแบ่งออกเป็น

1. Hand Amount คือตรวจสอบว่ามีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion หรือไม่ ซึ่งถ้ามีจะให้ค่าเท่ากับ 1 และถ้าไม่มีจะให้ค่าเท่ากับ 0

2. Grab Hand คือตรวจสอบว่ามีการกำมือหรือไม่ ถ้ามีการกำมือจะให้ค่าเท่ากับ 1 และถ้าแบบมืออยู่จะให้ค่าเท่ากับ 0

3. Keytap Index คือการตรวจสอบว่ามีการกระดิกนิ้วชี้หรือไม่ ซึ่งเมื่อมีการกระดิกนิ้วชี้ 1 ครั้ง จะทำเปลี่ยนค่าจาก 0 ไป 1 หรือ เปลี่ยนจาก 1 ไป 0 ใช้หลักการแบบ Toggle ค่าในการตรวจสอบ

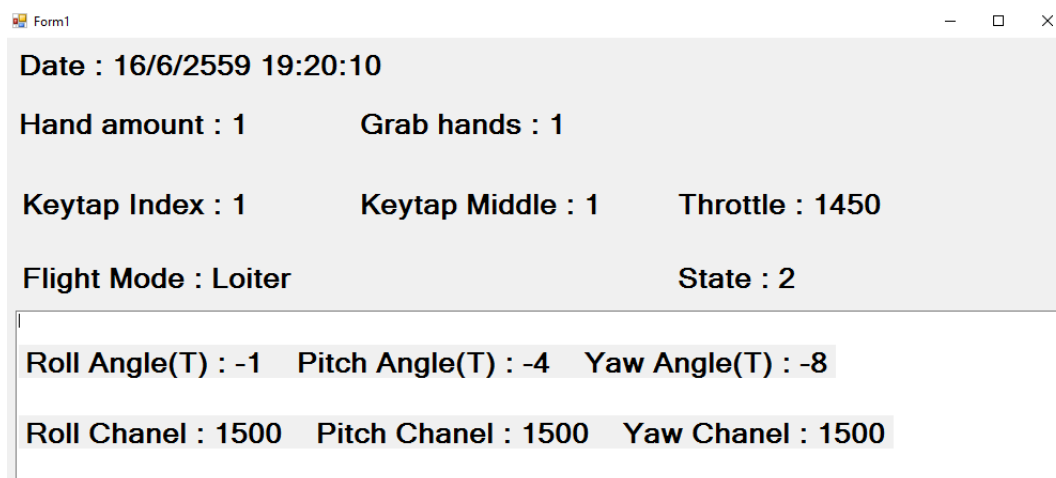
4. Keytap Middle คือการตรวจสอบว่ามีการกระดิกนิ้วกลางหรือไม่ ซึ่งเมื่อมีการกระดิกนิ้วกลาง 1 ครั้ง จะทำเปลี่ยนค่าจาก 0 ไป 1 หรือ เปลี่ยนจาก 1 ไป 0 ใช้หลักการแบบ Toggle ค่าในการตรวจสอบ

5. Roll Angle เป็นการตรวจสอบว่าการแบมือบนอุปกรณ์ Leap Motion ของผู้ใช้งานนั้น ตั้งตรงหรือมีการเอียงไปทางด้านซ้าย ขวาหรือไม่ เพื่อนำไปใช้ในการกำหนดค่าให้กับ Roll Channel ในฝั่งของส่วนควบคุมภาคอากาศ

6. Pitch Angle เป็นการตรวจสอบว่าการแบมือบนอุปกรณ์ Leap Motion ของผู้ใช้งานนั้น ตั้งตรงหรือมีการเอียงไปทางด้านหน้า หลังหรือไม่ เพื่อนำไปใช้ในการกำหนดค่าให้กับ Pitch Channel ในฝั่งของส่วนควบคุมภาคอากาศ

7. Yaw Angle เป็นการตรวจสอบว่าการแบมือบนอุปกรณ์ Leap Motion ของผู้ใช้งานนั้น ตั้งตรงหรือมีการหมุนไปในทิศทางตามเข็มนาฬิกาหรือไม่ เพื่อนำไปใช้ในการกำหนดค่าให้กับ Yaw Channel ในฝั่งของส่วนควบคุมภาคอากาศ

สามารถดูการแสดงผลลัพธ์บน Form จากท่าทางทั้ง 7 รูปแบบในภาพที่ 4-13 สำหรับกรณีที่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion และในภาพที่ 4-14 สำหรับกรณีที่ไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion



Date : 16/6/2559 19:20:10

Hand amount : 1 Grab hands : 1

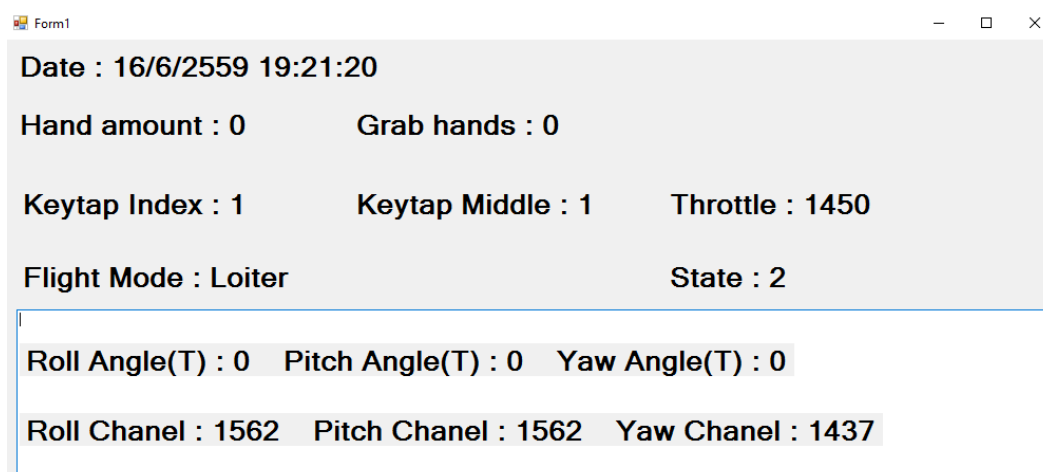
Keytap Index : 1 Keytap Middle : 1 Throttle : 1450

Flight Mode : Loiter State : 2

Roll Angle(T) : -1 Pitch Angle(T) : -4 Yaw Angle(T) : -8

Roll Chanel : 1500 Pitch Chanel : 1500 Yaw Chanel : 1500

ภาพที่ 4-13 จอแสดงผลของ Desktop Application โดยแสดงข้อมูลที่ใช้งานจากท่าทางต่าง ๆ ของมือ ในกรณีที่มีมือผู้ใช้งานควบคุม



Date : 16/6/2559 19:21:20

Hand amount : 0 Grab hands : 0

Keytap Index : 1 Keytap Middle : 1 Throttle : 1450

Flight Mode : Loiter State : 2

Roll Angle(T) : 0 Pitch Angle(T) : 0 Yaw Angle(T) : 0

Roll Chanel : 1562 Pitch Chanel : 1562 Yaw Chanel : 1437

ภาพที่ 4-14 จอแสดงผลของ Desktop Application โดยแสดงข้อมูลที่ใช้งานจากท่าทางต่าง ๆ ของมือ ในกรณีที่ไม่มีมือผู้ใช้งานควบคุม

