

PID Tunner an ArduCopter

รายวิชา 242-401 Computer Engineering Project II

ภาคการศึกษา 2 / 2560

รายชื่อผู้จัดทำ

นายจตุภัทร์ ปานน้อย รหัสนักศึกษา 5735512002

อาจารย์ที่ปรึกษา	อาจารย์ ดร.นพพล	เลิศชูวงศ์
อาจารย์ที่ปรึกษาร่วม	อาจารย์พัชรี	เทพนิมิตร
อาจารย์ที่ปรึกษาร่วม	อาจารย์กุลวรรณ	เชawanwathī

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยสงขลานครินทร์

ชื่อโครงการ

PID Tuning an ArduCopter

โดย

นายจตุภัทร์ ปานน้อย รหัสนักศึกษา 5735512002

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

รายงานนี้เป็นส่วนหนึ่งของวิชา

242-402 Computer Engineering Project II

อาจารย์ที่ปรึกษาโครงการ

(อาจารย์ ดร.นพพณ เลิศชูวงศ์)

อาจารย์ที่ปรึกษาร่วมโครงการ

(อาจารย์พัชรี เพพนิมิตร)

อาจารย์ที่ปรึกษาร่วมโครงการ

(อาจารย์กุลวรรณ เชawanwath)

ชื่อโครงการ PID Tunner an ArduCopter

ผู้จัดทำ นายจตุภร์ ปานน้อย 57355512002

ภาควิชา วิศวกรรมคอมพิวเตอร์

ปีการศึกษา 2560

อาจารย์ที่ปรึกษาโครงการ

คณะกรรมการสอบ

(อาจารย์ ดร.นพพณ เลิศช่วงศ่า)

(อาจารย์พัชรี เทพนิมิตร)

(อาจารย์กุลวรรณ์ เชวนวากิ)

โครงการนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตร์ปัจจุบัน

สาขาวิชาวิศวกรรมคอมพิวเตอร์

(ดร.คณสันต์ กาญจนสิทธิ์)

รองหัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์ วิทยาเขตภูเก็ต

ชื่อโครงการ การจูนระบบพีไอดีโดรนอาดูคอปเตอร์

ผู้จัดทำ นายจตุภัทร ปานน้อย 57355512002

ภาควิชา วิศวกรรมคอมพิวเตอร์

ปีการศึกษา 2560

บทคัดย่อ

โครงการการจูนระบบพีไอดีของโดรนอาดูคอปเตอร์ ซึ่งเป็นแบบมัลติคอปเตอร์ (MultiCopter) ด้วยการปรับปรุงโดรนที่มีอยู่แล้วให้มีสามารถใช้งานให้มีประสิทธิภาพสูงขึ้น เนื่องจากในปัจจุบันโดรนที่มีประสิทธิภาพสูงและสามารถใช้งานได้ดีนั้นย่อมมีราคาแพง ซึ่งผู้เล่นโดรนหรือมือสมัครเล่นส่วนมากยังไม่มีประสบการณ์ในการบังคับควบคุมโดรนมากนักอาจทำให้เกิดความเสียหายต่อตัวโดรนได้ ดังนั้นจึงได้ทำการทดลองที่จะช่วยให้การควบคุมโดรนให้เป็นเรื่องที่ง่ายโดยผู้เล่นหรือมือสมัครเล่นไม่จำเป็นต้องมีประสบการณ์ในการบังคับโดรนมาก่อน โดยจะทำการการปรับปรุงระบบต่าง ๆ ให้ดีขึ้น ทำการปรับปรุงซอฟต์แวร์เพื่อให้โดรนสามารถทำงานได้ดีขึ้น โดยการใช้ระบบควบคุมสัตว์ส่วน-ปริพันธ์-อนุพันธ์ (PID Controller) และการใช้ตัวกรองค่าล蔓น (Kalman Filter) เข้ามาช่วยในการปรับปรุงซอฟต์แวร์หรือซอฟต์แวร์ที่มีการนำระบบจีพีเอสมาประยุกต์ใน การควบคุมโดรน

สำหรับโดรนที่ได้รับการพัฒนาและปรับปรุงแล้ว จะมีความเสถียรภาพสูงและง่ายต่อการควบคุม แม้ผู้เล่นจะไม่มีประสบการณ์ในการเล่นก็ตาม โดยโดรนสามารถทรงตัวอยู่ได้โดยไม่ต้องควบคุมผ่านรีโมทและสามารถกำหนดตำแหน่งของตัวโดรนโดยใช้พิกัดละติจูดและลองติจัดผ่านโมดูลจีพีเอสได้ โดยโดรนจะบินไปตามตำแหน่งที่เรากำหนดได้ตามต้องการ โดยไม่ต้องใช้งานผ่านรีโมทเลย

Project Title	PID Tuning an ArduCopter
Author	Mr.Jatupat Pannoi 57355512002
Department	Computer Engineering
Academic Year	2017

Abstract

Projects PID tuning an ArduCopter. The multi-copter drones (MultiCopter) improved drone that are already available to provide higher efficiency. Currently drone are highly effective and can work well, it will be expensive. The drone players or amateurs, most of them do not have the experience to control drone so could cause damage to the drone had. Therefore, the project will help control the drone make it easy by players or amateurs do not need to have experience in a drone before. It will improve the systems that provide process improvement software so drones can perform better. By using a system of proportional control, integral, derivative (PID Controller) and the use of filtering Kalman (Kalman Filter) to help improve the software, or source code, and also the introduction of GPS applications in control drone.

For drone that have been developed and improved. It is highly stable and easy to control. Although players will not have the experience of playing time. The drone can be sustained without a remote control and can determine the position of the drone by using latitude and longitude continue taking it through the GPS module has. The drone will fly to the position that we define as needed. Without having to use a remote yet.

กิตติกรรมประกาศ

โครงการหัวข้อ PID Tuning an ArduCopter นี้ได้รับความเอื้อเพื่อและการดูแลเอาใจใส่จนทำให้ โครงการสำเร็จลุล่วงไปได้ด้วยดีจากอาจารย์ ดร.นพพณ เลิศชูวงศ์ อาจารย์ที่ปรึกษาโครงการที่ค่อยช่วยเหลือ ให้ คำปรึกษาและถ่ายทอดความรู้ที่เป็นประโยชน์แก่โครงการนี้มาตลอด ขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณครอบครัวที่ค่อยให้กำลังใจและสนับสนุนการทำโครงการนี้มาโดยตลอดและสุดท้ายนี้ ขอขอบคุณ พี่ ๆ และเพื่อน ๆ ทุกคนที่ได้ให้ความช่วยเหลือและให้กำลังใจในการทำโครงการนี้

นายจตุภัทร์ ปานน้อย

ผู้จัดทำ

25 มกราคม 2559

สารบัญ

เนื้อหา	หน้า
บทคัดย่อ	iv
Abstract	v
กิตติกรรมประกาศ.....	vi
บทที่ 1 บทนำ.....	3
1.1 ความเป็นมา	3
1.2 วัตถุประสงค์ของโครงการ.....	3
1.3 ขอบเขตของโครงการ	4
1.4 ขั้นตอนในการดำเนินงาน.....	5
1.5 ประโยชน์ที่คาดว่าจะได้รับ	5
1.6 สถานที่ทำงาน.....	5
1.7 เครื่องมือที่ใช้ในการพัฒนา	6
บทที่ 2 ความรู้พื้นฐาน.....	7
2.1 ความหมายของโดรน (Drone).....	7
2.2 ข้อดี - ข้อเสียของโดรน (Drone)	8
2.3 หลักการทำงานของโดรน (Drone)	8
2.4 การแสดงมุมเอียงและการเปลี่ยนแกนอ้างอิง	11
2.5 ชิ้นส่วนของโดรน	14
2.6 โปรแกรม Mission Planner	28
2.7 โปรแกรม Arduino IDE – ArduPilot.....	30
2.8 โปรแกรม Eclipse	31
2.9 โปรแกรม Matlab	32
2.10 โปรแกรม Virtual Studio	33
2.11 โปรแกรม Android Studio	34
2.12 ภาษา C+	35
2.13 การควบคุมแบบพีไอดี (PID Controller)	35
2.14 PID Tuning.....	39
2.15 หลักการ Kalman Filter.....	43
2.16 Google Maps API	47

สารบัญ (Continue)

เนื้อหา	หน้า
บทที่ 3 รายละเอียดการทำงาน.....	48
3.1 System Architecture	48
3.2 Model Mechanic.....	49
3.3 Model Controller	50
3.4 System Implementation.....	54
3.5 แผนการดำเนินงาน.....	75
บทที่ 4 การทดลอง.....	78
4.1 การทดลองที่ 1 การทำ Binding Remote กับ Receiver.....	78
4.2 การทดลองที่ 2 การโปรแกรมการทำงานของ Remote Flysky FS - TH9X	80
4.3 การทดลองที่ 3 การรีเซ็ตค่าเริ่มต้น (Default)ให้กับบอร์ด ArduPilotMega	82
4.4 การทดลองที่ 4 การติดตั้งเฟิร์มแวร์ให้กับบอร์ด ArduPilotMega	84
4.5 การทดลองที่ 5 การทำ Accel calibration ของบอร์ด ArduPilotMega.....	87
4.6 การทดลองที่ 6 ทำการ Calibrate Compass ของบอร์ด ArduPilotMega	90
4.7 การทดลองที่ 7 การ Calibrate ESC.....	94
4.8 การทดลองที่ 8 ทำการ Calibrate Radio ของบอร์ด ArduPilotMega	96
4.9 การทดลองที่ 9 ทำการตั้งค่าโหมดการบินของบอร์ด ArduPilotMega	97
4.10 การทดลองที่ 10 ทำการรับค่า roll, pitch, yaw ขณะที่บอร์ดอยู่กับที่	98
4.11 การทดลองที่ 11 การเตรียมความพร้อมของการทำ PID Tuning	101
4.12 การทดลองที่ 12 ทดสอบการหมุนของตัวโดรนขณะที่บอร์ดมีการเปลี่ยนแปลงตำแหน่ง.....	103
4.13 การทดลองที่ 13 ทดสอบการหมุนที่แกน Roll	105
4.14 การทดลองที่ 14 ทดสอบการหมุนที่แกน Pitch	110
4.15 การทดลองที่ 15 ทดสอบการบินของโดรนโดยไม่มีการจูนค่า	115
4.16 การทดลองที่ 16 ทดสอบการบินของโดรนโดยไม่มีแรงประท JACKER (จูนค่าแล้ว)....	116
4.17 การทดลองที่ 17 ทดสอบการบินของโดรนโดยมีแรงลมประท JACKER จากลมของพัดลม	117
4.18 การทดลองที่ 18 การปรับค่าแบบ PID โดยให้ค่ามีความละเอียดชี้.....	118
4.19 การทดลองที่ 19 ทดสอบการบินของโดรนในกรณีต่าง ๆ.....	120

สารบัญ (Continue)

เนื้อหา	หน้า
บทที่ 5 สรุปผลการดำเนินงาน.....	121
5.1 สรุปการทำงาน	121
5.2 ปัญหาและอุปสรรค	121
บรรณานุกรม.....	122

สารบัญรูปภาพ

รูปที่	หน้า
รูปที่ 2 - 1 ทิศทางการหมุนของมอเตอร์แบบ Hovering.....	9
รูปที่ 2 - 2 ทิศทางการหมุนของมอเตอร์แบบ Up-Down	9
รูปที่ 2 - 3 ทิศทางการหมุนมอเตอร์แบบ Roll.....	10
รูปที่ 2 - 4 ทิศทางการหมุนมอเตอร์แบบ Pitch.....	10
รูปที่ 2 - 5 ทิศทางการหมุนมอเตอร์แบบ Yaw	10
รูปที่ 2 - 6 การทำงานของ Strapdown INS operating concept	11
รูปที่ 2 - 7 หลักการ Assumptions และ Sign Convention.....	12
รูปที่ 2 - 8 Rotating Reference Frame	13
รูปที่ 2 - 9 Coordinate Transformation with Rotating Reference Frame	13
รูปที่ 2 - 10 รูปทรงใบพัด Chord line และ Trailing Edge.....	14
รูปที่ 2 - 11 Angle of Attack	15
รูปที่ 2 - 12 ใบพัด	15
รูปที่ 2 - 13 องค์ประกอบของมอเตอร์บล็อสเลส	16
รูปที่ 2 - 14 Motor Brushless	17
รูปที่ 2 - 15 Electronics speed controller	18
รูปที่ 2 - 16 ลำตัวเฟรม	18
รูปที่ 2 - 17 Receiver.....	19
รูปที่ 2 - 18 Remote	20
รูปที่ 2 - 19 โครงสร้างภายในของวิทยุ Flysky FS-TH9X.....	21
รูปที่ 2 - 20 โครงสร้างการทำงานหลักของวิทยุ Flysky FS-TH9X	22
รูปที่ 2 - 21 Board ArduCopterMega	23
รูปที่ 2 - 22 แผนภาพแสดง Sensor ที่มีภายในบอร์ด ArduCopterMega	24
รูปที่ 2 - 23 แผนภาพแสดง IC ภายในบอร์ด ArduCopterMega	24
รูปที่ 2 - 24 Li-Po Battery	26

สารบัญรูปภาพ (Continue)

รูปที่	หน้า
รูปที่ 2 - 25 Power Module	26
รูปที่ 2 - 26 GPS Module.....	26
รูปที่ 2 - 27 Telemetry Radio	27
รูปที่ 2 - 28 Program Mission Planner	28
รูปที่ 2 - 29 รายละเอียดต่าง ๆ ของโปรแกรม Mission Planner	29
รูปที่ 2 - 30 Program Arduino IDE – ArduPilot	30
รูปที่ 2 - 31 Program Eclipse	31
รูปที่ 2 - 32 Program Matlab	32
รูปที่ 2 - 33 Program Virtual Studio	33
รูปที่ 2 - 34 Program Android Studio.....	34
รูปที่ 2 - 35 ตัวควบคุม PID แบบขานาน.....	35
รูปที่ 2 - 36 ลักษณะสมบัติของการควบคุมแบบสัดส่วน	36
รูปที่ 2 - 37 quarter-amplitude decay.....	40
รูปที่ 2 - 38 กระบวนการของ Kalman Filter	43
รูปที่ 3 - 2 แผนภาพ diagram แสดงการทำงานโดยภาพรวมของโดรน	48
รูปที่ 3 - 3 ตัวลำที่ยังไม่ประกอบ	49
รูปที่ 3 - 4 ตัวลำที่ประกอบเรียบร้อยแล้ว	49
รูปที่ 3 - 5 ตัวลำที่ประกอบอุปกรณ์ครบ	50
รูปที่ 3 - 6 แผนภาพระบบโดยรวม	50
รูปที่ 3 - 7 แผนภาพของระบบโดยละเอียด	51
รูปที่ 3 - 8 แผนภาพการทำงานของระบบควบคุมเสถียรภาพ	51
รูปที่ 3 - 9 (a) แผนภาพการทำงานของระบบควบคุมแบบพี.....	52
รูปที่ 3 - 9 (b) แผนภาพการทำงานของระบบควบคุมแบบพีไอ	52
รูปที่ 3 - 9 (c) แผนภาพการทำงานของระบบควบคุมแบบพีไอดี	52

สารบัญรูปภาพ (Continue)

รูปที่	หน้า
รูปที่ 3 - 10 เว็บ GitHub.....	54
รูปที่ 3 - 11 คำสั่งในการดาวน์โหลด code.....	55
รูปที่ 3 - 12 Library ของ code ArduPilot.....	55
รูปที่ 3 - 13 Diagram ระบบการทำงานโดยรวมของบอร์ด APM.....	56
รูปที่ 3 - 14 หน้าเว็บดาวโหลด ArduPilot	72
รูปที่ 3 - 15 แฟลกไฟล์.zip	72
รูปที่ 3 - 16 เลือกที่อยู่ sketchbook.....	72
รูปที่ 3 - 17 Set Board Arduino	73
รูปที่ 3 - 18 Set port Arduino.....	73
รูปที่ 3 - 19 Set HAL Board Arduino	74
รูปที่ 3 - 20 ทดสอบการเบริน์ลงบอร์ด	74

สารบัญตาราง

ตารางที่	หน้า
ตารางที่ 2 - 1 Ziegler-Nichols Close-Loop Tuning.....	40
ตารางที่ 2 - 2 Damped Oscillation Close-Loop Tuning	41
ตารางที่ 3 - 1 การต่อหัวบอร์ด Arducopter 2.8	53
ตารางที่ 3 - 2 การต่อหัวรีเซวเวอร์ Flysky FS-R9B	54
ตารางที่ 3 - 3 วิธีการปรับแต่งค่าแบบ Ziegler–Nichols	71
ตารางที่ 3 - 4 แผนการดำเนินงาน <i>Project Preparation</i>	75
ตารางที่ 3 - 5 แผนการดำเนินงาน <i>Project I</i>	76
ตารางที่ 3 - 6 แผนการดำเนินงาน <i>Project II</i>	77

สารบัญคำย่อ

CW	Clock Wise
CCW	Counter Clock Wise
RC	Radio Control
RF	Radio Frequency
DC	Direct Current
GPS	Global Positioning System
ESC	Electronic Speed Control
TC	Thro Curve
PC	Pitch Curve
PPM	Pulse Position Modulation
PCM	Pulse Code Modulation
GFSK	Gaussian Frequency Shift Keying
AFHDS	Automatic Frequency Hopping Digital System
APM	Arduino Pilot Mega
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
PWM	Pulse With Modulated
BLDC	Brushless Direct Current
DCM	Direction Cosine Matrix

บทที่ 1 บทนำ

บทนี้จะเป็นการกล่าวถึงความเป็นมาของโครงการ PID Tuning an ArduCopter รวมไปถึงวัตถุประสงค์ ขอบเขต ขั้นตอนในการดำเนินงาน ประโยชน์ที่คาดว่าจะได้รับ สถานที่ทำโครงการ และเครื่องมือที่ใช้ในการพัฒนาของโครงการ

1.1 ความเป็นมา

ในปัจจุบันโดรน (Drone) หรือเฮลิคอปเตอร์หลายใบพัด (Multi-Helicopter) นั้นได้เป็นที่นิยมมาก ซึ่งสามารถนำไปใช้งานได้ในด้านต่าง ๆ ได้ เช่น การถ่ายภาพนิ่งทางอากาศ การถ่ายวิดีโอทางอากาศ รวมถึงการถ่ายทอดสดทางอากาศ (Streaming) การบินสำรวจจากที่สูง ซึ่งโดรนนั้นมีหลากหลายชนิด และแต่ละชนิดก็มีขนาดและราคาที่แตกต่างกันไปตามประสิทธิภาพของโดรนด้วยกัน ความสามารถที่ทำได้ก็ไม่เท่ากัน โดยส่วนมาก โดรนที่มีความสามารถ หรือประสิทธิภาพสูง ๆ นั้นจะมีราคาจะค่อนข้างแพง ยกตัวอย่างเช่น Phantom 4 Pro ซึ่งราคาเกือบแสนกว่าบาท ซึ่งมีฟังก์ชันการทำงานที่หลายหลาย ซึ่งแตกต่างจาก Cheerson CX-10D ราคาประมาณหนึ่งพันบาทโดยความสามารถที่ทำได้คือบินได้อย่างเดียว ไม่มีลูกเล่นเพิ่มเติม ซึ่งต่างจาก Phantom 4 Pro อย่างเห็นได้ชัด

ผู้จัดทำจึงได้คิดริเริ่ม ปรับปรุงโดรนให้มีประสิทธิภาพการทำงานสูงและราคาไม่แพง จึงนำระบบการควบคุมแบบสัดส่วน-ปริพันธ์-อนุพันธ์ (PID Controller) หรือการควบคุมการทรงตัวที่ดีและหลักการของตัวกรองคาลแมน (Kalman Filter) มาปรับใช้กับโดรนเพื่อที่จะทำให้โดรนนั้นมีประสิทธิภาพสูงและราคาถูก ช่วยประหยัดงบประมาณในการจัดซื้อดroneที่มีคุณภาพสูงมาใช้งาน หลังจากโครงการสำเร็จแล้วเราอาจจะได้โดรนที่เหมือนกับโดรนที่มีลักษณะตามที่มีขายตามห้องตลาดทั่วไป โดยจะแตกต่างกันตรงที่ราคาและยังมีประสิทธิภาพที่ดี มีความสามารถเทียบเท่าหรือสูงกว่าได้

1.2 วัตถุประสงค์ของโครงการ

- เพื่อปรับปรุงโครงสร้าง รวมถึงส่วนประกอบของโดรนเพื่อให้การทำงานมีประสิทธิภาพมากยิ่งขึ้น
- เพื่อพัฒนาและปรับปรุง code ที่ใช้ในการควบคุมโดรนโดยอาศัยหลักการทำงานของ PID Controller และ Kalman Filter เพื่อให้โดรนมีการทำงานอย่างเต็มประสิทธิภาพสูงสุด

1.3 ขอบเขตของโครงงาน

- นำระบบ PID Controller และ Kalman Filter มาใช้ในการควบคุมการทำงานของโดรน
- นำระบบ GPS มาช่วยในการควบคุมตำแหน่งของโดรน
- เขียนแอพพลิเคชันเพื่อใช้ในการควบคุมโดรนโดยไม่ใช้รีโมท

ในส่วนของพัฒนาโปรเจก

- ศึกษาการทำงานของโดรน
- ศึกษาเกี่ยวกับรายละเอียดของอุปกรณ์ของโดรน
- หาข้อมูลและจัดซื้ออุปกรณ์
- ศึกษาหลักการทำงานของ PID Controller และ Kalman Filter
- ศึกษาวิธีการติดตั้งโปรแกรมและวิธีการใช้งานโปรแกรมต่าง ๆ
- ศึกษาการทำงานของ Code ที่ใช้ในการควบคุมโดรน
- ทำการแก้ไข ปรับแต่ง การทำงานของ Code ที่ใช้ในการควบคุมตัวโดรนแบบง่าย ๆ

ในส่วนของโครงงาน I

- ศึกษาทฤษฎีการปรับแต่ง PID Controller และ Kalman Filter
- เพิ่มเติมส่วนของ Code สำหรับแยกการทำงานของ PID สำหรับแกน Roll และแกน Pitch
- ทำการปรับแต่งโดยใช้ PID Controller และ Kalman Filter และบันทึกผล (P, PI และ PID)
- ทำการจูน PID แบบ 2D (แกน Roll + Pitch)
- ศึกษาข้อมูลเกี่ยวกับตัว Telemetry
- หาข้อมูลเกี่ยวกับ Telemetry และจัดซื้ออุปกรณ์ Telemetry
- ศึกษาวิธีการเขียนแอพพลิเคชันบนระบบปฏิบัติการแอนดรอยด์
- ศึกษาการใช้งาน Google API

ในส่วนของโครงงาน II

- ปรับแต่งค่าและทำการจูนระบบควบคุม PID ให้สามารถต้านกระแสลม
- เขียนแอพพลิเคชันในส่วนรับภาพจากโดรน
- ปรับปรุง GUI ของแอพพลิเคชันให้ใช้งานได้สะดวก

1.4 ขั้นตอนในการดำเนินงาน

- ศึกษาและรวบรวมข้อมูลเกี่ยวกับการทำงานของโดรน
- ศึกษาและรวบรวมข้อมูลเกี่ยวกับส่วนประกอบของโดรน
- ทำการประกอบโดรนและติดตั้งอุปกรณ์ต่าง ๆ บนตัวโดรน
- ศึกษาวิธีการติดตั้งโปรแกรมและการใช้งานของโปรแกรมต่าง ๆ
- ศึกษาการทำงานของ Code ที่ใช้ในการควบคุมโดรน
- ศึกษาหลักการของ PID Controller และ Kalman Filter (P, PI และ PID)
- ทดสอบการบินของโดรนก่อนทำการปรับปรุง Code และก่อนการจูน PID เพื่อถูกความแตกต่าง
- เพิ่มเติมส่วนของ Code สำหรับแยกการทำงานของ PID สำหรับแกน Roll และแกน Pitch
- ทดสอบการบินของโดรนหลังทำการจูน PID ทั้งแบบ 1 มิติ และ 2 มิติเพื่อถูกความแตกต่าง
- ทำการทดสอบการบินในสถานการณ์ต่าง ๆ และทำการปรับแต่งตามความเหมาะสม
- ศึกษาในส่วนของการรับภาพจากโดรนผ่าน Camera (GoPro)
- ปรับปรุง interface ของแอพพลิเคชันให้ใช้งานได้สะดวก
- สรุปผลและจัดทำรายงานประกอบโครงการ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- สามารถเข้าใจหลักการทำงานพื้นฐานของโดรนได้
- สามารถสร้างโดรนที่มีประสิทธิภาพที่ดีและยังมีความเสถียรภาพสูงได้
- สามารถเขียนแอพพลิเคชันที่ในการควบคุมโดรนได้
- สามารถถ่ายภาพนิ่งมุมสูงและถ่ายทอดสดมุมสูงจากตัวโดรนได้

1.6 สถานที่ทำงาน

- ห้องพักนักศึกษาชาย มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตภูเก็ต
- ห้องปฏิบัติการฮาร์ดแวร์และห้องโครงการนักศึกษา ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตภูเก็ต

1.7 เครื่องมือที่ใช้ในการพัฒนา

Hardware

- Notebook Computer Lenovo Intel (R) Core(TM) 2 Duo T6600 @2.20 GHz, 4.00 GB of RAM
- Notebook Computer Samsung Intel (R) Atom(TM) N2600 @1.60 GHz, 2.00 GB of RAM
- Mobile Smartphone HUAWEI ASCENT MATE 7 Kirin925 @1.80GHz, 3.00 GB of RAM
- Mobile Smartphone SAMSUNG GALAXY NOTE 3 ARM Cortex A15 @1.90GHz, 3.00 GB of RAM
- Board ArduCopter APM 2.6 (Test before make project)
- Board ArduCopter APM 2.8 (Use in project)
- Board Raspberry Pi 2 (Test device)
- Board Arduino UNO (Test device)
- Board Arduino Nano (Test device)
- Remote Flysky FS-TH9X + Receiver FS-R8B (Radio)
- Telemetry Radio 433Mhz (Wireless Connected)
- Speed Controller Redcon 30A (Control motor)
- Motor Brushless 850KV (Motor)
- Servo Motor 5V (Test range radio)

Software

- Mission Planner 1.3.49 (Install firmware and PID Tuning)
- Arduino IDE (V.ArduPilot) 1.0.3 – gcc - 4.8.2 (Complier and Burning)
- Arduino IDE (V.Standard) 1.8.4 (Create Code and Library)
- Matlab 2017 (PID Tuner)
- Android Studio (Application Android)
- Virtual Studio 2016 (Camera Control)
- Eclipse (Create Code and Library for new version)

บทที่ 2 ความรู้พื้นฐาน

ในบทนี้กล่าวถึงความรู้พื้นฐานที่จำเป็นทั้งหมดสำหรับโครงงานนี้ ซึ่งจะแนะนำให้รู้จักกับหลักการทำงานของ Drone และอุปกรณ์ที่จำเป็นต่อการนำมาใช้ในการสร้าง Drone แนะนำโปรแกรม Mission Planner โปรแกรม Arduino IDE โปรแกรม Eclipse โปรแกรม Matlab โปรแกรม Android Studio โปรแกรม Virtual Studio แนะนำภาษา C++ เป้าหมาย ระบบควบคุม PID และหลักการ Kalman Filter

2.1 ความหมายของโดรน (Drone)

โดรน (Drone) หรือ ยูเอวี (Unmanned Aerial Vehicles: UAVs) เป็นเทคโนโลยีในสมัยใหม่ที่กำลังถูกจับตามองในขณะนี้ เพราะเป็นการควบคุมอากาศยานชนิดต่าง ๆ เช่น เครื่องบิน เฮลิคอปเตอร์ รวมถึงโดรนซึ่งเป็นเฮลิคอปเตอร์ชนิดหลายใบพัด โดยเป็นระบบควบคุมแบบอัตโนมัติ แล้ว โดรนยังเรียกว่า “อากาศยานไร้คนขับ” และในอนาคตอันใกล้จะมีการนำมาประยุกต์ใช้อย่างแพร่หลายครอบคลุมแทบทุกสายงาน และในสายงานอุตสาหกรรม เช่น การใช้โดรนเพื่อบันทึกภาพหรือเหตุการณ์จากมุมสูง การสำรวจพื้นที่การเกษตรและชลประทาน การสำรวจท่อส่งก๊าซ การเก็บข้อมูลสภาพอากาศ สภาพการจราจร และการลำเลียงขนส่ง เป็นต้น

โดยตามหลักการทำงานของโดรนจะใช้เพื่อเป็นตัวตรวจจับ ขนส่ง วิจัย โฉมตี ค้นหาและช่วยเหลือ โดยแบ่งการทำงานออกเป็น 3 ประเภท ได้แก่

2.1.1 Multirotor UAVs เป็นโดรนประเภทที่พบเห็นได้บ่อยที่สุด เคลื่อนตัวได้รวดเร็วและคล่องแคล่ว เนื่องจากมีทั้งแบบ 4, 6 และ 8 ใบพัด ไม่ต้องใช้รันเวย์ในการบิน แต่ขึ้นความเร็วของการบินน้อยกว่าโดรนรูปแบบอื่น ๆ ซึ่งมีชื่อเรียกแตกต่างกันไป เช่น แบบ 4 ใบพัดเรียกว่า Quad - Rotor แบบ 6 ใบพัดเรียกว่า Hexa - Rotor และแบบ 8 ใบพัดเรียกว่า Octa – Rotor

2.1.2 Fixed - wing drones เป็นโดรนที่มีลักษณะการทำงานคล้ายคลึงกับเครื่องบิน ต้องมีรันเวย์ ซึ่งโดรนประเภทนี้สามารถบินได้นานกว่าและเร็วกว่าโดรนในแบบที่แรก เมماะกับการใช้งานเพื่อสำรวจในพื้นที่กว้างใหญ่ แต่ยังสามารถบรรทุกของหนักได้ในระยะไกลและยังใช้พลังงานน้อยอีกด้วย

2.1.3 Hybrid model (tilt - wing) เป็นโดรนที่สามารถบินได้เร็วและไกลกว่า อีกทั้งยังและมีประสิทธิภาพมากกว่าโดรนในแบบที่สอง ไม่ต้องใช้รันเวย์ แต่โดรนประเภทนี้ยังมีอยู่น้อยในตลาดโลก

2.2 ข้อดี - ข้อเสียของโดรน (Drone)

2.2.1 ข้อดีของโดรน (Drone)

- รูปแบบของโดรนแบบ Quad - Rotor จะบินได้เง็งมากกว่าเฮลิคอปเตอร์แบบใบพัดเดียว เพราะว่ามีการใช้ใบพัดถึง 4 ใบ ซึ่งเป็นตัวสร้างแรงยก และในบางตัวใช้ใบพัดและใบถัง 6 - 8 ใบ จะทำให้มีแรงยกได้มากขึ้น และยังสามารถออกแบบให้ใช้ใบพัดที่มีขนาดเล็กกว่า เพื่อที่สามารถกระจายแรงยกไปยังพัดใบต่าง ๆ ได้ ทำให้เกิดแรงสั่นสะเทือนได้น้อยกว่า และใบพัดจะไม่ตีกับวัตถุอื่น ๆ จึงเหมาะสมที่จะนำมาสร้างเป็นโดรน
- กลไกการทำงานสามารถเข้าใจง่ายและยังสร้างได้ง่ายกว่า เพราะว่าถ้าเปรียบเทียบกับเฮลิคอปเตอร์ชนิดธรรมดากำจดมีใบพัดหลักชุดเดียว ซึ่งใบพัดอาจต้องมีขนาดใหญ่และตามหลักการถ้าของหนัก ๆ ทำการหมุนรีวิว ๆ จะทำให้เกิดแรงหนีศูนย์ได้ง่ายกว่า ทำให้ตัวเฮลิคอปเตอร์แบบใบพัดเดียวมีโอกาสสั่นสะเทือนได้มากกว่าโดรนที่มีหลายใบพัด
- กลไกของการทำงานใบพัดของโดรนชนิด Quad - Rotor ใช้กลไกที่มีซับช้อนไม่ยุ่งยาก และส่วนของเฮลิคอปเตอร์จะใช้วิธีการปรับ Pitch ใน การสร้างแรงยกในทิศทางต่าง ๆ ทำให้มีกลไกที่ซับซ้อนมากกว่า

2.2.2 ข้อเสียของโดรน (Drone)

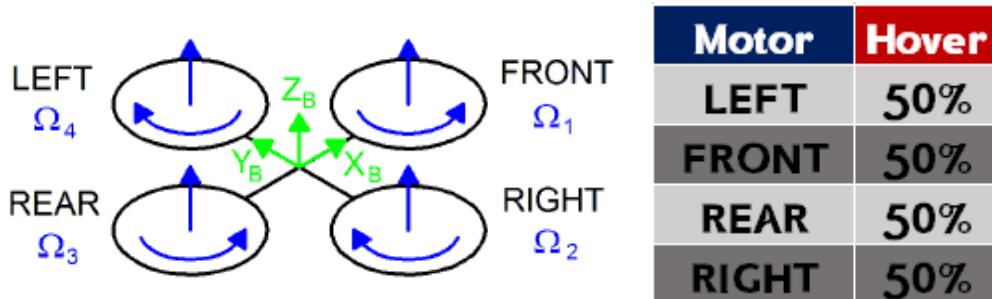
- ระบบควบคุมของโดรนชนิด Quad-Rotor จะซับซ้อนมากกว่าเฮลิคอปเตอร์ในปัจจุบันและมีระบบเซ็นเซอร์ที่ดีในการช่วยในการควบคุมตัวโดรน เช่น Gyro-meter หรือ Accelerometer
- ปัญหาเรื่องแหล่งพลังงานของโดรนแบบ Quad - Rotor ซึ่งจะมีการใช้มอเตอร์ทั้งหมดถึง 4 ตัว โดยต้องจ่ายพลังงานให้แก่มอเตอร์ทั้ง 4 ในปริมาณเท่า ๆ กันเพื่อที่จะทำให้ตัวโดรนสามารถบินได้ และในการบินหนึ่งครั้งของโดรนทำให้สามารถบินได้เพียง 10 – 15 นาทีเท่านั้น เนื่องจากมีข้อจำกัดในเรื่องแหล่งพลังงาน โดยสามารถบรรจุแหล่งพลังงานที่มีขนาดและน้ำหนักที่จำกัดบนตัวโดรน และถ้ามีน้ำหนักมากหรือขนาดใหญ่เกินไปก็ไม่สามารถนำไปติดตั้งบนตัวโดรนได้

2.3 หลักการทำงานของโดรน (Drone)

การบินของโดรน (Drone) หรือเฮลิคอปเตอร์สี่ใบพัด (Quad - Rotor) จะมีการเคลื่อนที่ 4 ทิศทาง หรือที่เรียกว่า 4 Channel ซึ่งประกอบด้วย การบินขึ้น-การบินลงแนวตั้ง การบินเดินหน้า-การบินถอยหลัง การบินเอียงซ้าย-การบินเอียงขวาและการบินหมุนซ้าย-การบินหมุนขวา

2.3.1 Hovering

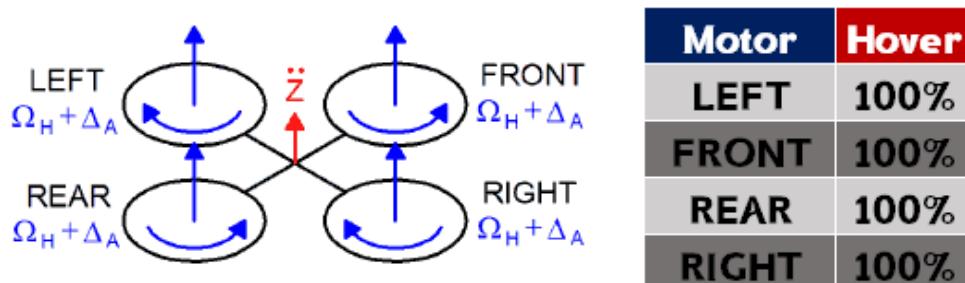
Hovering หรือการลอยตัวอยู่กับที่ สามารถทำได้โดยควบคุมให้ความเร็วของใบพัดทั้งสี่ตัวให้มีความเร็วให้เท่ากันเพื่อสร้างแรงบิด (torque) และหักล่างแรงบิด จากรูปที่ 2-1 จะเห็นว่า ใบพัดหน้า (FRONT) และใบพัดหลัง (REAR) จะหมุนตามเข็มนาฬิกา ใบพัดซ้าย (LEFT) และขวา (RIGHT) จะหมุนตามเข็มนาฬิกาจะทำให้เครื่องบินไม่หมุนตัว



รูปที่ 2 - 1 ทิศทางการหมุนของมอเตอร์แบบ Hovering

2.3.2 Throttle

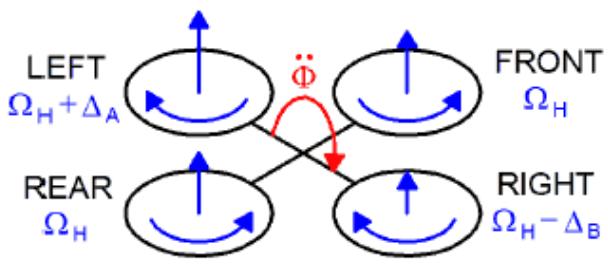
Throttle หรือการเร่งความเร็วให้โดรนสามารถบินขึ้น-สามารถบินลง จากรูปที่ 2-2 ใบพัดทั้งสี่ในจะต้องเพิ่มความเร็วของทุกใบพัดให้เท่ากัน จะทำให้โดรนสามารถลอยตัวขึ้นมาได้



รูปที่ 2 - 2 ทิศทางการหมุนของมอเตอร์แบบขึ้น-ลง

2.3.3 Roll

Roll หรือการบินเอียงตัวซ้าย-ขวา จากรูปที่ 2-3 ใบพัดหน้า (FRONT) และใบพัดหลัง (REAR) จะมีความเร็วเท่าเดิม แต่ความเร็วใบพัดซ้าย (LEFT) จะหมุนเร็วขึ้นทิศทางนี้จะยกตัวไปพัดขวา (Right) จะซ้ำกัน ทิศทางนี้จะตกลง จึงทำให้เกิดการเอียงตัวไปทางขวาได้ ส่วนเอียงตัวซ้ายก็ใช้วิธีคล้ายกัน

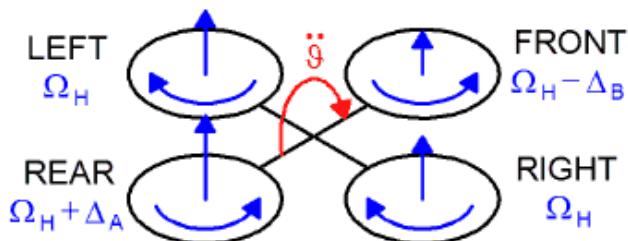


Motor	Hover
LEFT	100%
FRONT	50%
REAR	50%
RIGHT	25%

รูปที่ 2 - 3 ทิศทางการหมุนมอเตอร์แบบ Roll

2.3.4 Pitch

Pitch หรือการบินเอียงหน้าและหลัง ซึ่งจะคล้ายกับการบินแบบ Roll แต่เปลี่ยนเป็นใบพัดซ้าย (LEFT) และขวา (RIGHT) จะมีความเร็วคงที่ แต่ความเร็วใบพัดหลัง (REAR) จะหมุนเร็วขึ้นทางหลังจะยกใบพัดหน้า (FRONT) จะหมุนช้ากว่าทางหน้าจะตกล จึงทำให้เครื่องบินเอียงไปข้างหน้าดังรูปที่ 2-4

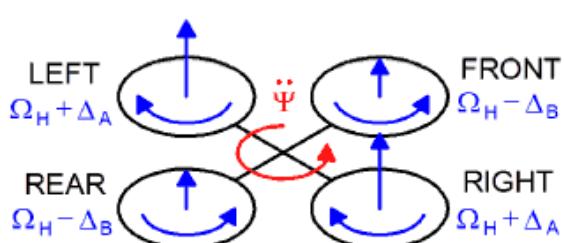


Motor	Hover
LEFT	50%
FRONT	25%
REAR	100%
RIGHT	50%

รูปที่ 2 - 4 ทิศทางการหมุนมอเตอร์แบบ Pitch

2.3.5 Yaw

Yaw หรือการบินหมุนตัว ให้ความเร็วของใบพัดหน้า (FRONT) ใบพัดหลังหลัง (REAR) ที่หมุนในทิศตามเข็มนาฬิกามากกว่าความเร็วใบพัดซ้าย (LEFT) ใบพัดขวา (RIGHT) ที่หมุนในทิศทวนเข็มนาฬิกา เพื่อให้แรงบิดด้านซ้ายมากกว่าด้านขวา จึงทำให้เครื่องบินหมุนตัวได้ดังรูปที่ 2-5



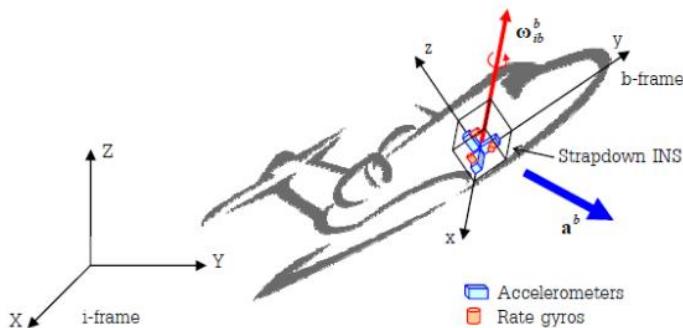
Motor	Hover
LEFT	100%
FRONT	25%
REAR	25%
RIGHT	100%

รูปที่ 2 - 5 ทิศทางการหมุนมอเตอร์แบบ Yaw

จากรูปแบบการบินแบบต่าง ๆ จะเห็นว่าการควบคุมการเคลื่อนที่นั้นจะเกิดจากการควบคุมความเร็วของมอเตอร์ทั้งสี่ตัว จะต้องทำการวิเคราะห์เพื่อที่จะทำให้มอเตอร์ทำงานได้อย่างถูกต้องและถ้ามอเตอร์ทำงานไม่ถูกต้อง จะเกิดการเคลื่อนที่แบบต่าง ๆ ทั้งระบบควบคุมหรือการใช้ sensor วัดความเอียงได้ไม่ดี เท่าที่ควรก็มีโอกาสทำให้โดรนหรือ Quad-rotor ตกหรือเสียหายได้

2.4 การแสดงมุมเอียงและการเปลี่ยนแกนอ้างอิง

Dead Reckoning คือหลักการทำงานที่นำแนวโน้มการเคลื่อนที่ไปทางทิศทางและระยะทางของการเคลื่อนที่ เช่น ถ้ารู้ว่าวัตถุ (Body) กำลังเคลื่อนที่ไปทางทิศเหนือด้วยความเร็ว 80 Km/hour หลังจากเวลาผ่านไปครึ่งชั่วโมงตำแหน่งของวัตถุคือทางทิศเหนือของจุดเริ่มต้นเป็นระยะ 40 Km



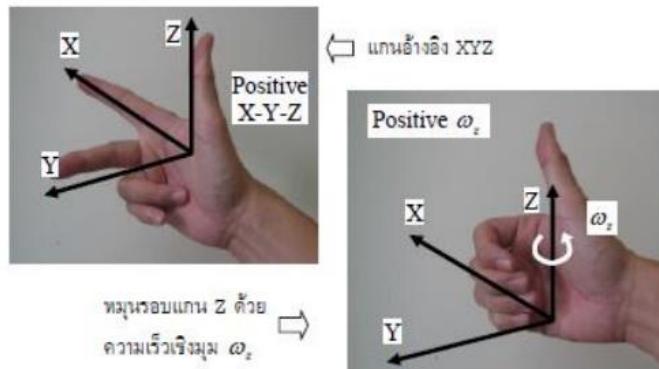
รูปที่ 2 - 6 การทำงานของ Strapdown INS operating concept

จากรูปที่ 2-6 แสดงหลักการทำงานของ Strapdown Inertial Navigation System (INS) ในการหาตำแหน่งของเครื่องบินใน Strapdown INS ประกอบด้วย Accelerometers และ Rate Gyros ซึ่งเป็นเซ็นเซอร์ใช้วัดความเร่งเชิงเส้น a^b และความเร็วเชิงมุม ω_{ib}^b ตามลำดับโดยหลักการแล้ว Strapdown INS นำข้อมูลความเร็วเชิงมุมที่ Rate Gyros วัดได้มาคำนวณหาทิศทางของการเคลื่อนที่ เพื่อใช้ประกอบกับข้อมูลความเร่งเชิงเส้นที่ Accelerometers วัดได้จากการคำนวณหาความเร็วและระยะทางของการเคลื่อนที่ โดยอาศัยหลักการ Dead Reckoning

นอกจากระบบนำร่องแบบ Strapdown INS มาใช้แล้ว การแสดงว่าวัตถุทำมุมเอียงอย่างไรกับแกนอ้างอิง(Orientation Representation) และการเปลี่ยนแกนอ้างอิงของเวกเตอร์ (Coordinate transformation) เป็นความรู้พื้นฐานที่สำคัญที่ถูกใช้งานในอีกหลายสาขา เช่น ในการแสดงภาพสามมิติ (Computer graphics) หรือในการวิเคราะห์ระบบที่ประกอบด้วยจุดเชื่อมต่อหลายจุด เช่น แขนกล เป็นต้น

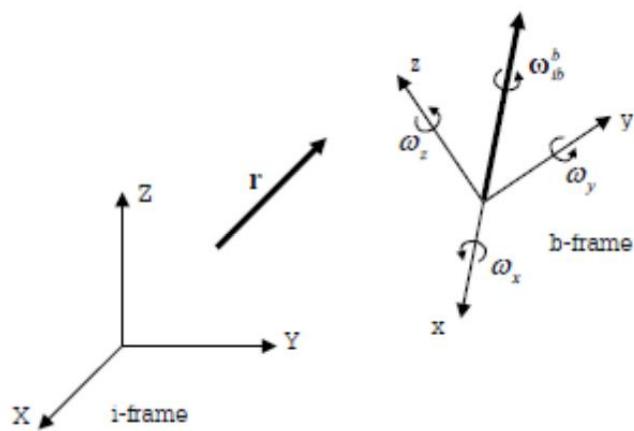
transformation ที่เป็นที่ยอมรับใช้งานอย่างกว้างขวาง 3 วิธีได้แก่ Direction Cosine Matrix ,Euler Angle และ Quaternion พร้อมทั้งเปรียบเทียบข้อดีข้อเสียของแต่ละวิธี

การวิเคราะห์เกี่ยวกับ Orientation และ Coordinate transformation โดยทั่วไปใช้สมมุติฐานของ Cartesian Coordinate (แกน X-Y-Z ตั้งฉากซึ่งกันและกัน) และกำหนดเครื่องหมาย (Sign Convention) ตาม Right Hand Rule กล่าวคือให้ XYZ เป็นแกนอ้างอิง (Reference Frame) แบบ Cartesian ทิศบวกของแกน X-Y-Z จะซึ่งเป็นทิศทางของนิ้วซ้าย นิ้วกางและนิ้วโป้งของมือขวาตามลำดับ เมื่อใช้มือขวาทำการรอบแกนที่หมุนโดยให้นิ้วโป้งซี้ไปในทิศทางของแกนที่หมุนนั้น การนิ้วโป้งซี้ไปในทิศทางของแกนที่หมุนนั้นการหมุนมีค่าเป็นบวกเมื่อหมุนตามทิศของนิ้วที่เหลือทั้ง 4 ดังแสดงในภาพที่ 2-7 หลักการ Assumptions และ Sign Convention



รูปที่ 2 - 7 หลักการ Assumptions และ Sign Convention

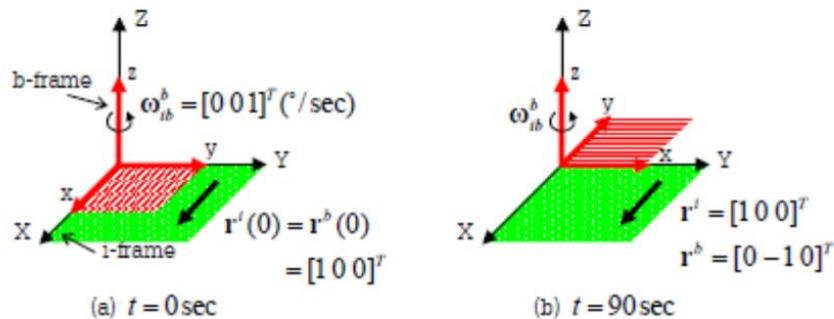
การวิเคราะห์ปัญหาของ Orientation และ Coordinate Transformation สามารถพิจารณาได้ดังนี้ กำหนดให้มีแกนอ้างอิง 2 แกน คือ I-frame (XYZ) และ b-frame (xyz) ดังแสดงในรูปที่ 2.4 โดยที่ I-frame หยุดนิ่งไม่เคลื่อนที่ (Inertia), b-frame หมุนด้วยความเร็วเชิงมุม $\omega_{ib}^b = [\omega_x \ \omega_y \ \omega_z]^T$ (subscript Ib ใช้ระบุว่าเป็นความเร็วเชิงมุมของ b-frame สัมพัทธ์กับ I-frame, superscript b ใช้ระบุว่าเป็นเวกเตอร์ที่เขียนบรรยายใน b-frame และ superscript T หมายถึง Transpose) ดังนั้นเวกเตอร์ ω_{ib}^b มีขนาดขององค์ประกอบในแกน x, y และ z คือตามลำดับให้ r คือเวกเตอร์อิสระใด ๆ ปัญหาของ Coordinate Transformation คือ จะสามารถเขียนบรรยายเวกเตอร์อิสระ r ใน I-frame (r^i) และ b-frame (r^b) ได้



รูปที่ 2 - 8 Rotating Reference Frame

รูปที่ 2-8 แสดงตัวอย่างการเปลี่ยนแกนอ้างอิง โดยกำหนดให้ตอนเริ่มต้น ($t = 0$) i-frame และ b-frame ซ้อนทับกันโดยที่แกน X ทับแกน x แกน Y ทับแกน y และแกน Z ทับแกน z พอดี ให้ r เป็นเวกเตอร์ อิสระมีขนาด 1 หน่วยในทิศบวกของ X เราสามารถเขียน r ใน i-frame และ b-frame ณ เวลา $t = 0$ จะได้ $r(0) = r^b(0) = [1 \ 0 \ 0]^T$ ดังแสดงในรูปที่ 2-9 (a) กำหนดให้ b-frame หมุนด้วยความเร็วเชิงมุม 1 องศาต่อวินาที รอบแกน z ($\omega_{ib}^b = [0 \ 0 \ 1]^T$)

ดังนั้นที่เวลา $t = 90\text{sec}$ b-frame หมุนไปได้ 90 องศา ดังแสดงในภาพที่ 2-9 (b) ทำให้สามารถเขียน เวกเตอร์ r ใน b-frame ได้ใหม่เป็น $r^b(90) = [0 \ -1 \ 0]^T$



รูปที่ 2 - 9 Coordinate Transformation with Rotating Reference Frame

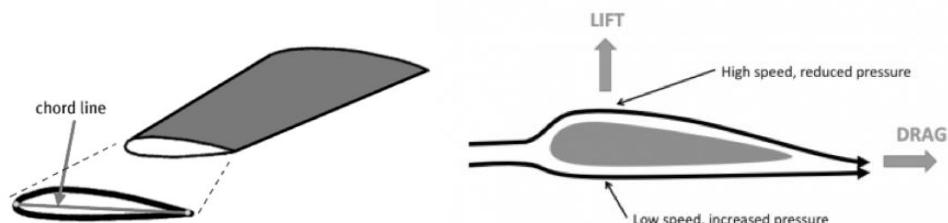
2.5 ชิ้นส่วนของโดรน

2.5.1 ใบพัด

ใบพัด เป็นอุปกรณ์ที่ทำให้เกิดมวลลมใต้ปีกพอใบพัดเริ่มหมุนเร็วขึ้นเรื่อยๆ จนสร้างมวลลม 'ได้มากพอลมใต้ปีกนี้ก็จะเป็นแรงยกทำให้โดรนสามารถบินขึ้นได้และในทางกลับกันเมื่อทำการลดความเร็วใบพัดลงใบพัดเริ่มที่จะหมุนช้าลงมวลลมใต้ปีกจะมีน้อยลงโดรนก็จะค่อยๆ บินลดระดับลงจนกระทั่งลงจอด

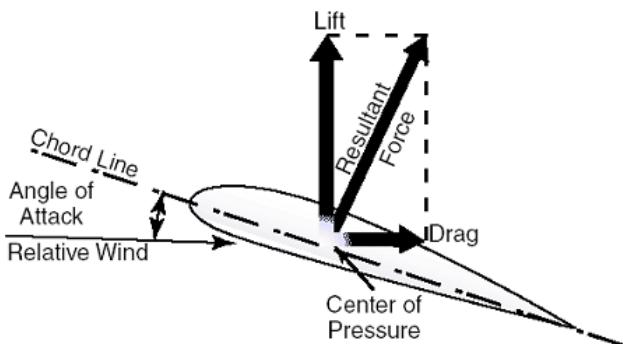
แม้ว่าการทำงานของใบพัดจะเป็นเรื่องง่ายอีกทั้งยังเป็นอุปกรณ์ชั้นเล็กๆ ไม่มี并发症จริงที่ใช้ควบคุมอีก แต่รายละเอียดของใบพัดสำหรับโดรนนั้นมีความแตกต่างกันได้ในหลาย ๆ จุด เช่น รูปทรง (Shape) หรือเส้นความยาวคอร์ด (Chord Line) ที่เป็นเส้นตรงซึ่งวัดจากด้านหนึ่งไปยังอีกด้านหนึ่งของภาคตัดขวางใบพัด เป็นต้น

ปัจจุบันใบพัดมีให้เลือกนำมาใช้มากมาย ซึ่งแต่ละแบบก็จะถูกดีไซน์มาเพื่อวัตถุประสงค์ในการบินที่แตกต่างกันออกໄไปโดยสิ่งที่ทำให้เราแยกใบพัดได้ชัดเจนที่สุดก็คงจะเป็นรูปทรงของ Airfoil และยังเป็นตัวการที่ทำให้เกิดแรงยกพาตัวลำโดรนบินขึ้นไปทำหน้าที่เหมือนกับปีกของเครื่องบินโดยรูปทรงของ Airfoil ที่แนะนำกันส่วนใหญ่คือคราวให้ Front Edge หรือ Leading Edge ที่มีความหนาแล้วค่อยๆ เรียวเล็กลงไปจนถึงหาง (Trailing Edge) เพราะแบบนี้จะมีแรงต้าน (Drag) น้อยและเกิดแรงยก (Lift) ได้ดีแสดงดังรูปที่ 2-10



รูปที่ 2 - 10 รูปทรงใบพัด Chord line และ Trailing Edge

ส่วนความยาวคอร์ดของใบพัดจะมีผลต่อประสิทธิภาพการบินโดยสัมพันธ์กับมุมปะทะหรือที่ในวงการนักบินเรียกว่า Angle of Attack หรือ AOA ที่โดยพื้นฐานแล้วจะเป็นตัวกำหนดแรงยกและแรงต้านในขณะที่ใบพัดหมุนซึ่งแรงยกนั้นจะเปลี่ยนแปลงไปตาม AOA คือยิ่งเป็นใบพัดที่มี AOA มากแรงยกก็จะมากแต่ถ้ามีมากเกินไป หมายถึงใบพัดมีความโค้งจนสร้างแรงยกได้มากไปได้จนถึงจุดวิกฤต (Critical angle of attack) ก็จะทำให้เกิดการสูญเสียแรงยกหรือที่เรียกว่า stall จนทำให้โดรนเกิดการร่วงหล่นได้ ดังนั้นใบพัดที่ดีจึงต้องผ่านการทดสอบให้มีแรงยกที่สัมพันธ์กับ AOA อย่างเหมาะสมแล้วแสดงดังรูปที่ 2-11



รูปที่ 2 - 11 Angle of Attack

โดยใบพัดจะมีหลากหลายประเภทที่นำไปใช้งาน และใบพัดที่นำมาใช้ในการประกอบโดรนจะเป็นใบพัด Airfoil โดยใบพัดจะเป็นตัวบวกขนาดความยาว X การเปิดองศาของใบพัดจะใช้ทั้งแบบหมุนตามเข็มนาฬิกาและวนเข็มนาฬิกาวัดดูจะมีให้เลือกใช้หลากหลาย เช่น พลาสติกธรรมชาติ พลาสติก ABS กึ่งคาร์บอนไฟเบอร์และคาร์บอนไฟเบอร์แท้ ซึ่งแสดงดังรูปที่ 2-12



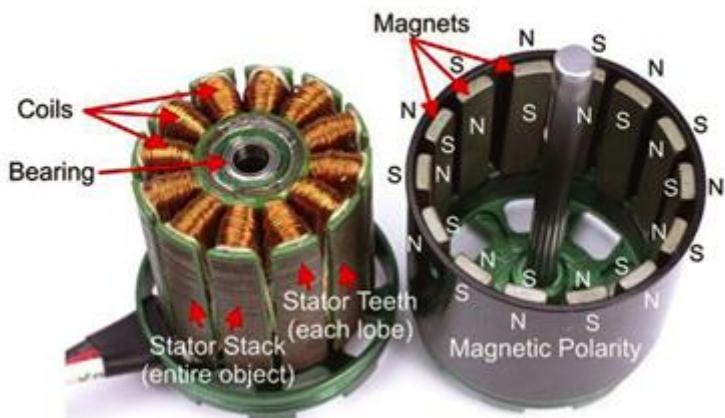
รูปที่ 2 - 12 ใบพัดขนาด 10 X 4.5

2.5.2 Motor

มอเตอร์ไฟฟ้ากระแสตรงชนิดไร้แปรงถ่าน (Brushless DC หรือ BLDC) เป็นมอเตอร์ไฟฟ้ากระแสสลับแบบ synchronous (AC Synchronous) แบบสามเฟสหรือจะเป็นแบบ permanent magnet synchronous motor (PMSM) ซึ่งต้องใช้ไฟสามเฟสในการขับ ซึ่งในอดีตเราจะเห็นการใช้งานมอเตอร์ AC Synchronous นี้ในโรงงานอุตสาหกรรมเนื่องจากมีแหล่งจ่ายไฟสามเฟสมอเตอร์เหล่านี้จะหมุนด้วยความเร็วที่เป็นจำนวนเท่าของความถี่ของไฟสามเฟสที่จ่ายและข้อจำกัดในการปรับความเร็วรอบจึงทำให้มอเตอร์ประเภทนี้ไม่ถูกใช้งานมากนักในอุปกรณ์เครื่องใช้ไฟฟ้าทั่ว ๆ ไปในครัวเรือนด้วยเทคโนโลยีปัจจุบันเราสามารถควบคุมความเร็วรอบที่ต่ำต่าง ๆ ซึ่งทำงานโดยการรับแรงดันไฟฟ้ากระแสตรงมาสร้างเป็นแรงดันไฟฟ้ากระแสสลับสามเฟสจ่ายให้กับมอเตอร์ ซึ่ง DC มอเตอร์ก็ใช้วิธีสลับทิศทางการจ่ายกระแส (commute) จากแหล่งจ่ายไฟกระแสตรงจ่ายให้ขาด漉ดผ่าน Commutator ซึ่งใช้แปรถ่ายในการทำหน้าที่นี้แต่ในกรณีของ BLDC จะออกแบบให้ขาด漉ดอยู่นิ่ง (stator) และมีแม่เหล็กถาวรหมุนได้แล้วใช้วิธีการสลับ

ทิศทางกระแสจ่ายให้ขาด漉ดด้วยวงจรไฟฟ้าแทนการใช้แปรงถ่าน จึงเรียกชื่อว่า “มอเตอร์ไฟฟ้ากระแสตรง ชนิดไร้แปรงถ่าน”

Motor Brushless คือ เครื่องจักรกลไฟฟ้าแบบไม่ใช้แปรงถ่านในการส่งผ่านกระแสไฟฟ้าเข้าหนึ่หรือออกขาด漉ดไฟฟ้าที่ตัวที่เคลื่อนที่ซึ่งจะรวมไปถึงการเปลี่ยนจากขาด漉ดไฟฟ้าไปใช้เป็นแม่เหล็กถาวร ตัวอย่างของเครื่องจักรกลไฟฟ้าแบบ Brushless ก็ได้แก่ พัดลมระบบความร้อนคอมพิวเตอร์ (Brushless DC) เครื่องกำเนิดไฟฟ้าและมอเตอร์ประเภทต่าง ๆ จากรูปที่ 2-13 เป็นตัวอย่างมอเตอร์



รูปที่ 2 - 13 องค์ประกอบของมอเตอร์บลสเลส

ข้อมูลทางเทคนิคของมอเตอร์บลสเลส

- ฐานมอเตอร์ได้ajan 22.0 mm
- ฐานมอเตอร์บนajan 14.0 mm
- กำลังของมอเตอร์ (KV) 850KV-RPM / Volt =>9435 รอบต่อนาที
- Idle Current 0.03 A
- ขนาด ESC ที่รองรับ 30-40 A
- ขนาดเซลล์แบตเตอรี่ 3 Cell
- Propeller 10 x 4.5
- น้ำหนัก 52 g
- ฐานมอเตอร์รอบนอก 27.8 mm



รูปที่ 2 - 14 Motor Brushless REDCON (2214/850KV)

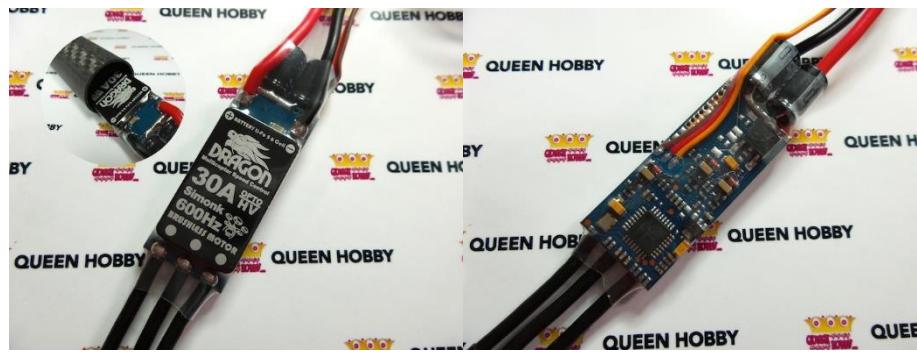
2.5.3 Electronic Speed Controller (ESC)

Electronics Speed Controller หรือ ESC คือ อุปกรณ์อิเล็กทรอนิกที่ใช้ในการส่งคำสั่งเพื่อควบคุมมอเตอร์ให้สามารถเร่งประสีทิชิภาพของมอเตอร์ได้อย่างเต็มกำลัง มักมีสายไฟออกมาทั้งสองด้านด้านหนึ่งจะใช้จ่ายกระแสไฟฟ้าให้แก่มอเตอร์ส่วน ส่วนอีกด้านหนึ่งก็จะมีสายไฟออกมาอีก 2 เส้นเพื่อรับกระแสไฟจากแบตเตอรี่และในด้านที่มีสายไฟนี้ก็จะมีสายไฟเส้นเล็กพร้อมแจ็คอีก 1 เส้นเพื่อเสียบเข้ากับเครื่องรับวิทยุหรือรีซีฟเวอร์ เพื่อรับคำสั่งจากเครื่องส่งวิทยุผ่านรีซีฟเวอร์ เมื่อเรากดที่เครื่องส่งวิทยุเครื่องรับวิทยุจะรับคำสั่งจากเครื่องโดยส่งผ่านสปีดคอนโทรลเลอร์ เพื่อจ่ายกระแสมากน้อยให้มอเตอร์เร่งหรือเดินเบาได้ตามความต้องการ

DRAGON ESC 30A ถูกออกแบบให้มีโปรแกรมต่าง ๆ ใช้งานตามแต่ชนิดของมอเตอร์และช่วยในเรื่องการปรับแต่งและให้ speed ส่งกำลังไฟฟ้าให้กับมอเตอร์หมุนเต็มที่และส่งผลให้มอเตอร์ทำงานได้สูงสุด 100% โดยไม่สูญเสียกำลังสามารถใช้กับมอเตอร์ที่มีค่าต้านทานต่ำ ๆ ได้มีระบบป้องกัน over-load มีตัดการทำงานของมอเตอร์เมื่อสัญญาณวิทยุหายไป มีระบบป้องกันมอเตอร์ล็อกใช้กับ multi-copter ได้โดยตรงและรองรับแบตเตอรี่ได้ตั้งแต่ 2- 6 เซลล์ จากที่ 2-15 เป็นตัวอย่าง Speed Controller

ข้อมูลทางเทคนิคของ ESC

- สามารถใช้กับมอเตอร์ที่มีค่า R ต่ำ ๆ ได้
- มีระบบป้องกัน over-load
- มีระบบตัดการทำงานของมอเตอร์เมื่อสัญญาณวิทยุหายไป
- รองรับมอเตอร์รอบสูง ๆ เนื่องจากใช้ CPU ที่เร็วกว่าสปีดทัวร์ไป
- มีระบบป้องกันมอเตอร์ล็อก
- มีโปรแกรมพิเศษสำหรับใช้กับ MultiCopter ได้โดยตรง
- ไม่ต้องโปรแกรมใด ๆ เพียงแค่ Calibrate แล้วสามารถใช้งานได้ทันที
- รองรับแบตเตอรี่ได้ตั้งแต่ 2- 6 เซลล์ Li-Po



รูปที่ 2 - 15 DRAGON ESC 30A

2.5.4 ลำตัวเฟรม

ลำตัวเฟรม เป็นเฟรมขนาด 450 mm วัสดุเป็นไฟเบอร์กลาส ที่ตัวเฟรมและวัสดุเป็น polymide - nylon สามารถประกอบได้ง่ายมีเท่นจับที่ใหญ่สามารถรองรับการติดกล้องหรือ อุปกรณ์เสริมอื่น ๆ ได้มีแรงงdrag ทำให้สามารถต่ออุปกรณ์ต่าง ๆ ได้โดยไม่ต้องเดินสายไฟมาก มีขา 2 สี เพื่อให้ง่ายต่อการแยกและขณะทำการบิน จากรูปที่ 2-16 เป็นตัวอย่างลำตัวเฟรม



รูปที่ 2 - 16 ลำตัวเฟรมขนาด 450 mm.

ข้อมูลทางเทคนิค

- กว้าง : 450mm
- สูง : 55mm
- น้ำหนัก : 270g (w/out electronics)
- ขนาดมอเตอร์ที่สามารถติดตั้งได้: 16/19mm

2.5.5 Receiver Flysky FS-R9B

ตัวรับสัญญาณวิทยุ หรือ Receiver สามารถรับสัญญาณได้ 1-8 ช่องสัญญาณ โดยใช้คู่กับรีโมท Flysky FS-TH9X จากรูปที่ 2-17 เป็นตัวอย่าง Receiver Flysky FS-R9B



รูปที่ 2 - 17 Receiver Flysky FS-R9B

ข้อมูลทางเทคนิค

- ความต้องการจากแหล่งจ่ายไฟ : 4.5-6.5V
- ขนาด : 2.04 x 1.37 x 0.59" (52 x 35 x 15mm)
- น้ำหนัก : 0.635oz (18g)
- ช่องสัญญาณ : 8channels
- ความไวต่อการรับสัญญาณ : -105dBm
- ย่านความถี่ : 2.4-2.48GHz
- ความกว้างของความถี่ : 500KHz
- ช่วงคลื่น : 160
- มาตรฐาน : GFSK(Gaussian Frequency Shift-Keying)
- โหมดการทำงาน : Automatic Frequency Hopping Digital System
- ความยาวเสาสัญญาณ : 26mm.

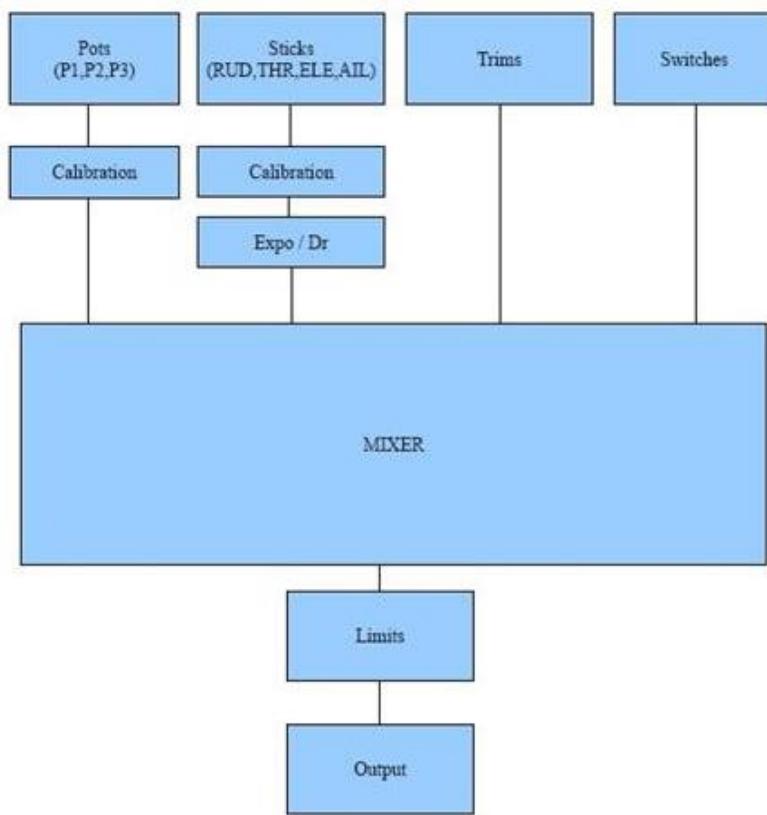
2.5.6 รีโมท Flysky FS-TH9X



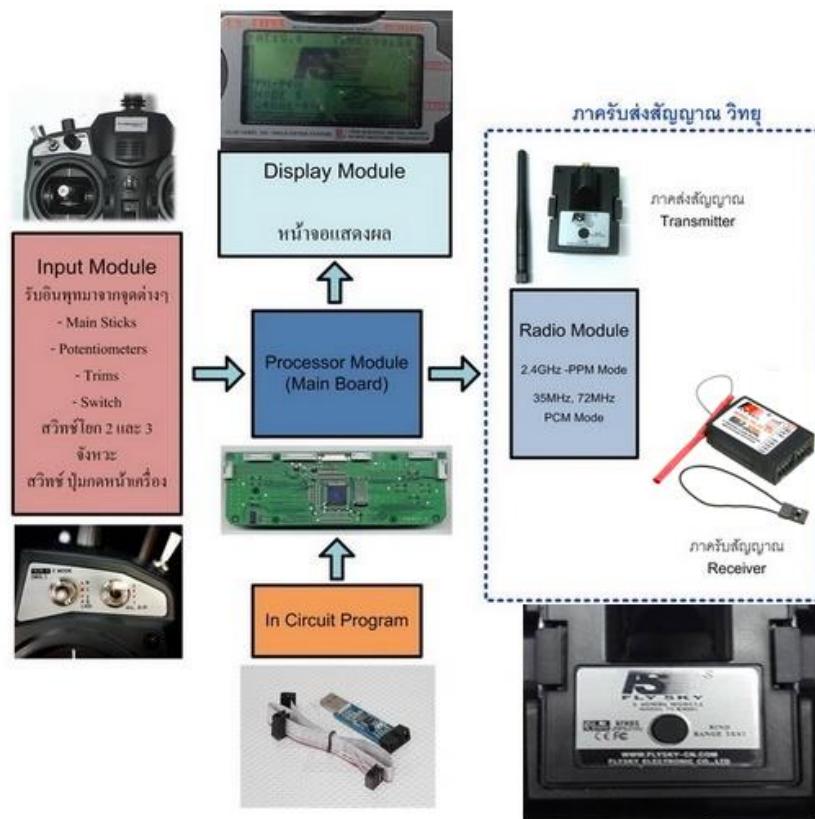
รูปที่ 2 - 18 Remote Flysky FS-TH9X

จากรูปที่ 2-18 เป็นรีโมทวิทยุที่ใช้คลื่นความถี่ 2.4 GHz โดยใช้ระบบ AFHDS รองรับการส่งสัญญาณ 9 ช่องสามารถใช้กับตัว receiver สามารถตั้งโปรแกรมการทำงานในรูปแบบต่าง ๆ ได้มีหน้าจอควบคุมแบบกราฟิกเมนูสามารถเลือกควบคุมอินพุต-เอาต์พุตได้อย่างอิสระและสามารถถอดโ模式ลภาคส่งสัญญาณวิทยุได้ทำงานได้ทั้งหมด Pulse Position Modulation (PPM) และ Pulse Code Modulation (PCM) บันทึกเก็บข้อมูลเครื่องบินได้การออกแบบในลักษณะ Module ที่สามารถถอดเปลี่ยนได้มีอัตราค่าถูกที่สำคัญ คือ Firmware หรือโปรแกรมที่ทำให้เครื่องวิทยุนี้ทำงานได้ เป็นประเภท Open Source สามารถนำมาดัดแปลงแก้ไขหาข้อบกพร่องในตัวได้

โครงสร้างของวิทยุรุ่มนี้ออกแบบมาด้วยหลักการที่มีภาคการทำงานต่าง ๆ แยกออกจากกันเป็น Module โดยมีหัวใจหลักสำคัญคือ Processor ควบคุมการทำงานหลักอยู่ 1 ตัวหรือมี Main Board และ CPU คล้าย ๆ คอมพิวเตอร์โดยมีโปรแกรมหรือ Firmware เป็นตัวสั่งให้ Main Board ทำงานโดยที่ Main Board นี้จะมีส่วนเชื่อมต่อกับ Module ต่าง ๆ ในตัวเครื่อง เช่น อินพุตที่รับเข้ามา ไฟเลี้ยงของวงจร ส่วนการแสดงผลจกราฟฟิกและเอาต์พุตส่งที่ออกไป



รูปที่ 2 - 19 โครงสร้างภายในของวิทยุ Flysky FS-TH9X



รูปที่ 2 - 20 โครงสร้างการทำงานหลักของวิทยุ Flysky FS-TH9X

ภาคอินพุตที่รับเข้ามาประกอบไปด้วย 4 ส่วนหลัก ได้แก่

1. Main Sticks (สติ๊กโยกหลัก 4 ตัว – Rudder, Throttle, Elevator และ Aileron)
2. Potentiometers (อินพุตแบบปรับค่าได้ แบบ Volume คือ P1 ,P2 ,P3)
3. Trims (trim อินพุตจาก RudderTrims, ThrottleTrims, ElevatorTrims และ AileronTrims)
4. Switch สวิตช์คันโยก 2-3 จังหวะ และสวิตช์ปุ่มกดหน้าเครื่อง

อินพุตที่เข้ามาทั้งหมดจะถูกส่งเข้ามาประมาณผลสถานะของมัน โดยการตรวจสอบเงื่อนไขต่าง ๆ รวมถึงโปรแกรมควบคุมส่วนผสม MIXER แล้วทำการเข้ารหัสส่งไปควบคุมภาคเอาต์พุตต่อไป

ข้อมูลทางเทคนิค

- ช่องสัญญาณ : 9 channels
- ชนิดการควบคุม : helicopter ,airplane, glider
- กำลังในการส่งสัญญาณวิทยุ: less than 20dB
- มาตรฐาน : GFSK
- ชนิดการถอดรหัส : PPM/PCM
- ชนิดการแสดงผล : 128*64 dot
- ระบบเตือนพลังงานต่ำ : have
- แหล่งพลังงาน : 12V DC (1.5 AA X 8)
- น้ำหนัก : 680g
- ความยาวเสาอากาศ : 26mm
- ขนาด : 190 X 80 X 240mm

การมอดูเลตแบบ (Gaussian Frequency Shift Keying, GFSK)

การมอดูเลตแบบ GFSK จะมีลักษณะการมอดูเลตเหมือนกับการมอดูเลตแบบ FSK (Frequency Shift Keying) แต่จะมีการปรับปรุงพัลส์ที่เข้ามาเรียกว่า Gaussian Filter เพื่อลดการสิ้นเปลืองแบนวิธจึงสามารถให้ถ่ายโอนข้อมูลได้เร็วขึ้นโดยที่ถ้าสัญญาณดิจิตอลมีค่าเป็น “1” ก็จะถูกส่งด้วยพัลส์ที่มีค่าความถี่ของคลื่น파หะที่ค่าความถี่ค่านึงและถ้าสัญญาณดิจิตอลมีค่าเป็น “0” ก็จะถูกส่งด้วยพัลส์ที่มีค่าความถี่ของคลื่น파หะที่ค่าความถี่อีกค่านึง

โหมดการทำงาน (Mode): Automatic Frequency Hopping Digital System (AFHDS)

Automatic Frequency Hopping Digital System (AFHDS) เป็นการใช้หลักการกระโดดแบบอัตโนมัติของความถี่ในระบบดิจิตอล เป็นระบบที่ถูกพัฒนาขึ้นเพื่อป้องกันการรบกวนความสามารถในการใช้พลังงานที่ต่ำมากและมีความไวต่อการรับสัญญาณสูงเป็นมาตรฐานใหม่ของระบบวิทยุในช่วงความถี่ 2.4 Ghz ของ Flysky Radio

Pulse Position Modulation (PPM)

Pulse Position Modulation (PPM) จะใช้การจัดข้อมูลโดยใช้ระยะเวลาความกว้างของลูกคลื่น

Pulse Code Modulation (PCM)

Pulse Code Modulation (PCM) เป็นรูปแบบการเข้ารหัสของสัญญาณโดยใช้การจัดข้อมูลในรูปแบบดิจิตอลที่เป็นเลขฐานสอง ซึ่งใช้การจัดข้อมูลโดยใช้ระยะเวลาความกว้างของลูกคลื่น

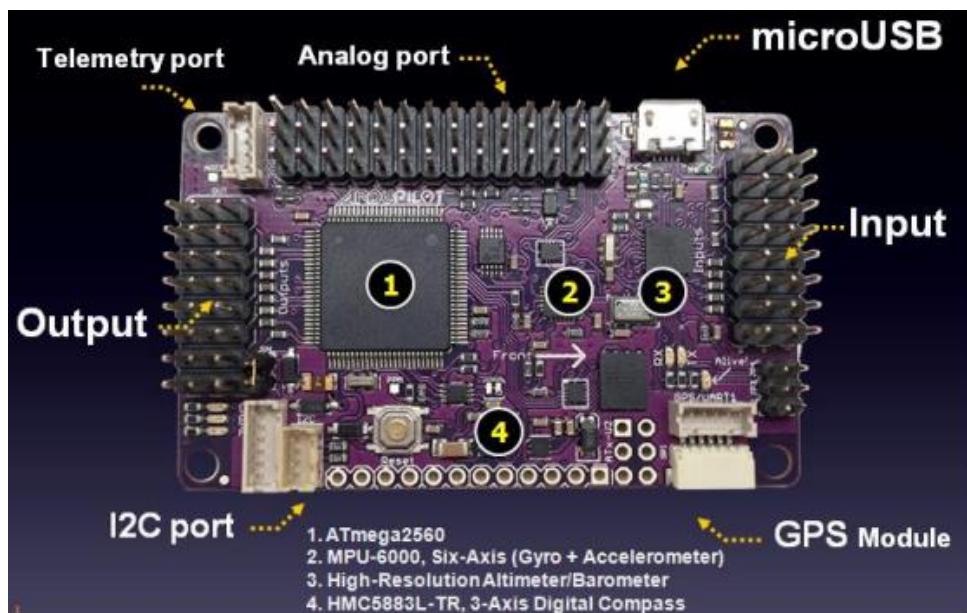
ดังนั้น PPM และ PCM เป็นรูปแบบการเข้ารหัสสัญญาณการควบคุมไปกับคลื่นความถี่วิทยุ โดย PPM นั้นจะระบุตำแหน่งการควบคุมจากความยาวคลื่น ส่วน PCM นั้นจะใช้การตีความหมายอ กมาให้อยู่ในรูปของสัญญาณ Digital ซึ่งทำให้การส่งสัญญาณแบบ PCM เมื่อมีสัญญาณที่มีความแรงเพียงพอความถูกต้องของสัญญาณจะมากกว่า ไม่มีการแก่งของสัญญาณซึ่งทำให้การควบคุมเกิดอาการสั่น ๆ ไปมา นอกจากนี้ PCM ยังมีการตรวจความถูกต้องของสัญญาณในระดับหนึ่งทำให้มีสัญญาณที่ไม่ถูกต้องเข้ามา จะทำการเพิกเฉยต่อสัญญาณที่เข้ามาในชุดนั้นทำให้มีสัญญาณอ่อนหรือถูกกรอง จะไม่มีคำสั่งในการควบคุม