4.1.4 การทดสอบโปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application

เป็นส่วนสำหรับติดต่อผู้ใช้งาน ซึ่งมีหน้าที่หลัก 4 อย่างด้วยกัน

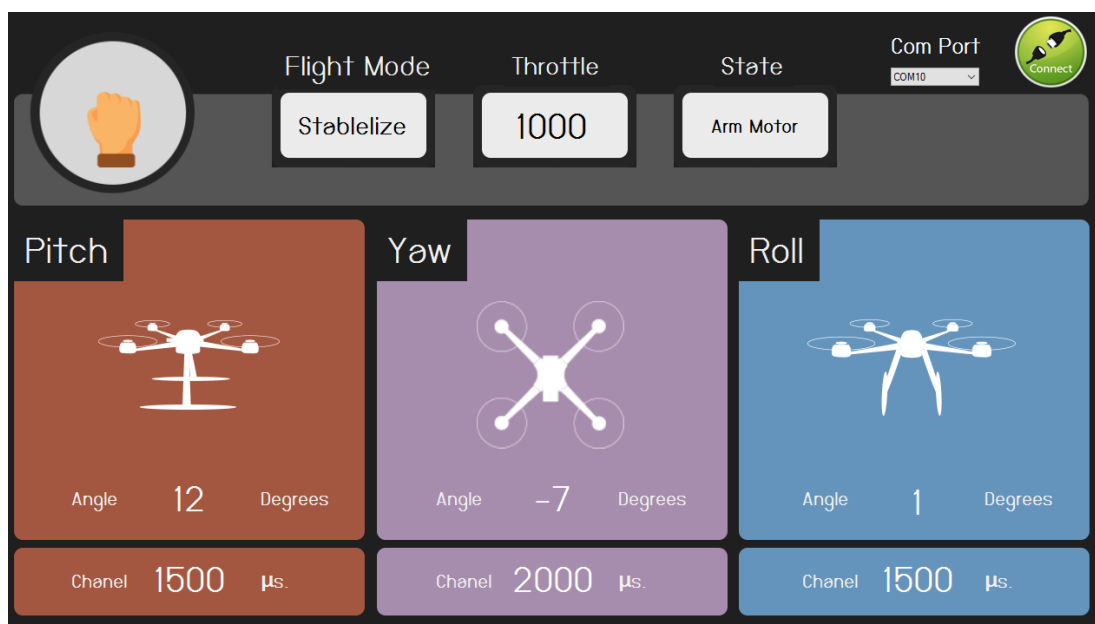
1. กำหนดการเริ่มต้นส่งข้อมูลไปยังส่วนควบคุมภาคอากาศ โดยสามารถกดที่ปุ่มในตำแหน่งด้านมุมบนขวาจากภาพที่ 4-15 และ 4-16 ซึ่งถ้าเปิดโปรแกรมมาครั้งแรก ปุ่มนี้จะมีสีแดง และเขียนว่า Disconnect แต่ถ้ามีการกดเพื่อเริ่มต้นส่งข้อมูลจะเปลี่ยนเป็นสีเขียว และเขียนว่า Connect

2. ส่วนแสดงสถานะของมือผู้ใช้งาน โดยถ้าไม่มีมืออยู่บนอุปกรณ์จะแสดงรูปท่าทางการแบ่มือแต่จะเป็นสีเทา และถ้ามีมืออยู่บนอุปกรณ์ Leap Motion จะเป็นสีเนื้อคล้าย ๆ สีของมือจริง โดยถ้ามีการกำมือจะเปลี่ยนรูปสำหรับแสดง จากท่าทางการแบ่มือเป็นท่าทางการกำมือ ซึ่งตำแหน่งในการแสดงจะอยู่ในมุมบนด้านซ้าย นอกจากนั้นจะมีการแสดงองศาของมือผู้ใช้งานในส่วนของมุม Roll, Pitch และ Yaw ซึ่งจะแสดงภายใต้หัวข้อ Angle และมีหน่วยเป็น Degrees หรือองศา

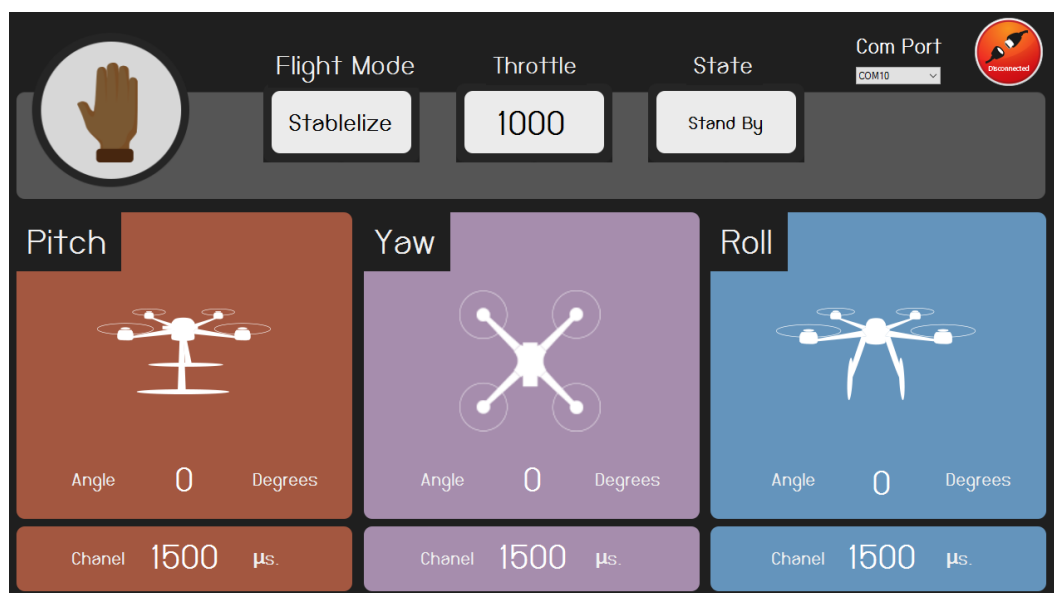
3. ส่วนแสดงค่าของ Roll Channel, Pitch Channel, Yaw Channel ภายใต้หัวข้อ Channel และมีหน่วยเป็นไมโครวินาที และส่วนสำหรับแสดง State การทำงานเพื่อให้ผู้ใช้งานรับรู้ได้ว่ากำลังสั่งให้มอเตอร์ทำงานอะไรอยู่ ซึ่งจะอยู่ภายใต้หัวข้อ State และยังมีส่วนสำหรับแสดง Throttle Channel ภายใต้หัวข้อ Throttle ซึ่งจะมีหน่วยเป็นไมโครวินาที และสุดท้ายเป็นส่วนสำหรับแสดง Mode สำหรับการบินของมอเตอร์ภายใต้หัวข้อ Flight Mode ซึ่งสำหรับหัวข้อนี้ทุก ๆ ค่าที่นำมาแสดงจะเป็นการรับค่ากลับมาจากฝั่งของส่วนควบคุมภาคอากาศ