2.5.7 บอร์ด APM 2.8 (ArduPilotMega version 2.8)

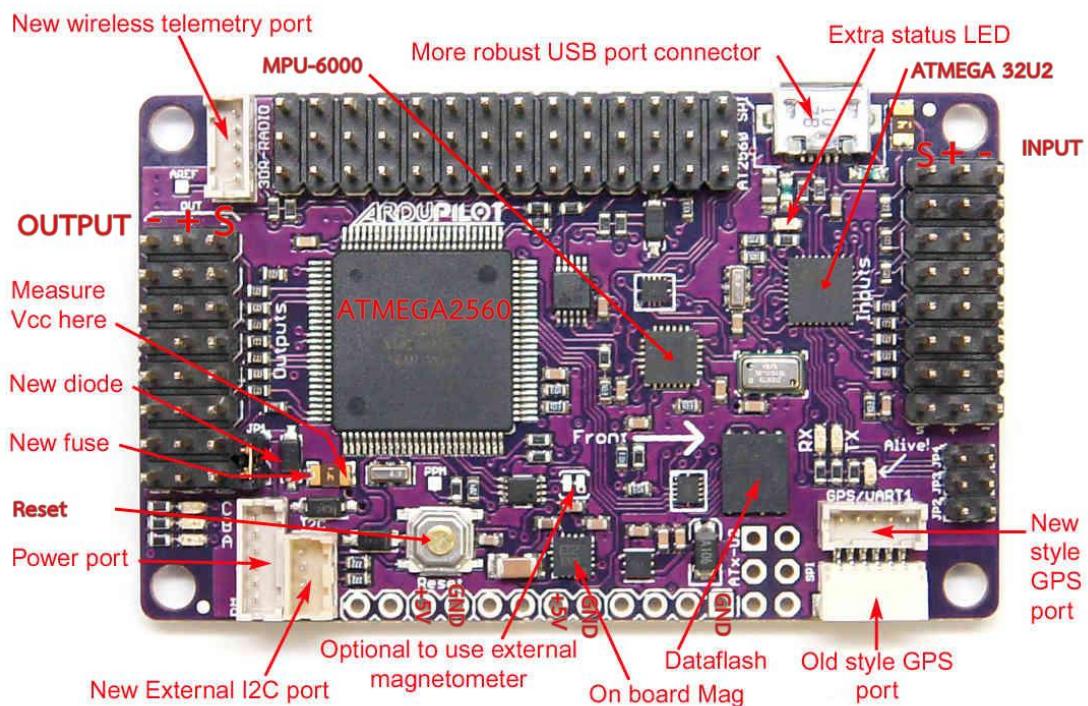
APM เป็นบอร์ดสำเร็จรูปที่ไม่ต้องประกอบและมี firmware ลงมาภายในตัวบอร์ดเรียบร้อย ซึ่งแสดงในรูปที่ 2-21 เป็นตัวอย่างบอร์ด APM 2.8 (ArduPilotMega 2.8)



รูปที่ 2 - 21 Board APM 2.8



รูปที่ 2 - 22 แผนภาพแสดง Sensor ที่มีภายในบอร์ด APM



รูปที่ 2 - 23 แผนภาพแสดง IC ภายในบอร์ด APM 2.8

จากรูปที่ 2-23 พบร่วมกับภายในเป็น Microcontroller AVR ATMEGA 2560 และประกอบด้วย Sensor คือ 3-AXIS Gyrometer, 3-AXIS Accelerometer, Barometer

ข้อมูลทางเทคนิคของ AVR ATMEGA 2560

- Parameter Value
- หน่วยความจำ Flash (Kbytes) : 256 Kbytes
- ขา Pin ที่นับได้ : 100
- Max. Operating Freq. (MHz) : 16 MHz
- CPU : 8-bit AVR
- Hardware QTouch Acquisition : No
- I/O Pins สูงสุด : 86
- Ext Interrupts : 32
- SPI : 5
- หน่วยความจำ SRAM (Kbytes) : 8
- หน่วยความจำ EEPROM (Bytes) : 4096
- Output Compare Channel : 16
- Input Capture Channel : 4
- ช่องสัญญาณ PWM : 15

MPU-6000 เป็น sensor 3-AXIS gyro และ 3-Axis Accelerometer มีช่วงการทำงานที่ $\pm 250, \pm 500, \pm 1000$, and $\pm 2000^{\circ}/\text{sec}$ (dps)

ข้อมูลทางเทคนิคของ MS5611 เป็นเซนเซอร์บารอ米เตอร์ใช้สำหรับวัดความสูง

- ไม่ต้องมีความละเอียดสูง, 10cm
- กินกำลังไฟต่ำ, 1 μA (standby < 0.15 μA)
- เชื่อมโยงความดันแบบดิจิตอลในตัว (24 bit $\Delta\Sigma$ ADC)
- I²C และ SPI interface สูงถึง 20 MHz
- ไม่มีส่วนประกอบภายนอก (มีวงจร oscillator ภายใน)

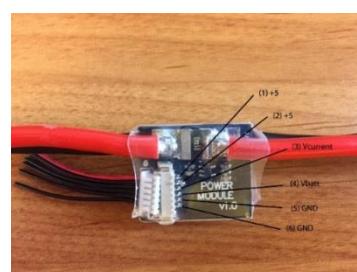
2.5.8 Battery Li-Po 3-Cell



รูปที่ 2 - 24 Li-Po Battery

จากรูปที่ 2-24 เป็น Battery 3 Cell ชนิด Lithium Polymer ขนาดความจุ 2200mA ขนาดแรงดัน 11.1 V ซึ่งภายในประกอบด้วยก้อนถ่านจำนวน 3 ก้อน (3เซลล์)

2.5.9 Power Module



รูปที่ 2 - 25 Power Module

จากรูปที่ 2-25 เป็นโมดูลที่ใช้สำหรับแปลงแรงดันจาก 12 โวลต์ เป็น 5 โวลต์เพื่อจ่ายให้แก่บอร์ด APM 2.8 และตัว Receiver

2.5.10 GPS Module



รูปที่ 2 - 26 GPS Module

จากรูปที่ 2-26 เป็น GPS Module อุปกรณ์สำหรับระบุตำแหน่งของตัวโดรนโดย GPS module จะทำการติดต่อกับดาวเทียม GPS อย่างน้อย 3 จุด เพื่อที่จะสามารถระบุตำแหน่ง Latitude และ Longitude ณ ตำแหน่งปัจจุบันได้

โมดูลจีพีเอส u-blox NEO-7N เป็นโมดูลที่ใช้พลังงานต่ำ ใช้พลังงานอยู่ที่ 2.7 V - 3.6 V ส่งข้อมูลผ่าน UART โดยโมดูลจะรับค่าจากดาวเทียมและส่งข้อมูลค่า GPS ออกมาเป็น 6 โปรโตคอลคือ GPRMC, GPVTG, GPGGA, GPGSA, GPGSV และ GPGLL

ข้อมูลทางเทคนิค

- ขนาด : 60x11.5mm
- ความยาว : 140mm
- ความยาวสาย Cable : 200mm
- น้ำหนัก : 26g
- ระยะที่มีความแม่นยำ : 2m
- รองรับช่องสัญญาณ : 56

2.5.11 Telemetry Radio



รูปที่ 2 - 27 Telemetry Radio

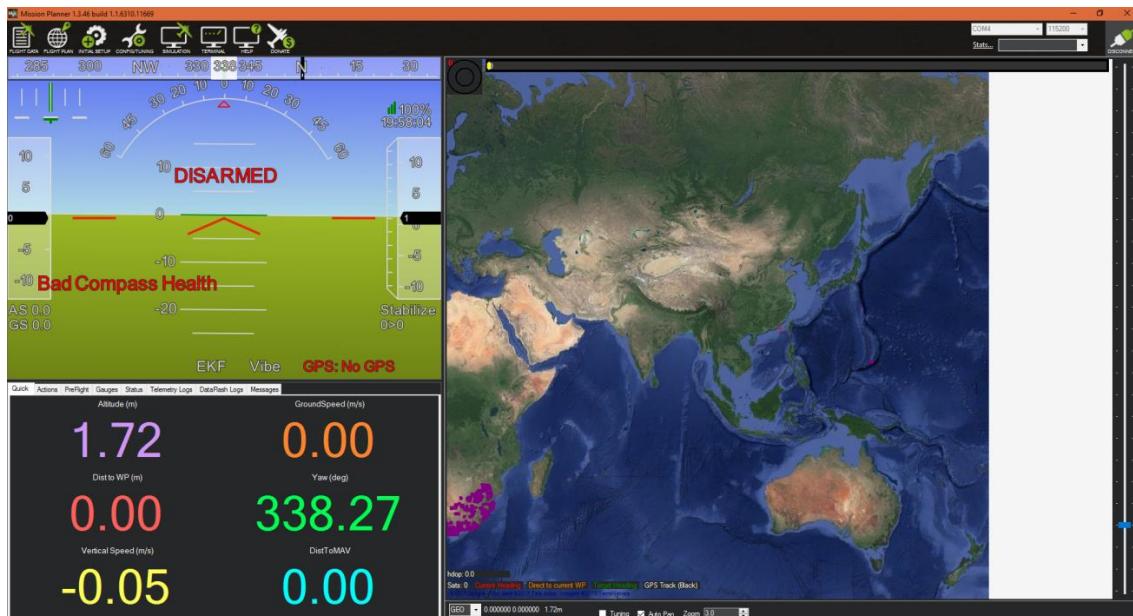
จากรูปที่ 2-27 Telemetry Radio เป็นอุปกรณ์ที่ใช้ในการรับส่งสัญญาณที่ส่งออกมาจากตัวบอร์ด APM ไปยังตัวรับปลายทาง ซึ่งตัว Telemetry Radio จะทำงานกับพอร์ต USB ดังนั้นอุปกรณ์ที่ทำงานร่วมกับ Telemetry Radio จะต้องรองรับการทำงานผ่านพอร์ต USB

ข้อมูลทางเทคนิค

- คลื่นความถี่ 433 MHz
- มีกรอบการรายงานโปรโตคอล MAVLink และมีการรายงานสถานะ
- อัตราการส่งข้อมูลในอากาศ 250kbps
- กำลังในการส่ง 100 mW
- ช่วงที่มีประสิทธิภาพ 1km

2.6 โปรแกรม Mission Planner

Mission Planner เป็นโปรแกรมสำหรับทำการตั้งค่าบอร์ด APM ที่ใช้สำหรับควบคุมโดรนหรืออากาศยานจากภาคพื้นและยังเป็น open-source ground station application อีกด้วย อีกทั้งยังสามารถ configure ค่าต่างของบอร์ด APM ได้และยังสามารถตั้งค่าการทำงานของ GPS และสามารถรองรับการทำงานบน Windows, Mac OS-X และ Linux



รูปที่ 2 - 28 Program Mission Planner

จากรูปที่ 2-28 เป็นตัวอย่างโปรแกรม Mission Planner ซึ่งจะบอกค่าต่าง ๆ ของ Drone ที่ทำการเข้ามายังหน้าจอ



รูปที่ 2 - 29 รายละเอียดต่าง ๆ ของโปรแกรม Mission Planner

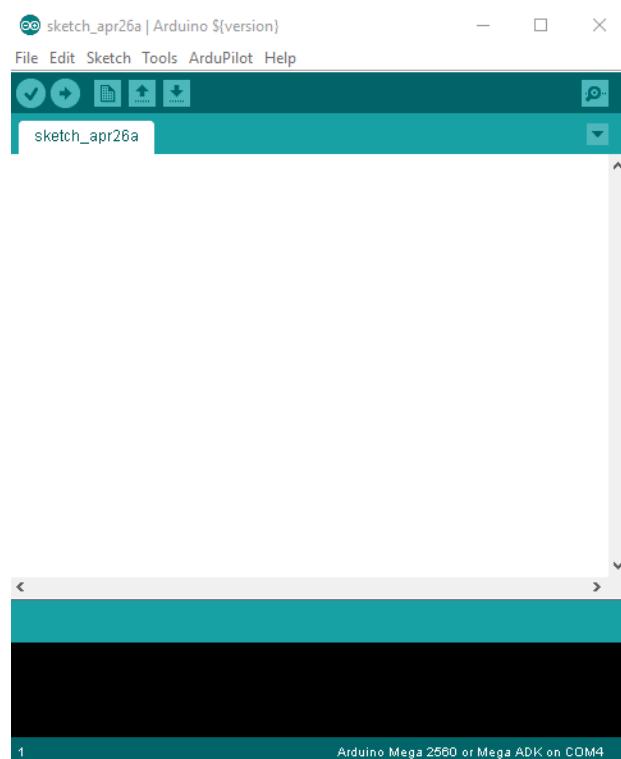
จากรูปที่ 2-29 เป็นส่วนต่าง ๆ ของโปรแกรม Mission Planner ในขณะที่ทำการบิน Drone อยู่

เมนูพื้นฐานบนโปรแกรม

Fight Data	คือ ข้อมูลเกี่ยวกับการบินโดยจะส่งมาแสดงเป็น GUI ให้เราเห็นความเคลื่อนไหวของ Drone
Fight Plan	คือ การแสดงตำแหน่งของการบินและสั่งงานการบินเป็น Way point
Initial Setup	คือ การตั้งค่าต่าง ๆ เกี่ยวกับบอร์ด APM พื้นฐาน
Config/Tuning	คือ การตั้งค่าต่าง ๆ เกี่ยวกับตัวโปรแกรมและการจุนค่าต่าง ๆ
Simulation	คือ การทดสอบค่าการบินต่าง ๆ โดยการจำลองการบิน
Firmware	คือ การการโหลดตัว firmware version ใหม่จากทาง Diydroner
Terminal	คือ การเปิดหน้าต่างการ Config แบบพิมพ์ Command ใช้ตั้งค่าต่าง ๆ
Help	คือ การเปิดหน้าต่างขอความช่วยเหลือ
Donate	คือ การเปิดหน้าต่างสั่งซื้อ
Connect	คือ การทำการเชื่อมต่อกับบอร์ด APM

2.7 โปรแกรม Arduino IDE – ArduPilot

Arduino IDE เป็นเครื่องมือการเขียนโปรแกรมที่มีใช้งานได้กับบอร์ด Arduino ได้ทุกรุ่น โดยภายในจะมีเครื่องมือที่จะเป็นสำหรับติดต่อ Arduino เช่น การค้นหา Arduino ที่ติดต่อกับเครื่องคอมพิวเตอร์ การเลือกรุ่น Arduino ที่ต้องอยู่เพื่อตรวจสอบว่าขนาดของโปรแกรมที่เขียนหรือไลบรารีต่าง ๆ ซับพอร์ตกับ Arduino รุ่นนั้น ๆ อีกทั้งยังมีโปรแกรมติดต่อผ่านซีเรียลโดยตรงสำหรับคอมพิวเตอร์และยังเป็นโปรแกรมโอเพ่นซึ่งสามารถนำไปใช้งานได้พร้อม ๆ อีกทั้งมีซอฟต์แวร์ที่ต้องติดต่อกับ Arduino ที่ชื่อ ArduPilot ที่สามารถใช้ในการเขียนโปรแกรมโดยเป็นเวอร์ชันที่พัฒนาพิเศษมาจากการพัฒนา ArduPilot ให้สามารถใช้งานกับบอร์ด ArduPilotMega โดยเฉพาะ



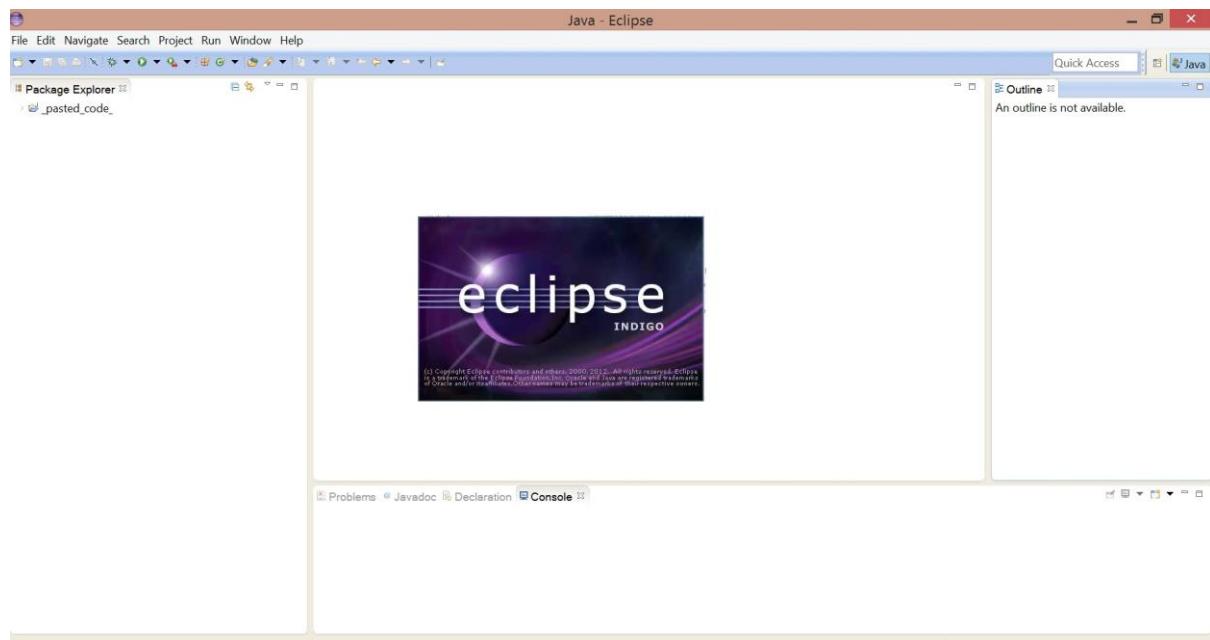
รูปที่ 2 - 30 Program Arduino IDE – ArduPilot

2.8 โปรแกรม Eclipse

Eclipse คือ โปรแกรมที่ใช้สำหรับพัฒนาภาษา Java ซึ่งโปรแกรม Eclipse เป็นโปรแกรมหนึ่งที่ใช้ในการพัฒนา Application Server ได้อย่างมีประสิทธิภาพและเนื่องจาก Eclipse เป็นซอฟต์แวร์ OpenSource ที่พัฒนาขึ้นเพื่อใช้โดยนักพัฒนาเอง ทำให้ความก้าวหน้าในการพัฒนาของ Eclipse เป็นไปอย่างต่อเนื่องและรวดเร็ว

Eclipse มีองค์ประกอบหลักที่เรียกว่า Eclipse Platform ซึ่งให้บริการพื้นฐานหลักสำหรับรวมเครื่องมือต่าง ๆ จากภายนอกให้สามารถเข้ามาทำงานร่วมกันในสภาพแวดล้อมเดียวกัน และมีองค์ประกอบที่เรียกว่า Plug-in Development Environment (PDE) ซึ่งใช้ในการเพิ่มความสามารถในการพัฒนาซอฟต์แวร์มากขึ้น เครื่องมือภายนอกจะถูกพัฒนาในรูปแบบที่เรียกว่า Eclipse plug-ins ดังนั้น หากต้องการให้ Eclipse ทำงานได้เพิ่มเติม ก็เพียงแต่พัฒนา plugin สำหรับงานนั้นขึ้นมาและนำ Plug-in นั้นมาติดตั้งเพิ่มเติมให้กับ Eclipse ที่มีอยู่เท่านั้น Eclipse Plug-in ที่มีมาพร้อมกับ Eclipse เมื่อเรา download มาครั้งแรกก็คือองค์ประกอบที่เรียกว่า Java Development Toolkit (JDT) ซึ่งเป็นเครื่องมือในการเขียนและ Debug โปรแกรมภาษา Java

ข้อดีของโปรแกรม Eclipse คือ ติดตั้งง่าย สามารถใช้ได้กับ J2SDK ได้ทุกเวอร์ชัน รองรับภาษาต่างประเทศ อีกหลายภาษา มี plugin ที่ใช้เสริมประสิทธิภาพของโปรแกรม สามารถทำงานได้กับไฟล์หลายชนิด เช่น HTML, Java, C, JSP, EJB, XML และ GIF และ ใช้งานได้กับระบบปฏิบัติการ Windows, Linux และ Mac OS



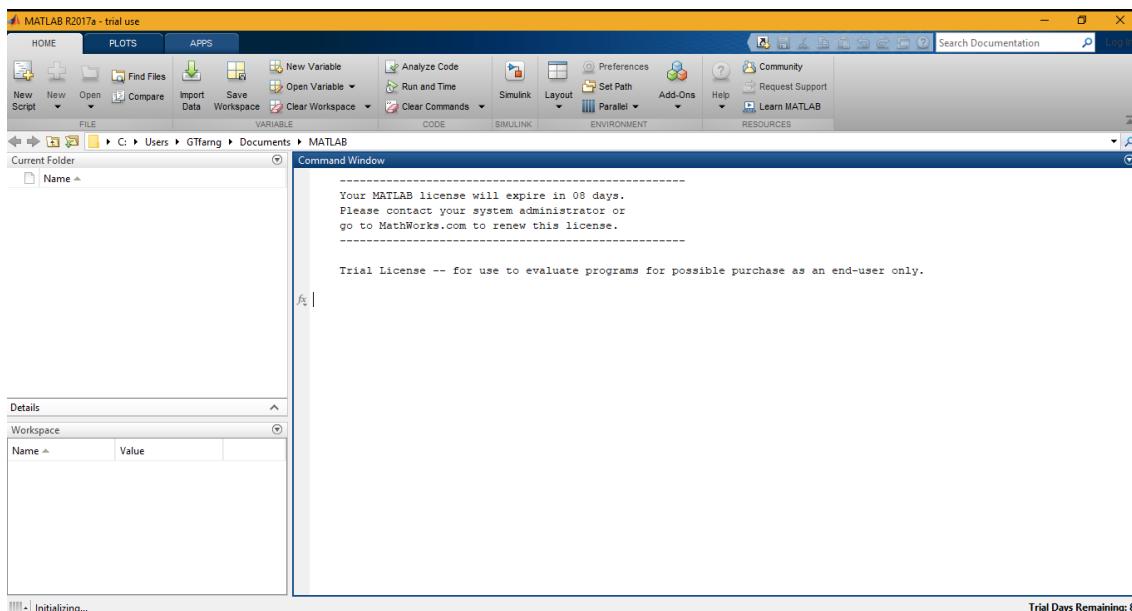
รูปที่ 2 - 31 Program Eclipse

2.9 โปรแกรม Matlab

Matlab เป็นภาษาคอมพิวเตอร์ระดับสูงที่มาพร้อมด้วยสภาพแวดล้อมการทำงานเชิงโต้ตอบ (คล้าย ๆ เครื่องคิดเลข) ซึ่งสามารถคำนวณคณิตศาสตร์ที่ซับซ้อนได้อย่างรวดเร็วมากกว่าภาษาคอมพิวเตอร์สมัยก่อน เช่น ภาษา C, C++ หรือ Fortran

Matlab เป็นภาษาคอมพิวเตอร์ระดับสูงที่ใช้สำหรับคำนวณเชิงตัวเลข แสดงผลกราฟฟิกและเขียนแอปพลิเคชัน ทำให้เราสามารถคำนวณผลลัพธ์ พัฒนาอัลกอริทึม สร้างแบบจำลอง และแอปพลิเคชันได้ง่าย ๆ และรวดเร็วมาก ภายใต้ Matlab ประกอบด้วยภาษาคอมพิวเตอร์ ทูลบ็อกซ์ (Toolbox: กลุ่มฟังก์ชันสำเร็จรูป ในแต่ละสาขาวิชา) และฟังก์ชันพื้นฐานจำนวนมาก ทำให้การวิเคราะห์ทำได้หลากหลายวิธี พร้อมกับคำตอบที่รวดเร็วกว่าโปรแกรมตารางคำนวณ (Spreadsheet) หรือภาษาคอมพิวเตอร์สมัยก่อน เช่น C, C++, Fortran, Java และอื่น ๆ

Matlab สามารถนำไปประยุกต์ใช้งานได้หลายสาขามาก ทั้งการประมวลผลสัญญาณ (Signal Processing) การสื่อสาร (Communication) การประมวลผลภาพและวิดีโอ (Image and Video Processing) ระบบควบคุม (Control System) การวัดและควบคุม (Instruments and Control) การคำนวณทางเศรษฐศาสตร์ (Economic) การคำนวณทางชีววิทยา (Biology) และอื่น ๆ มีนักวิทยาศาสตร์และวิศวกรหลายล้านคนทั่วโลกที่ใช้ Matlab ในการคำนวณเชิงตัวเลข

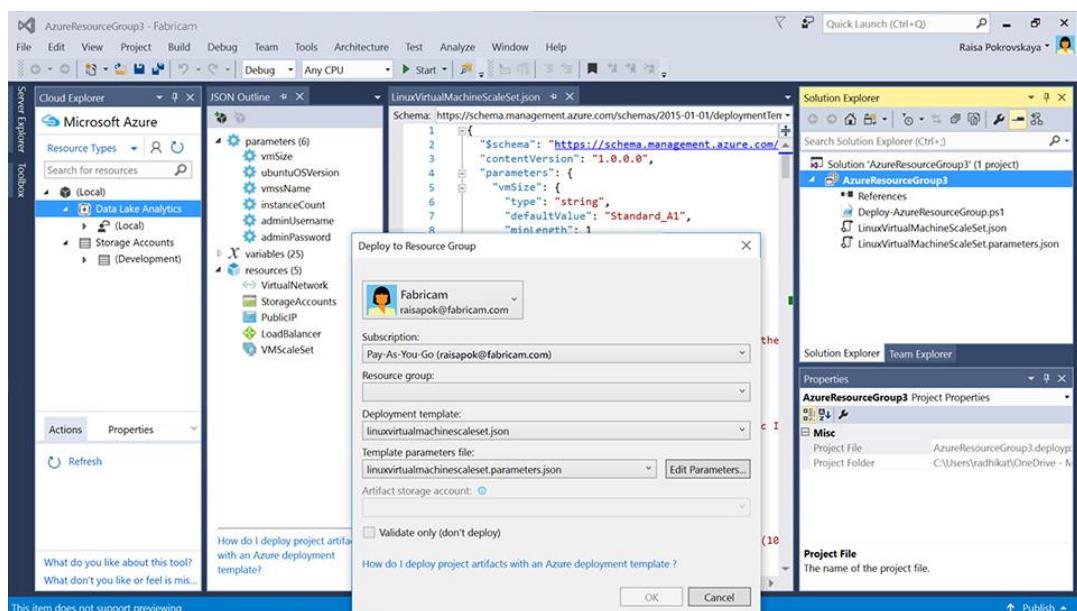


รูปที่ 2 - 32 Program Matlab

2.10 โปรแกรม Virtual Studio

Visual Studio (วิชวลสตูดิโอ) คือ โปรแกรมตัวหนึ่งที่เป็นเครื่องมือที่ช่วยพัฒนาซอฟต์แวร์และระบบต่าง ๆ ซึ่งสามารถติดต่อสื่อสารพูดคุยกับคอมพิวเตอร์ได้ในระดับหนึ่งแล้ว แต่ยังไม่สามารถพัฒนาเป็นระบบเองได้ หมายความสำหรับภาษา VB และ VB.NET เนื่องจากไม่ครอบคลุมได้พัฒนาโปรแกรมและภาษาขั้นมาควบคู่กัน เพื่อให้ใช้งานได้ซึ่งกันและกัน ซึ่งนักโปรแกรมเมอร์จะนำเครื่องมือมาใช้ในการพัฒนาต่อ�อดให้เกิดเป็นระบบต่าง ๆ หรือเป็นเว็บไซต์ และแอปพลิเคชันต่าง ๆ

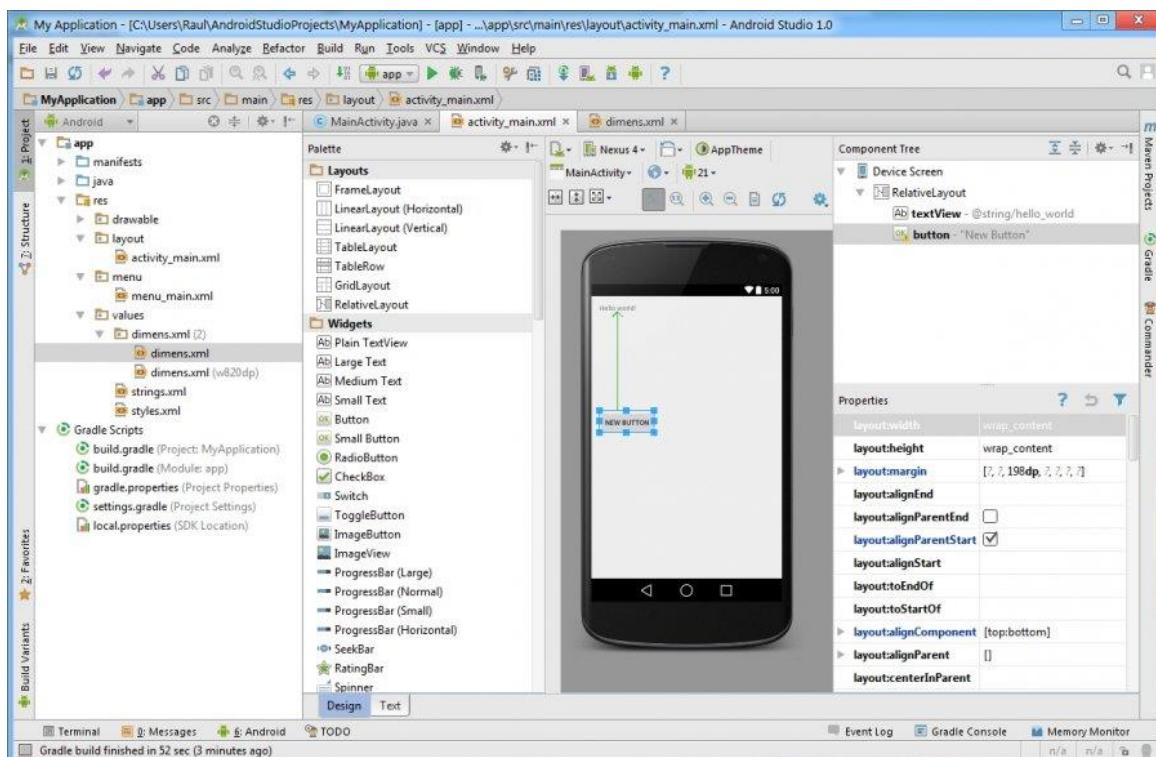
ดังนั้น วิชวลสตูดิโอจึงเป็นโปรแกรมตัวหนึ่งที่เป็นเครื่องมือที่ช่วยพัฒนาซอฟต์แวร์และระบบต่าง ๆ ซึ่งโปรแกรมได้มีการติดต่อสื่อสารกับคอมพิวเตอร์ในระดับหนึ่งแล้ว แต่ไม่สามารถพัฒนาเป็นระบบได้ด้วยตนเอง นักพัฒนาจะนำเครื่องมือของโปรแกรมมาใช้พัฒนาต่อให้เกิดเป็นซอฟต์แวร์หรือระบบต่าง ๆ เพื่อช่วยอำนวยความสะดวก สะดวก และลดเวลาการทำงานและข้อผิดพลาดได้เป็นอย่างมาก



รูปที่ 2 - 33 Program Virtual Studio

2.11 โปรแกรม Android Studio

Android Studio เป็น IDE Tool จาก Google ไว้พัฒนา Android สำหรับ Android Studio เป็น IDE Tools ล่าสุดจาก Google ไว้พัฒนาโปรแกรม Android โดยเฉพาะ โดยพัฒนาแนวคิดพื้นฐานมาจาก IntelliJ IDEA คล้าย ๆ กับการทำงานของ Eclipse และ Android ADT Plugin โดยวัตถุประสงค์ของ Android Studio คือต้องการพัฒนาเครื่องมือ IDE ที่สามารถพัฒนา App บน Android ให้มีประสิทธิภาพมากขึ้น ทั้งด้านการออกแบบ GUI ที่ช่วยให้สามารถ Preview ตัว App บุกมองที่แตกต่างกันบน Smart Phone แต่ละรุ่น สามารถแสดงผลบางอย่างได้ทันทีโดยไม่ต้องทำการรัน App บน Emulator รวมทั้งยังแก้ไขปรับปรุงในเรื่องของความเร็วของ Emulator ที่ยังเจอบัญหา กันอยู่ในปัจจุบัน



รูปที่ 2 - 34 Program Android Studio

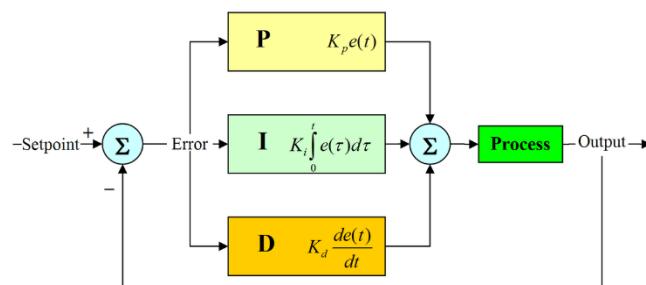
จากรูปที่ 2-34 เป็นหน้าแสดงการทำงานของโปรแกรม Android Studio โดยจะแบ่งเป็น 3 ส่วนคือ ด้านซ้าย ตรงกลาง และด้านขวา ซึ่งด้านซ้ายเป็นส่วนที่แสดง รายชื่อไฟล์และโฟลเดอร์ที่เก็บไฟล์ที่ใช้ในการทำงาน ในแต่ละโปรเจค ตรงกลางเป็นส่วนแสดงข้อมูลในไฟล์ หรือก็คือส่วนที่ใช้เขียนโค้ดต่าง ๆ ส่วนด้านขวาเป็นส่วนที่ จะแสดงตอนออกแบบ จัดตกแต่งหน้าตา GUI ของ Application Android ที่ทำการออกแบบไว้ โดยสามารถสร้างโดยเขียนโค้ดขึ้นมาหรือใช้ Tool ของโปรแกรมช่วย เพื่อสะดวกต่อการสร้าง

2.12 ภาษา C++

ภาษาซีพลัสพลัส (C++) เป็นภาษาโปรแกรมคอมพิวเตอร์อเนกประสงค์ มีโครงสร้างภาษาที่มีการจัดชนิดข้อมูลแบบ statically typed และสนับสนุนรูปแบบการเขียนโปรแกรมที่ได้แก่ การโปรแกรมเชิงกระบวนการคำสั่ง การนิยามข้อมูล การโปรแกรมเชิงวัตถุ และการโปรแกรมแบบเจเนริก ซึ่งมีประสิทธิภาพและความยืดหยุ่นในการออกแบบโปรแกรมสูง และสามารถพัฒนาได้ในหลาย ๆ แพลตฟอร์มถูกออกแบบมาเพื่อเป็นภาษาสำหรับการเขียนโปรแกรมทั่วไปสามารถรองรับการเขียนโปรแกรมในระดับภาษาเครื่องได้ และเป็นภาษาที่มีการเปิดกว้าง และยังเป็นภาษาที่มีความซับซ้อนมากกว่าภาษาซี แต่ได้รับการออกแบบเพื่อเข้ากันได้กับภาษาซีในเกือบทุกรณี

2.13 การควบคุมแบบพีไอดี (PID Controller)

ในระบบควบคุมมีตัวควบคุมหลายชนิด ตัวควบคุมส่วนใหญ่ที่ใช้ในการควบคุมกระบวนการเป็นแบบ PID โดยต้องนุ่มนวลกับระบบที่ต้องการควบคุม ดังแสดงในรูปที่ 2-34 สัญญาณออกจากตัวควบคุม PID สามารถบรรยายได้ดังนี้



รูปที่ 2 - 35 ตัวควบคุม PID แบบขนาน

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right)$$

โดย $u(t)$ คือสัญญาณควบคุม $e(t)$ คือค่าความคลาดเคลื่อนของสัญญาณออกจากตัวกำหนด

การควบคุม PID ประกอบไปด้วยเทคนิคการควบคุมพื้นฐาน 3 แบบ แบบสัดส่วน (Proportional หรือ P) แบบอินทิเกรล (Integral หรือ I) และแบบอนุพันธ์ (Derivative หรือ D) แต่ละแบบสามารถนำมาประกอบกันเพื่อให้ได้ตัวควบคุมที่ต้องการ ตัวควบคุมมีพารามิเตอร์ 3 ตัว คือค่าอัตราขยายแบบสัดส่วน (K_p) ค่า integral time (T_i) และ derivative time (T_d) ซึ่งรายละเอียดของแต่ละแบบมีดังนี้

2.13.1 Proportional Action

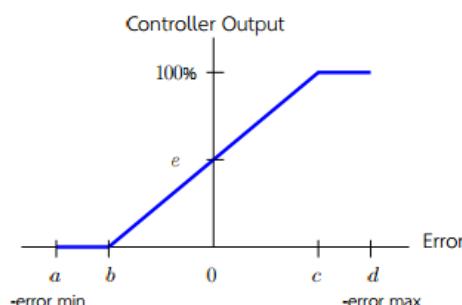
การควบคุมแบบสัดส่วนเป็นเทคนิคที่ง่ายที่สุด หลักการคือสัญญาณควบคุม ($u(t)$) จากตัวควบคุมที่ส่งไปปรับกระบวนการมีค่าเป็นสัดส่วนกับความคลาดเคลื่อน ซึ่งสามารถเขียนได้ในรูป

$$u(t) = K_p e(t)$$

โดยที่ K_p คือค่าอัตราขยายและ $e(t)$ = ความคลาดเคลื่อน = ค่ากำหนด - ค่าวัด

ตัวควบคุมบางตัวสัญญาณเข้าและสัญญาณออกอาจมีหน่วยต่างกัน เช่นการเปลี่ยนแปลงของอุณหภูมิที่ทำให้เกิดการเปลี่ยนแปลง ความดัน เพื่อหลีกเลี่ยงการแปลงหน่วย ความสัมพันธ์ระหว่างสัญญาณออกและสัญญาณเข้าของตัวควบคุมอาจแสดงเป็นแบบสัดส่วน (Proportional Band หรือ %PB) โดยที่แบบสัดส่วนคือพิสัยของสัญญาณเข้าที่ทำให้ตัวควบคุมปฏิบัติงานเต็มพิสัยการทำงานหรือถ้ามองจากตัวควบคุม แบบสัดส่วนคือช่วงความคลาดเคลื่อนที่ทำให้สัญญาณออกของตัวควบคุมเปลี่ยนแปลงจากค่าสูงสุดไปต่ำสุด โดยแสดงเป็นเปอร์เซ็นต์ของพิสัยสัญญาณเข้าตัวควบคุม ความสัมพันธ์ระหว่างอัตราขยายและเปอร์เซ็นต์แบบสัดส่วนคือ

$$K_p = \frac{100}{\% PB}$$



รูปที่ 2 - 36 ตัวบทแบบสัมบัติของการควบคุมแบบสัดส่วน

จากรูปที่ 2-35 ค่าเปอร์เซ็นต์แบบสัดส่วน (%PB) คือระยะ bc เม็ดความคลาดเคลื่อนเป็นศูนย์ ยังมีสัญญาณค่าหนึ่งออกจากตัวควบคุมที่ป้อนให้กับกระบวนการ ค่านี้叫做 *bias* ทำให้ระบบทำงานที่จุดทำงานต่อไปได้ โดยทั่วไป สัญญาณค่านี้มักจะถูกตั้งให้เท่ากับ 50% ของสัญญาณขาออกสูงสุดของตัวควบคุม นั่นคือ

$$\text{สัญญาณออก} = \left(\frac{\% \text{ ความคลาดเคลื่อน}}{\% \text{ แบบสัดส่วน}} \right) + 50\%$$

นอกจากนี้ ตัวควบคุมมี责任ทำงานที่เป็นเชิงเส้นช่วงหนึ่ง โดยทำหน้าที่เป็นตัวขยาย (amplifier) และถ้าความคลาดเคลื่อนมีมากเกินระดับหนึ่ง ตัวขยายจะอิ่มตัวทำให้สัญญาณออกมีค่าคงที่ การควบคุมแบบสัดส่วนนี้สามารถควบคุมระบบได้ดีพอสมควร หมายความกับกระบวนการที่ต้องการผลตอบสนองรวดเร็วและยอมให้เกิดความคลาดเคลื่อนขนาดคงที่ขนาดหนึ่ง อย่างไรก็ตาม หากในกระบวนการเกิดมีการเปลี่ยนแปลงพารามิเตอร์ อาจทำให้เกิดปัญหา เช่น มีความคลาดเคลื่อนในสภาพะอยู่ตัว (steady-state error) หรือที่เรียกว่าอฟเซต (offset) ตัวควบคุมแบบสัดส่วนไม่สามารถแก้ไขให้หมดได้

แนวทางการแก้ไขปัญหาที่เกิดขึ้นทำได้ 2 วิธีคือวิธีแรกคือ เพิ่มอัตราขยาย (gain) ของตัวควบคุมเพื่อเพิ่มผลของความคลาดเคลื่อนที่มีต่อระบบ ถึงแม้ความคลาดเคลื่อนที่เกิดขึ้นจะมีค่าน้อยลงแต่ก็จะทำให้สัญญาณออกจากตัวควบคุมที่เหมาะสมกับกระบวนการขณะนั้นได้ อย่างไรก็ตามการเพิ่มผลของความคลาดเคลื่อนมากเกินไปก็อาจทำให้ระบบแกว่งได้เนื่องจากระบบมีความไว วิธีที่สองคือ ปรับค่าใบแอกซของตัวควบคุมใหม่ด้วยมือ ซึ่งทำให้ตัวควบคุมเลื่อนจุดทำงานไปยังจุดที่ให้สัญญาณออกที่เหมาะสมกับกระบวนการในขณะนั้นได้ ปัญหาของวิธีหลังอยู่ตรงที่ต้องปรับค่าใบแอกซของตัวควบคุมทุกรั้งที่มีการเปลี่ยนแปลงพารามิเตอร์ของกระบวนการ

2.13.2 Integral Action

ผลตอบของการควบคุมแบบสัดส่วนรวมกับการควบคุมแบบอินทิเกรชันทิกรัล สามารถอธิบายได้ในสมการ

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau \right)$$

เมื่อ K_p คืออัตราขยาย $e(t)$ คือความคลาดเคลื่อน และ T_i คือ integral time (วินาที)

เมื่อเปรียบเทียบกับสมการของตัวควบคุมแบบสัดส่วน ความแตกต่างอยู่ตรงที่เพิ่มไปแอกซันน์คือตัวควบคุมแบบสัดส่วนถูกจำกัดด้วยส่วนใบแอกซเป็นค่าคงที่ ส่วนการควบคุมแบบอินทิเกรชันทิกรัลมีการสะสมความคลาดเคลื่อนในการปรับแต่งใบแอกซ (นั่นคือ ทำหน้าที่เป็นตัวอินทิเกรชัน) และจะหยุดสะสมเมื่อความคลาดเคลื่อนของระบบเป็นศูนย์ เมื่อผลตอบเข้าที่สมบูรณ์แล้ว เพิ่มไปแอกซของระบบจะมีค่าน้อยเพียงใดขึ้นอยู่กับลักษณะของการรบกวน (disturbance) การทำงานในลักษณะเช่นนี้มีลักษณะคล้ายกับฟังก์ชันรีเซตด้วยมือ (manual-reset function) ดังนั้นในบางครั้งจึงเรียกตัวอินทิเกรชันทิกรัลว่าฟังก์ชันรีเซต (reset function)

คุณสมบัติของตัวอินทิกรัลในการกำจัดความคลาดเคลื่อน (หรืออฟเซต) เป็นข้อดีอย่างมาก จึงเป็นที่นิยมใช้กับระบบควบคุมป้อนกลับ อย่างไรก็ตามตัวอินทิกรัลก็มีข้อเสีย นั่นคือทำให้เกิดการล้าหลัง (capacity-like lag) และทำให้ช่วงเวลาของการแกว่งยาวนานขึ้น โดยที่ไปรับแบบสัดส่วนรวมกับอินทิกรัลจะมีช่วงเวลาของการแกว่งนานกว่าระบบเชิงสัดส่วนอย่างเดียว 50% หรือ $TPI = 1.5TP$ สำหรับระบบที่มีค่าคงตัวเวลา (time constant) น้อย (เช่น ระบบควบคุมอัตราการ ไฟล) ปัญหานี้จะไม่มีผลมากนักแต่สำหรับระบบที่มีค่าคงตัวเวลามาก (เช่น ระบบควบคุมระดับ) ปัญหานี้อาจมีผลมากจนทำให้ระบบเข้าสู่จุดวิกฤติที่ไม่สามารถรับได้

การควบคุมแบบอินทิกรัลมีลักษณะเช่นเดียวกับการควบคุมสัดส่วนตรงผลกระทบของการเพิ่มอัตราขยายของตัวควบคุม หากอัตราขยายมีค่ามากเกินไปจะทำให้ผลตอบของระบบมีการแกว่ง โดยที่ไป Integral time ($T_i = 1/K_i \text{ sec}$ โดยที่ $K_i = \text{repeats/sec}$) เป็นตัวแสดงว่าอัตราการตอบสนองของระบบการต่อสัญญาณการควบคุม ค่า T_i ที่น้อยกว่าจะทำให้ตัวควบคุมมีการตอบสนองที่เร็วกว่าในระยะเริ่มต้น โดยที่ความคลาดเคลื่อนยังเป็นค่าบวกอยู่ ดังนั้นกว่าความคลาดเคลื่อนจะเป็นศูนย์ (ซึ่งทำให้เทอม $\int t_0 e(t) dt$ หยุดทำงาน) เทอมไปแอกส์จะมีค่าสูงกว่าที่ต้องการ ดังนั้นผลตอบสนองจึงเกิดส่วน พุ่งเกิน (overshoot) สูงกว่าค่ากำหนด เป็นผลให้ตัวอินทิกรัลทำหน้าที่ปรับให้ความคลาดเคลื่อนมีค่าลดลง การใช้ตัวอินทิกรัลในการควบคุมควรระวังในเรื่องของความคลาดเคลื่อนขนาดใหญ่ (เช่นเกิดการเปลี่ยนแปลงค่ากำหนดขนาดใหญ่) เพราะจะทำให้เกิดปัญหา integral windup ถึงแม้ว่า T_i มีค่าถูกต้องในสภาพการทำงานธรรมดា แต่สัญญาณควบคุมอาจถึงจุดอิ่มตัวขณะผลตอบเกิดส่วนพุ่งเกิน

2.13.3 Derivative Action

ตัวควบคุมแบบสัดส่วนและแบบอินทิกรัลต่างก็มีข้อจำกัดอยู่ที่ความคลาดเคลื่อนขนาดใหญ่ ซึ่งเป็นปัญหาต่อการควบคุมกระบวนการ แต่ความคลาดเคลื่อนขนาดใหญ่นี้สามารถรู้ได้ช่วงหน้าโดยพิจารณาจากแนวโน้มของความคลาดเคลื่อนหรืออัตราการเปลี่ยนแปลงของสัญญาณนั้นเอง ตัวอนุพันธ์มีหลักการทำงานคือตัวควบคุมตอบสนองต่ออัตราการเปลี่ยนแปลงของความคลาดเคลื่อน ถึงแม้ว่าความคลาดเคลื่อนมียังค่าเล็กอยู่ สัญญาณออกของตัวอนุพันธ์ไม่ได้สัมพันธ์กับขนาดของความคลาดเคลื่อน แต่เป็นอัตราการเปลี่ยนแปลงของความคลาดเคลื่อน ถ้าความคลาดเคลื่อนมีค่าคงที่ ตัวอนุพันธ์จะให้สัญญาณออกเป็นศูนย์ คุณลักษณะข้อนี้มีผลต่อตัวควบคุมจะมีผลตอบสนองที่เกิดก่อนที่ความคลาดเคลื่อนจะเพิ่มมากขึ้นและทำให้ระบบมีผลตอบสนองที่เร็วขึ้น ตัวควบคุมแบบอนุพันธ์สามารถเขียนได้ดังนี้

$$u(t) = K_p \left(e(t) + T_d \frac{de(t)}{dt} \right)$$

โดย derivative time (T_d) เป็นเวลาที่แสดงถึงผลตอบสนองเนื่องจากตัวอนุพันธ์การเพิ่ม T_d จะทำให้ผลตอบสนองของตัวอนุพันธ์มีค่ามากขึ้น เนื่องจากตัวอนุพันธ์มีความไวต่อการเปลี่ยนแปลงมาก ดังนั้นจึงนิยมใช้กับค่าที่รัดได้เท่านั้น แต่ไม่ใช้กับค่ากำหนด เพราะการเปลี่ยนค่ากำหนดมักจะเป็นแบบขั้น (step) ทำให้ผลตอบสนองของตัวอนุพันธ์เป็นพลัสและทำให้เกิดการกระแทก (bounce) ของอุปกรณ์ในกระบวนการสำหรับค่ากำหนดใช้เฉพาะกับตัวควบคุมสัดส่วนและอินทิกรัล

ตัวอนุพันธ์คือตัวควบคุมที่ก่อให้เกิดผลตรงข้ามกับตัวอินทิกรัล ดังนั้นจึงใช้ในการปรับปรุงกระบวนการที่มีการล้าหลังทางเวลา (time lag) มากๆ ทำให้ผลตอบสนองรวดเร็วขึ้นและช่วงเวลาการแก่งที่สั้นลง ข้อเสียของตัวอนุพันธ์คือ มีความไวต่อสัญญาณรบกวนเป็นอย่างมาก เพราะมีผลตอบสนองโดยตรงต่ออัตราการเปลี่ยนแปลงของสัญญาณที่รัดได้ ดังนั้นแม้สัญญาณรบกวนจะมีขนาดเล็กแต่ก็อาจก่อให้เกิดการเปลี่ยนแปลงต่อสัญญาณออกของตัวควบคุม จึงเป็นไปไม่ได้ที่จะใช้ตัวอนุพันธ์ในการควบคุมผลของสัญญาณรบกวน ยิ่งไปกว่านั้นระบบใดที่มีสัญญาณรบกวนมากจะไม่สามารถใช้ตัวอนุพันธ์ ในการอุตสาหกรรมส่วนใหญ่นิยมใช้เพียงตัวควบคุม PI เท่านั้น

2.14 PID Tuning

จากสมการ PID เราจะมีค่าคงที่อยู่ 3 ตัวคือ K_p , K_i และ K_d ที่ต้องกำหนดค่าเข้าไป การเลือกค่าที่เหมาะสมจะทำให้ระบบสมดุลและได้ผลลัพธ์ตรงตามต้องการมากที่สุด ซึ่งก็คือการจูน PID ส่วนวิธีการจูน PID สามารถแบ่งได้สองแบบใหญ่ๆ คือ Close loop tuning และ Open loop tuning

- Closed loop คือการจูน PID controller ในโหมด Auto และกำลังคอนโทรลระบบอยู่
- Open loop คือการจูนโดยอาศัย Process model และการตอบสนองของระบบต่อการเปลี่ยนแปลงของ CV แบบ Step โดยขณะที่ PID controller อยู่ในโหมด Manual

Trial & Error Close-Loop Tuning

วิธีนี้คือการทดลองป้อน K_p และ K_i เข้าคอนโทรล แล้วสังเกตค่าที่ทำให้ระบบสมดุลขึ้นตอนการจูนโดยมีวิธีการดังนี้

1. เริ่มพล็อตกราฟของ Process variable (PV)
2. เช็ต K_i และ K_d เป็นศูนย์
3. เช็ต K_p ค่าน้อย ๆ
4. เปลี่ยน PID Controller ให้อยู่ในโหมด Auto
5. เทระบบโดยเปลี่ยนค่า Set point แล้วสังเกตว่า damping ของระบบ
6. ปรับค่า K_p เพิ่มขึ้นเรื่อยๆ จนระบบมี damping ตามต้องการหรือให้เป็น quarter-amplitude decay (Decay Ratio ~ 0.25) ดังรูป 2-37 (b)
7. ถ้าระบบมี Offset ระหว่าง Set point และ Process variable ให้ปรับ K_i เพิ่มขึ้นจนไม่มี Offset

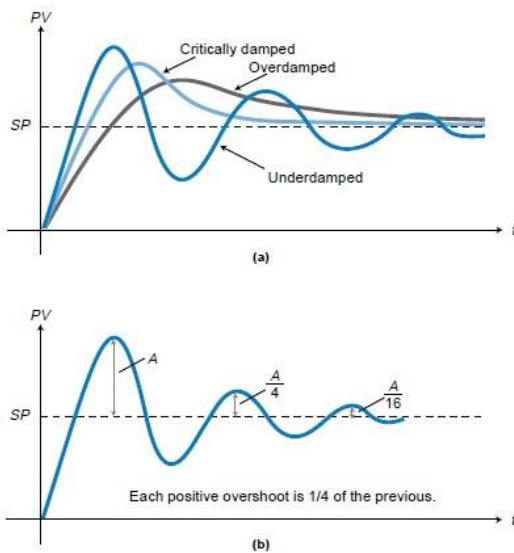


Figure 15-77. Process variable responses: (a) overdamped, critically damped, and (b) quarter-amplitude.

รูปที่ 2 - 37 quarter-amplitude decay

Ziegler-Nichols Close-Loop Tuning

หลักของวิธีนี้คือหาค่า gain ที่ทำให้ระบบเกิด Oscillation แบบ amplitude คงที่ ซึ่งเรียกว่า Ultimate proportional gain (K_{pu}) และค่าการสั่น ซึ่งเรียกว่า Ultimate period (T_u) จากนั้นนำทั้งสองค่านี้ไปคำนวณหา K_p , K_i และ K_d ตามตารางด้านล่าง

ตารางที่ 2 - 1 Ziegler-Nichols Close-Loop Tuning

Type of Controller	Loop Tuning Constant	Tuning Equation Dependent	Tuning Equation Independent
Proportional (P)	K_p	$K_p = 0.5 * K_{pu}$	$K_p = 0.5 * K_{pu}$
Proportional-Integral (PI)	K_p K_i	$K_p = 0.45 * K_{pu}$ $K_i = 1.2 / T_u$	$K_p = 0.45 * K_{pu}$ $K_i = (1.2 * K_p) / T_u$
Proportional-Integral-Derivative (PID)	K_p K_i K_d	$K_p = 0.6 * K_{pu}$ $K_i = 2 / T_u$ $K_d = T_u / 8$	$K_p = 0.6 * K_{pu}$ $K_i = (2 * K_p) / T_u$ $K_d = (T_u * K_p) / 8$

Damped Oscillation Close-Loop Tuning

บางระบบไม่ต้องการให้เกิด Oscillation แบบ amplitude คงที่ วิธีนี้จึงหาค่า gain ที่ทำให้ระบบมี damping เป็น quarter-amplitude decay ซึ่งเรียกว่า Damping proportional gain (K_{dp}) และค่าการสั่น ซึ่งเรียกว่า Damping period (T_{dp}) และคำนวณค่า K_p , K_i และ K_d จากตารางด้านล่าง

ตารางที่ 2 - 2 Damped Oscillation Close-Loop Tuning

Type of Controller	Loop Tuning Constant	Tuning Equation Dependent	Tuning Equation Independent
Proportional (P)	K_p	$K_p = 1.1 * K_{dp}$	$K_p = 1.1 * K_{dp}$
Proportional-Integral (PI)	K_p	$K_p = 1.1 * K_{dp}$	$K_p = 1.1 * K_{dp}$
	K_i	$K_i = 2.6 / T_{dp}$	$K_i = (2.6 * K_p) / T_{dp}$
Proportional-Integral-Derivative (PID)	K_p	$K_p = 1.1 * K_{pu}$	$K_p = 1.1 * K_{pu}$
	K_i	$K_i = 3.6 / T_{dp}$	$K_i = (3.6 * K_p) / T_{dp}$
	K_d	$K_d = T_{dp} / 9$	$K_d = (T_{dp} * K_p) / 9$

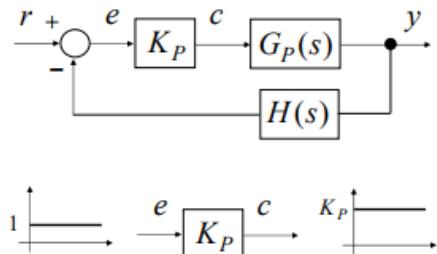
Open Loop Tuning of Self-Regulating process

ในการจูนแบบ Open loop สำหรับระบบที่มี Steady state (Self-regulation) เราจูนโดยอาศัยการคำนวณค่าจากกราฟการตอบสนองของระบบต่อการเปลี่ยนแปลงของ Control variable (CV) ซึ่งมีอยู่หลายวิธี ได้แก่

- Ziegler-Nichols open-loop ให้ผลลัพธ์ของระบบเป็นแบบ Quarter amplitude damping
- Cohen-Coon ให้ผลลัพธ์ของระบบเป็นแบบ Quarter amplitude damping

PID Controller

1. Proportional Mode of Control (P-Control)



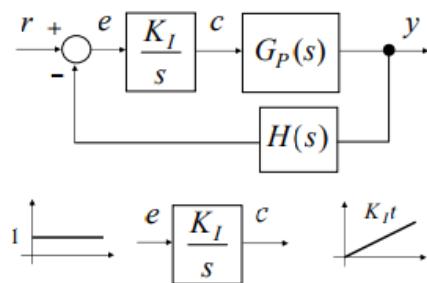
ข้อดีของตัวควบคุมแบบสัดส่วน

1. การเพิ่มค่า Gain ให้สูงขึ้น จะทำให้ได้ผลตอบสนองที่ไวขึ้น

ข้อด้อยของตัวควบคุมแบบสัดส่วน

1. ถ้าใช้งานเดียว ๆ กับระบบที่มี System Type 0 จะไม่สามารถจัด Steady State Error ได้
2. ซึ่งสามารถลดผลกระทบได้ด้วยการเพิ่มค่า Gain ใหม่ค่าสูง ๆ
3. แต่สำหรับ 2nd Order Plant ขึ้นไป การเพิ่มค่า Gain ให้มีค่าสูงจะทำให้เกิด overshoot ที่สูงขึ้นด้วย
4. และการเพิ่มค่า Gain ให้มีค่าสูงอาจจะทำไม่ได้ในทางปฏิบัติ เพราะ output มีค่าจำกัด

2. Integral Mode of Control (I-Control)



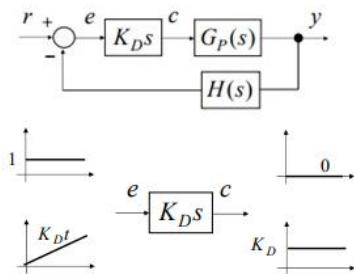
ข้อดีของตัวควบคุมแบบปริพันธ์

1. ขัด steady-state error ได้
2. สามารถใช้งานเดี่ยว ๆ ได้แต่ต้องใช้ค่า Gain ที่เหมาะสม

ข้อด้อยของตัวควบคุมแบบปริพันธ์

1. ไม่สามารถขัด overshoot ได้
2. การใช้ค่า Gain ที่ไม่เหมาะสม อาจทำให้ผลตอบสนองเกิดการแกว่งตัวได้
3. ระบบจะมี Order ที่สูงขึ้น และอาจทำให้ระบบมีผลตอบสนองที่ไม่เร็วประس่งค์

3. Derivative Mode of Control (D-Control)



ข้อดีของตัวควบคุมแบบอนุพันธ์

1. ลดผลตอบสนองที่เป็น overshoot

ข้อด้อยของตัวควบคุมแบบอนุพันธ์

1. ไม่สามารถใช้งานเดี่ยว ๆ ได้ต้องใช้งานร่วมกับตัวควบคุมแบบสัծสวน
2. ไม่สามารถขัด steady-state error ได้
3. ทำให้ผลตอบสนองช้าลง

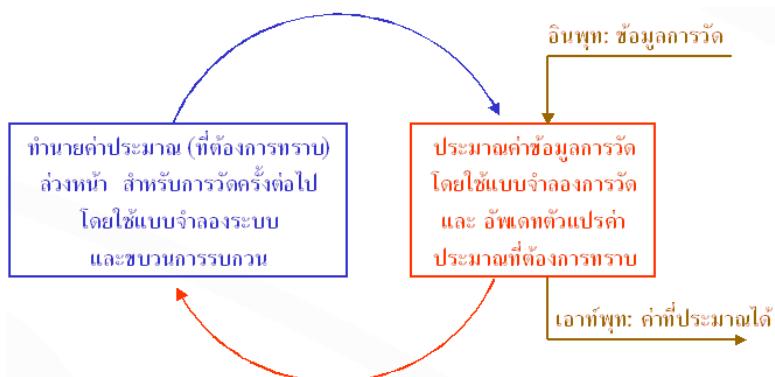
2.15 หลักการ Kalman Filter

Kalman filter เป็นวิธีการที่ใช้เพื่อประมาณค่าหรือสถานะของระบบ โดยนำข้อมูลเกี่ยวกับความไม่แน่นอน เช่นความคลาดเคลื่อนของเซ็นเซอร์ จ่ายอยู่ในรูปแบบของเมตริกซ์ เพื่อให้ใช้งานกับระบบที่มีสัญญาณเข้าออกหลายสัญญาณโดยอาศัยทฤษฎีการประมาณค่า (estimation theory) ที่ใช้พื้นฐานทางคณิตศาสตร์ที่เรียกว่า "stochastic" ซึ่งประกอบด้วยทฤษฎีความน่าจะเป็น (probability) และขบวนการสุ่ม (random process)

โครงสร้างของตัวกรองคัลเมน์มีแบบทั่วไปดังนี้

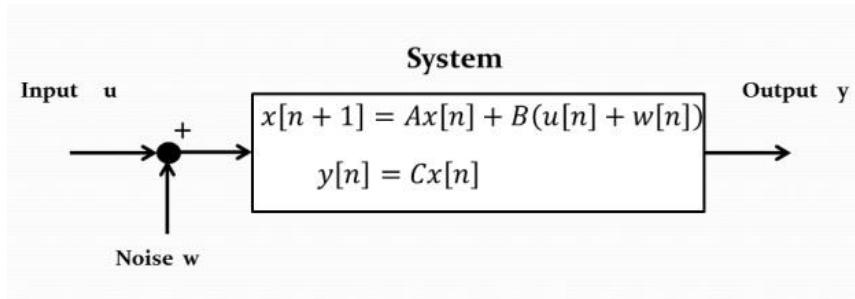
- 1) แบบจำลองระบบ (system model)
- 2) แบบจำลองการวัด (measurement model)
- 3) แบบจำลองขบวนการรบกวน (process noise model)

ตัวกรองคัลเมน์จะทำงานในลักษณะที่เรียกว่า "recursive" โดยใช้แบบจำลองระบบและแบบจำลองขบวนการรบกวนในการค่าประมาณ (ที่ต้องการทราบ) ล่วงหน้าสำหรับการวัดครั้งต่อไปที่จะเกิดขึ้น จากนั้นจะนำค่าประมาณที่ได้ไปใช้ในแบบจำลองการวัด เพื่อทำการประมาณค่าข้อมูลการวัดขึ้น และนำไปเปรียบเทียบค่าข้อมูลการวัดที่ได้จริง การหมุนเวียนของการคำนวณแบบ recursive จะมีจำนวนครั้งเท่ากับจำนวนการวัดในหนึ่งช่วงเวลาที่เราสนใจ



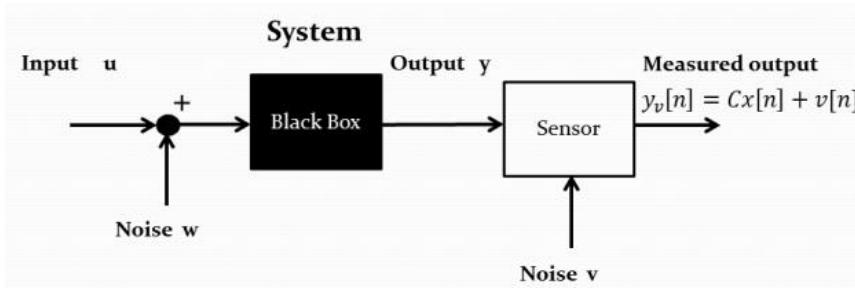
รูปที่ 2 - 38 กระบวนการของ Kalman Filter

การทำงานของ Kalman Filter ใช้แบบจำลองทางคณิตศาสตร์ของระบบในการประมาณสถานะของระบบล่วงหน้า โดยที่ว่าไปแล้วคุณลักษณะของระบบทางกายภาพ (Physical System) สามารถแสดงด้วยแบบจำลองทางคณิตศาสตร์แบบ State Space Representation ที่ประกอบด้วยสถานะของระบบ (State Variable) สัญญาณรบกวน (Noise) และ Input กับ Output ในรูปของสมการอนุพันธ์ (Differential Equation) ได้ตามภาพ



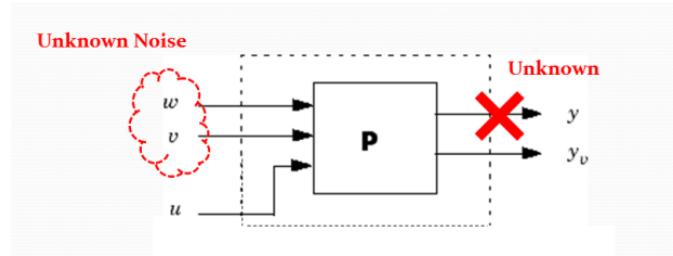
โดยที่ $x[n]$, $u[n]$, $w[n]$ และ $y[n]$ ในรูปของ Vector ตัวแปร คือ สถานะ (State Variables), สัญญาณ Control Input, สัญญาณรบกวน (Noise) และ Output ของระบบ ส่วน A , B และ C ในรูปของ Matrix ค่าคงที่ คือ System Matrix, Input Matrix และ Output Matrix ตามลำดับ โดยที่ $x[n+1]$ คือค่าสถานะที่เปลี่ยนแปลงไปเนื่องจากสถานะก่อนหน้า และ Input ที่ป้อนเข้าไปในระบบ รวมถึงสัญญาณรบกวนในระบบด้วย

ค่าคงที่ใน System Matrix เป็นค่าที่ได้จากการจำลองคุณลักษณะของระบบส่วนสมการคณิตศาสตร์ ซึ่งอาจมีความคลาดเคลื่อนได้ นอกจากนี้การวัดค่า Output ของระบบ จะเป็นต้องใช้อุปกรณ์ในการตรวจวัด ซึ่งอาจมีความคลาดเคลื่อนในการตรวจวัดทำให้มีค่าที่แท้จริง โดยสามารถแสดงเป็น State Space Diagram ได้ดังนี้

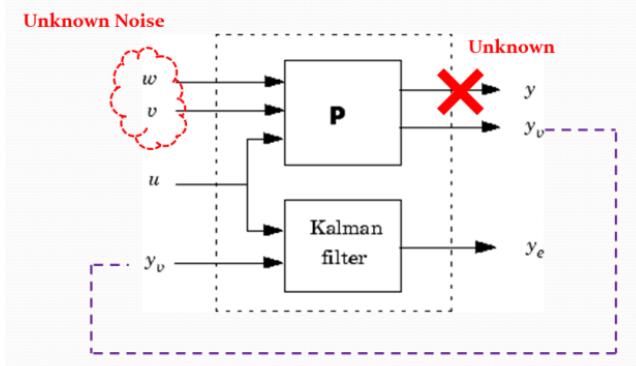


โดย $y_v[n]$ คือค่า Output ที่วัดได้จากอุปกรณ์ตรวจวัด และ $v[n]$ คือความคลาดเคลื่อนหรือสัญญาณรบกวนในอุปกรณ์ตรวจวัด

จากภาพดังกล่าว สามารถยุบรวม System Block และ Sensor Block เข้าด้วยกันเป็น Plant P ที่รับค่าจาก Input $u[n]$ และสัญญาณรบกวน $w[n]$ และ $v[n]$ กับให้ค่า Output ที่วัดได้คือ $y_v[n]$ ซึ่งจากค่าตั้งกล่าว เราจะยังไม่สามารถแยกแยะได้ว่าค่าที่แท้จริงของระบบคืออะไร



Kalman Filter นำเอาค่า Input $u[n]$ และ Measured Output $y_v[n]$ ไปคำนวณเพื่อหาค่าประมาณของ Output หรือ $y_e[n]$ ที่ใกล้เคียงกับค่าจริงมากที่สุดตามภาพ



Kalman Filter ใช้แบบจำลอง System Matrix, Input Matrix และ Output Matrix (A , B และ C) ของระบบ เพื่อใช้ประมาณค่าสถานะต่อไปของระบบ $x_e[n+1]$ และค่าประมาณ Output $y_e[n]$ รวมทั้งใช้ค่า Measured Output $y_v[n]$ เพื่อนำไปคำนวณ Filter Gain ที่สามารถลด Error Covariance ของค่าประมาณ $x_e[n+1]$ ได้ดีที่สุด จากการคำนวณดังนี้

$$x_e[n + 1] = Ax_e[n] + Bu[n] + M(y_v[n] - Cx_e[n])$$

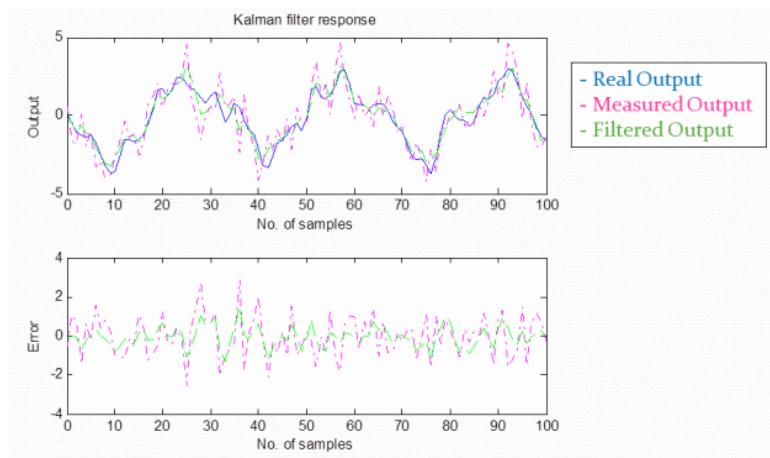
$$y_e[n] = Cx_e[n]$$

$$\text{Filter gain: } M = P[n|n - 1]C'(CP[n|n - 1]C' + R[n])^{-1}$$

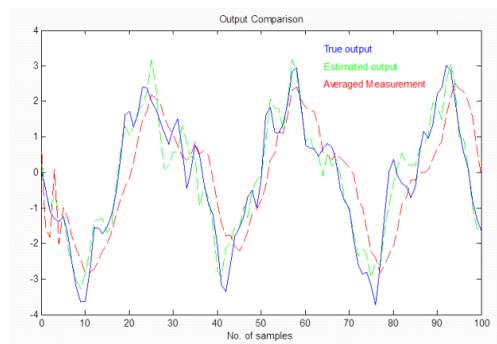
P, R -> error and noise covariance

หลักการอีกอย่างหนึ่งของ Kalman Filter คือการคำนวณแบบทำข้า ระหว่างการประมาณสถานะล่วงหน้า (Predict) กับการใช้ค่าที่วัดได้เพื่อนำมาปรับปรุงค่าประมาณปัจจุบัน (Correct) ด้วยการใช้ Filter Gain M ร่วมกับความแตกต่างระหว่างค่า Output ที่วัดได้กับค่าประมาณ ($y_v[n] - Cx_e[n]$) ซึ่งทำให้ Kalman Filter มีความแม่นยำสูงและทนต่อความคลาดเคลื่อนในระบบ โดยการคำนวณค่า Filter Gain M จะให้น้ำหนักกับค่าที่มีความน่าเชื่อถือมากกว่า (หรือมี Error Covariance ต่ำกว่า)

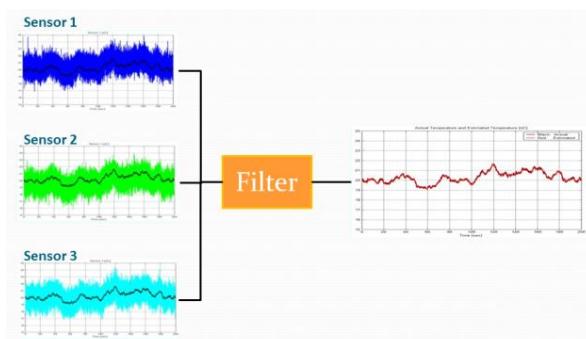
การทดลองเปรียบเทียบค่า Measured Output $y_v[n]$ ที่ได้จากการตรวจวัด กับ Estimated Output $y_e[n]$ ที่ได้จาก Kalman Filter พบร่วมค่า $y_e[n]$ มีค่าใกล้เคียงกับค่า $y[n]$ หรือ Real Output โดยตรงจากระบบมากกว่า ซึ่งในกรณีของการใช้ Kalman Filter ในระบบ Inertial Navigation จะสามารถช่วยให้ค่าที่มีความคลาดเคลื่อนที่จริงน้อยกว่าค่าที่วัดได้จากระบบโดยตรง



เมื่อเปรียบเทียบ Kalman Filter กับ Averaging Filter จะสังเกตได้ว่าผลที่ได้จาก Kalman Filter มีค่าใกล้เคียงกับค่า True Output มากกว่า โดยการใช้ Averaging Filter แสดงให้เห็นผลข้างเคียง คือ การ Smooth Out และ Time Delay ของผลที่ได้



นอกจากการใช้ Kalman Filter ในการประมาณค่าในระบบ Inertial Navigation แล้ว ยังสามารถประยุกต์ใช้ Kalman Filter กับการทำ Data Fusion จากอุปกรณ์ตรวจวัดหลายประเภทได้ โดย Kalman Filter จะคำนวณค่า Filter Gain ที่ให้น้ำหนักกับข้อมูลจากอุปกรณ์ตรวจวัดที่มี Error Covariance ต่ำกว่า ซึ่งจะให้ผลการประมาณ Sensor Output ที่มีความแม่นยำ



2.16 Google Maps API



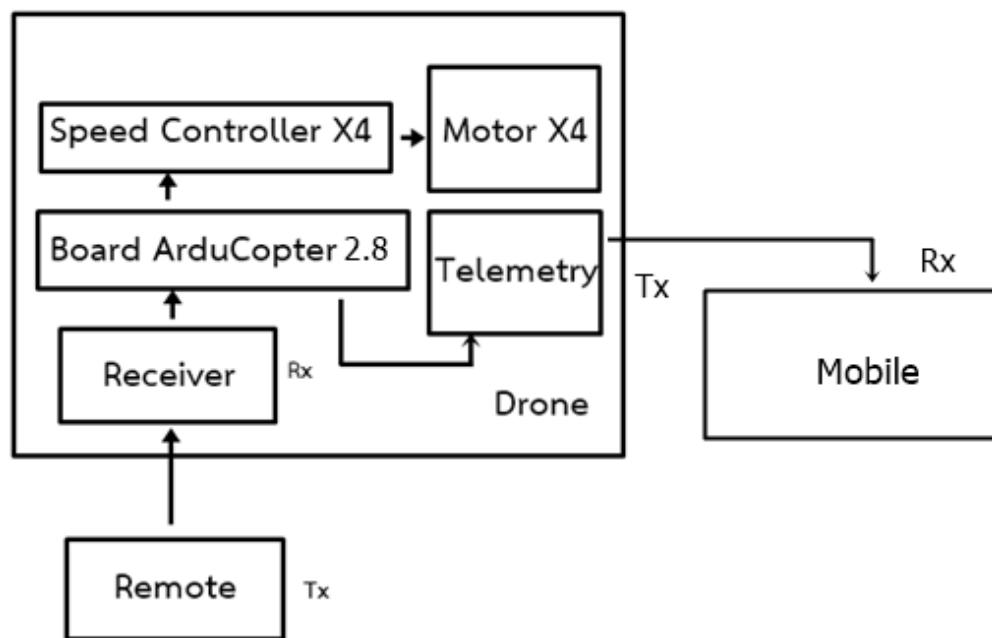
Google Maps API เป็นชุด API ของ Google สำหรับพัฒนา web application และ mobile application (Android, iOS) ไว้สำหรับเรียกใช้แผนที่และชุด service ต่าง ๆ ของ Google เพื่อพัฒนา Application ได้เหมือนกับที่ Google โดยแผนที่ยัง features ต่าง ๆ มากมายให้เรียกใช้

- การปรับแต่งแผนที่ (Styled Map)
- ชุดควบคุมแผนที่ (Map Control)
- ชุดเครื่องมือวาดภาพบนแผนที่ (Drawing)
- การนำทางจากจุดหนึ่งไปยังอีกจุดหนึ่ง (Directions Service)
- การคำนวณความสูงของจุดพิกัด (Elevation Service)
- การแปลงที่อยู่เป็นพิกัด Latitude และ Longitude (GeoCoding Service)
- การดึงข้อมูล POI (Point of Interest) คือข้อมูลสถานที่ต่าง ๆ ที่ Google รวบรวมไว้ให้ เช่น โรงแรม ห้างสรรพสินค้า โรงเรียน สถานที่ราชการต่าง ๆ และสถานที่อื่น ๆ อีกมากมายมาใช้งานในแอปพลิเคชันของเรา
- Street View

บทที่ 3 รายละเอียดการทำงาน

ในบทนี้จะกล่าวถึงรายละเอียดการทำงานในเรื่อง System Architecture, Model Mechanic, Model Controller และ System Implementation

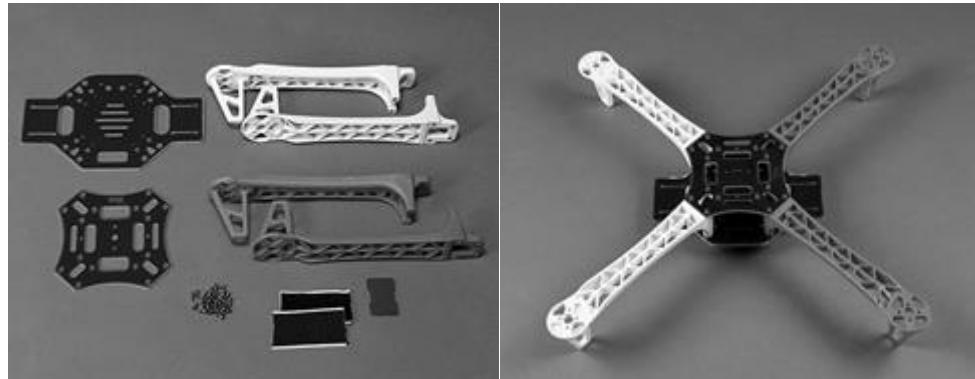
3.1 System Architecture



รูปที่ 3 - 1 แผนภาพ diagram แสดงการทำงานโดยภาพรวมของโดรน

จากรูปที่ 3-1 จะเห็นได้ว่าการทำงานจะเริ่มจากตัว Transmitter โดยผู้ใช้จะทำการควบคุม Drone โดยใส่ input ต่าง ๆ ซึ่งจะทำการส่งผ่าน Antenna ไปที่ตัว Drone ในฝั่งของ Drone ก็จะมี Receiver รับข้อมูลแล้วทำการส่งข้อมูลไปประมวลผลที่ Board ArduPilotMega หลังจากได้ output แล้วก็จะส่งข้อมูลไปที่ Speed Controller และทำการแปลง output นั้นเป็นแรงดันไปขับมอเตอร์แต่ละตัวให้หมุน