4. ส่วนแสดงรูปการเคลื่อนที่ของมอเตอร์ในทิศทางที่แตกต่างกันทั้ง 3 ค่า โดยพิจารณาจากค่าของ Roll Channel, Pitch Channel, Yaw Channel ที่รับกลับมาจากฝั่งของส่วนควบคุมภาคอากาศ

สำหรับหัวข้อ 4.1.4 สามารถแสดงภาพที่แตกต่างกัน 2 รูปแบบด้วยกันคือ ภาพที่ 4-15 สำหรับกรณีที่มีการเริ่มต้นการส่งข้อมูลและมีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion และสำหรับภาพที่ 4-16 สำหรับกรณีที่ยังไม่เริ่มต้นส่งข้อมูลและไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion



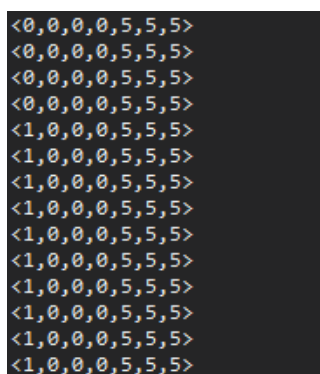
ภาพที่ 4-15 โปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application โดยแสดงข้อมูลในกรณีที่ยัง
เริ่มต้นส่งข้อมูลและมีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion



ภาพที่ 4-16 โปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application โดยแสดงข้อมูลในกรณีที่ยัง
ไม่เริ่มต้นส่งข้อมูลและไม่มีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion

4.1.5 การทดสอบโปรแกรมสำหรับส่งข้อมูลไปยังส่วนควบคุมภาคอากาศ

สำหรับส่วนนี้จะเป็นการใช้คำสั่ง `write()` ซึ่งสำหรับ `SerialPort Class` ในภาษา C# คำสั่งนี้จะเป็นการส่งข้อมูลออกไปผ่าน `SerialPort` ที่ได้ตั้งค่าเอาไว้เรียบร้อยแล้ว โดยในส่วนของการส่งข้อมูลนี้ผู้จัดทำได้ใช้รูปแบบการส่งข้อมูลแบบ “<X1, X2, X3, X4, X5, X6, X7>” โดยมีเครื่องหมาย “<” เป็น Start Message และ “>” เป็น End Message สำหรับข้อมูลแต่ละชุด สามารถดูตัวอย่างข้อมูลที่ส่งออกไปยังส่วนควบคุมภาคอากาศได้จากภาพที่ 4-17



```
<0,0,0,0,5,5,5>
<0,0,0,0,5,5,5>
<0,0,0,0,5,5,5>
<0,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
<1,0,0,0,5,5,5>
```

ภาพที่ 4-17 ตัวอย่างของชุดข้อมูลที่ส่งออกไปให้กับส่วนควบคุมภาคอากาศ โดยแสดงผลผ่านส่วน Output ของโปรแกรม Microsoft Visual Studio

4.2 ผลการดำเนินงานของส่วนควบคุมภาคอากาศ

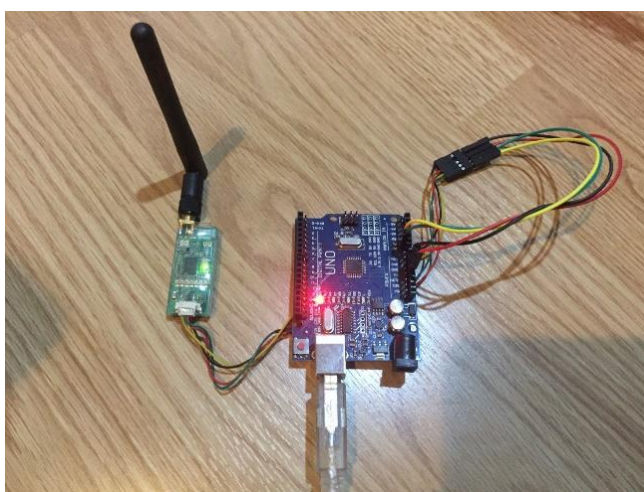
สำหรับผลการดำเนินงานของส่วนควบคุมภาคอากาศนั้น สามารถแบ่งผลการดำเนินงานตามลำดับการพัฒนาโปรแกรมได้เป็น

1. การทดสอบระบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคอากาศ
2. การทดสอบโปรแกรมรับข้อมูลจากส่วนควบคุมภาคพื้นดิน และตรวจสอบตำแหน่งข้อมูล
3. การทดสอบโปรแกรมตรวจสอบความยาวข้อมูลและหาค่าของข้อมูลที่เหมือนกันมากที่สุด
ในบัฟเฟอร์
4. การทดสอบโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่มีมือ
5. การทดสอบโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่ไม่มีมือหรือ
สัญญาณขาดหาย
6. การทดสอบโปรแกรมสำหรับสร้างสัญญาณ PWM
7. การทดสอบโปรแกรมสำหรับส่งข้อมูลไปยังส่วนควบคุมภาคพื้นดิน

4.2.1 การทดสอบระบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคอากาศ

การทดสอบระบบฮาร์ดแวร์สำหรับส่วนควบคุมภาคอากาศประกอบด้วย 2 ส่วน ได้แก่

1. ส่วนสำหรับรับส่งข้อมูลกับส่วนควบคุมภาคพื้นดิน ซึ่งจะประกอบด้วยอุปกรณ์ Wireless Datalink และ Arduino UNO โดยในส่วนนี้จะทำหน้าที่เสมือน Radio Receiver ในการรับข้อมูลจากส่วนควบคุมภาคพื้นดินเข้ามายังส่วนควบคุมภาคอากาศ และมีหน้าที่ส่งข้อมูลบางส่วนกลับไปที่เพื่อแสดงผลบนโปรแกรม Desktop Application โดยจะมีตัวอย่างการเชื่อมต่อตามภาพที่ 4-18



ภาพที่ 4-18 การเชื่อมต่อระหว่าง Arduino UNO กับ Wireless Datalink (915 MHz)

2. ส่วนของมอเตอร์ ซึ่งก็คือส่วนสำหรับรับสัญญาณ PWM ที่บอร์ด Arduino UNO สร้างขึ้นมาและนำสัญญาณนั้นไปประมวลผลเพื่อควบคุมการทำงานของมอเตอร์ทั้ง 4 ตัว ซึ่งส่งผลโดยตรงต่อลักษณะการบินของมอเตอร์ โดยในส่วนนี้จะประกอบด้วยอุปกรณ์ดังต่อไปนี้

- APM2.6 (ArduPilotMega 2.6)
- GPS 7N for APM
- ESC (Electronic Speed Controller) Dragon Speed 30 A Opto
- Brushless Motor 4010/850 KV
- Power Module (RCtimer for APM Max 30 V, 90 A)
- Battery 3Cell (4300 mAh, 2200 mAh)