3.2 Model Mechanic



รูปที่ 3 - 2 ตัวลำที่ยังไม่ประกอบ

จากรูปที่ 3-2 จะนำเฟรมมาประกอบขึ้นโครงเป็นลำตัวให้ Drone ซึ่งจะเข้มด้วยนอตต่าง ๆ ในแต่ละจุดที่เป็นข้อต่อซึ่งหลังจากประกอบเสร็จแล้วจะได้ดังรูปข้างต้น



รูปที่ 3 - 3 ตัวลำที่ประกอบเรียบร้อยแล้ว

จากรูปที่ 3-3 เป็นการนำไปพัดประกอบเข้ากับมอเตอร์และนำไบคิดติดกับเฟรมที่ประกอบไว้ก่อนหน้านี้ด้วยนอต จากนั้นนำ Speed Controller มาต่อ กับ มอเตอร์แล้วยืดด้วย Cable Tie ให้แน่น

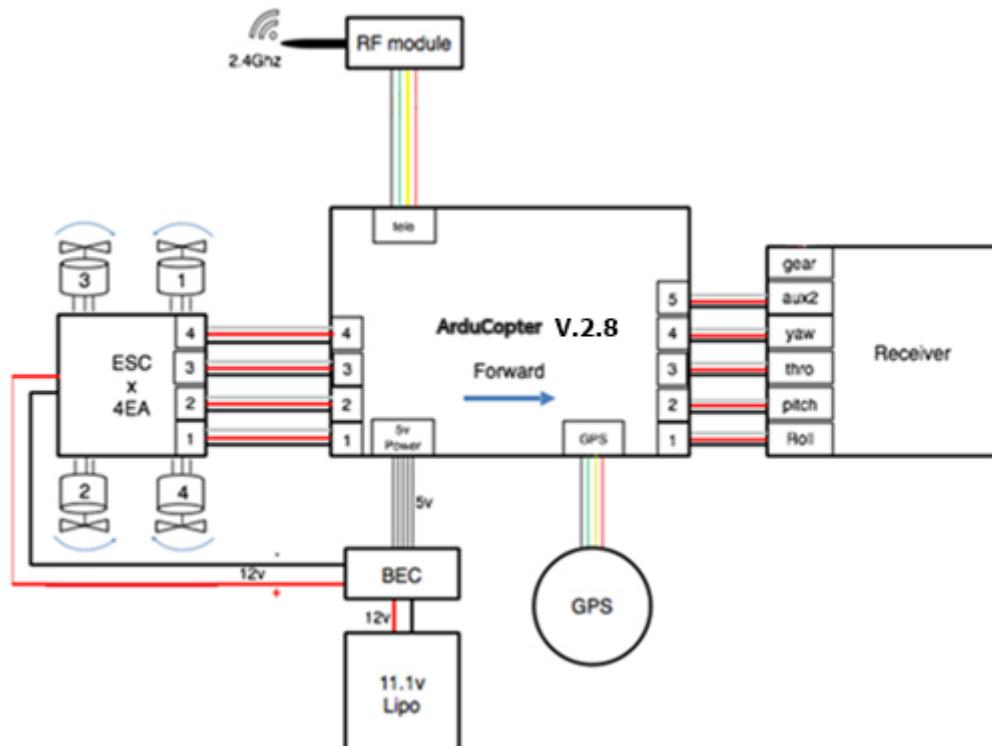


รูปที่ 3 - 4 ตัวลำที่ประกอบอุปกรณ์ครบ

จากรูปที่ 3-4 นำบอร์ดต่อ กับ ลำตัวของ Drone จากนั้นทำการต่อสายจาก Speed Controller เข้า กับ เอ้าต์พุตบอร์ดของ ArduCopter และทำการต่อ Receiver เข้าอินพุตของบอร์ด ArduCopter จากนั้นทำการ ต่อส่วนที่เหลือ เช่น Module GPS , Module แปลงไฟ และอุปกรณ์เสริมต่าง ๆ เข้ากับบอร์ด

3.3 Model Controller

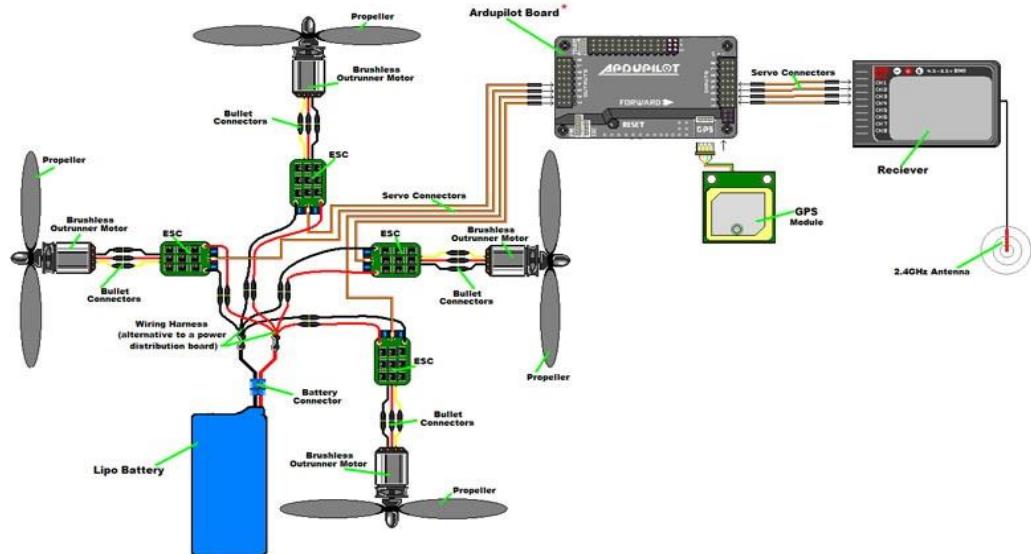
- แผนภาพระบบโดยรวม



รูปที่ 3 - 5 แผนภาพระบบโดยรวม

จากรูปที่ 3-5 เป็นแผนภาพของอุปกรณ์หลัก ๆ ที่มีการเชื่อมต่อกันโดยจะมีการส่งข้อมูลมาจากการ Transmitter โดยที่บอร์ดจะมี Receiver ในการรับข้อมูลนั้น จากนั้นจะทำการส่งข้อมูลไปยังบอร์ด บอร์ดจะทำการประมวลผลที่รับแล้วก็จะส่งข้อมูลต่อไปยัง Speed Controller (ESC) ซึ่งจะทำการแปลงสัญญาณที่ได้รับมาเป็นแรงดันจ่ายให้มอเตอร์ทำงานต่อไป

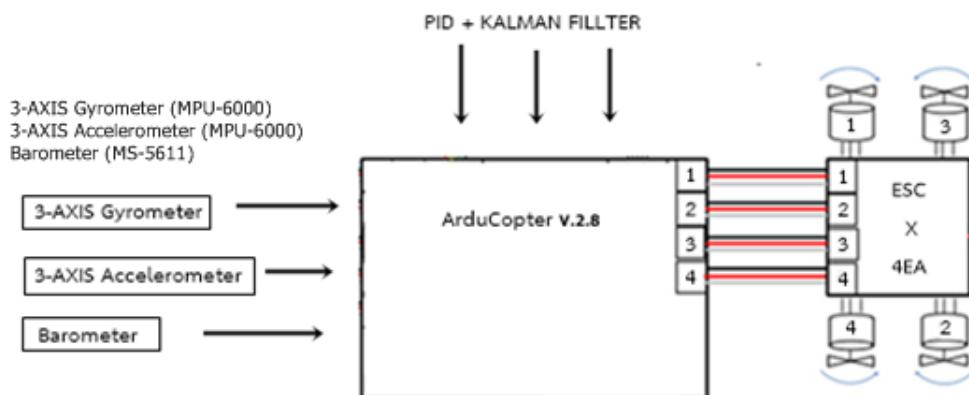
- แผนภาพของระบบโดยละเอียด



รูปที่ 3 - 6 แผนภาพของระบบโดยละเอียด

จากรูปที่ 3-6 เป็นแผนภาพของอุปกรณ์ทั้งหมดที่มี โดยจะมีการเพิ่มในส่วนของอุปกรณ์อยู่ ๆ เช่น โมดูล GPS ตัว telemetry และแบตเตอรี่ เป็นต้น

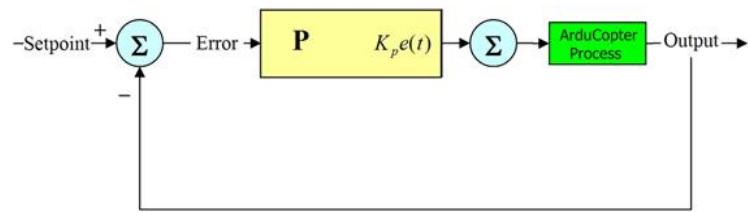
- ระบบควบคุมเสถียรภาพ



รูปที่ 3 - 7 แผนภาพการทำงานของระบบควบคุมเสถียรภาพ

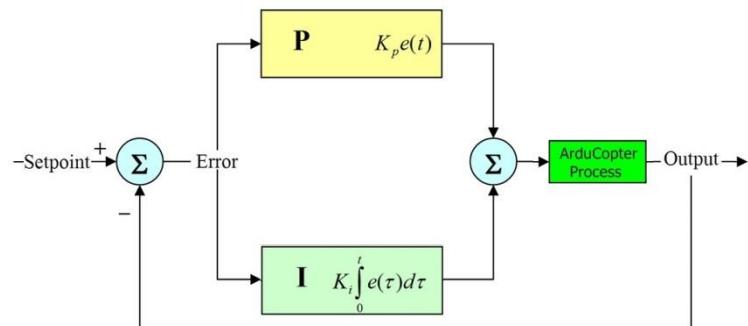
จากรูปที่ 3 - 7 เป็นรูปแบบที่แสดงถึงการนำเข้าเซ็นเซอร์ 3-AIX Gyrometer (MPU-6000), 3-AXIS Accelerometer (MPU-6000) และ Barometer (MS-5611) มาใช้ในการปรับเสถียรภาพในการบินของโดรนโดยอาศัยหลักการเสถียรภาพหรือ PID และ Kalman Filter มาช่วยในการควบคุมการทรงตัวของโดรนเพื่อให้มีประสิทธิภาพสูงสุดในด้านการควบคุมและการทรงตัวที่ดีของโดรน

- ระบบควบคุมแบบพี (P)



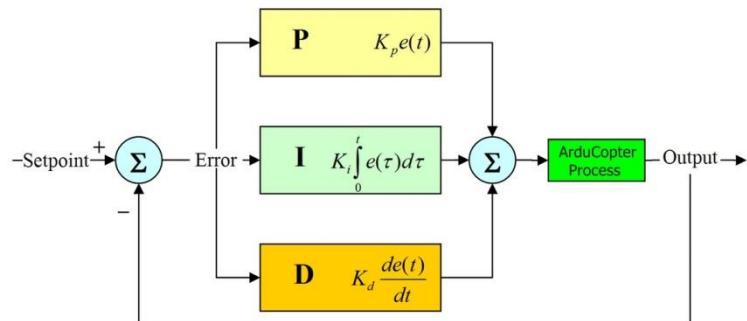
รูปที่ 3 - 8 (a) แผนภาพการทำงานของระบบควบคุมแบบพี

- ระบบควบคุมแบบพีไอ (P+ I)



รูปที่ 3 - 8 (b) แผนภาพการทำงานของระบบควบคุมแบบพีไอ

- ระบบควบคุมแบบพีไอดี (P+ I + D)



รูปที่ 3 - 8 (c) แผนภาพการทำงานของระบบควบคุมแบบพีไอดี

3.3.1 การต่อบอร์ด ARDUCOPTER 2.8



ตารางที่ 3 - 1 การต่อที่บอร์ด Arducopter 2.8

ชนิดของขาต่อ	ช่องเสียบ	อุปกรณ์ที่นำมาต่อ
OUTPUT (Speed Controller)	1	Speed Controller หน้า,ขวา
	2	Speed Controller หลัง,ซ้าย
	3	Speed Controller หน้า,ซ้าย
	4	Speed Controller หลัง,ขวา
INPUT (Receiver)	1	Roll
	2	Pitch
	3	Throttle
	4	Yaw
	5	Aux
GPS	GPS	GPS Module
I2C	I2C	Compass Module
PM	PM	3DR Power Module
Telemetry Radio	Telemetry Radio	Telemetry Module

จากตารางที่ 3-1 แสดงการต่ออุปกรณ์โดยละเอียดของบอร์ด Arducopter 2.8

3.3.2 การต่อ Receiver Flysky FS-R9B



ตารางที่ 3 - 2 การต่อที่ Receiver Flysky FS-R9B

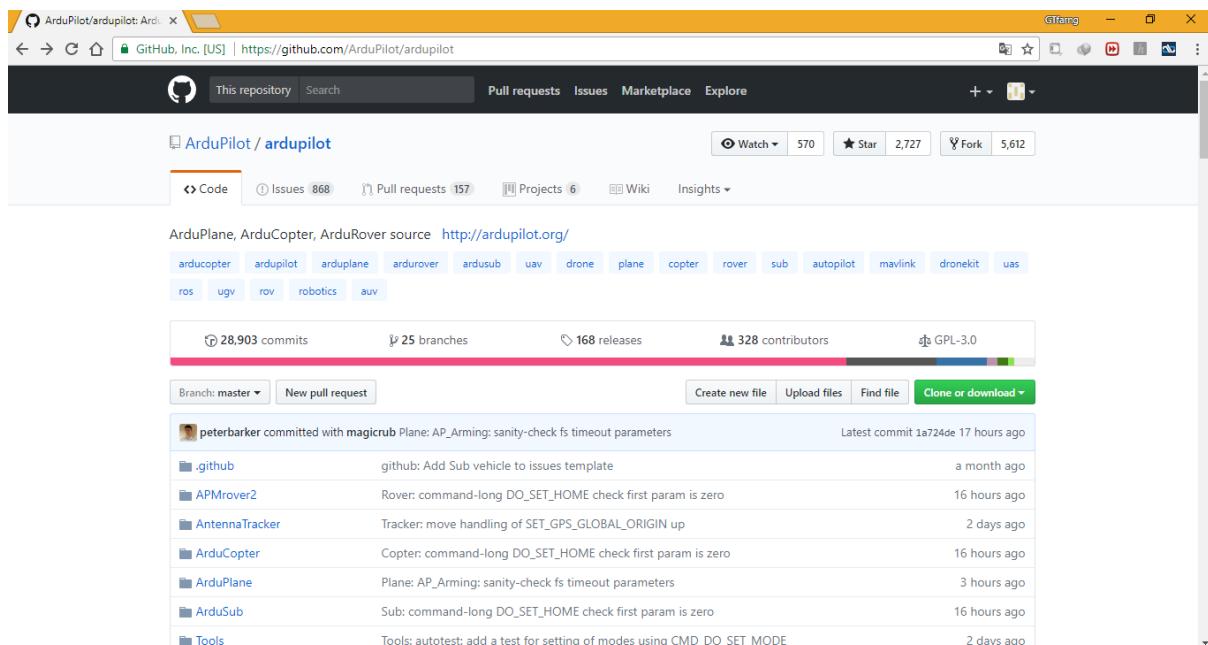
BIND – จุนหาสัญญาณเข้ากับรีโมท	Signal (S)	VCC (+)	Ground (-)
BAT – แบตเตอรี่ (4.5V - 6.5V)	Signal (S)	VCC (+)	Ground (-)
CH8 – ช่องสัญญาณ 8	Signal (S)	VCC (+)	Ground (-)
CH7 – ช่องสัญญาณ 7	Signal (S)	VCC (+)	Ground (-)
CH6 – ช่องสัญญาณ 6	Signal (S)	VCC (+)	Ground (-)
CH5 – ช่องสัญญาณ 5	Signal (S)	VCC (+)	Ground (-)
CH4 – ช่องสัญญาณ 4	Signal (S)	VCC (+)	Ground (-)
CH3 – ช่องสัญญาณ 3	Signal (S)	VCC (+)	Ground (-)
CH2 – ช่องสัญญาณ 2	Signal (S)	VCC (+)	Ground (-)
CH1 – ช่องสัญญาณ 1	Signal (S)	VCC (+)	Ground (-)

จากตารางที่ 3-2 แสดงการต่ออุปกรณ์โดยละเอียดของ Receiver Flysky FS-R9B

3.4 System Implementation

- การนำ Code ArduPilot มาใช้งานกับบอร์ด ArduPilotMega

ในส่วนของการนำ code มาใช้งานกับบอร์ด APM 2.8 จะใช้ Github ในการดึง code จากผู้พัฒนามา ทำความสะอาดเข้าใจและปรับปรุง code ให้ดีขึ้น



รูปที่ 3 - 9 เว็บ GitHub

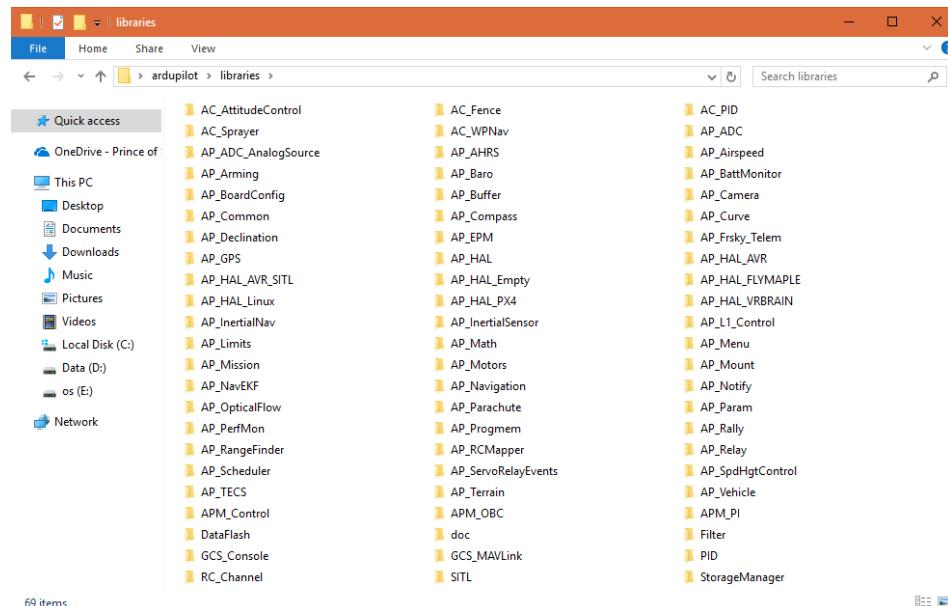
จากรูปที่ 3-9 เป็นหน้าเว็บ GitHub ของโค้ด ArduPilot สามารถดาวน์โหลดลงเครื่องโดยการกดที่ Clone or download

```
MINGW32:/c/GIT/Ardupilot1
$ git clone git://github.com/ArduPilot/ardupilot.git
Cloning into 'ardupilot'...
remote: Counting objects: 195527, done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 195527 (delta 13), reused 14 (delta 8), pack-reused 195490
Receiving objects: 100% (195527/195527), 89.96 MiB | 25.00 KiB/s, done.
Resolving deltas: 100% (142477/142477), done.
Checking out files: 100% (2664/2664), done.

MINGW32:/c/GIT/Ardupilot1
$
```

รูปที่ 3 - 10 คำสั่งในการดาวน์โหลด code

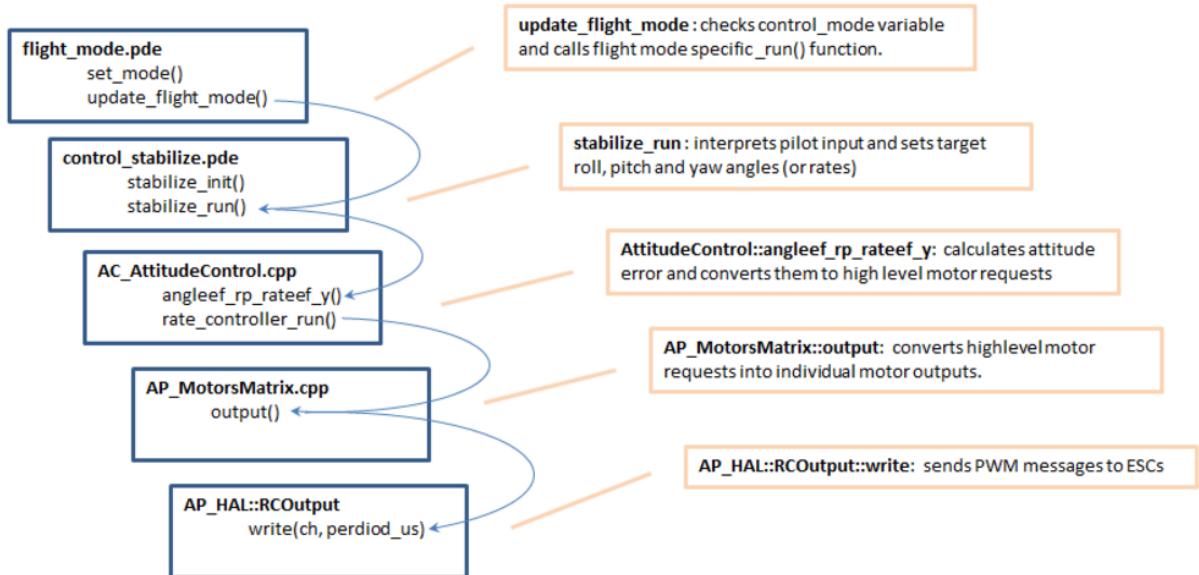
จากรูปที่ 3-10 เป็นการใช้คำสั่ง Git เพื่อโหลด code ลงมาเว็บไว้ในเครื่อง



รูปที่ 3 - 11 Library ของ code ArduPilot

จากรูปที่ 3-11 เป็นส่วนของ library ที่มีให้เลือกใช้งานกับบอร์ด ArduCopter

- ระบบการทำงานโดยรวมของบอร์ด APM



รูปที่ 3 - 12 Diagram ระบบการทำงานโดยรวมของบอร์ด APM

จากรูปที่ 3-12 จะเป็นระบบการทำงานโดยรวมของบอร์ด APM ขณะทำการบิน โดยโปรแกรมจะเริ่มตรวจสอบ Flight Mode ก่อนว่าเป็นโหมดใด ในที่นี้จะใช้โหมด Stabilize หรือโหมดคงตัวเพื่อทำการปรับแต่งค่า PID ให้สมบูรณ์ หลังจากนั้นฟังก์ชัน `control_stabilize` จะเริ่มทำงาน ฟังก์ชันนี้ก็ส่งค่าไปฟังก์ชัน `AC_Attitude Control` เพื่อทำการควบคุมรวมถึงอ่านค่าจากเซนเซอร์ อ่านค่าจากวิทยุ และนำไปยัง PID เพื่อคำนวณและส่งออกมาเป็น `output` ไปยัง `AP_MotorMatrix` ฟังก์ชันนี้จะทำการแปลงค่า `output` ที่ต้องการสั่งมอเตอร์เป็นค่าที่มอเตอร์ต้องการ และฟังก์ชันสุดท้ายจะเป็นฟังก์ชัน `AP_HAL` ฟังก์ชันนี้จะเป็นฟังก์ชันในระบบฮาร์ดแวร์จะทำการส่งสัญญาณเป็น PWM ไปยัง ESCs

- ลำดับการทำงานพื้นฐานของ code ArduCopter

- ไฟล์ flight_mode.cpp

```

bool Copter::set_mode(control_mode_t mode, mode_reason_t reason)
{
    // boolean to record if flight mode could be set
    bool success = false;
    bool ignore_checks = !motors.armed(); // allow switching to any mode if disarmed. We rely on the arming check to perform

    // return immediately if we are already in the desired mode
    if (mode == control_mode) {
        prev_control_mode = control_mode;
        prev_control_mode_reason = control_mode_reason;

        control_mode_reason = reason;
        return true;
    }

    switch(mode) {

        case STABILIZE:
            #if FRAME_CONFIG == HELI_FRAME
                success = heli_stabilize_init(ignore_checks);
            #else
                success = stabilize_init(ignore_checks);
            #endif
            break;
    }

    // update flight mode
    if (success) {
        // perform any cleanup required by previous flight mode
        exit_mode(control_mode, mode);

        prev_control_mode = control_mode;
        prev_control_mode_reason = control_mode_reason;

        control_mode = mode;
        control_mode_reason = reason;
        DataFlash.Log_Write_Mode(control_mode, control_mode_reason);

        #if AC_FENCE == ENABLED
            // pilot requested flight mode change during a fence breach indicates pilot is attempting to manually recover
            // this flight mode change could be automatic (i.e. fence, battery, GPS or GCS failsafe)
            // but it should be harmless to disable the fence temporarily in these situations as well
            fence.manual_recovery_start();
        #endif
    }

    void Copter::update_flight_mode()
    {
        // Update EKF speed limit - used to limit speed when we are using optical flow
        ahrs.getEkfControlLimits(ekfGndSpdLimit, ekfNavVelGainScaler);

        switch (control_mode) {

            case STABILIZE:
                #if FRAME_CONFIG == HELI_FRAME
                    heli_stabilize_run();
                #else
                    stabilize_run();
                #endif
                break;
        }
    }
}

```

Code ข้างต้นเป็นการเริ่มต้นเข้าสู่โหมดการทำงานต่าง ๆ โดยจะทำการตรวจสอบการทำงานก่อนว่ามีการ ARM หรือไม่ และเมื่อมีการ ARM ให้ทำการปรับปรุงพารามิเตอร์เป็นโหมดในปัจจุบันและเริ่มเข้าสู่โหมดการทำงานของ Stabilize ทันทีและมีการเรียกใช้งานไฟล์ Control_stabilize เป็นลำดับการทำงานขั้นต่อไป

2. ไฟล์ Control_stabilize.cpp

```
// stabilize_init - initialise stabilize controller
bool Copter::stabilize_init(bool ignore_checks)
{
    // if landed and the mode we're switching from does not have manual throttle and the throttle stick is too high
    if (motors.armed() && ap.land_complete && !mode_has_manual_throttle(control_mode) && (get_pilot_desired_throttle(channel_throttle->control_in) > get_non_takeoff_throttle())) {
        return false;
    }
    // set target altitude to zero for reporting
    pos_control.set_alt_target(0);

    return true;
}

// stabilize_run - runs the main stabilize controller
// should be called at 100hz or more
void Copter::stabilize_run()
{
    float target_roll, target_pitch;
    float target_yaw_rate;
    float pilot_throttle_scaled;

    // if not armed set throttle to zero and exit immediately
    if (!motors.armed() || ap.throttle_zero || !motors.get_interlock()) {
        motors.set_desired_spool_state(AP_Motors::DESIRED_SPIN_WHEN_ARMED);
        attitude_control.set_throttle_out_unstabilized(0,true,g.throttle_filt);
        return;
    }

    // call attitude controller
    attitude_control.input_euler_angle_roll_pitch_euler_rate_yaw_smooth(target_roll,
    target_pitch, target_yaw_rate, get_smoothing_gain());
```

Code ข้างต้นเป็นการทำงานโดยเริ่มต้นของโหมด stabilize โดยจะมีตัวแปรรับค่าจากเริ่มที่ roll, pitch, yaw และ throttle และจะทำการตรวจสอบว่ามีการ arm แล้วหรือไม่ หากมีการ arm แล้วคือพร้อมบิน ก็จะ initial motor ให้เริ่มรับคำสั่งและเข้าสู่ simple_mode ในโหมดนี้จะเป็นโหมดพื้นฐานรองรับมาที่ไม่ต้องใช้ตัวอื่น ๆ จะทำการตามคำสั่งของเริ่มทอย่างเดียวในการเคลื่อนที่แบบพื้นฐานต่าง ๆ เช่น การเอียงซ้าย เอียงขวา เป็นต้น และมีการเรียกใช้งานไฟล์ AC_AttitudeControl เป็นลำดับการทำงานขั้นต่อไป

3. ไฟล์ AC_AttitudeControl.cpp

```
void AC_AttitudeControl::relax_bf_rate_controller()
{
    // Set reference angular velocity used in angular velocity controller equal
    // to the input angular velocity and reset the angular velocity integrators.
    // This zeros the output of the angular velocity controller.
    _ang_vel_target_rads = _ahrs.get_gyro();
    get_rate_roll_pid().reset_I();
    get_rate_pitch_pid().reset_I();
    get_rate_yaw_pid().reset_I();

    // Write euler derivatives derived from vehicle angular velocity to
    // _att_target_euler_rate_rads. This resets the state of the input shapers.
    ang_vel_to_euler_rate(Vector3f(_ahrs.roll,_ahrs.pitch,_ahrs.yaw), _ang_vel_target_rads, _att_target_euler_rate_rads);
}
```

Code ข้างต้นเป็นการทำงานของการคำนวณค่าต่าง ๆ ที่รับมาจากค่า roll, pitch และ yaw รวมถึงการนำฟังก์ชัน P, PI และ PID รวมถึง Kalman Filter มาประกอบเพื่อช่วยในการคำนวณ จากนั้นทำการปรับปรุงค่าในพารามิเตอร์ให้เป็นปัจจุบันและทำการเรียกใช้งาน AP_MotorMatrix เป็นลำดับการทำงานขั้นต่อไป

ในส่วนของการนำ K_p , K_i และ K_d มาใช้งานนั้นสามารถทำได้โดยการเข็ตค่าเริ่มต้นก่อน ซึ่งการเข็ตค่าเริ่มต้นนั้นจะมีการทำงานทั้ง 3 รูปแบบ คือ จะมีการทำงานแบบ P การทำงานแบบ PI และ การทำงานแบบ PID ไฟล์ที่ใช้ในการเข็ตค่าจะมีไฟล์ AC_P.cpp สำหรับการทำงานแบบ P, AC_PI_2D.cpp สำหรับการทำงานแบบ PI และ AC_PID.cpp สำหรับการทำงานแบบ PID หลังจากนั้นก็จะทำการเรียกการใช้งาน PID ซึ่งเป็นการประมวลค่าต่าง ๆ ที่ได้ทำการส่งจากไฟล์ข้างต้น

- การใช้งาน code ในส่วนของ PID

1. การเข็ตค่าเริ่มต้นของ PID

```

void operator() (const float p,
                  const float i,
                  const float d,
                  const int16_t imaxval) {
    _kp = p; _ki = i; _kd = d; _imax = imaxval;
}

float kP() const {
    return _kp.get();
}
float kI() const {
    return _ki.get();
}
float kD() const {
    return _kd.get();
}
int16_t imax() const {
    return _imax.get();
}

void kP(const float v) {
    _kp.set(v);
}
void kI(const float v) {
    _ki.set(v);
}
void kD(const float v) {
    _kd.set(v);
}

```

2. ส่วนของ Filter จะเป็น library ที่จะ filter ของทุกแกนก่อนจะไปเข้า PID

```

void AC_PID::set_input_filter_all(float input)
{
    if (!isfinite(input)) {
        return;
    }

    if (_flags._reset_filter) {
        _flags._reset_filter = false;
        _input = input;
        _derivative = 0.0f;
    }

    float input_filt_change = get_filt_alpha() * (input - _input);
    _input = _input + input_filt_change;
    if (_dt > 0.0f) {
        _derivative = input_filt_change / _dt;
    }
}

```

3. Code การทำงานในส่วนของ PID

```

float PID::get_pid(float error, float scaler)
{
    uint32_t tnow = AP_HAL::millis();
    uint32_t dt = tnow - _last_t;
    float output      = 0;
    float delta_time;

    if (_last_t == 0 || dt > 1000) {
        dt = 0;
        reset_I();
    }
    _last_t = tnow;

    delta_time = (float)dt / 1000.0f;

    // Compute proportional component
    output += error * _kp;

    // Compute derivative component if time has elapsed
    if ((fabsf(_kd) > 0) && (dt > 0)) {
        float derivative;

        if (isnan(_last_derivative)) {
            // we've just done a reset, suppress the first derivative
            // term as we don't want a sudden change in input to cause
            // discrete low pass filter, cuts out the
            // high frequency noise that can drive the controller crazy
            float RC = 1/(2*M_PI*_fCut);
            derivative = _last_derivative +
                ((delta_time / (RC + delta_time)) *
                 (derivative - _last_derivative));
        }

        // update state
        _last_error      = error;
        _last_derivative = derivative;

        // add in derivative component
        output           += _kd * derivative;
    }

    // scale the P and D components
    output *= scaler;

    // Compute integral component if time has elapsed
    if ((fabsf(_ki) > 0) && (dt > 0)) {
        _integrator      += (error * _ki) * scaler * delta_time;
        if (_integrator < -_imax) {
            _integrator = -_imax;
        } else if (_integrator > _imax) {
            _integrator = _imax;
        }
        output           += _integrator;
    }

    return output;
}

```

5. เมื่อได้ค่ามาแล้วระบบจะทำการ update simple mode คือโหมดในการคำสั่งอย่างง่าย

```
void Copter::update_simple_mode(void)
{
    float rollx, pitchx;

    if (ap.simple_mode == 0 || !ap.new_radio_frame) {
        return;
    }

    // mark radio frame as consumed
    ap.new_radio_frame = false;

    if (ap.simple_mode == 1) {

        rollx = channel_roll->control_in*simple_cos_yaw - channel_pitch->control_in*simple_sin_yaw;
        pitchx = channel_roll->control_in*simple_sin_yaw + channel_pitch->control_in*simple_cos_yaw;
    }else{

        rollx = channel_roll->control_in*super_simple_cos_yaw - channel_pitch->control_in*super_simple_sin_yaw;
        pitchx = channel_roll->control_in*super_simple_sin_yaw + channel_pitch->control_in*super_simple_cos_yaw;
    }

    channel_roll->control_in = rollx*ahrs.cos_yaw() + pitchx*ahrs.sin_yaw();
    channel_pitch->control_in = -rollx*ahrs.sin_yaw() + pitchx*ahrs.cos_yaw();
}
```

4.ไฟล์ AP_MotorMatrix.cpp

```
void AP_MotorsMatrix::Init()
{
    // setup the motors
    setup_motors();

    // enable fast channels or instant pwm
    set_update_rate(_speed_hz);
}

void AP_MotorsMatrix::output_to_motors()
{
    int8_t i;
    int16_t motor_out[AP_MOTORS_MAX_NUM_MOTORS]; // final pwm values sent to the motor

    switch (_multicopter_flags.spool_mode) {
        case SHUT_DOWN:
            // sends minimum values out to the motors
            // set motor output based on thrust requests
            for (i=0; i<AP_MOTORS_MAX_NUM_MOTORS; i++) {
                if (motor_enabled[i]) {
                    motor_out[i] = _throttle_radio_min;
                }
            }
            break;
        case SPIN_WHEN_ARMED:
            // sends output to motors when armed but not flying
            for (i=0; i<AP_MOTORS_MAX_NUM_MOTORS; i++) {
                if (motor_enabled[i]) {
                    motor_out[i] = constrain_int16(_throttle_radio_min + _throttle_low_end_pct
                        * _min_throttle, _throttle_radio_min, _throttle_radio_min + _min_throttle);
                }
            }
            break;
    }
}
```

```

        case SPOOL_UP:
        case THROTTLE_UNLIMITED:
        case SPOOL_DOWN:
            // set motor output based on thrust requests
            for (i=0; i<AP_MOTORS_MAX_NUM_MOTORS; i++) {
                if (motor_enabled[i]) {
                    motor_out[i] = calc_thrust_to_pwm(_thrust_rpyt_out[i]);
                }
            }
            break;
        }

        // send output to each motor
        hal.rcout->cork();
        for (i=0; i<AP_MOTORS_MAX_NUM_MOTORS; i++) {
            if (motor_enabled[i]) {
                rc_write(i, motor_out[i]);
            }
        }
        hal.rcout->push();
    }
}

```

5.

Code ข้างต้นเป็นการทำงานของการคำนวนคลื่น PWM เพื่อส่งให้มอเตอร์ทำการบินแบบต่าง ๆ ซึ่งการบินในแต่ละแบบมอเตอร์จะมีการทำงานที่แตกต่างกันไป โดยจะมีการแปลงค่าต่าง ๆ โดยมีการแปลงค่าที่ส่งมาจาก AC_AttitudeControl เพื่อแปลงเป็นข้อมูล PWM เพื่อทำการเรียกใช้ AP_HAL เป็นลำดับการทำงานต่อไป

6. ไฟล์ AP_HAL.cpp

```

/*
 * Output a single channel, possibly grouped with previous writes if
 * cork() has been called before.
 */
virtual void    write(uint8_t ch, uint16_t period_us) = 0;

```

Code ข้างต้นเป็นการทำงานโดยการเรียกใช้งานฟังก์ชันเพื่อทำการส่งข้อมูล PWM เพื่อไปสู่การทำงานที่ ESC ให้เริ่มการมอเตอร์ทำงาน

- ไฟล์ที่เกี่ยวข้องการปรับแต่ง PID และ Kalman Filter

1. ไฟล์ AC_P.cpp

```

const AP_Param::GroupInfo AC_P::var_info[] = {
    // @Param: P
    // @DisplayName: PI Proportional Gain
    /* @Description: P Gain which produces an output value
     * that is proportional to the current error value*/
    AP_GROUPINFO("P",      0, AC_P, _kp, 0),
    AP_GROUPEnd
};

float AC_P::get_p(float error) const
{
    return (float)error * _kp;
}

void AC_P::load_gains()
{
    _kp.load();
}

```

Code ข้างต้นเป็นการคืนค่า Error ของ K_p เพื่อที่จะทำการโหลดไปประมวลผลในไฟล์ PID.cpp สำหรับการทำงานเฉพาะแบบพี (แบบสัดส่วน)

2. ไฟล์ AC_PI_2D.cpp

```
const AP_Param::GroupInfo AC_PI_2D::var_info[] = {
    // @Param: P
    // @DisplayName: PID Proportional Gain
    /* @Description: P Gain which produces an output
    value that is proportional to the current error value*/
    AP_GROUPINFO("P",     0, AC_PI_2D, _kp, 0),

    // @Param: I
    // @DisplayName: PID Integral Gain
    /* @Description: I Gain which produces an output
    that is proportional to both the magnitude and the duration of the error*/
    AP_GROUPINFO("I",     1, AC_PI_2D, _ki, 0),

    AP_GROUPEnd
};

Vector2f AC_PI_2D::get_pi()
{
    return get_p() + get_i();
}

void AC_PI_2D::load_gains()
{
    _kp.load();
    _ki.load();
    _imax.load();
    _imax = fabsf(_imax);
    _filt_hz.load();

    // calculate the input filter alpha
    calc_filt_alpha();
}
```

Code ข้างต้นเป็นการคืนค่า Error ของ K_p และ K_i เพื่อที่จะทำการโหลดไปประมวลผลในไฟล์ PID.cpp สำหรับการทำงานเฉพาะแบบพีไอ (แบบสัดส่วน+ปริพันธ์)

3. ไฟล์ AC_PID.cpp

```
const AP_Param::GroupInfo AC_PID::var_info[] = {
    // @Param: P
    // @DisplayName: PID Proportional Gain
    // @Description: P Gain which produces an output
    // value that is proportional to the current error value
    AP_GROUPINFO("P",     0, AC_PID, _kp, 0),

    // @Param: I
    // @DisplayName: PID Integral Gain
    /* @Description: I Gain which produces an output that is
    proportional to both the magnitude and the duration of the error*/
    AP_GROUPINFO("I",     1, AC_PID, _ki, 0),

    // @Param: D
    // @DisplayName: PID Derivative Gain
    /* @Description: D Gain which produces an output that is
    proportional to the rate of change of the error*/
    AP_GROUPINFO("D",     2, AC_PID, _kd, 0),

    AP_GROUPEnd
};
```

```

float AC_PID::get_pid()
{
    return get_p() + get_i() + get_d();
}

void AC_PID::load_gains()
{
    _kp.load();
    _ki.load();
    _kd.load();
    _imax.load();
    _imax = fabsf(_imax);
    _filt_hz.load();
}

```

Code ข้างต้นเป็นการคืนค่า Error ของ K_p , K_i และ K_d เพื่อที่จะทำการโหลดไปประมวลผลในไฟล์ PID.cpp สำหรับการทำงานเฉพาะแบบพีไอดี (แบบสัดส่วน+ปริพันธ์+อนุพันธ์)

4. ไฟล์ PID.cpp

```

float PID::get_pid(float error, float scaler)
{
    uint32_t tnow = AP_HAL::millis();
    uint32_t dt = tnow - _last_t;
    float output      = 0;
    float delta_time;

    if (_last_t == 0 || dt > 1000) {
        dt = 0;

        // if this PID hasn't been used for a full second then zero
        // the integrator term. This prevents I buildup from a
        // previous fight mode from causing a massive return before
        // the integrator gets a chance to correct itself
        reset_I();
    }
    _last_t = tnow;

    delta_time = (float)dt / 1000.0f;

    // Compute proportional component
    output += error * _kp;

    // Compute derivative component if time has elapsed
    if ((fabsf(_kd) > 0) && (dt > 0)) {
        float derivative;

        if (isnan(_last_derivative)) {
            // we've just done a reset, suppress the first derivative
            // term as we don't want a sudden change in input to cause
            // a large D output change
            derivative = 0;
            _last_derivative = 0;
        } else {
            derivative = (error - _last_error) / delta_time;
        }

        // discrete low pass filter, cuts out the
        // high frequency noise that can drive the controller crazy
        float RC = 1/(2*M_PI*fCut);
        derivative = _last_derivative +
                    ((delta_time / (RC + delta_time)) *
                     (derivative - _last_derivative));

        // update state
        _last_error      = error;
        _last_derivative = derivative;

        // add in derivative component
        output          += _kd * derivative;
    }

    // scale the P and D components
    output *= scaler;
}

```

```

// scale the P and D components
output *= scaler;

// Compute integral component if time has elapsed
if ((fabsf(_ki) > 0) && (dt > 0)) {
    _integrator += (error * _ki) * scaler * delta_time;
    if (_integrator < -_imax) {
        _integrator = -_imax;
    } else if (_integrator > _imax) {
        _integrator = _imax;
    }
    output += _integrator;
}

return output;
}

```

Code ข้างต้นเป็นการทำงานของพีไอดี โดยจะทำการประมวลผลค่าที่ต่าง ๆ ที่ได้รับจากการทำงานในก่อนหน้านี้

5. ไฟล์ AP_NavEK2.cpp

โดยไฟล์ AP_NavEK2.cpp เป็นไฟล์ในส่วนของการทำงานของ Kalman Filter ซึ่งหลังจากทำการประมวลผลแล้วก็จะทำการส่งค่าไปยัง AP_HAL เพื่อทำการส่งค่าให้ ESC สร้างคลื่น PWM สั่งให้มอเตอร์ทำงาน

- ส่วนของ Code ที่ได้ทำการปรับปรุง

ไฟล์ my_AC_AttitudeControl.cpp

```

void AC_AttitudeControl::relax_bf_rate_controller()
{
    _ang_vel_target_rads = _ahrs.get_gyro();
    get_rate_roll_pid().reset_I();
    get_rate_pitch_pid().reset_I();
    get_rate_yaw_pid().reset_I();

    ang_vel_to_euler_rate(Vector3f(_ahrs.roll,_ahrs.pitch,_ahrs.yaw), _ang_vel_target_rads, _att_target_euler_rate_rads);
}

float AC_AttitudeControl::rate_bf_to_motor_roll(float rate_target_rads)
{
    float current_rate_rads = _ahrs.get_gyro().x;
    float rate_error_rads = rate_target_rads - current_rate_rads;

    get_rate_roll_pid().set_input_filter_d(rate_error_rads);
    get_rate_roll_pid().set_desired_rate(rate_target_rads);

    float integrator = get_rate_roll_pid().get_integrator();

    if (!_motors.limit.roll_pitch || ((integrator > 0 && rate_error_rads < 0) || (integrator < 0 && rate_error_rads > 0))) {
        integrator = get_rate_roll_pid().get_i();
    }

    float output = get_rate_roll_pid().get_p() + integrator + get_rate_roll_pid().get_d();

    return constrain_float(output, -1.0f, 1.0f);
}

float AC_AttitudeControl::max_rate_step_bf_roll()
{
    float alpha = get_rate_roll_pid().get_filt_alpha();
    float alpha_remaining = 1-alpha;
    return AC_ATTITUDE_RATE_RP_CONTROLLER_OUT_MAX/((alpha_remaining*alpha_remaining*alpha_remaining*alpha*get_rate_roll_pid().kD())/_dt + get_rate_roll_pid().kP());
}

```

```

float AC_AttitudeControl::rate_bf_to_motor_pitch(float rate_target_rads)
{
    float current_rate_rads = _ahrs.get_gyro().y;
    float rate_error_rads = rate_target_rads - current_rate_rads;

    get_rate_pitch_pid().set_input_filter_d(rate_error_rads);
    get_rate_pitch_pid().set_desired_rate(rate_target_rads);

    float integrator = get_rate_pitch_pid().get_integrator();

    if (!motors.limit.roll_pitch || ((integrator > 0 && rate_error_rads < 0) || (integrator < 0 && rate_error_rads > 0))) {
        integrator = get_rate_pitch_pid().get_i();
    }

    float output = get_rate_pitch_pid().get_p() + integrator + get_rate_pitch_pid().get_d();

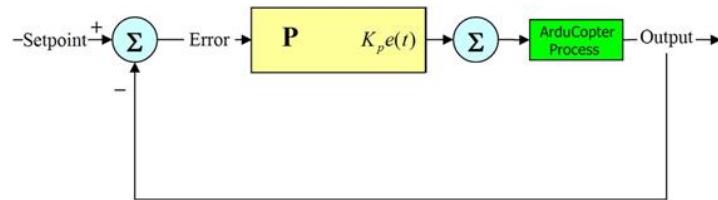
    return constrain_float(output, -1.0f, 1.0f);
}

float AC_AttitudeControl::max_rate_step_bf_pitch()
{
    float alpha = get_rate_pitch_pid().get_filt_alpha();
    float alpha_remaining = 1-alpha;
    return AC_ATTITUDE_RATE_RP_CONTROLLER_OUT_MAX*((alpha_remaining*alpha_remaining*alpha*get_rate_pitch_pid().kD())/_dt + get_rate_pitch_pid().kP());
}

```

Code ข้างต้นเป็นการปรับปรุงไฟล์ที่ใช้สำหรับเรียกการทำงานของระบบควบคุมแบบพี ระบบควบคุมแบบพีโอลและระบบควบคุมแบบพีโอดี โดยจะเริ่มตั้งแต่การรีเซ็ตค่าของระบบ การโหลดค่าของระบบมาใช้งานและการส่งค่าเข้าสู่ระบบ เพื่อให้มีการทำงานในลำดับต่อไป

การทำงานของระบบควบคุมแบบพี (ปรับเฉพาะค่าสัดส่วนหรือ K_p)



ไฟล์ my_P.cpp เป็นไฟล์ที่ใช้ในระบบควบคุมแบบพีที่ชึ้นได้จากการปรับปรุง

```
#define roll_P 0.15f
#define pitch_P 0.125f
```

set ค่าของ K_p ของแกน roll และแกน pitch

```

float AC_P::get_roll(float error) const
{
    return (float)error * roll_P;
}

void AC_P::load_gains_roll()
{
    roll_P.load();
}

void AC_P::save_gains_roll()
{
    roll_P.save();
}
```

การทำงานของระบบควบคุมแบบพีของแกน roll

```

float AC_P::get_pitch(float error) const
{
    return (float)error * pitch_P;
}

void AC_P::load_gains_pitch()
{
    pitch_P.load();
}

void AC_P::save_gains_pitch()
{
    pitch_P.save();
}
```

การทำงานของระบบควบคุมแบบพีของแกน pitch

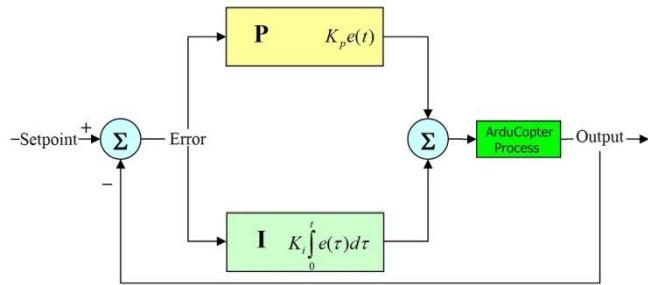
```

void AC_P::get_p()
{
    return get_p();
}
```

คืนค่าการทำงานของระบบควบคุมแบบพี

Code ข้างต้นเป็นการปรับปรุงไฟล์ซึ่งใช้สำหรับกำหนดค่าเริ่มต้นสำหรับการทำงานของระบบควบคุมแบบพีด้วยจะมีการหาค่า error หลังจากนั้นก็จะทำการโหลดค่าและทำการบันทึกค่าเพื่อนำไปใช้สำหรับการทำงานในกระบวนการต่อ ๆ ไปโดยการ return

การทำงานของระบบควบคุมแบบพีไอ (ปรับค่าสัดส่วนและปริพันธ์ หรือ K_p และ K_i)



ไฟล์ my_PI.cpp เป็นไฟล์ที่ใช้ในระบบควบคุมแบบพีไอที่ชึ้นได้จากการปรับปรุง

```

#define roll_P 0.1f
#define roll_I 0.25f
#define roll_IMAX 100

#define pitch_P 0.1f
#define pitch_I 0.23f
#define pitch_IMAX 100
  
```

set ค่าของ K_p และ K_i ของแกน roll และแกน pitch

```

void AC_PI::operator_roll() (float p, float i, float imaxval, float input_filt_hz, float dt)
{
    roll_kp = p;
    roll_ki = i;
    roll_imax = fabsf(imaxval);
    roll_filt_hz = input_filt_hz;      // กำหนดพารามิเตอร์เริ่มต้นของแกน roll
    roll_dt = dt;

    calc_filt_alpha();
}

void AC_PI_roll::load_gains()
{
    roll_kp.load();
    roll_ki.load();
    roll_imax.load();
    roll_imax = fabsf(_imax);
    roll_filt_hz.load();

    calc_filt_alpha();
}                                     // การทำงานของระบบควบคุม
                                         // แบบพีไอของแกน roll

void AC_PI_roll::save_gains()
{
    roll_kp.save();
    roll_ki.save();
    roll_imax.save();
    roll_filt_hz.save();
}

void AC_PI::operator_pitch() (float p, float i, float imaxval, float input_filt_hz, float dt)
{
    pitch_kp = p;
    pitch_ki = i;
    pitch_imax = fabsf(imaxval);
    pitch_filt_hz = input_filt_hz;      // กำหนดพารามิเตอร์เริ่มต้นของแกน pitch
    pitch_dt = dt;

    calc_filt_alpha();
}
  
```

```

void AC_PI_pitch::load_gains()
{
    pitch_kp.load();
    pitch_ki.load();
    pitch_imax.load();
    pitch_imax = fabsf(_imax);
    pitch_filt_hz.load();

    calc_filt_alpha();
}

void AC_PI_pitch::save_gains()
{
    pitch_kp.save();
    pitch_ki.save();
    pitch_imax.save();
    pitch_filt_hz.save();
}

void AC_PI::set_input(const Vector2f &input)
{
    if (!isfinite(input.x) || !isfinite(input.y)) {
        return;
    }

    if (_flags._reset_filter) {           การตรวจสอบการรับอินพุตผ่านตัวกรอง
        _flags._reset_filter = false;      การทำงานของระบบควบคุมแบบฟ์ไอ
        _input = input;
    }

    Vector2f input_filt_change = (input - _input) * _filt_alpha;
    _input = _input + input_filt_change;
}

Vector2f AC_PI::get_roll() const
{
    return (_input * roll_kp);          การทำงานของระบบควบคุม
}                                      แบบฟ์ไอของแกน roll
                                         แบบฟ์ไอของแกน roll
Vector2f AC_PI::get_i_roll()
{
    if(!is_zero(roll_ki) && !is_zero(roll_dt)) {
        _integrator += (_input * roll_ki) * roll_dt;
        float integrator_length = _integrator.length();
        if ((integrator_length > roll_imax) && (integrator_length > 0)) {
            _integrator *= (roll_imax / integrator_length);
        }
        return _integrator;
    }
    return Vector2f();
}

Vector2f AC_PI::get_p_pitch() const
{
    return (_input * pitch_kp);         การทำงานของระบบควบคุม
}                                      แบบฟ์ไอของแกن pitch
                                         แบบฟ์ไอของแกน pitch
Vector2f AC_PI::get_i_pitch()
{
    if(!is_zero(pitch_ki) && !is_zero(pitch_dt)) {
        _integrator += (_input * pitch_ki) * pitch_dt;
        float integrator_length = _integrator.length();
        if ((integrator_length > pitch_imax) && (integrator_length > 0)) {
            _integrator *= (pitch_imax / integrator_length);
        }
        return _integrator;
    }
    return Vector2f();
}

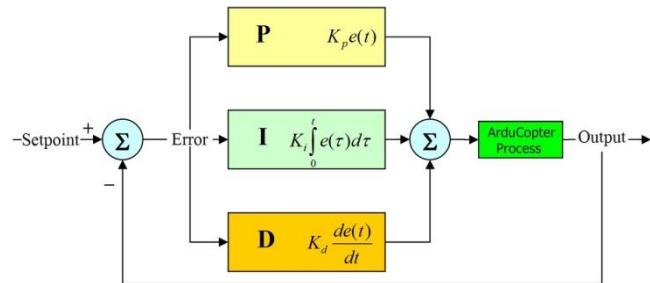
Vector2f AC_PI::get_p()
{
    return get_p();
}

Vector2f AC_PI::get_pi()
{
    return get_p() + get_i();           ศึกษาการทำงานของ
}                                      ระบบควบคุมแบบฟ์ไอ

```

Code ข้างต้นเป็นการปรับปรุงไฟล์ซึ่งใช้สำหรับกำหนดค่าเริ่มต้นสำหรับการทำงานของระบบควบคุมแบบฟ์ไอโดยจะมีการหาค่า error หลังจากนั้นก็จะทำการโหลดค่า จากนั้นก็ทำการคำนวณหาค่าและเข้าสู่เงื่อนไขต่าง ๆ และทำการบันทึกค่าเพื่อนำไปใช้สำหรับการทำงานในกระบวนการการต่อ ๆ ไปโดยการ return

การทำงานของระบบควบคุมแบบพีไอดี (ปรับค่าสัดส่วน ปริพันธ์และอนุพันธ์ หรือ K_p , K_i และ K_d)



ไฟล์ my_PID.cpp เป็นไฟล์ที่ใช้ในระบบควบคุมแบบพีไอดีที่ซึ่งได้จากการปรับปรุง

```

#define roll_P 0.015f
#define roll_I 0.1f
#define roll_D 0.004f
#define roll_IMAX 100
#define roll_FILTER 5.0f
#define roll_DT 0.01f
#define roll_INITIAL_FF 0.0f

#define pitch_P 0.15f
#define pitch_I 0.6f
#define pitch_D 0.0125f
#define pitch_IMAX 100
#define pitch_FILTER 5.0f
#define pitch_DT 0.01f
#define pitch_INITIAL_FF 0.0f

set ค่าของ Kp Ki และ Kd
ของแกน roll และแกน pitch

void AC_PID::operator_roll() (float p, float i, float d, float imaxval, float input_filt_hz, float dt)
{
    roll_kp = p;
    roll_ki = i;
    roll_kd = d;
    roll_imax = fabsf(imaxval);
    roll_filt_hz = input_filt_hz;
    roll_dt = dt;
}

void AC_PID::load_gains_roll()
{
    roll_kp.load();
    roll_ki.load();
    roll_kd.load();
    roll_imax.load();
    roll_imax = fabsf(_imax);
    roll_filt_hz.load();
}

void AC_PID::save_gains_roll()
{
    roll_kp.save();
    roll_ki.save();
    roll_kd.save();
    roll_imax.save();
    roll_filt_hz.save();
}

void AC_PID::operator_pitch() (float p, float i, float d, float imaxval, float input_filt_hz, float dt)
{
    pitch_kp = p;
    pitch_ki = i;
    pitch_kd = d;
    pitch_imax = fabsf(imaxval);
    pitch_filt_hz = input_filt_hz;
    pitch_dt = dt;
}
  
```

กำหนดพารามิเตอร์เริ่มต้นของแกน roll

การทำงานของระบบควบคุมแบบพีไอดีของแกน roll

กำหนดพารามิเตอร์เริ่มต้นของแกน pitch

```

void AC_PID::load_gains_pitch()
{
    pitch_kp.load();
    pitch_ki.load();
    pitch_kd.load();
    pitch_imax.load();
    pitch_imax = fabsf(_imax);
    pitch_filt_hz.load();
}

void AC_PID::save_gains_pitch()
{
    pitch_kp.save();
    pitch_ki.save();
    pitch_kd.save();
    pitch_imax.save();
    pitch_filt_hz.save();
}

void AC_PID::set_input_filter_d(float input)
{
    // don't process inf or NaN
    if (!isfinite(input)) {
        return;
    }

    // reset input filter to value received
    if (_flags._reset_filter) {
        _flags._reset_filter = false;    การตรวจสอบการอินพุตผ่านตัวกรอง
        _derivative = 0.0f;             การทำงานของระบบควบคุมฟ์ไอดี
    }

    // update filter and calculate derivative
    if (_dt > 0.0f) {
        float derivative = (input - _input) / _dt;
        _derivative = _derivative + get_filt_alpha() * (derivative - _derivative);
    }

    _input = input;
}

float AC_PID::get_p_roll()
{
    _pid_info.P = (roll_input * roll_kp);
    return _pid_info.P;
}

float AC_PID::get_i_roll()
{
    if(!is_zero(roll_ki) && !is_zero(roll_dt)) {
        _integrator += ((float)_input * roll_ki) * roll_dt;
        if (_integrator < -roll_imax) {
            _integrator = -roll_imax;
        } else if (_integrator > roll_imax) {
            _integrator = roll_imax;
        }
        _pid_info.I = _integrator;           การทำงานของระบบควบคุม
                                            แบบฟ์ไอดีของแกน roll
    }
    return 0;
}

float AC_PID::get_d_roll()
{
    _pid_info.D = (roll_kd * _derivative);
    return _pid_info.D;
}

float AC_PID::get_p_pitch()
{
    _pid_info.P = (_input * pitch_kp);
    return _pid_info.P;
}

float AC_PID::get_i_pitch()
{
    if(is_zero(pitch_ki) && is_zero(pitch_dt)) {
        _integrator += ((float)_input * pitch_ki) * pitch_dt;
        if (_integrator < -pitch_imax) {
            _integrator = -pitch_imax;
        } else if (_integrator > pitch_imax) {
            _integrator = pitch_imax;
        }
        _pid_info.I = _integrator;           การทำงานของระบบควบคุม
                                            แบบฟ์ไอดีของแกน pitch
    }
    return 0;
}

float AC_PID::get_d_pitch()
{
    _pid_info.D = (pitch_kd * _derivative);
    return _pid_info.D;
}

```

การทำงานของระบบควบคุม
แบบฟ์ไอดีของแกน pitch

```

float AC_PID::get_p()
{
    return get_p();
}

float AC_PID::get_pi()
{
    return get_p() + get_i();
}

float AC_PID::get_pid()
{
    return get_p() + get_i() + get_d();
}

```

ศีนค่าการทำงานของ
ระบบควบคุมแบบพื้นดิน

Code ข้างต้นเป็นการปรับปรุงไฟล์ซึ่งใช้สำหรับกำหนดค่าเริ่มต้นสำหรับการทำงานของระบบควบคุมแบบพื้นดินโดยจะมีการหาค่า error หลังจากนั้นก็จะทำการโหลดค่า จากนั้นก็ทำการคำนวณหาค่าและเข้าสู่เงื่อนไขต่าง ๆ และทำการบันทึกค่าเพื่อนำไปใช้สำหรับการทำงานในกระบวนการต่อ ๆ ไปโดยการ return

● การปรับแต่ง PID

การปรับแต่งด้วยวิธี Ziegler–Nichols โดยใช้ตารางตารางที่ 3-2 เพื่อค่าหา P I D ที่จะนำไปใช้จริง

ตารางที่ 3 - 3 แสดงวิธีการปรับแต่งค่าแบบ Ziegler–Nichols

Control Type	K _p	K _i	K _d
P	0.50 K _c	-	-
PI	0.45 K _c	1.2 K _p / K _c	-
PID	0.60 K _c	2 K _p / K _c	K _p K _c / 8

- ขั้นตอนการหาค่า P I D ที่เหมาะสมสำหรับระบบนี้
 1. กำหนดค่า K_p และ K_i ให้ = 0 ในแรก x และทำการทดสอบบิน
 2. ปรับค่า K_p เพิ่มขึ้นเรื่อยๆจาก 0 จนระบบแก่วง
 3. นำค่า K_c และ P_c ไปแทนในตารางที่ 3-2

- การ burn code ลงบอร์ด APM

1. ดาวน์โหลดโปรแกรม ArduPilot-Arduino จากเว็บ <http://firmware.ardupilot.org/Tools/Arduino/ArduPilot-Arduino-1.0.3-gcc-4.8.2-windows.zip> ตามรูปที่ 3-13

Install ArduPilot-Arduino

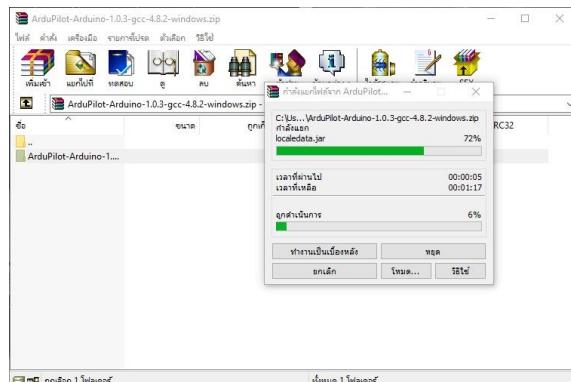
Download and unzip the ArduPilot Arduino package:

<http://firmware.ardupilot.org/Tools/Arduino/ArduPilot-Arduino-1.0.3-gcc-4.8.2-windows.zip>

This can be unzipped directly to the C: drive or C:\Program Files\

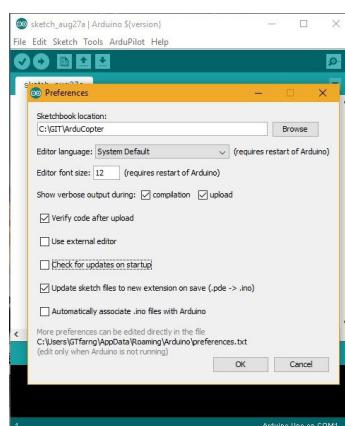
รูปที่ 3 - 13 หน้าเว็บดาวโหลด ArduPilot

2. จะได้ไฟล์ .zip มาและให้ทำการแตกไฟล์ ArduPilot-Arduino-1.0.3-gcc-4.8.2-windows.zip ตามรูปที่ 3-14



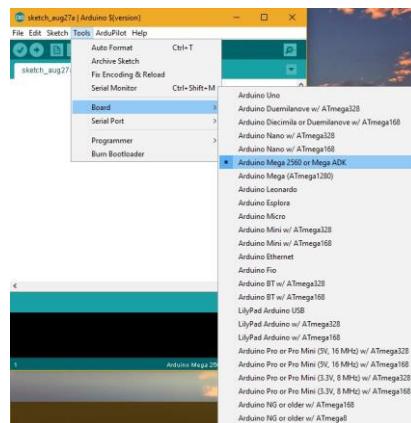
รูปที่ 3 - 14 แตกไฟล์.zip

3. ทำการเปิดโปรแกรมและทำการตั้ง sketchbook ตามโฟลเดอร์ที่ได้โหลด code มาเก็บไว้เพื่อให้โปรแกรมสามารถเปิดไฟล์ code ได้สะดวกตามรูปที่ 3-15



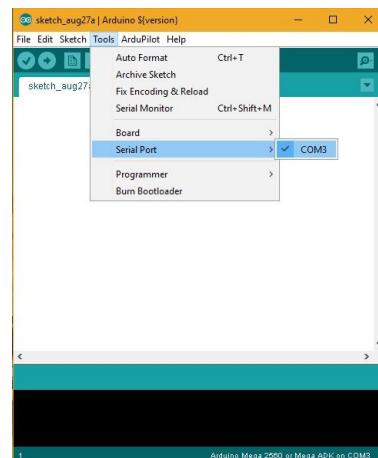
รูปที่ 3 - 15 เลือกที่อยู่ sketchbook

4. ทำการเซ็ตบอร์ดเป็น Arduino Mega 2560 or Mega ADK



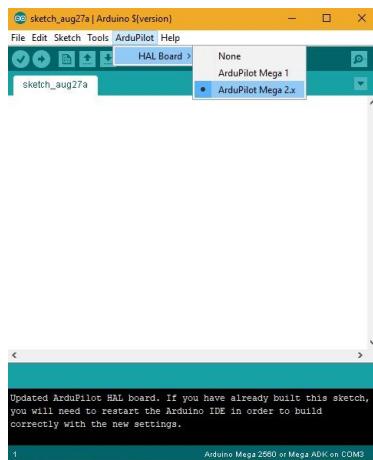
รูปที่ 3 - 16 Set Board Arduino

5. ทำการเลือก Serial port ให้ตรงกับไดร์เวอร์ที่ทำการติดตั้ง



รูปที่ 3 - 17 Set port Arduino

6. ทำการเลือก HAL Board เป็น ArduPilot Mega 2.X



รูปที่ 3 - 18 Set HAL Board Arduino

7. หลังจากนั้นสามารถเปิดไฟล์ code เพื่อคุณแล้วแก้ไขได้ คลิกที่เครื่องหมายถูกเพื่อ Compile code และ burn ลงบอร์ดจากรูปที่ 3-18

```

AC_PID_test | Arduino [version]
File Edit Sketch Tools ArduPilot Help
AC_PID_test
/*
 * Example of PID library.
 * 2012 Code by Jason Short, Randy Hackey, DIYDrones.com
 */
#include <AP_Common.h>
#include <AP_Program.h>
#include <AP_HAL_ATM.h>
#include <AP_HAL_ATP.h>
#include <AP_Math.h>
#include <AP_Parm.h>
#include <AP_Sensor.h>
#include <AC_PID.h>
#include <AC_MCU_PID.h>

const AP_HAL::HAL* hal = AP_HAL::BOARD_DRIVER;

// default PID values
#define TEST_P 1.0
#define TEST_I 0.0
#define TEST_D 0.2
#define TEST_MAX 10

// setup function
void setup()
{
    hal.console->println("ArduPilot Mega ac_PID library test");
}

Done compiling
C:\Users\Markbook\AppData\Local\Temp\build028677444668543102.tmp\AC_PID_test.cpp.eif
C:\Users\Markbook\AppData\Local\Temp\build028677444668543102.tmp\AC_PID_test.cpp.hex
Binary sketch size: 21,536 bytes (of a 256,048 byte maximum)

```

รูปที่ 3 - 19 ทดสอบการเบิร์นลงบอร์ด

3.5 แผนการดำเนินงาน

ตารางที่ 3 - 4 แผนการดำเนินงาน Project Preparation

การดำเนินงาน / ระยะเวลา	ปี พ.ศ.2560																			
	มกราคม				กุมภาพันธ์				มีนาคม				เมษายน				พฤษภาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ศึกษาการทำงานของโดรน																				
ศึกษาข้อมูลเกี่ยวกับอุปกรณ์ของโดรน																				
ศึกษาข้อมูลเกี่ยวกับอุปกรณ์และจัดซื้ออุปกรณ์																				
ศึกษาหลักการ PID และหลักการที่เกี่ยวข้องเพิ่มเติม																				
ศึกษาวิธีการติดตั้งโปรแกรมและวิธีการใช้งานโปรแกรม																				
ศึกษาการทำงานของ Code ที่เกี่ยวข้อง																				
ทดลองปรับแต่ง PID แบบง่าย ๆ																				

ตารางที่ 3 - 5 แผนการดำเนินงาน Project I

การดำเนินงาน / ระยะเวลา	ปี พ.ศ.2560																			
	สิงหาคม				กันยายน				ตุลาคม				พฤษจิกายน				ธันวาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ศึกษาทฤษฎีการปรับแต่ง PID และทฤษฎีที่เกี่ยวข้องแบบต่าง ๆ																				
ศึกษาทฤษฎีการปรับแต่งแบบ Ziegler-Nichols																				
ปรับปรุง Code ที่เกี่ยวกับการทำงานของ P, PI และ PID																				
ทดลองปรับแต่ง PID และบันทึกผลของแกน Roll																				
เพิ่มการปรับแต่ง PID และบันทึกผลของแกน Pitch																				
ศึกษาข้อมูลเกี่ยวกับ Telemetry และทำการจัดซื้อ																				
ศึกษาเกี่ยวกับการเขียน Android application																				
ศึกษาการใช้งาน Google Maps Api																				

ตารางที่ 3 - 6 แผนการดำเนินงาน Project II

การดำเนินงาน / ระยะเวลา	ปี พ.ศ.2561																			
	มกราคม				กุมภาพันธ์				มีนาคม				เมษายน				พฤษภาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ศึกษาข้อมูลที่เป็นปัจจัยที่ส่งผลต่อการทรงตัวของโดรน																				
ทำการพัฒนาโดรนให้สามารถทรงตัวและต้านลมได้																				
ทำการปรับน้ำหนักและจุด洼อุปกรณ์บนตัวโดรนให้สมดุล																				
ทำการปรับและจูนระบบควบคุม PID ให้สมดุลมากยิ่งขึ้น																				
ทดสอบการบินในสถานที่ต่าง ๆ																				
ศึกษาการใช้งานระบบ Tx และ Rx ของกล้อง																				
เขียน android application ของการรับส่งภาพ																				
ปรับปรุง GUI ของแอพพลิเคชันให้ใช้งานสะดวก																				

บทที่ 4 การทดลอง

4.1 การทดลองที่ 1 การทำ Binding Remote กับ Receiver

อุปกรณ์

เพื่อทำการจูนสัญญาณของ Remote กับ Receiver ให้ตรงกัน

อุปกรณ์

1. Remote Flysky FS-TH9X
2. Receiver Flysky FS-R9B

วิธีการทดลอง

1. ทำการต่อสาย Binding ที่ช่อง BIND ของ Receiver
2. ทำการต่อ Battery ที่ช่อง BAT ของ Receiver



3. จะสังเกตเห็นไฟที่ Receiver กระพริบ



4. จากนั้นกดที่ปุ่ม Bind Range Test ที่ด้านหลังรีโมทค้างไว้แล้วเปิดสวิตช์ที่รีโมท



5. จะสังเกตเห็นไฟที่ Receiver หยุดกระพริบเป็นอันว่าสามารถใช้งานได้



6. จากนั้นทำการถอดสาย Binding และสาย Battery ออกจากตัว Receiver

7. ทำการปิดสวิตช์ที่รีโมท

8. ทำการต่อ Battery ที่ Receiver จากนั้นทำการเปิดสวิตช์ที่รีโมท

9. สังเกตไฟที่ตัว Receiver ถ้าติดก็แสดงว่าใช้งานได้ปกติ

จากการทดลองจนสัญญาณ

ปรากฏว่า Remote กับ Receiver สามารถเชื่อมต่อกันได้

4.2 การทดลองที่ 2 การโปรแกรมการทำงานของ Remote Flysky FS - TH9X

อุปกรณ์

เพื่อที่จะทำให้สามารถทำการ Calibrate ได้ง่ายยิ่งขึ้น

อุปกรณ์

1. Remote Flysky FS-TH9X

วิธีการทดลอง

1. ทำการเปิดสวิตซ์ตัวรีโมท และทำการดันสวิตซ์ทุกตัวให้อยู่ตำแหน่งเริ่มต้นแล้วจะแสดงหน้าจอดังภาพ



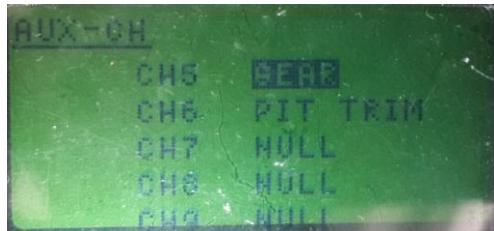
2. ทำการเลือกโหมดของการส่งสัญญาณเข้าสู่ตัวรับสัญญาณ โดยให้ทำการเลือกเป็น PPM (Pulse Position Modulation) เนื่องจากต้องทำการส่งสัญญาณเพื่อใช้ในการควบคุม ESC



3. ทำการเลือกโหมดการบิน โดยให้ทำการเลือกโหมดเป็น HELI -> HELI 1 ซึ่งจะมีการควบคุม 4 ทิศทาง หรือ 4 ช่องสัญญาณ Ch1 -> Roll , Ch2 -> Pitch , Ch3 -> Throttle และ Ch4 -> Yaw และทำการเลือกตำแหน่งของตำแหน่งควบคุม



4.ทำการเลือกช่องสัญญาณเพิ่มเติม ซึ่งใช้ในการเป็นสวิทช์ควบคุมการใช้งานเป็นโหมดต่าง ๆ เช่น Stabilize , Loiter , RTL เป็นต้น



5.จากนั้นทำการตรวจสอบว่าหลังจากการโปรแกรมข้างต้นนั้นทำงานถูกต้องหรือไม่ โดยสามารถดูได้ที่ DISPLAY ของรีโมทได้เบื้องต้น ซึ่งจะทำการ Calibrate Radio ซึ่งจะแสดงช่วงของสัญญาณที่ส่งออกจากตัวรีโมทเข้าสู่ตัวรับสัญญาณว่ามีค่าเท่าใดบ้าง



4.3 การทดลองที่ 3 การรีเซ็ตค่าเริ่มต้น (Default)ให้กับบอร์ด ArduPilotMega

จุดประสงค์

เพื่อเป็นทำการรีเซ็ตค่าเริ่มต้นให้เป็นค่า Default และ reset boot loader เพื่อป้องกันการเกิด error ต่าง ๆ

อุปกรณ์

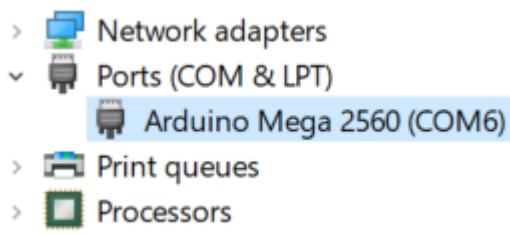
- 1.บอร์ด ArduPilotMaga
- 2.สาย Micro USB
- 3.โปรแกรม Arduino IDE

วิธีการทดลอง

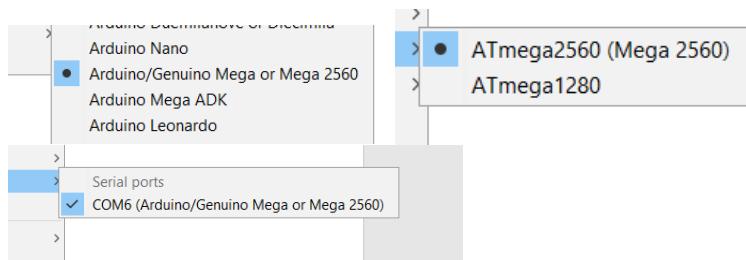
- 1.ทำการเสียบสาย Micro USB ระหว่างตัวบอร์ดกับพอร์ต USB
- 2.ทำการเปิดโปรแกรม Arduino IDE ขึ้นมา



- 3.ทำการติดตั้ง Driver ของบอร์ด ArduPilotMega



- 4.ทำการเลือก board, processor และ port ให้ตรงกับที่ใช้โดยไปที่เมนู Tools



- 5.หลังจากตั้งค่าในโปรแกรมแล้วก็ให้ไปที่เมนู File -> Examples -> EEPROM -> eeprom_clear

The screenshot shows the Arduino IDE interface with the sketch named 'eeprom_clear'. The code is as follows:

```
#include <EEPROM.h>

void setup() {
    // initialize the LED pin as an output.
    pinMode(13, OUTPUT);

    /**
     * Iterate through each byte of the EEPROM storage.
     * Larger AVR processors have larger EEPROM sizes.
     * - Arduino Duemilanove: 512b EEPROM storage.
     */
}
```

The status bar at the bottom indicates: Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6.

6. จากนั้นทำการ compile และ burn ลงบอร์ด ArduPilotMega

The screenshot shows the Arduino IDE terminal window with the message 'Done uploading.' and the following statistics:

```
Sketch uses 1302 bytes (0%) of program storage space
Global variables use 9 bytes (0%) of dynamic memory
```

The status bar at the bottom indicates: Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6.