และสามารถดูการเชื่อมต่อจากฮาร์ดแวร์ของจริงได้จากภาพที่ 4-19



ภาพที่ 4-19 การเชื่อมต่ออุปกรณ์ที่ใช้งานบนส่วนของมัลติโรเตอร์

4.2.2 การทดสอบโปรแกรมรับข้อมูลจากส่วนควบคุมภาคพื้นดินและตรวจสอบตำแหน่งข้อมูล

สำหรับการส่งข้อมูลแบบ Serial จะเป็นการส่งข้อมูลออกมาเรื่อย ๆ ตามที่ได้กำหนดไว้คือ 2 Hz และจากการทดสอบพบว่าข้อมูลที่เข้ามาบาง Byte นั้นเป็นข้อมูลขยะ ยกตัวอย่างเช่น เป็นอักขระที่ไม่ได้นำมาใช้งาน เป็นอักขรภาษาไทยหรือภาษาอังกฤษ ซึ่งล้วนแล้วเป็นข้อมูลที่ไม่สามารถนำมาใช้งานจริงได้ ทางผู้จัดทำได้สร้างวิธีแก้ไขขึ้นมาโดยให้มีการตรวจสอบตำแหน่งข้อมูล ซึ่งเป็นการตรวจสอบในขั้นตอนที่ 1 ก่อนจะเริ่มเก็บข้อมูลลง Buffer ซึ่งปัญหานี้เกิดเฉพาะฝั่งของ Arduino UNO ที่รับข้อมูลเข้ามาเท่านั้น จึงทำการแก้ไขเฉพาะที่ส่วนของ Arduino UNO โดยทำการรับข้อมูลเข้ามาทุก Byte ตามปกติ แต่จะไม่นำข้อมูลทุก Byte เข้ามาเก็บลง Buffer ที่ใช้สำหรับเข้ากระบวนการตรวจสอบส่วนถัดไป โดยสามารถดูได้จากภาพที่ 4-20 จะเห็นได้ว่าข้อมูลจะถูกเก็บลง Buffer ได้นั้นจะต้องเจอข้อมูล หรือ INPUT FROM GCS เป็นเครื่องหมาย "<" ก่อนเท่านั้น เมื่อพบแล้วจึงตรวจสอบในแต่ละตำแหน่งถัดไปของข้อมูลที่เข้าว่าตรงตามข้อกำหนดหรือไม่ ถ้าตรงจึงนำข้อมูล Byte นั้นไปส่งไปยัง Buffer จากภาพที่ 4-20 จะเห็นได้ว่าบรรทัดสุดท้ายสามารถเก็บข้อมูลได้ครบทั้ง 13 ตัวตามที่ต้องการ และข้อมูลแต่ละตำแหน่งสามารถนำไปใช้งานต่อ

ได้ตามที่วางแผนเอาไว้ สำหรับการแก้ปัญหาเช่นนี้จะมีความถูกต้องเป็นหลัก แต่จะส่งผลเสียในเรื่องของความเร็วในการนำค่าที่ได้จากขั้นตอนนี้ไปกำหนดค่าให้กับสัญญาณ PWM ได้ เพราะถ้าข้อมูลชุดนี้ที่เริ่มเก็บลง Buffer แล้ว และเจอข้อมูลเสียกลางทางจะทำให้ต้องเคลีย Buffer และรอไปอีก 0.5 วินาทีกว่าที่ข้อมูลชุดใหม่จะถูกส่งมา

```

INPUT FROM GCS : <
Result INPUT :
INPUT FROM GCS : 0
Result INPUT : 0
INPUT FROM GCS : ,
Result INPUT : 0,
INPUT FROM GCS : 0
Result INPUT : 0,0
INPUT FROM GCS : ,
Result INPUT : 0,0,
INPUT FROM GCS : 0
Result INPUT : 0,0,0
INPUT FROM GCS : ,
Result INPUT : 0,0,0,
INPUT FROM GCS : 0
Result INPUT : 0,0,0,0
INPUT FROM GCS : ,
Result INPUT : 0,0,0,0,
INPUT FROM GCS : 5
Result INPUT : 0,0,0,0,5
INPUT FROM GCS : ,
Result INPUT : 0,0,0,0,5,
INPUT FROM GCS : 4
Result INPUT : 0,0,0,0,5,4
INPUT FROM GCS : ,
Result INPUT : 0,0,0,0,5,4,
INPUT FROM GCS : 5
Result INPUT : 0,0,0,0,5,4,5

```

ภาพที่ 4-20 การรับข้อมูลจากส่วนควบคุมภาคพื้นดินและตรวจสอบความถูกต้องในขั้นตอนที่ 1
(ตรวจสอบตำแหน่งข้อมูล) บน Serial Monitor ของโปรแกรม Arduino Sketch

4.2.3 การทดสอบโปรแกรมตรวจสอบความยาวข้อมูลและหาค่าของข้อมูลที่เหมือนกันมากที่สุดในปีเฟอร์

หลังจากที่ได้ข้อมูลจากขั้นตอนการตรวจสอบขั้นที่ 1 เรียบร้อยแล้ว ข้อมูลจะถูกเก็บอยู่ในรูปแบบ X1, X2, X3, X4, X5, X6, X7 ซึ่งถ้ามีความยาวเท่ากับ 13 ตัวอักษร จึงจะสามารถผ่านขั้นตอนการตรวจสอบความยาวข้อมูลได้ ซึ่งเป็นการตรวจสอบในขั้นตอนที่ 2 แล้วจึงทำการแยกค่าของข้อมูลทั้ง 7 ชนิดออกจากกัน โดยไม่นับรวมเครื่องหมาย “,” และเมื่อนำข้อมูลลง Buffer เรียบร้อยแล้ว จะเข้าสู่การตรวจสอบในขั้นตอนที่ 3 นั่นก็คือการตรวจสอบหาค่าที่เหมือนกันมากที่สุดในการเก็บข้อมูล

ยกตัวอย่างการอธิบาย สำหรับการตรวจสอบว่ามีมือของผู้ใช้งานอยู่บนอุปกรณ์ Leap Motion หรือไม่ ถ้าดูจากภาพที่ 4-21 ในหัวข้อ INPUT FOR Hand สำหรับค่านี้การที่แยกข้อมูลเช่นนี้ได้แสดงว่าต้องผ่านการตรวจสอบในขั้นตอนที่ 2 เรียบร้อยแล้วเช่นกัน และนำไปใส่ลงสู่ Buffer สำหรับเก็บข้อมูลการตรวจสอบมือ เมื่อใน Buffer มีข้อมูลเข้ามาเรียบร้อยแล้ว จะเข้าสู่การตรวจสอบใน

ขั้นตอนที่ 3 ทั้งนี้ นั่นคือการตรวจสอบหาจำนวนของค่าที่เก็บใน Buffer ที่เหมือนกันมากที่สุด โดยข้อมูลไม่จำเป็นต้องมีข้อมูลเต็ม Buffer ซึ่งจากภาพที่ 4-21 ในหัวข้อ INPUT FOR Hand คือข้อมูลล่าสุดที่ส่วนควบคุมภาคพื้นดินส่งขึ้นมา และในหัวข้อ Buffer for Hand ที่ตามด้วยตัวเลข ซึ่งคือตำแหน่งของบัพเฟอร์ที่แตกต่างกัน โดยจะมีหน้าที่หลักสำหรับเก็บข้อมูล และผลลัพธ์ที่ได้จะอยู่ในหัวข้อ Result for Hand ซึ่งเท่ากับ 0 นั่นเอง

```

INPUT FOR Hand : 0
Buffer for Hand 0 : 0
Buffer for Hand 1 :
Buffer for Hand 2 :
Result for Hand : 0
Hand Amount=0 : State 0 Roll=1500 Pitch=1500 Throttle=1000 Yaw=1500 FM=1000
INPUT FOR Hand : 0
Buffer for Hand 0 : 0
Buffer for Hand 1 : 0
Buffer for Hand 2 :
Result for Hand : 0
Hand Amount=0 : State 0 Roll=1500 Pitch=1500 Throttle=1000 Yaw=1500 FM=1000
INPUT FOR Hand : 0
Buffer for Hand 0 : 0
Buffer for Hand 1 : 0
Buffer for Hand 2 : 0
Result for Hand : 0
Hand Amount=0 : State 0 Roll=1500 Pitch=1500 Throttle=1000 Yaw=1500 FM=1000
INPUT FOR Hand : 0
Buffer for Hand 0 : 0
Buffer for Hand 1 : 0
Buffer for Hand 2 : 0
Result for Hand : 0

```

ภาพที่ 4-21 การตรวจสอบในขั้นตอนที่ 2 (ตรวจสอบความยาวข้อมูล) และ 3 (ตรวจสอบค่าที่เหมือนกันมากที่สุดในบัพเฟอร์) พร้อมแสดงผลลัพธ์สุดท้าย บน Serial Monitor ของโปรแกรม Arduino Sketch

จากภาพที่ 4-22 เป็นอีกตัวอย่างสำหรับตรวจสอบว่ามีมือหรือไม่อยู่บนอุปกรณ์ Leap Motion โดยสำหรับตัวอย่างนี้เป็นช่วงระหว่างการเปลี่ยนจากมีมือเป็นไม่มีมืออยู่บนอุปกรณ์ Leap Motion จะเห็นได้ว่าข้อมูลที่มีค่าเท่ากับ 0 จะต้องถูกใส่ลงใน Buffer มากกว่าครึ่งของขนาด Buffer ซึ่งในการทดลองนี้ผู้จัดทำได้กำหนดขนาด Buffer ไว้ที่ 3 เพราะฉะนั้นข้อมูลที่มีค่า 0 จึงต้องใส่ลงมา 2 ตัว จึงจะได้ผลลัพธ์สุดท้ายออกมาเท่ากับ 0 ซึ่งจะทำให้เกิดการล่าช้าไป 1-1.5 วินาทีต่อการอัปเดตการเปลี่ยนแปลงต่าง ๆ

```

INPUT FOR Hand : 1
Buffer for Hand 0 : 1
Buffer for Hand 1 : 1
Buffer for Hand 2 : 1
Result for Hand : 1
Hand Amount=1 : State 0 Roll=1500 Pitch=1500 Throttle=1000 Yaw=1500 FM=1000
INPUT FOR Hand : 1
Buffer for Hand 0 : 1
Buffer for Hand 1 : 1
Buffer for Hand 2 : 1
Result for Hand : 1
Hand Amount=1 : State 0 Roll=1500 Pitch=1500 Throttle=1000 Yaw=1500 FM=1000
INPUT FOR Hand : 0
Buffer for Hand 0 : 0
Buffer for Hand 1 : 1
Buffer for Hand 2 : 1
Result for Hand : 1
Hand Amount=1 : State 0 Roll=1500 Pitch=1500 Throttle=1000 Yaw=1500 FM=1000
INPUT FOR Hand : 0
Buffer for Hand 0 : 0
Buffer for Hand 1 : 0
Buffer for Hand 2 : 1
Result for Hand : 0
Hand Amount=0 : State 0 Roll=1500 Pitch=1500 Throttle=1000 Yaw=1500 FM=1000

```

ภาพที่ 4-22 การหาผลลัพธ์สุดท้ายขณะมีการเปลี่ยนแปลงระหว่างมีมือเป็นขณะไม่มีมือ
บน Serial Monitor ของโปรแกรม Arduino Sketch

4.2.4 การทดสอบโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่มีมือ

สำหรับส่วนนี้จะแบ่งการทำงานออกเป็น 4 การทำงานที่แตกต่างกันได้ดังนี้

1. ขั้นตอน Stand by เป็นขั้นตอนการนำมือไปไว้บนอุปกรณ์ Leap Motion และอยู่ในท่าแบบมือ โดยสามารถควบคุมการสร้างสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel, Yaw Channel ได้ตามการเปลี่ยนแปลงองศาในมุม Roll, Pitch, Yaw ของมือ แต่สำหรับค่าของ Throttle Channel และ Flight Mode Channel จะถูกกำหนดไว้ที่ 1000 สามารถดูการกำหนดสัญญาณ PWM ในขั้นตอนนี้ได้จากภาพที่ 4-23 ซึ่งสำหรับขั้นตอน Stand by จะไปตรงกับ State 0 ในโปรแกรม

Hand Amount=1 :	State 0	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=1000
Hand Amount=1 :	State 0	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=1000
Hand Amount=1 :	State 0	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=1000
Hand Amount=1 :	State 0	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=1000
Hand Amount=1 :	State 0	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1687	FM=1000
Hand Amount=1 :	State 0	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1687	FM=1000
Hand Amount=1 :	State 0	Roll=1500	Pitch=1437	Throttle=1000	Yaw=1687	FM=1000
Hand Amount=1 :	State 0	Roll=1500	Pitch=1375	Throttle=1000	Yaw=1687	FM=1000
Hand Amount=1 :	State 0	Roll=1500	Pitch=1375	Throttle=1000	Yaw=1687	FM=1000
Hand Amount=1 :	State 0	Roll=1500	Pitch=1375	Throttle=1000	Yaw=1687	FM=1000
Hand Amount=1 :	State 0	Roll=1500	Pitch=1375	Throttle=1000	Yaw=1687	FM=1000

ภาพที่ 4-23 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Stand by บน Serial Monitor ของโปรแกรม Arduino Sketch

2. ขั้นตอน Arm Motor เป็นขั้นตอนการทำท่าทางการกำมือค้างไว้บนอุปกรณ์ Leap Motion โดยสามารถควบคุมการสร้างสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel ได้ตามการเปลี่ยนแปลงองศาในมุม Roll, Pitch ของมือ แต่สำหรับค่าของ Throttle Channel และ Flight Mode Channel จะถูกกำหนดไว้ที่ 1000 และสำหรับค่า Yaw Channel จะกำหนดไว้ที่ 2000 เพื่อเป็นการสั่ง Arm Motor โดยสามารถดูการกำหนดสัญญาณ PWM ในขั้นตอนนี้ได้จากภาพที่ 4-24 ซึ่งสำหรับขั้นตอน Arm Motor จะไปตรงกับ State 1 ในโปรแกรม

Hand Amount=1 :	State 1	Roll=1437	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000
Hand Amount=1 :	State 1	Roll=1250	Pitch=1375	Throttle=1000	Yaw=2000	FM=1000

ภาพที่ 4-24 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Arm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch

3. ขั้นตอน Control Flight เป็นขั้นตอนการแบมืออยู่บนอุปกรณ์ Leap Motion โดยสามารถควบคุมการสร้างสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel, Yaw Channel ได้ตามการเปลี่ยนแปลงองศาในมุม Roll, Pitch, Yaw ของมือ แต่สำหรับค่าของ Throttle Channel จะเปลี่ยนแปลงตามการกระดิกนิ้วระหว่างนิ้วชี้ เพื่อเพิ่มค่าของ Throttle Channel และนิ้วกลาง เพื่อลด