7. เมื่อขึ้นคำว่า Done uploading แสดงว่า burn สำเร็จแล้ว ไฟตระหง่านจะขึ้นสีเขียวค้างไว้และไม่มีการกระพริบใด ๆ เนื่องจากไม่มีตัว Boot loader บอร์ดจึงไม่สามารถทำงานได้ จำเป็นต้องทำการลงเฟิร์มแวร์ใหม่ก่อน

4.4 การทดลองที่ 4 การติดตั้งเฟิร์มแวร์ให้กับบอร์ด ArduPilotMega

จุดประสงค์

เพื่อเป็นการติดตั้งเฟิร์มแวร์เพื่อให้บอร์ด ArduPilotMega สามารถทำงานได้

อุปกรณ์

- 1.บอร์ด ArduPilotMaga
- 2.สาย Micro USB
- 3.โปรแกรม Mission Planner

วิธีการทดลอง

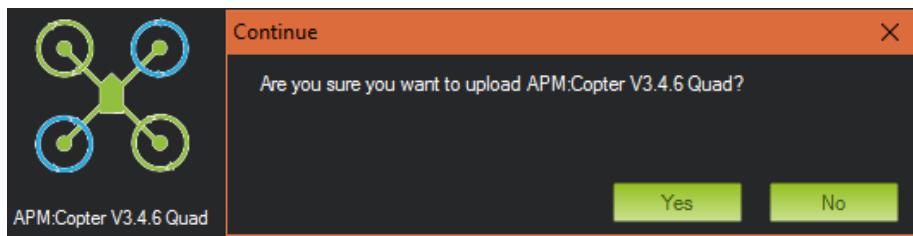
- 1.ทำการเสียบสาย micro USB ตรงตัวบอร์ดและเสียบเข้า Port USB
- 2.ทำการเปิดโปรแกรม Mission Planner ขึ้นมา



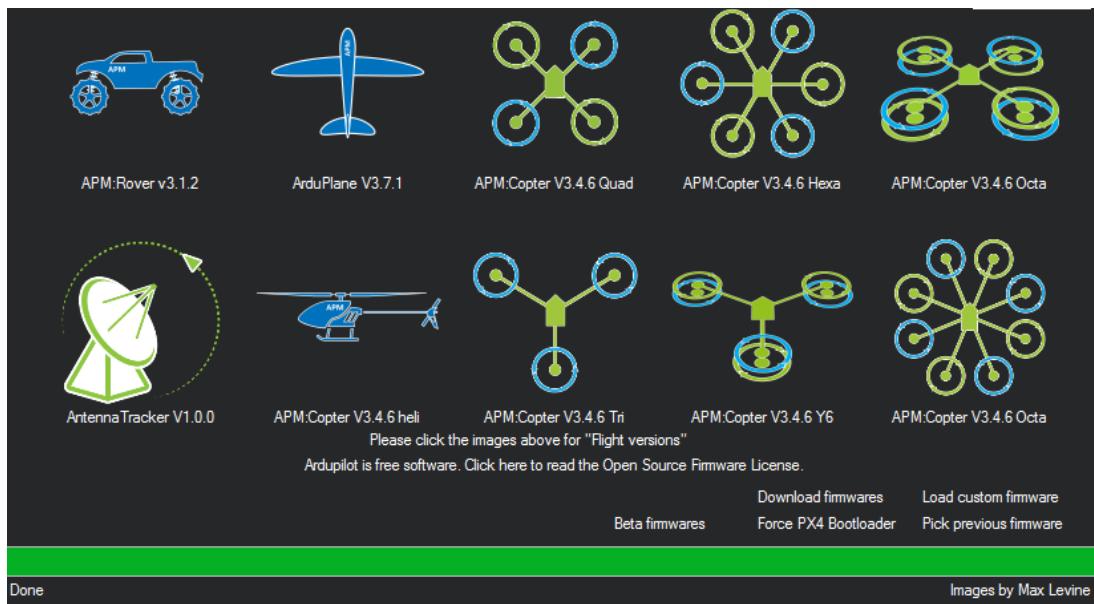
- 3.จากนั้นคลิกที่เมนู INITIAL SETUP



- 4.คลิกที่คำสั่ง Install Firmware จากนั้นเลือกชนิดของอุปกรณ์ โดยในที่นี้จะทำการเลือกเป็น Copter Quad จากนั้นคลิกที่ Yes เพื่อทำการลง Firmware



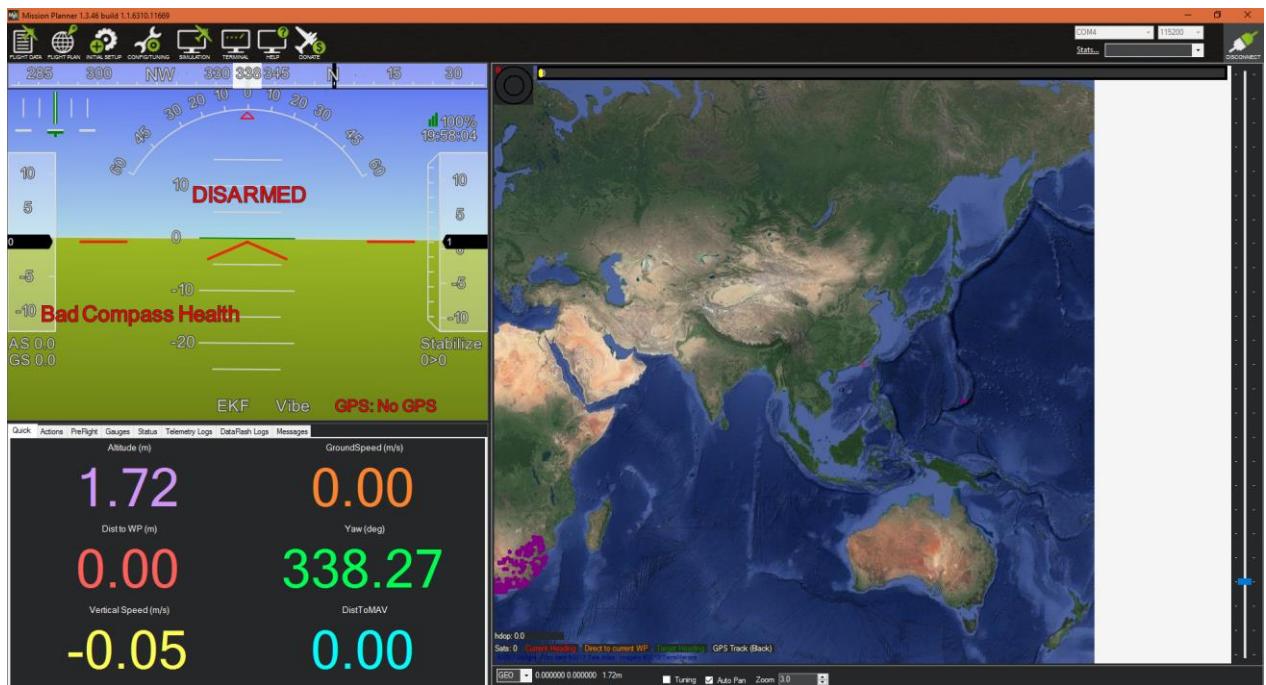
5. จากนั้น รอให้ล็อก firmware ให้เรียบร้อยจนสังเกตเห็นคำว่า “Done” ก็เป็นอันเสร็จ



6. จากนั้นคลิกที่แท็บเมนู FIGHT DATA ทำการเลือก port ที่เชื่อมต่อจากนั้นคลิกที่ Connect

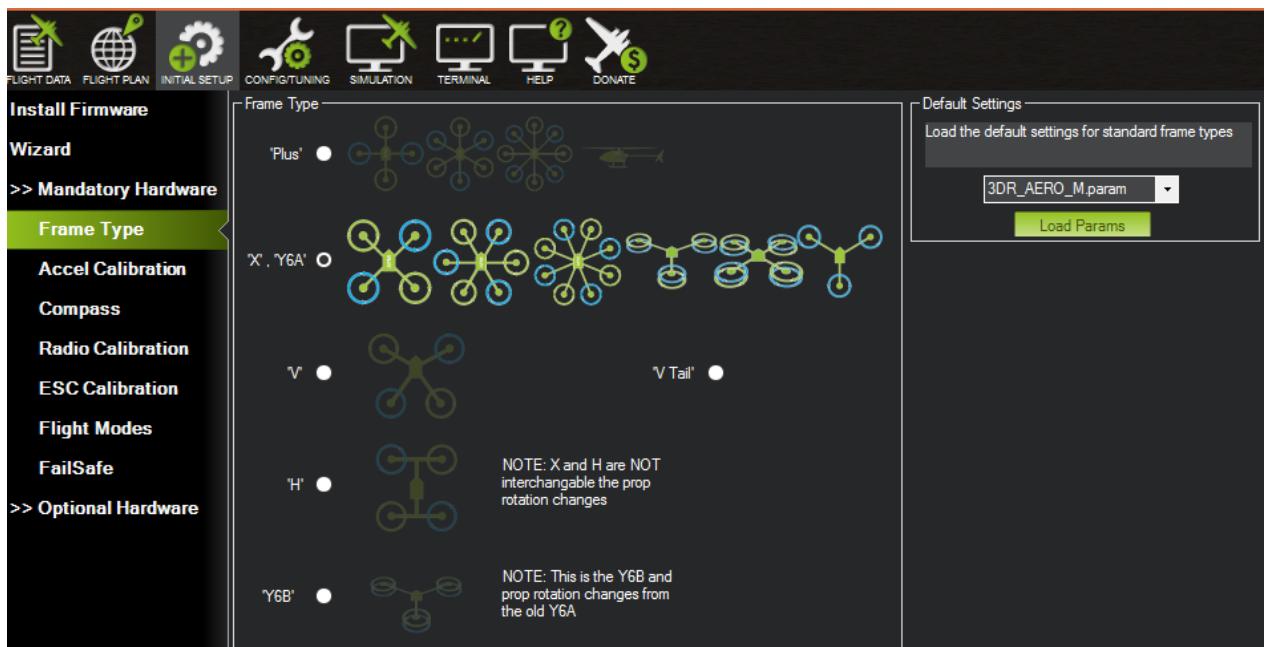


7. จะปรากฏข้อมูลต่าง ๆ ของบอร์ดแสดงในโปรแกรมดังรูป



8. จากนั้นกลับไปที่เมนู INITIAL SETUP เพื่อทำการตั้งค่าของบอร์ด

9. สามารถตั้งค่าได้ที่เมนู >>Mandatory Hardware เช่น การตั้งค่า Frame Type(ชนิดของเฟรม)



10. ก็เป็นอันเสร็จสำหรับการลงเฟิร์มแวร์ให้กับบอร์ด ArduPilotMega

4.5 การทดลองที่ 5 การทำ Accel calibration ของบอร์ด ArduPilotMega

อุปกรณ์

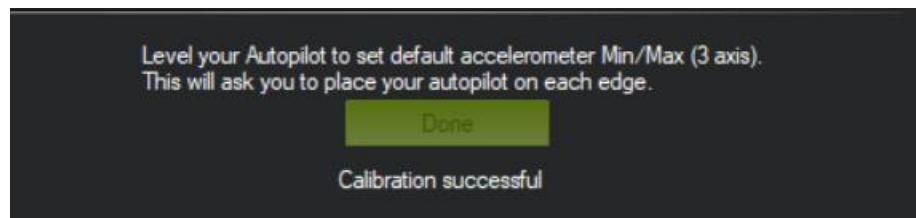
เพื่อทำการ Accel Calibrate ของบอร์ด ArduPilotMaga เพื่อเป็นการกำหนดค่าเริ่มต้นของแต่ละแกน
อุปกรณ์

- 1.บอร์ด ArduPilotMaga
- 2.สาย Micro USB
- 3.โปรแกรม MissionPlanner

วิธีการทดลอง

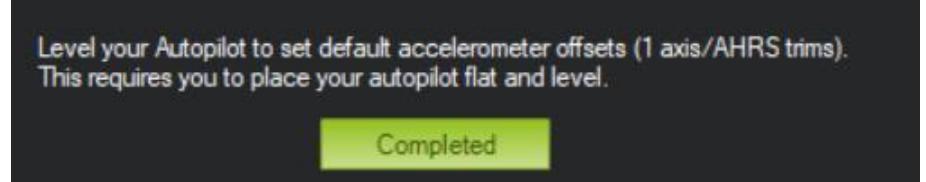
- 1.ทำการเสียบสาย micro USB ตรงตัวบอร์ดและเสียบเข้า Port USB
 - 2.ทำการเชื่อมต่อเข้ากับโปรแกรม Mission Planner
 - 3.ไปที่เมนู INITIAL SETUP -> Mandatory Hardware -> Calibrate Accel
- ในส่วนแรก Calibrate Accel (3-axis) คือ การให้ตัวบอร์ดจำแกนในแต่ละด้านของบอร์ด เช่น front, left, right, down, up และ back
 - 4.จากนั้นทำการพลิกบอร์ดตามที่โปรแกรมกำหนดโดยให้หันลูกศรที่ติดอยู่ที่บอร์ดไปตามทิศทางต่าง ๆ





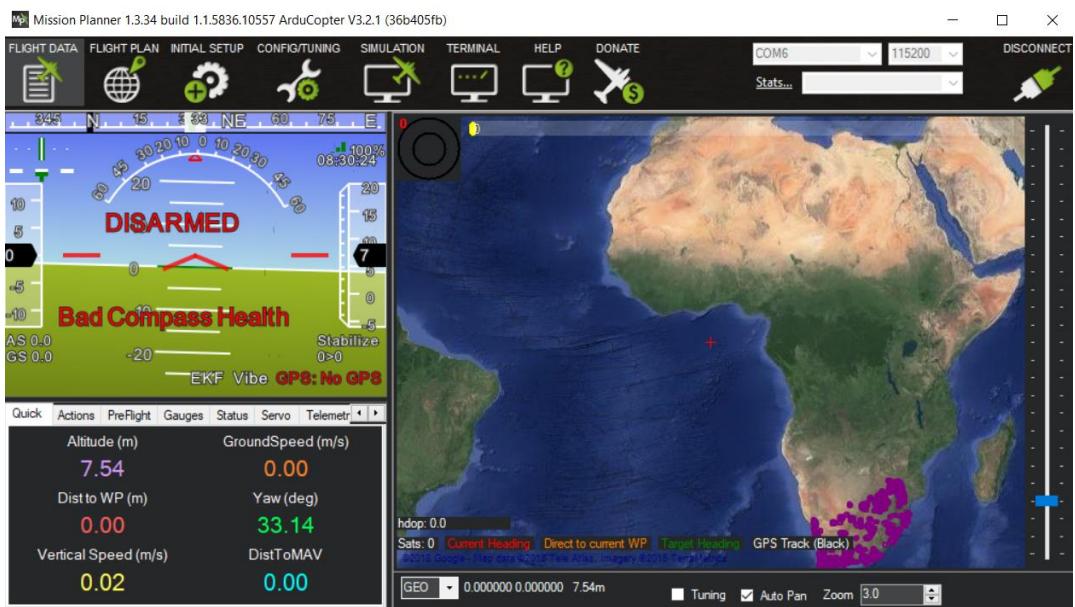
5. หากทำการ calibrate สำเร็จแล้วก็จะขึ้นข้อความว่า Calibration successful

- ในส่วนที่สอง Calibrate Level (1-axis) คือ การให้ตัวบอร์ดจำแนก (0,0,0) โดยจะต้องตั้งบอร์ดให้ราบกับพื้น



6. หากทำการ calibrate สำเร็จแล้วก็จะขึ้นข้อความว่า Completed

- 7. จากนั้นกลับมาที่เมนู FLIGHT DATA และทำการ connect เพื่อตรวจสอบความถูกต้องหลังจากการทำการ calibrate



6. จะเห็นว่าหลังจากการทำ calibrate ที่ (0,0,0) แล้วโปรแกรมสามารถอ่านค่าจาก Sensor ได้อย่างถูกต้องและมีความแม่นยำมากขึ้น

roll	0.795522	vz	0	groundspeed2	0	ay	4
pitch	-1.348084	vlen	0	groundcourse2	0	az	-1000
yaw	116.8339	altoffsethome	0	satcountB	0	accelsq	1.000552
SSA	0	gpssatus	1	gpsitime	1/1/2513 (gx	-1
AOA	0	gpshdop	99.99	altd1000	0.00202	gy	1
groundcourse	0	satcount	0	altd100	0.0202	gz	2
lat	0	lat2	0	airspeed	0	gyrosq	2.44949
lng	0	lng2	0	targetairspeed	0	mx	0
alt	2.02	altasl2	0	lowairspeed	False	my	0
altasl	-17	gpssatus2	0	asratio	0	mz	0
vx	0	gpshdop2	0	groundspeed	0	magfield	0
vy	0	satcount2	0	ax	-33	ax2	0

4.6 การทดลองที่ 6 ทำการ Calibrate Compass ของบอร์ด ArduPilotMega

จุดประสงค์

เพื่อทำการเช็คค่าเริ่มต้นให้กับเข็มทิศและจีพีเอสในทิศทางการหมุนด้านต่าง ๆ ของบอร์ด

อุปกรณ์

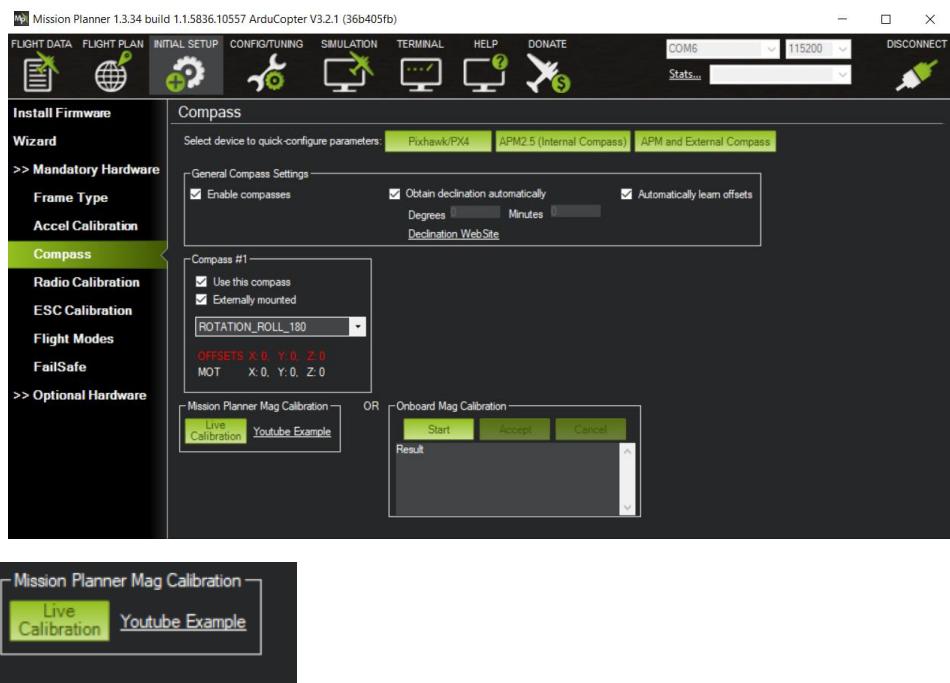
- 1.บอร์ด ArduPilotMaga
- 2.Module GPS NEO-7N
- 3.สาย Micro USB
- 4.โปรแกรม MissionPlanner

วิธีการทดลอง

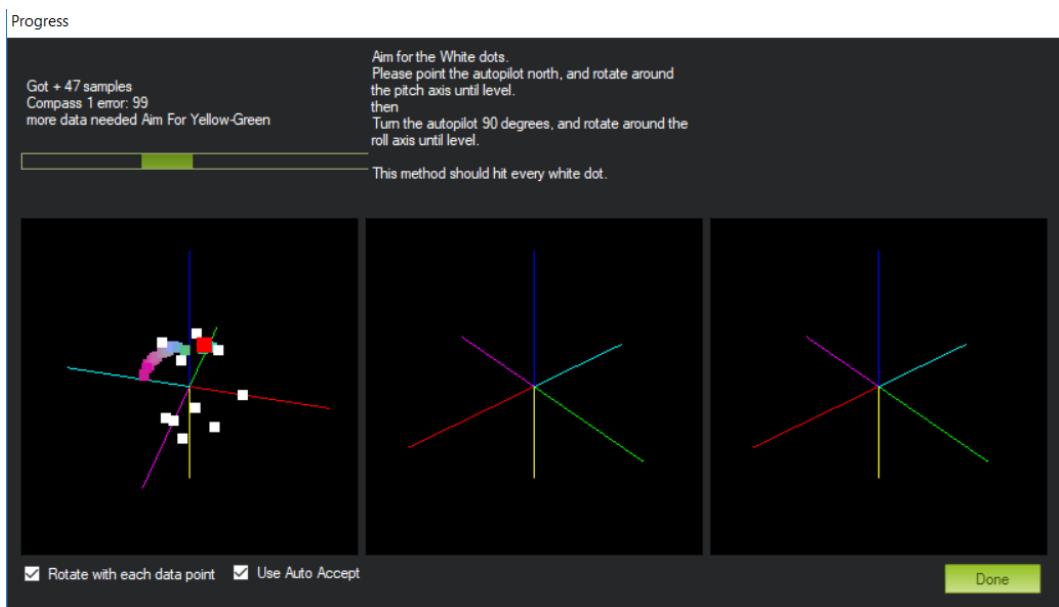
- 1.ทำการเสียบโมดูล GPS ที่มี compass ภายนอกเข้ากับ port I2C และ port GPS
- 2.ปรับให้ลูกศรของบอร์ด ArduPilotMega กับ Module ไปในทิศทางเดียวกัน



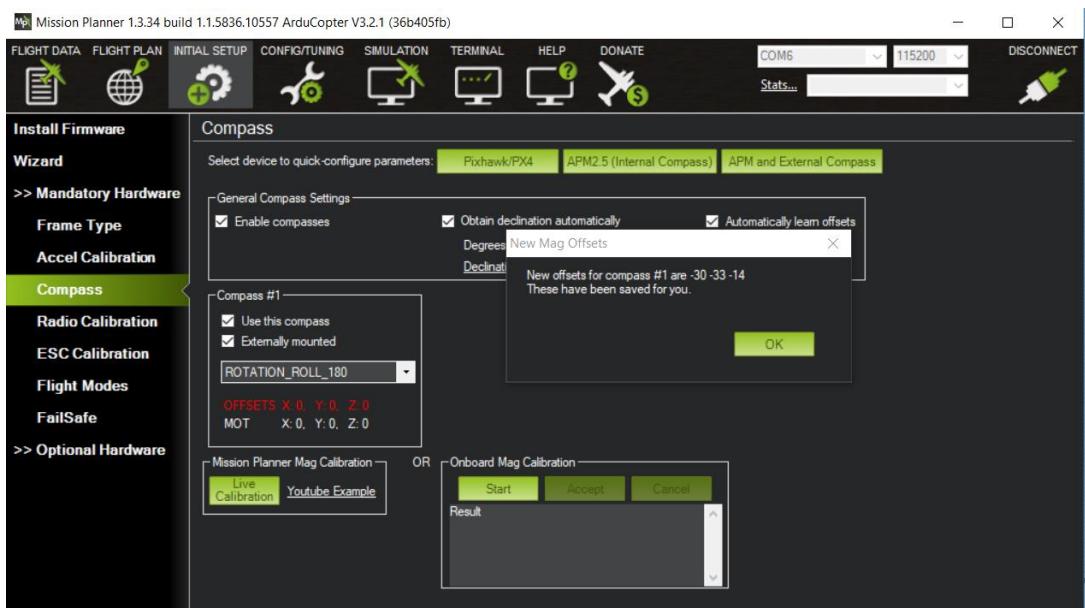
- 3.จากนั้นไปที่เมนู INITIAL SETUP -> Mandatory Hardware -> Compass -> APM and External Compass



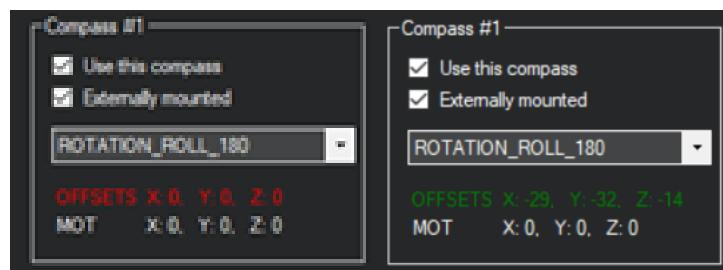
4. จากนั้นทำการกดที่ Live Calibration



5. ให้ทำการจับบอร์ดและหมุนไปในแนวต่าง ๆ เพื่อให้บอร์ดลดจำค่าของแนวและด้านต่าง ๆ ของเข็มทิศ



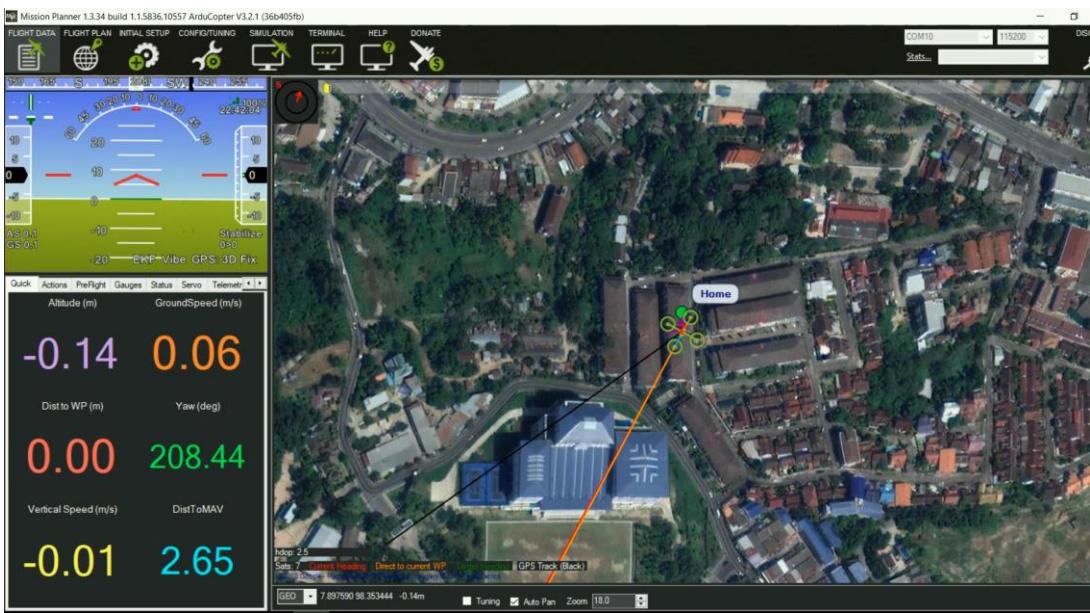
6. เพื่อสำหรับเจ้ารีโมทมีข้อความขึ้นมาแจ้งเตือนและระบุตำแหน่งของเข็มทิศ



ก่อนและหลังการ Calibrate Compass



7. จะเห็นว่าคำว่า bad compass หายไปเนื่องจากทำการ calibrate compass เรียบร้อยแล้ว และที่แสดง GPS : no fix เนื่องจากอยู่ภายในอาคารจึงไม่สามารถรับสัญญาณจากดาวเทียมได้



8. เมื่อ GPS สามารถสัญญาณจากดาวเทียมได้จะแสดงสถานะ GPS : 3D FIX พร้อมทั้งแสดงตำแหน่งที่อยู่ ณ ปัจจุบันบนแผนที่ และยังสามารถแสดงถึงเขตการป้องกันหรือเขตห้ามทำการบินได้อีกด้วย โดยจะอยู่ใกล้ ๆ กับสนามบินหรือกองบินต่าง ๆ เพื่อเป็นการป้องกันจากการบุกรุกจากอากาศยานไร้คนขับหรือโดรน

4.7 การทดลองที่ 7 การ Calibrate ESC

อุปกรณ์

เพื่อทำการเซ็ตค่าเริ่มต้นของช่วงสัญญาณที่ส่งให้ ESC เพื่อใช้ในการควบคุมมอเตอร์เพื่อจะทำให้มอเตอร์สามารถทำงานได้พร้อมกัน

อุปกรณ์

1. Remote Flysky FS-TH9X
2. Receiver Flysky R9B
3. ESC 30A
4. Battery 11.1 V (3 cell)
5. Motor Blussless
6. สาย Signal

วิธีการทดลอง

1. ทำเป้ารีโมทแล้วทำการดัน Throuttle ขึ้นมาสุด จากนั้นเสียบแบตเตอรี่แล้วจะยินเสียงจาก ESC จะมีเสียงบีบ หรือ บีบ บีบ



2. จากนั้นทำการดัน Throuttle ลงมาสุด แล้วจะได้ยินเสียงตอบรับจาก ESC แล้วจะได้ยินเสียง บีบ หรือ บีบ บีบ

2. ก็เป็นอันเสร็จ และสามารถทดสอบได้โดยการดัน Throuttle ดูว่ามอเตอร์หมุนทุกตัวหรือไม่ หรือสามารถดู output ที่ได้ในตัวโปรแกรม mission planner



ผลการทดลอง



จะเห็นได้ว่าเมื่อทำการ calibrate ESC และ %Interferent หรือการที่มอเตอร์หมุนไม่พร้อมกัน ซึ่งเกิดจากช่วงสัญญาณเริ่มต้นที่ตัว ESC ไม่เท่ากันแล้วจะทำให้เกิดการทำงานที่ไม่พร้อมเพียงกันแต่เมื่อทำการ calibrate esc มอเตอร์ทุกตัวก็สามารถทำงานได้พร้อมเพียงกันเนื่องจากมีช่วงสัญญาณเริ่มต้นที่เท่ากัน ซึ่งจะมีค่าเป็น 0 หรือ 50 %

4.8 การทดลองที่ 8 ทำการ Calibrate Radio ของบอร์ด ArduPilotMega

จุดประสงค์

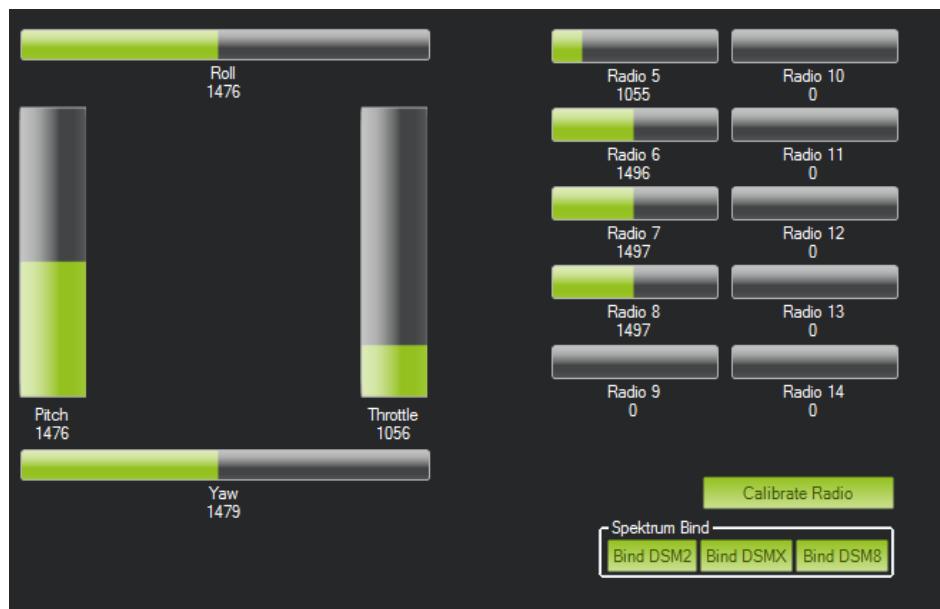
เพื่อทำการเช็คค่าเริ่มต้นของสัญญาณ PWM และสูงสุดที่ส่งมาจาก receiver กับบอร์ด ArduPilotMega อุปกรณ์

- 1.บอร์ด ArduPilotMaga
- 2.Module GPS NEO-7N
- 3.สาย Micro USB
- 4.โปรแกรม MissionPlanner
- 5.Receiver + Remote
- 6.สาย Signal

วิธีการทดลอง

1.ทำการเสียบสายสัญญาณระหว่าง Receiver กับ board ArduPilotMega

2.ไปที่เมนู INTIAL SETUP -> Mandatory Hardware -> Radio Calibration



การ calibrate วิทยุเป็นการทำเพื่อให้ตัวโดรนทราบว่าช่วงความกว้างของสัญญาณวิทยุมีค่าเป็นเท่าใด โดยค่าที่โปรแกรมบันทึกลงไว้มี 5 ช่องสัญญาณคือ ช่อง Roll, ช่อง Pitch, ช่อง Yaw , ช่อง Throttle และช่อง AUX ใช้สำหรับการปรับโหมดในการควบคุมของโดรน โดยเราสามารถยกคันโยกที่ตัววิทยุเพื่อดูค่าต่าง ๆ จากโปรแกรมได้

4.9 การทดลองที่ 9 ทำการตั้งค่าโหมดการบินของบอร์ด ArduPilotMega

จุดประสงค์

เพื่อทำการเช็คค่าเริ่มต้นของสัญญาณ PWM และสูงสุดที่ส่งมาจาก receiver กับบอร์ด ArduPilotMega

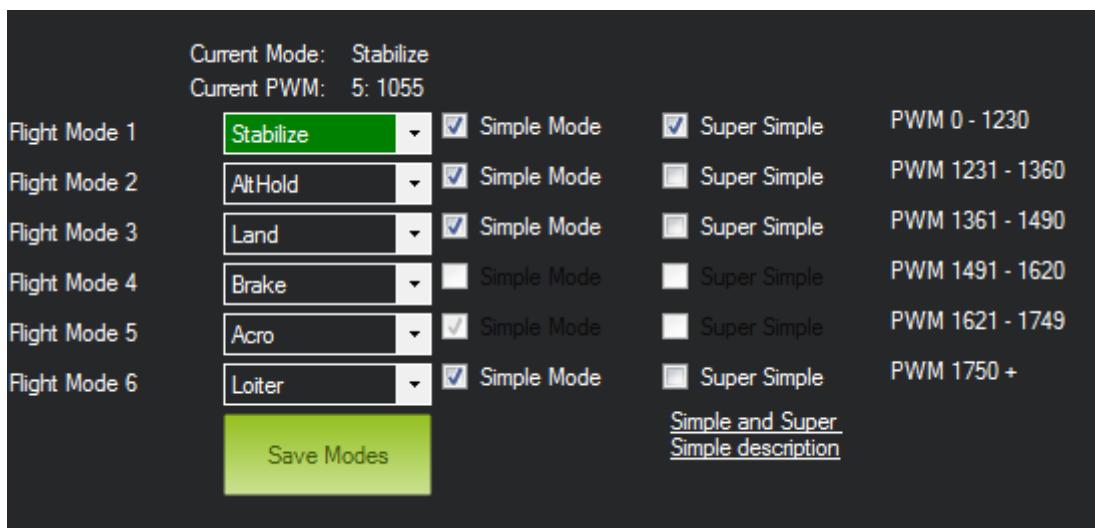
อุปกรณ์

- 1.บอร์ด ArduPilotMaga
- 2.Module GPS NEO-7N
- 3.สาย Micro USB
- 4.โปรแกรม MissionPlanner
- 5.Receiver + Remote
- 6.สาย Signal

วิธีการทดลอง

1.ทำการเสียบสายสัญญาณระหว่าง Receiver กับ board ArduPilotMega

2.ไปที่เมนู INITIAL SETUP -> Mandatory Hardware -> flight mode



โหมด Stabilize จะเป็นโหมดที่ทำให้ตัวโดรนเข้าสู่การที่โดรนมีเสถียรภาพสูง

โหมด Loiter จะเป็นการล็อกตัวโดรนโดยพิกัด GPS

โหมด RTL จะเป็นการใช้งาน Return to home โดยตั้งพิกัด GPS เพื่อจอด

โหมด LAND จะเป็นการใช้งานการ Landing ของตัวโดรน

รูปดังกล่าวเป็นหน้าโปรแกรมที่แสดงสถานะของโดรนตอนปัจจุบันหลังจากการ Setup เป็นต้น เพื่อให้โดรนสามารถบินได้ โดยโปรแกรมจะแสดงระดับของตัวโดรน ความเร็ว องศาการหมุน โดยเมื่อทำการ config บอร์ด APM สำเร็จแล้วก็จะสามารถทำการ ARM ได้ โดยทำการเลื่อนตำแหน่งคันบังคับที่รีโมทลง จะขึ้นสถานะ ARM ซึ่งเป็นการบ่งบอกว่าโดรนของเราสามารถบินได้แล้ว

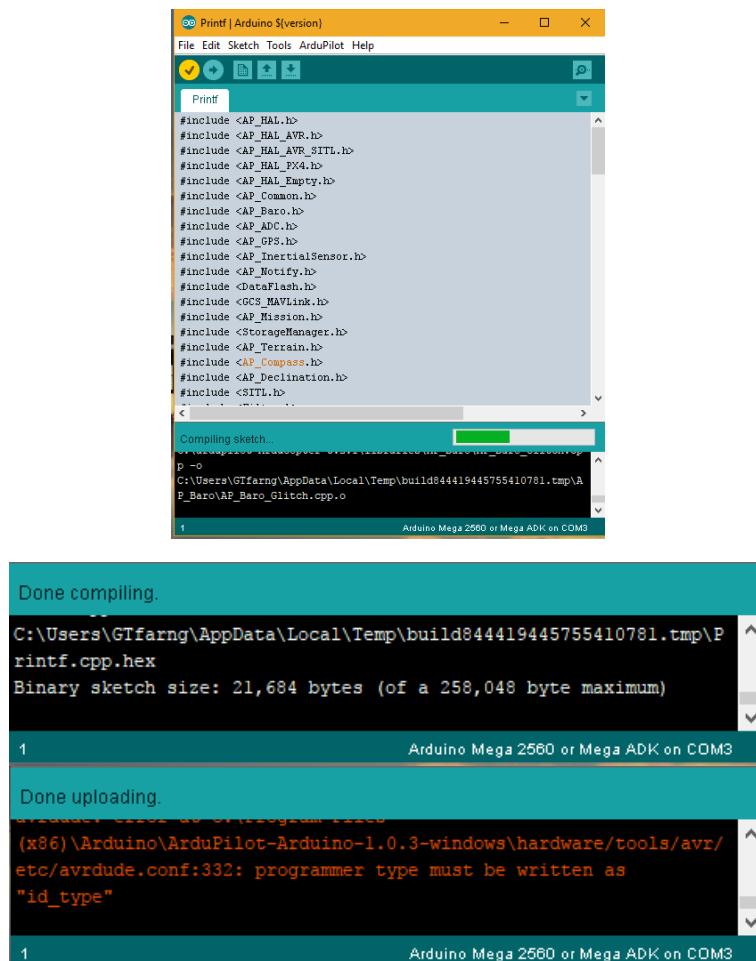
4.10 การทดลองที่ 10 ทำการรับค่า roll, pitch, yaw ขณะที่บอร์ดอยู่กับที่จุดประสงค์

เพื่อทำการทดลองรับค่า roll, pitch, yaw จาก sensor ประมาณ 2 นาทีแล้วพล็อตกราฟ อุปกรณ์

- 1.บอร์ด ArduPilotMega 2.8
- 2.สาย Micro USB
- 3.โปรแกรม Arduino IDE สำหรับ ArduPilot

วิธีการทดลอง

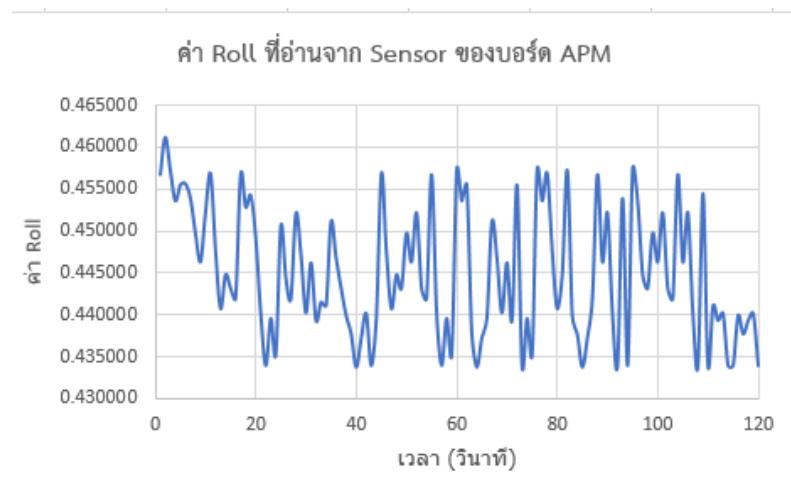
- 1.ทำการเสียบสาย micro USB ตรงตัวบอร์ดและเสียบเข้า Port USB
- 2.ทำการเปิดโปรแกรม Arduino IDE ขึ้นมา
- 3.ทำการเบริน code ที่เตรียมไว้สำหรับทดสอบบอร์ดโดยการกด compile และ upload



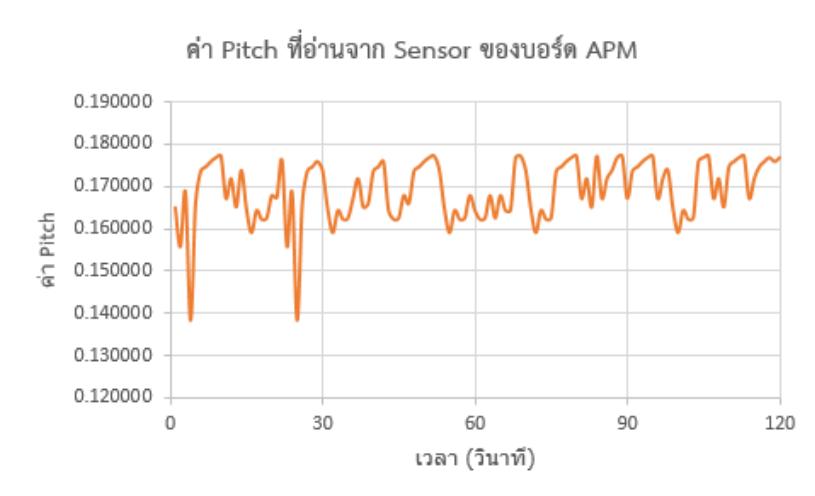
4.หลังจากขั้น Done uploading ให้ทำการเกิดไฟล์ data.txt ซึ่งได้จากการอ่านค่า roll, pitch และ yaw และเขียนลง text file

5. ทำการพล็อตโดยให้แกน X เป็นเวลา มีหน่วยเป็นวินาที และให้แกน Y เป็นค่า Roll, Pitch หรือ Yaw มีหน่วยเป็นองศา

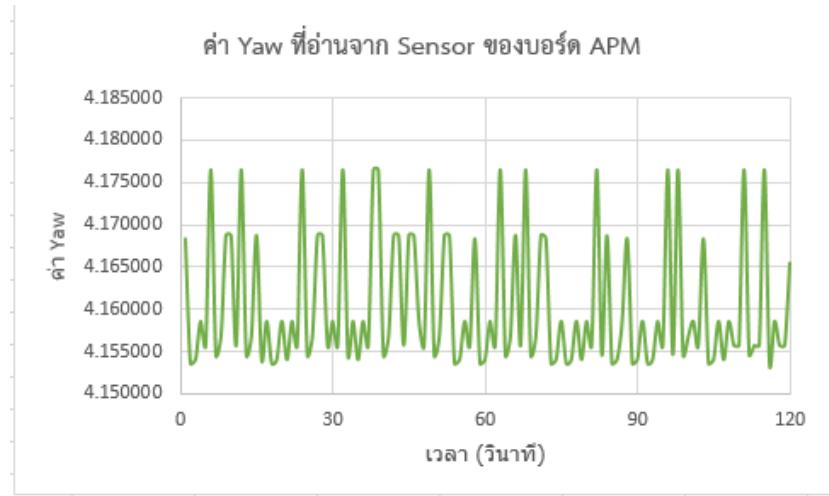
กราฟแสดงค่า Roll ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ในเวลา 2 นาที