ค่าของ Throttle Channel และในส่วนของ Flight Mode Channel จะถูกกำหนดไว้ที่ 2000 เพื่อให้โหมดสำหรับการบินอยู่ที่ Loiter Mode โดยสามารถดูการกำหนดสัญญาณ PWM ในขั้นตอนนี้ได้จากภาพที่ 4-25 ซึ่งสำหรับขั้นตอน Control Flight จะไปตรงกับ State 2 ในโปรแกรม

Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1437	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000

ภาพที่ 4-25 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Control Flight บน Serial Monitor ของโปรแกรม Arduino Sketch

4. ขั้นตอน Disarm Motor เป็นขั้นตอนการทำท่าทางการกำมือค้างไว้บนอุปกรณ์ Leap Motion โดยสามารถควบคุมการสร้างสัญญาณ PWM ในส่วนของ Roll Channel, Pitch Channel ได้ตามการเปลี่ยนแปลงองศาในมุม Roll, Pitch ของมือ แต่สำหรับค่าของ Throttle Channel กับ Yaw Channel จะถูกกำหนดไว้ที่ 1000 เพื่อทำการสั่ง Disarm Motor และสำหรับค่า Flight Mode Channel จะกำหนดไว้ที่ 2000 เพื่อกำหนดให้โหมดการบินยังอยู่ที่ Loiter Mode โดยสามารถดูการกำหนดสัญญาณ PWM ในขั้นตอนนี้ได้จากภาพที่ 4-26 ซึ่งสำหรับขั้นตอน Disarm Motor จะไปตรงกับ State 3 ในโปรแกรม

Hand Amount=1 :	State 3	Roll=1312	Pitch=1437	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1437	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1437	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1375	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1375	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1375	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1375	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1437	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1437	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1312	Pitch=1437	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=1 :	State 3	Roll=1250	Pitch=1437	Throttle=1000	Yaw=1000	FM=2000

ภาพที่ 4-26 การกำหนดค่าของสัญญาณ PWM ขณะมีมือควบคุมในขั้นตอน Disarm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch

4.2.5 การทดสอบโปรแกรมสำหรับกำหนดค่าให้กับสัญญาณ PWM ในกรณีที่ไม่มีมือหรือสัญญาณขาดหาย

สำหรับส่วนนี้จะต้องมีการตรวจสอบด้วยว่าขณะที่มีสัญญาณควบคุมอยู่นั้นดำเนินการถึงขั้นตอนใดแล้ว และเมื่อไม่มีมือควบคุมการบินหรือสัญญาณการเชื่อมต่อขาดหายจะต้องตรวจสอบว่าตรงกับการดำเนินการในขั้นตอนใด ซึ่งจะแบ่งเป็นฟังก์ชันสำหรับใช้งานในแต่ละขั้นตอนของการทำงานในโหมด Stand By, Arm Motor, Control Flight และ Disarm Motor โดยจะมีการทำงานที่แตกต่างกันออกไปดังนี้

1. ถ้าดำเนินการอยู่ในขั้นตอน Stand by จะทำการเรียกใช้งานฟังก์ชัน failSafeStateFlightControl_0() โดยจะมีการกำหนดสัญญาณของ PWM ตามภาพที่ 4-27 ซึ่งจะทำให้การกำหนดสัญญาณของแต่ละ Channel ให้เสมือนตอนกำลังถือ Transmitter Radio ไว้เฉย ๆ โดยไม่ควบคุมค่าใด ๆ

Signal is Disconnect :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Signal is Disconnect :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Signal is Disconnect :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Signal is Disconnect :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Signal is Disconnect :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Hand Amount=0 :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Hand Amount=0 :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Hand Amount=0 :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Hand Amount=0 :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Hand Amount=0 :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000
Hand Amount=0 :	State 0	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1500	FM=1000

ภาพที่ 4-27 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Stand by บน Serial Monitor ของโปรแกรม Arduino Sketch

2. ถ้าดำเนินการอยู่ในขั้นตอน Arm Motor จะทำการเรียกใช้งานฟังก์ชัน failSafeStateFlightControl_1() โดยจะมีการกำหนดสัญญาณของ PWM ตามภาพที่ 4-28 ซึ่งจะเป็นการกำหนดสัญญาณของแต่ละ Channel ให้เสมือนกำลังทำการ Disarm Motor อยู่เพื่อป้องกันอันตราย

Signal is Disconnect :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Signal is Disconnect :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Signal is Disconnect :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Signal is Disconnect :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Signal is Disconnect :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Signal is Disconnect :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Hand Amount=0 :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Hand Amount=0 :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Hand Amount=0 :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000
Hand Amount=0 :	State 1	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=1000

ภาพที่ 4-28 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Arm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch

3. ถ้าดำเนินการอยู่ในขั้นตอน Control Flight จะทำการเรียกใช้งานฟังก์ชัน failSafeStateFlightControl_2() โดยจะมีการกำหนดสัญญาณของ PWM ที่แตกต่างกัน 2 กรณี สำหรับกรณีที่ 1. จะเป็นการกำหนดสัญญาณของแต่ละ Channel ให้มอเตอร์สามารถลอยนิ่งอยู่บนอากาศได้ สำหรับกรณีที่ตรวจสอบพบค่าของ Throttle Channel มากกว่า 1350 ขึ้นไป แต่ถ้าค่าของ Throttle Channel น้อยกว่า 1350 จะเป็นรูปแบบสำหรับกรณีที่ 2 ซึ่งจะกำหนดสัญญาณแต่ละ Channel ให้เสมือนกำลังทำการ Disarm Motor เพื่อป้องกันอันตรายใด ๆ ตามภาพที่ 4-29

Signal is Disconnect :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 2	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000

ภาพที่ 4-29 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Control Flight บน Serial Monitor ของโปรแกรม Arduino Sketch

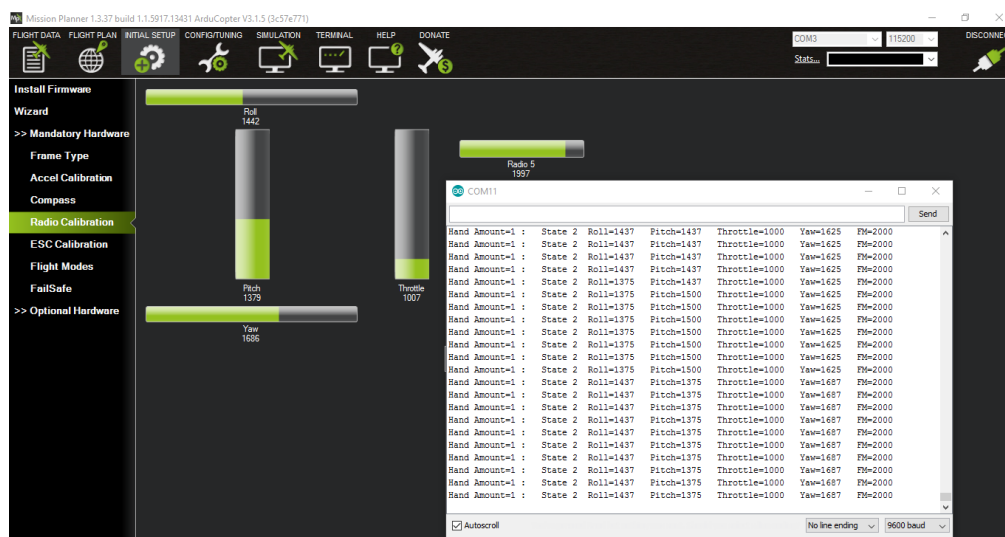
4. ถ้าดำเนินการอยู่ในขั้นตอน Disarm Motor จะทำการเรียกใช้งานฟังก์ชัน failSafeStateFlightControl_3() โดยจะมีการกำหนดสัญญาณของ PWM ตามภาพที่ 4-30 ซึ่งจะเป็นการกำหนดสัญญาณของแต่ละ Channel ให้เสมือนกำลังทำการ Disarm Motor ต่อเนื่องไปเลย เพื่อสั่งให้มอเตอร์หยุดทำงาน

Signal is Disconnect :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Signal is Disconnect :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000
Hand Amount=0 :	State 3	Roll=1500	Pitch=1500	Throttle=1000	Yaw=1000	FM=2000

ภาพที่ 4-30 การกำหนดค่าของสัญญาณ PWM ขณะไม่มีมือควบคุมในขั้นตอน Disarm Motor บน Serial Monitor ของโปรแกรม Arduino Sketch

4.2.6 การทดสอบโปรแกรมสำหรับสร้างสัญญาณ PWM

เป็นการเรียกใช้งานคำสั่ง `delayMicrosecond()` เพื่อค้างสถานะลอจิกตามเวลาที่ต้องการ โดยจะรับค่าของ Roll, Pitch, Throttle, Yaw, Flight Mode จากขั้นตอนในหัวข้อ 4.2.4 และ 4.2.5 มากำหนดให้กับ Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel ตามลำดับ โดยสามารถอ่านค่าของสัญญาณ PWM ที่สร้างขึ้นได้จากโปรแกรม Mission Planner เพื่อตรวจสอบความถูกต้องของสัญญาณที่สร้างขึ้นผ่าน Digital Pin และส่งออกไปให้กับ Input Pin ของ Flight Controller สามารถดูตัวอย่างได้จากภาพที่ 4-31 ซึ่งจะเป็นการแสดงค่าของสัญญาณ PWM จากฝั่งของ Arduino ที่ทำการสร้างสัญญาณขึ้น บน Serial Monitor และตรวจสอบค่าของสัญญาณ PWM ที่ Flight Controller อ่านได้ผ่านโปรแกรม Mission Planner



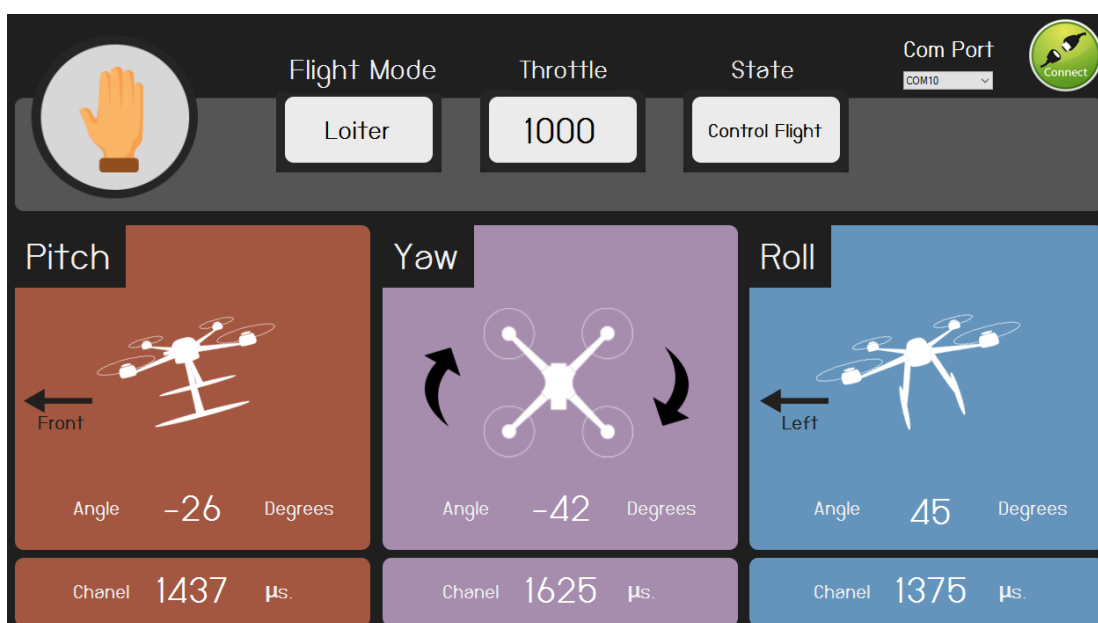
ภาพที่ 4-31 ค่าของสัญญาณ PWM ที่สร้างขึ้น บน Serial Monitor ของโปรแกรม Arduino Sketch และตรวจสอบค่าของสัญญาณ PWM ผ่าน โปรแกรม Mission Planner

4.2.7 การทดสอบโปรแกรมสำหรับส่งข้อมูลไปยังส่วนควบคุมภาคพื้นดิน

สำหรับส่วนนี้จะเป็นการส่งข้อมูลของค่าที่กำหนดให้กับ Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel พร้อมทั้งขั้นตอนการทำงานทั้ง 4 ขั้นตอนที่กำลังทำงาน ณ ขณะนั้น ออกไปให้กับส่วนควบคุมภาคพื้นดินเพื่อนำไปแสดงบน Desktop Application สามารถดูได้จากภาพที่ 4-32 ซึ่งคือภาพที่ฝั่งของ Arduino UNO สร้างสัญญาณในส่วน Roll Channel, Pitch Channel, Throttle Channel, Yaw Channel, Flight Mode Channel พร้อมทั้งระบุขั้นตอนการทำงานซึ่ง State เท่ากับ 2 แสดงว่าอยู่ในขั้นตอนการ Control Flight ตามที่ได้อธิบายไว้ในหัวข้อ 3.2.8 โดยแสดงผลบน Serial Monitor ของโปรแกรม Arduino Sketch และภาพที่ 4-33 คือภาพแสดงส่วนของ Desktop Application ที่รับค่าดังกล่าวจากฝั่งของส่วนควบคุมภาคอากาศเข้ามาแสดง

Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000
Hand Amount=1 :	State 2	Roll=1375	Pitch=1437	Throttle=1000	Yaw=1625	FM=2000

ภาพที่ 4-32 ค่าของสัญญาณ PWM ทั้ง 5 Channel ที่สร้างขึ้นพร้อมระบุขั้นตอนการทำงานขณะนั้นบน Serial Monitor ของโปรแกรม Arduino Sketch



ภาพที่ 4-33 โปรแกรมแสดงผลข้อมูลในรูปแบบ Desktop Application ที่รับค่ามาจากส่วนควบคุมภาคอากาศ

บทที่ 5

สรุปผล ข้อเสนอแนะ และปัญหา

5.1 สรุปผล

จากการพัฒนาระบบสำหรับควบคุมการบินของมัลติโรเตอร์ด้วยอุปกรณ์ Leap Motion โดยการใช้มือควบคุมเพียงมือเดียว เพื่อให้การควบคุมการบินของมัลติโรเตอร์นั้นง่ายขึ้นและลดความซับซ้อนในการควบคุมลง โดยสามารถช่วยลดระยะเวลาในการเรียนรู้การควบคุมการบินของมัลติโรเตอร์ให้มีระยะเวลาในการเรียนรู้ที่น้อยลงจากเดิม เนื่องจากการใช้มือในการควบคุมการบินของมัลติโรเตอร์ด้วยท่าทางต่าง ๆ ของมือนั้น เป็นสิ่งที่มนุษย์ทุกคนสามารถทำได้และไม่จำเป็นต้องเรียนรู้วิธีการทำท่าทางต่าง ๆ เพียงแต่ต้องจำให้ได้ก่อนว่าท่าใดใช้สำหรับทำงานในขั้นตอนใดและสามารถทำอะไรได้บ้างในขั้นตอนนั้น

5.2 ปัญหาที่พบ

5.2.1 การเคลื่อนไหวของมือเพียงเล็กน้อย ส่งผลต่อการเปลี่ยนแปลงของมุม Roll, Pitch, Yaw แก้ไขโดยกำหนดช่วงขององศามุม Roll, Pitch, Yaw ให้มีช่วงที่กว้างขึ้นก่อนจะเกิดการเปลี่ยนแปลง เช่น 20 องศา ต่อการเปลี่ยนแปลง 1 องศา เป็นต้น

5.2.2 การรับข้อมูลที่ฝั่งของ Arduino UNO จากส่วนควบคุมภาคพื้นดิน มีอัตราความผิดพลาดสูงเนื่องจาก Wireless Datalink มีอัตราการส่งข้อมูลค่อนข้างเร็ว แก้ไขโดยกำหนดความถี่ในการส่งข้อมูลที่ส่วนควบคุมทางภาคพื้นดินให้มีความถี่ 2Hz และที่ฝั่ง Arduino UNO มีการตรวจสอบค่าก่อนนำไปใช้งาน

5.2.3 สัญญาณ PWM ที่จ่ายให้กับ Input Pin ของ Flight Controller ต้องมีการทำงานที่มีความถี่ 400-500Hz หรือ 50Hz ซึ่งถ้าใช้คำสั่ง analogWrite() จะต้องใช้ Pin ที่สามารถสร้างสัญญาณ PWM ได้และต้องมีความถี่ PWM ตามช่วงดังกล่าว ซึ่งจะสามารถใช้ได้เพียง Digital Pin 3, 9, 10, 11 ซึ่งไม่เพียงพอต่อการใช้งาน แก้ไขโดยใช้ Servo Library ในการสร้างสัญญาณ PWM แบบ Soft PWM แทน ซึ่งมีความถี่ของสัญญาณ PWM เท่ากับ 50 Hz จึงสามารถทดแทนได้

5.2.4 สัญญาณ PWM ที่จ่ายให้กับ Input Pin ของ Flight Controller มีจังหวะเพิ่มหรือลดเองจากค่ากำหนดไว้ด้วยบอร์ด Arduino UNO

5.2.5 มีข้อจำกัดการใช้งานสำหรับควบคุมการทำงานของแต่ละ Channel ด้วยมือเพียงมือเดียว ซึ่งถ้าต้องการควบคุมการทำงานของ Gimbal ด้วยจะไม่สามารถทำได้ เนื่องจากบางท่าทางไม่สามารถตรวจจับได้โดยง่าย เช่น การกระดิกนิ้วโป้ง หรือบางท่าเมื่อทำแล้วจะส่งผลเสียหายมากกว่าผลดี เช่น การทำท่าหมุนมือเป็นวงกลม จะทำให้ค่าของมุม Roll, Pitch, Yaw เกิดการเหวี่ยงมากเกินไป หรือการกระดิกนิ้วกลางกับนิ้วนาง มักจะทำงานพร้อม ๆ กัน

5.3 ข้อเสนอแนะ

5.3.1 ปรับเปลี่ยนการควบคุมจากการควบคุมเพียงมือเดียวเป็นสองมือ เพื่อให้สามารถควบคุมการทำงานได้หลาย Channel มากขึ้น

5.3.2 เปลี่ยน Controller จาก Arduino UNO เป็นบอร์ดรุ่นอื่นหรือชนิดอื่นที่ประมวลผลได้เร็วขึ้น เพื่อการตอบสนองที่เร็วขึ้นของระบบ

เอกสารอ้างอิง

1. Servo Library. Arduino Reference. [Online].
Available from : <https://www.arduino.cc/en/Reference/Servo>
2. SoftwareSerial Library. Arduino Reference. [Online].
Available from : <https://www.arduino.cc/en/Reference/SoftwareSerial>
3. PWM Frequency. Arduino Reference. [Online].
Available from : <https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM>
4. Leap Motion. Leap Motion SDK Document. [Online].
Available from : <https://developer.leapmotion.com/documentation/v2/csharp/index.html>
5. C# Programing. MSDN C# Reference. [Online].
Available from : [https://msdn.microsoft.com/library/618ayhy6\(VS.110\).aspx](https://msdn.microsoft.com/library/618ayhy6(VS.110).aspx)
6. Stack Overflow Community. Q&A Programing Technical. [Online].
Available from : <http://stackoverflow.com/>
7. Arduino Pinout Diagram. Arduino Uno Board. [Online].
Available from : <http://foros.giltesa.com/otros/arduino/fc/docs/pinout/uno.jpg>
8. APM Setup. APM Mission Planner. [Online].
Available from : <http://ardupilot.org/planner2/index.html>

ประวัติผู้แต่ง

ปรินญาณพนธ์เรื่อง : การควบคุมอากาศยานไร้คนขับอัจฉริยะด้วยลิปโมชั้น
 สาขาวิชา : วิศวกรรมคอมพิวเตอร์
 ภาควิชา : วิศวกรรมไฟฟ้าและคอมพิวเตอร์
 คณะ : วิศวกรรมศาสตร์
 ชื่อ : นางสาวพรกมล ชูชุมพร

ประวัติ

เกิดวันที่ 30 มิถุนายน พ.ศ. 2535 อยู่บ้านเลขที่ 77/4 หมู่บ้านสวนตะไคร้ ถนนสวนตะไคร้
 ซอย 3 ตำบลสนามจันทร์ อำเภอเมือง จังหวัดนครปฐม สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย
 แผนการเรียนวิทยาศาสตร์-คณิตศาสตร์ โรงเรียนราชินีบูรณะ ปีการศึกษา 2553 และสำเร็จการศึกษา
 ในระดับปริญญาตรี สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
 คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปีการศึกษา 2558

ชื่อ : นายรอยบุญ ลือพัศตรา

ประวัติ

เกิดวันที่ 13 สิงหาคม พ.ศ. 2535 อยู่บ้านเลขที่ 4/163 หมู่ที่ 5 หมู่บ้านอริยวัฒน์ ถนนยิงเป้าใต้
 ตำบลสนามจันทร์ อำเภอเมือง จังหวัดนครปฐม สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย
 แผนการเรียนวิทยาศาสตร์-คณิตศาสตร์ โรงเรียนสิรินธรราชวิทยาลัย ปีการศึกษา 2553 และสำเร็จ
 การศึกษาในระดับปริญญาตรี สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและ
 คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปี
 การศึกษา 2558