กราฟแสดงค่า Pitch ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ในเวลา 2 นาที



กราฟแสดงค่า Yaw ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ในเวลา 2 นาที



จากการทดลองดังกล่าวพบว่า สามารถอ่านค่า Roll, Pitch และ Yaw ผ่าน Gyrometer Sensor ของบอร์ดได้โดยค่าที่อ่านมานั้นมีความคลาดเคลื่อน ± 0.001 ถึง ± 0.01 ดังกราฟที่แสดงด้านบน

4.11 การทดลองที่ 11 การเตรียมความพร้อมของการทำ PID Tuning

จุดประสงค์

เพื่อทำการติดตั้งอุปกรณ์ก่อนที่จะทำการจูนค่า PID

อุปกรณ์

1. ตัวโดรนที่ประกอบเสร็จแล้ว
2. เชือกสำหรับขึงตัวโดรนกับเสา
3. เสาหรือคานที่มีความแข็งแรงระดับหนึ่ง
4. โปรแกรม Mission Planner



วิธีการทดลอง

1. ทำการขึงตัวโดรนด้วยเชือกกับเสาทั้งสองข้าง โดยถ่วงน้ำหนักโดยให้โดรนมี balanced ไม่เอียงข้างใดข้างหนึ่ง



2. ทำการ Connect กับโปรแกรม Mission planner เพื่อทำการดูว่าในโปรแกรมตัวโดรนมี balanced หรือไม่



จากการทดลองดังกล่าว เป็นการเตรียมความพร้อมเพื่อที่จะทำการจูนค่าของระบบควบคุมแบบต่าง ๆ และจากรูปด้านบนเป็นการใช้งานโหมด Tuning ในโปรแกรม Mission Planner ซึ่งจะแสดงค่าที่รับจาก Sensor แบบ real-time

4.12 การทดลองที่ 12 ทดสอบการหมุนของตัวโดรนขณะที่บอร์ดมีการเปลี่ยนแปลงตำแหน่ง (ก่อนการปรับค่า K_p , K_i , k_d)

อุปกรณ์

เพื่อทำการรับค่า roll ในขณะทำการจับตัวโดรนที่ 90 องศาแล้วปล่อย และพล็อตกราฟ

อุปกรณ์

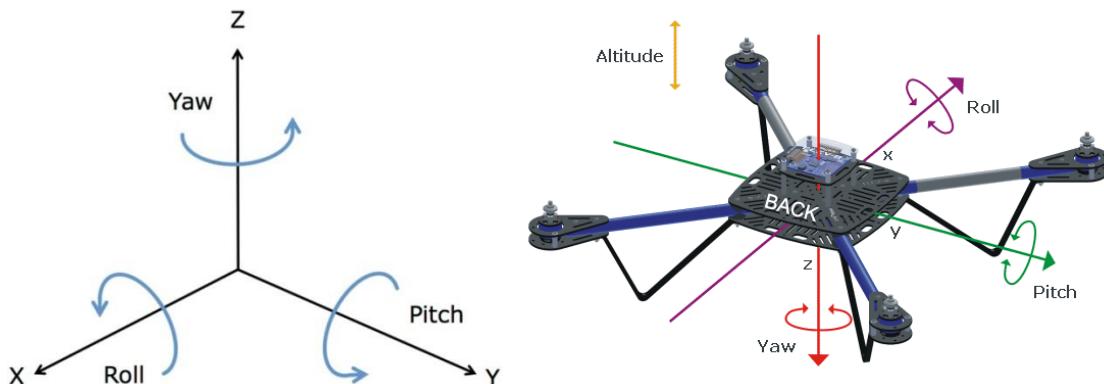
- 1.บอร์ด ArduPilotMega 2.8
- 2.สาย Micro USB
- 3.โปรแกรม Arduino IDE สำหรับ ArduPilot

วิธีการทดลอง

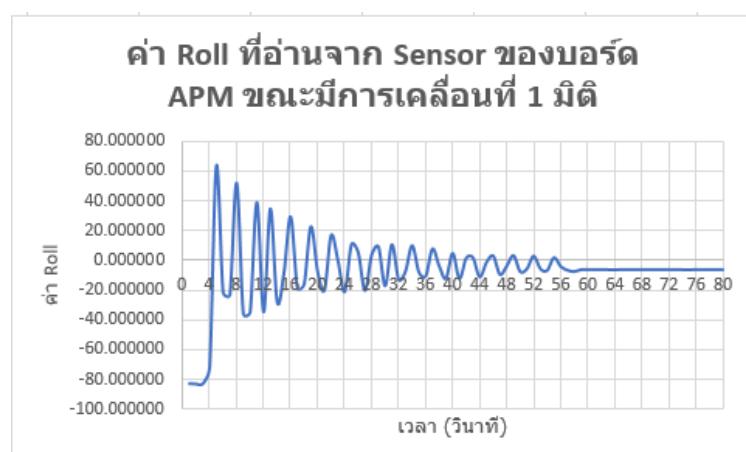
1.หลังจากการเตรียมการทดลองที่ 5 เรียบร้อยแล้ว จากนั้นก็ทำการทดลองรับค่า Roll, Pitch, Yaw ขณะที่ตัวโดรนมีการเปลี่ยนแปลงตำแหน่ง

2.ทำการจับตัวโดรนให้อุ่น 90 องศาแล้วปล่อยให้ตัวโดรนกลับสู่สมดุล

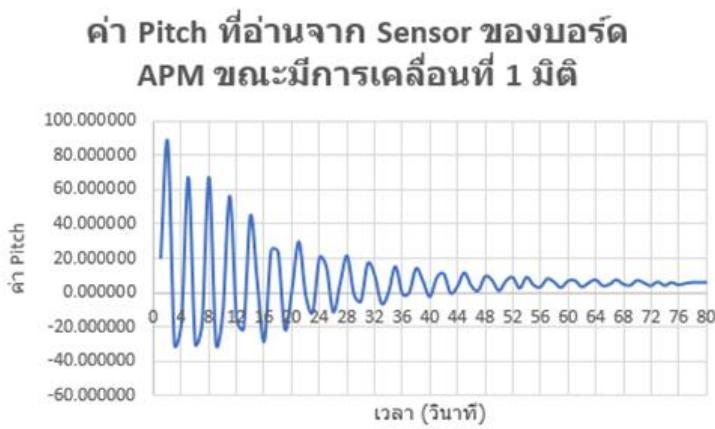
3.ทำการเขียนโปรแกรมแสดง output เป็น text file และทำการ plot กราฟ



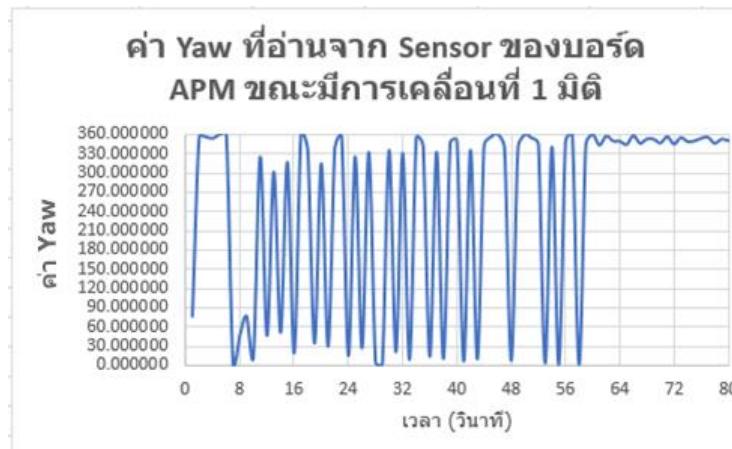
กราฟแสดงค่า Roll ที่มีการเปลี่ยนแปลงจากตำแหน่งที่ตัวโดรนอยู่ที่ 90 องศาจนถึงตำแหน่งสมดุล



กราฟแสดงค่า Pitch ที่มีการเปลี่ยนแปลงจากตำแหน่งที่ตัวโดรนอยู่ที่ 90 องศาจนถึงตำแหน่งสมดุล



กราฟแสดงค่า Yaw ที่มีการเปลี่ยนแปลงจากตำแหน่งที่ตัวโดรนอยู่ที่ 90 องศาจนถึงตำแหน่งสมดุล



จากการทดลองดังกล่าวเป็นการทดสอบการรับค่า Roll ขณะที่บอร์ดเอียง 90 องศา จนกว่าที่บอร์ดจะอยู่ในสภาพสมดุลหรือ Stable เปรียบเสมือนที่โดรนกำลังบินตั้งฉากกับพื้นโลก

ซึ่งเป็นการจำลองการทำงานของโดรนเมื่อยู่ในเหตุการณ์ที่ไม่คาดคิดคือโดรนบินตั้งฉากกับพื้นโลก โดยใบพัดของโดรนทั้งด้านหน้าและด้านหลังจะต้องหมุนสลับกันไปมาเพื่อที่จะให้โดรนกลับสู่ภาวะปกติหรือ Stable

4.13 การทดลองที่ 13 ทดสอบการหมุน ที่แกน Roll ของตัวโดรนขณะที่บอร์ดมีการเปลี่ยนแปลงตำแหน่ง (มีการปรับค่า K_p , K_i , K_d)

จุดประสงค์

เพื่อทำการจูนและหาค่า K_p , K_i , K_d ที่ทำให้เกิดความสมดุลหรือ stable

อุปกรณ์

- 1.บอร์ด ArduPilotMaga 2.8
- 2.สาย Micro USB
- 3.โปรแกรม Arduino IDE สำหรับ ArduPilot
- 4.โปรแกรม Missionplanner

วิธีการทดลอง

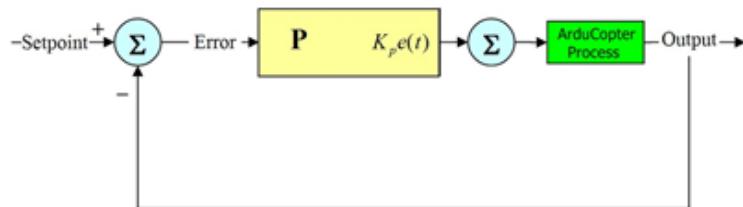
1.ทำการปรับค่า K_p , K_i หรือ K_d เริ่มต้นจาก 0 จากนั้นทำการเขียนโปรแกรมแสดง output เป็น text file และทำการ plot กราฟ

2.ทำการเพิ่มหรือเปลี่ยนแปลงค่า K_p , K_i หรือ K_d ไปเรื่อยๆ จนกว่าจะพบจุดที่เป็น Stable หรือจุดที่เกิด overshoot

3.ทำการทดลองทั้ง 3 แบบ คือ แบบ P โดยเปลี่ยนแปลงค่า K_p , แบบ PI โดยเปลี่ยนแปลงค่า K_p และ K_i และแบบ PID โดยเปลี่ยนแปลงค่า K_p , K_i และ K_d

ตอนที่ 1 การจูนและปรับค่าเฉพาะ K_p (แบบ P)

โดยการให้ค่า $K_p = 0, 0.05, 0.1, 0.15, 0.2$



การทำงานของระบบควบคุมแบบพี (Proportional)

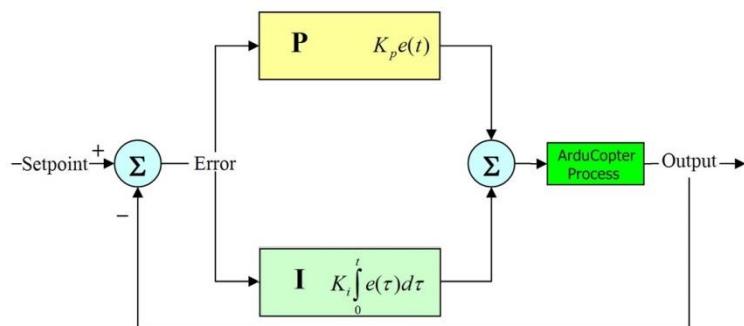
ผลการทดลอง

การทำงานของระบบ	K_p	กราฟแสดงการทำงานณ สภาวะสมดุล	เวลาที่ใช้ (วินาที)
แบบสัดส่วน (P)	0.15		29 (***) เร็วที่สุด

ตอนที่ 2 การจูนและปรับค่าเฉพาะ K_p และ K_i (แบบ PI)

โดยการให้ค่า $K_p = 0, 0.05, 0.1, 0.15$

โดยการให้ค่า $K_i = 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3$



การทำงานของระบบควบคุมแบบพีไอ (Proportional + Integral)

ผลการทดลอง

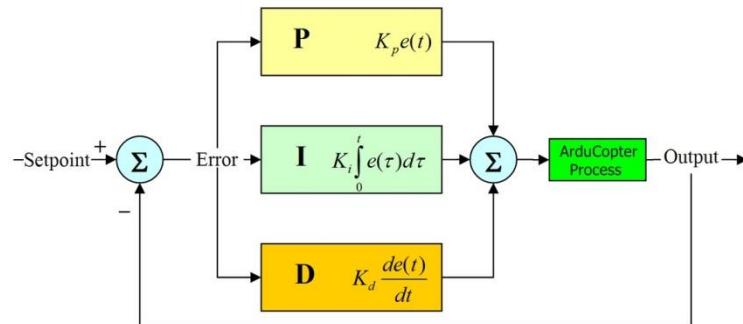
การทำงานของระบบ	K_p	K_i	กราฟแสดงการทำงานณ สภาวะสมดุล	เวลาที่ใช้ (วินาที)
แบบสัดส่วน (P) + แบบปริพันธ์ (I)	0.1	0.25		25 (***) เร็วที่สุด

ตอนที่ 3 การจูนค่าและปรับค่า K_p , K_i และ K_d (แบบ PID)

โดยการให้ค่า $K_p = 0, 0.1, 0.15$

โดยการให้ค่า $K_i = 0, 0.1, 0.2$

โดยการให้ค่า $K_d = 0, 0.002, 0.004, 0.008, 0.016$



การทำงานของระบบควบคุมแบบพีไอดี (Proportional + Integral + Derivative)

ผลการทดลอง

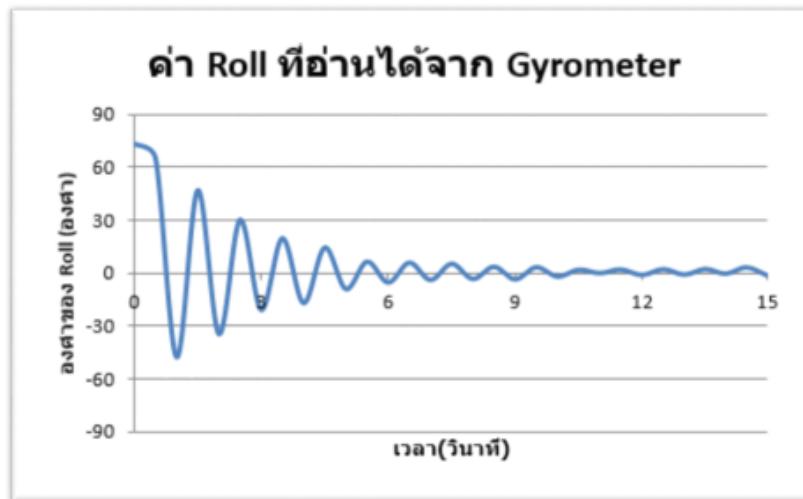
การทำงานของระบบ	K_p	K_i	K_d	กราฟแสดงการทำงานณ สภาวะสมดุล	เวลาที่ใช้ (วินาที)
แบบสัดส่วน (P) + แบบปริพันธ์ (I) + แบบอนุพันธ์ (D)	0.15	0.1	0.004		23 (***) เร็วที่สุด

ค่าที่สามารถนำไปใช้จริงได้

การทำงาน	K_p	K_i	K_d
แบบ P	0.15	-	-
แบบ PI	0.1	0.25	-
แบบ PID	0.15	0.1	0.004

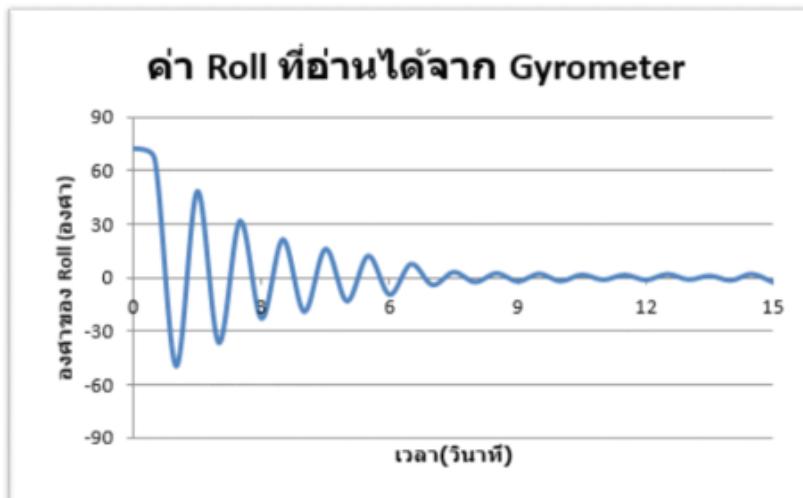
การทำงานของระบบควบคุมโดยการปรับค่า K_p อย่างเดียว (แบบ P)

กราฟแสดงค่า Roll ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ณ สภาวะสมดุล



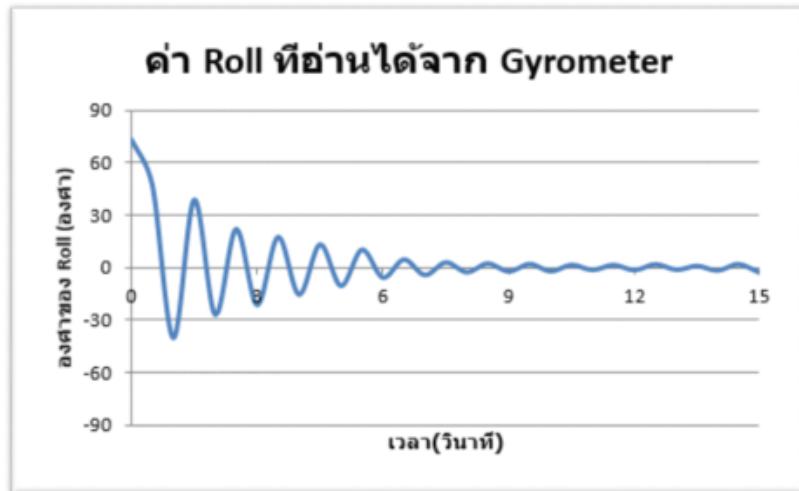
การทำงานของระบบควบคุมโดยการปรับค่า K_p และ K_i (แบบ PI)

กราฟแสดงค่า Roll ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ณ สภาวะสมดุล



การทำงานของระบบควบคุมโดยการปรับค่า K_p , K_i และ K_d (แบบ PID)

กราฟแสดงค่า Roll ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ณ สภาวะสมดุล



สรุปผลการทดลอง

จากการทดลองพบว่า การปรับแบบ P หรือปรับเฉพาะค่า K_p จะเห็นว่าระบบจะเข้าสู่สภาวะสมดุลช้า และเกิด Overshoot เยอะมาก ส่วนการปรับแบบ PI หรือทำการเพิ่มค่า K_i เข้ามาจะเห็นว่าระบบจะเข้าสู่สภาวะสมดุลช้ากว่าแบบ P แต่ Overshoot จะมีการลดลงมาก และอันสุดท้ายการปรับแบบ PID ซึ่งเป็นการนำค่า K_p , K_i และ K_d เข้ามาใช้ในการคำนวณระบบควบคุม จะทำให้ระบบเข้าสู่สภาวะสมดุลเร็วมากและไม่มีการเกิด overshoot หรือมีการเกิดที่น้อยมาก

4.14 การทดลองที่ 14 ทดสอบการหมุน ที่แกน Pitch ของตัวโดรนขณะที่บอร์ดมีการเปลี่ยนแปลงตำแหน่ง (มีการปรับค่า K_p , K_i , K_d)

จุดประสงค์

เพื่อทำการจูนและหาค่า K_p , K_i , K_d ที่ทำให้เกิดความสมดุลหรือ stable

อุปกรณ์

- 1.บอร์ด ArduPilotMega 2.8
- 2.สาย Micro USB
- 3.โปรแกรม Arduino IDE สำหรับ ArduPilot
- 4.โปรแกรม Missionplanner

วิธีการทดลอง

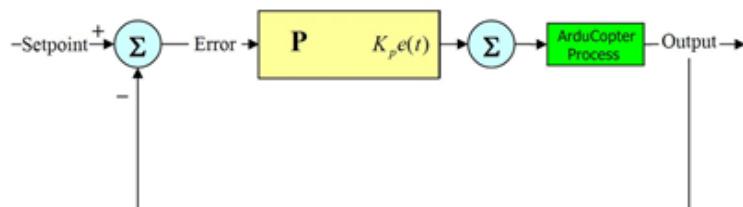
1.ทำการปรับค่า K_p , K_i หรือ K_d เริ่มต้นจาก 0 จากนั้นทำการเขียนโปรแกรมแสดง output เป็น text file และทำการ plot กราฟ

2.ทำการเพิ่มหรือเปลี่ยนแปลงค่า K_p , K_i หรือ K_d ไปเรื่อยๆ จนกว่าจะพบจุดที่เป็น Stable หรือจุดที่เกิด overshoot

3.ทำการทดลองทั้ง 3 แบบ คือ แบบ P โดยเปลี่ยนแปลงค่า K_p , แบบ PI โดยเปลี่ยนแปลงค่า K_p และ K_i และแบบ PID โดยเปลี่ยนแปลงค่า K_p , K_i และ K_d

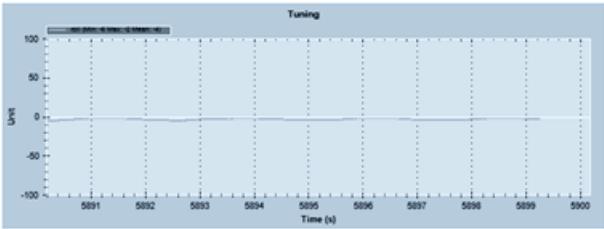
ตอนที่ 1 การจูนและปรับค่าเฉพาะ K_p (แบบ P)

โดยการให้ค่า $K_p = 0, 0.05, 0.1, 0.125, 0.15$



การทำงานของระบบควบคุมแบบพี (Proportional)

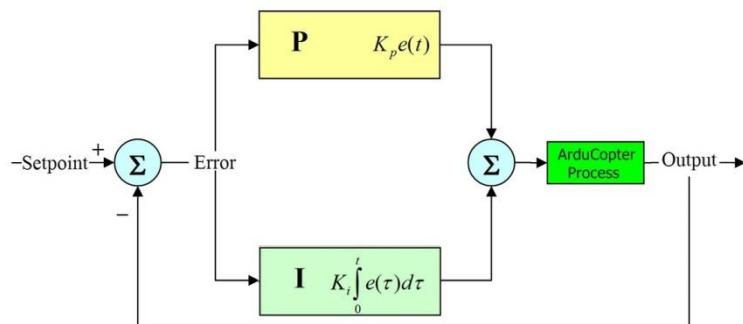
ผลการทดลอง

การทำงานของระบบ	K_p	กราฟแสดงการทำงาน ณ สภาวะสมดุล	เวลาที่ใช้ (วินาที)
แบบสัดส่วน (P)	0.125		27 (***) เร็วที่สุด

ตอนที่ 2 การจูนและปรับค่าเฉพาะ K_p และ K_i (แบบ PI)

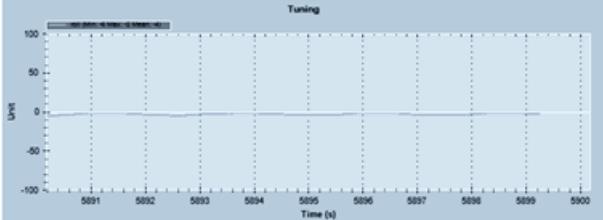
โดยการให้ค่า $K_p = 0, 0.05, 0.1, 0.15$

โดยการให้ค่า $K_i = 0, 0.05, 0.1, 0.15, 0.2, 0.23, 0.25$



การทำงานของระบบควบคุมแบบพิโอล (Proportional + Integral)

ผลการทดลอง

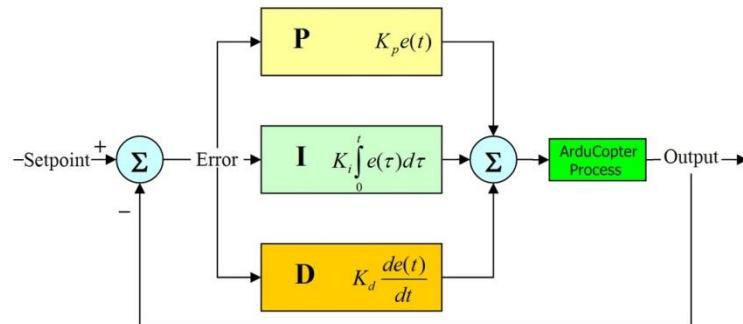
การทำงานของระบบ	K_p	K_i	กราฟแสดงการทำงาน ณ สภาวะสมดุล	เวลาที่ใช้ (วินาที)
แบบสัดส่วน (P) + แบบปริพันธ์ (I)	0.1	0.23		23 (***) เร็วที่สุด

ตอนที่ 3 การจูนค่าและปรับค่า K_p , K_i และ K_d (แบบ PID)

โดยการให้ค่า $K_p = 0, 0.1, 0.15$

โดยการให้ค่า $K_i = 0, 0.3, 0.6$

โดยการให้ค่า $K_d = 0, 0.0025, 0.0050, 0.01, 0.0125$



การทำงานของระบบควบคุมแบบพีไอดี (Proportional + Integral + Derivative)

ผลการทดลอง

การทำงานของระบบ	K_p	K_i	K_d	กราฟแสดงการทำงาน ณ สภาวะสมดุล	เวลาที่ใช้ (วินาที)
แบบสัตส่วน (P) + แบบปริพันธ์ (I) + แบบอนุพันธ์ (D)	0.15	0.6	0.0125		22 (***) เร็วที่สุด

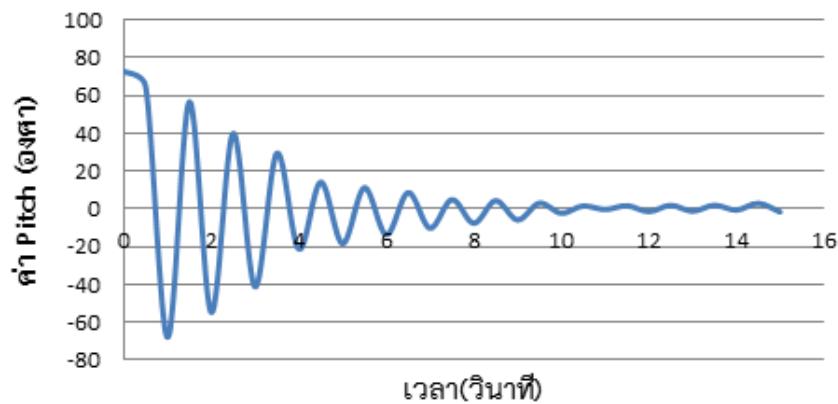
ค่าที่สามารถนำไปใช้จริงได้

การทำงาน	K_p	K_i	K_d
แบบ P	0.125	-	-
แบบ PI	0.1	0.23	-
แบบ PID	0.15	0.6	0.0125

การทำงานของระบบควบคุมโดยการปรับค่า K_p อย่างเดียว (แบบ P)

กราฟแสดงค่า Roll ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ณ สภาวะสมดุล

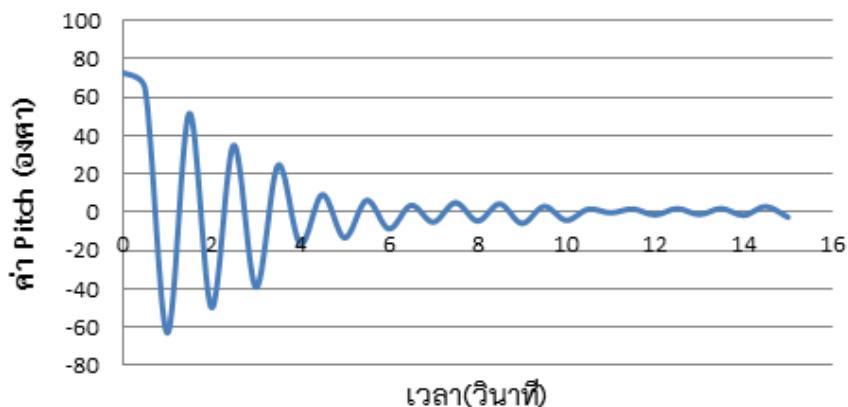
ค่า Pitch ที่อ่านได้จาก Gyrometer



การทำงานของระบบควบคุมโดยการปรับค่า K_p และ K_i (แบบ PI)

กราฟแสดงค่า Roll ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ณ สภาวะสมดุล

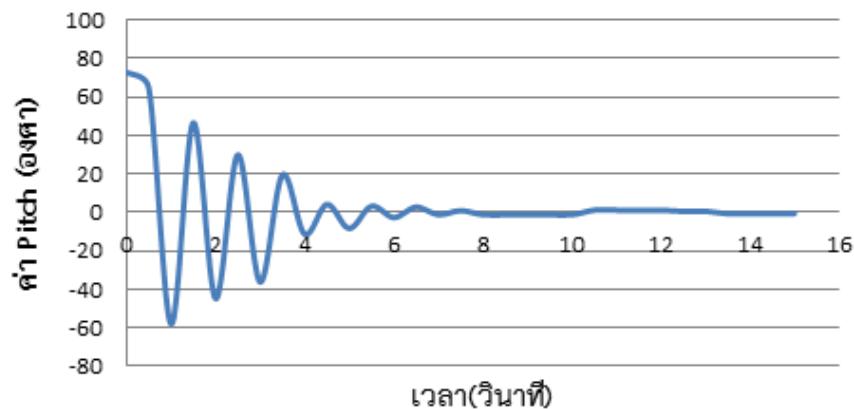
ค่า Pitch ที่อ่านได้จาก Gyrometer



การทำงานของระบบควบคุมโดยการปรับค่า K_p , K_i และ K_d (แบบ PID)

กราฟแสดงค่า Roll ที่อ่านจาก Gyrometer ของบอร์ด ArduPilotMega ณ สภาวะสมดุล

ค่า Pitch ที่อ่านได้จาก Gyrometer



สรุปผลการทดลอง

จากการทดลองพบว่า การปรับแบบ P หรือปรับเฉพาะค่า K_p จะเห็นว่าระบบจะเข้าสู่สภาวะสมดุลช้า และเกิด Overshoot เยอะมาก ส่วนการปรับแบบ PI หรือทำการเพิ่มค่า K_i เข้ามาจะเห็นว่าระบบจะเข้าสู่สภาวะสมดุลช้ากว่าแบบ P แต่ Overshoot จะมีการลดลงมาก และอันสุดท้ายการปรับแบบ PID ซึ่งเป็นการนำค่า K_p , K_i และ K_d เข้ามาใช้ในการคำนวณระบบควบคุม จะทำให้ระบบเข้าสู่สภาวะสมดุลเร็วมากและไม่มีการเกิด overshoot หรือมีการเกิดที่น้อยมาก

4.15 การทดลองที่ 15 การทดสอบการบินของโดรนโดยไม่มีการจูนค่า

จุดประสงค์

เพื่อทำการเช็คการบินของตัวโดรนก่อนที่จะทำการปรับจูนค่า

อุปกรณ์

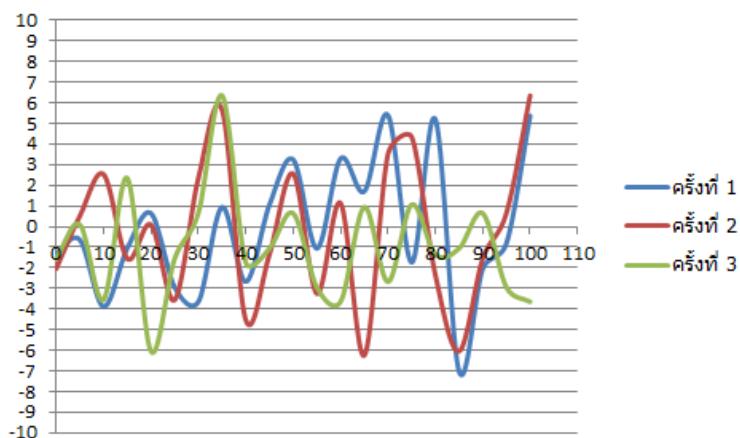
- 1.ตัวโดรนที่ทำการประกอบเสร็จเรียบร้อยแล้ว

วิธีการทดลอง

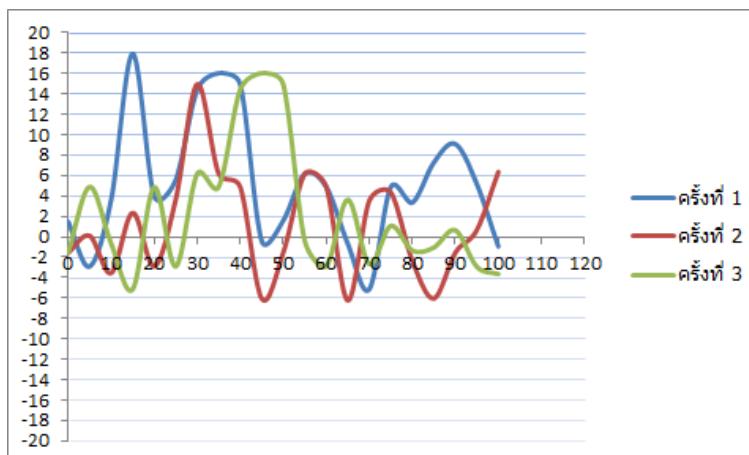
- 1.ทำการตรวจสอบอุปกรณ์ที่ตัวโดรนให้เรียบร้อย
- 2.ทำการ ARM เพื่อเป็นการปลดล็อคระบบความปลอดภัยของโดรนเพื่อป้องกันการเกิดอุบัติเหตุ
- 3.เริ่มทำการบินโดยเก็บค่า Roll กับ Pitch ผ่านตัว Telemetry

ผลการทดลอง

ที่แกน Roll



ที่แกน Pitch



4.16 การทดลองที่ 16 การทดสอบการบินของโดรนโดยไม่มีแรงจากแรงลม (จุนค่าแล้ว)

จุดประสงค์

เพื่อทำการเช็คการบินของตัวโดรนหลังจากการปรับจุนค่าเรียบร้อยแล้ว(ไม่มีแรงลมประท)
อุปกรณ์

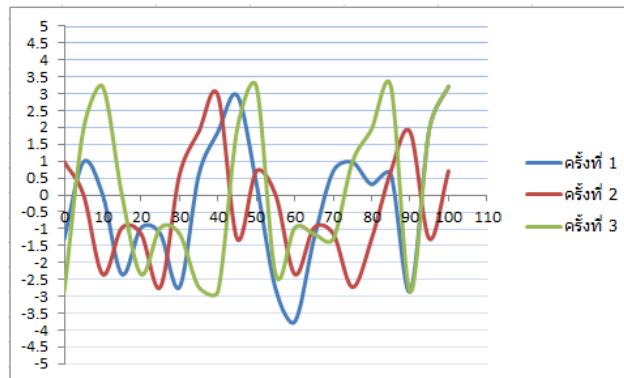
- 1.ตัวโดรนที่ทำการประกอบเสร็จเรียบร้อยแล้ว

วิธีการทดลอง

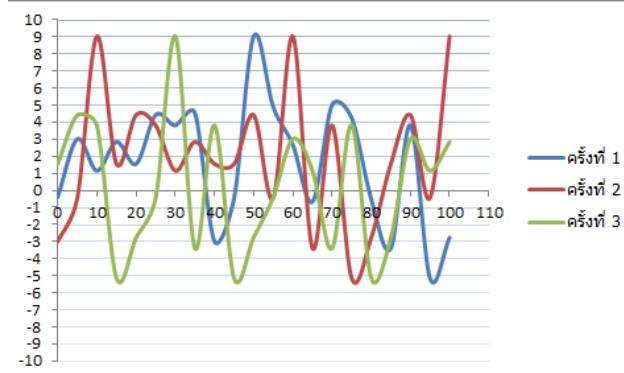
- 1.ทำการตรวจสอบอุปกรณ์ที่ตัวโดรนให้เรียบร้อย
- 2.ทำการ ARM เพื่อเป็นการปลดล็อคระบบความปลอดภัยของโดรนเพื่อป้องกันการเกิดอุบัติเหตุ
- 3.เริ่มทำการบินโดยเก็บค่า Roll กับ Pitch ผ่านตัว Telemetry

ผลการทดลอง

ที่แกน Roll



ที่แกน Pitch



4.17 การทดลองที่ 17 การทดสอบการบินของโดรนโดยมีแรงลมปะทะจากลมของพัดลม

จุดประสงค์

เพื่อทำการเช็คการบินของตัวโดรนหลังจากการปรับจุนค่าเรียบร้อยแล้ว โดยมีการปะทะจากแรงลมของพัดลม

อุปกรณ์

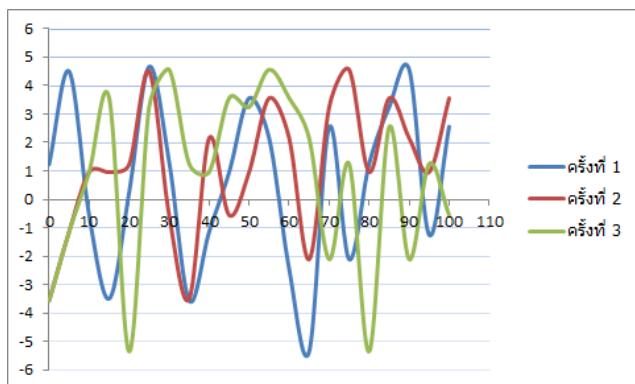
- 1.ตัวโดรนที่ทำการประกอบเสร็จเรียบร้อยแล้ว
- 2.พัดลมขนาด 12 นิ้ว

วิธีการทดลอง

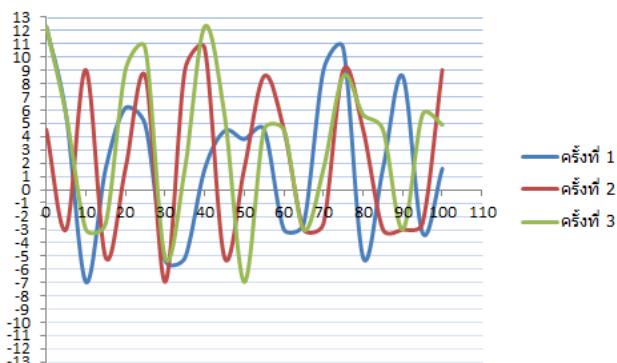
- 1.ทำการตรวจสอบค่าอุปกรณ์ที่ตัวโดรนให้เรียบร้อย
- 2.ทำการ ARM เพื่อเป็นการปลดล็อคระบบความปลอดภัยของโดรนเพื่อป้องกันการเกิดอุบัติเหตุ
- 3.เริ่มทำการบินโดยและทำการเพิ่มแรงลมเข้ามากระทบกับตัวโดรนทั้งด้านบนในพัดและด้านใต้ใบพัด
- 4.ทำการเก็บค่า Roll กับ Pitch ผ่านตัว Telemetry

ผลการทดลอง

ที่แกน Roll



ที่แกน Pitch



4.18 การทดลองที่ 18 การปรับค่าแบบ PID โดยให้ค่ามีความละเอียดขึ้น

ຈຸດປະສົງ

เพื่อทำการจุนและหาค่า K_p , K_i , k_d และเพื่อตัดความแตกต่าง

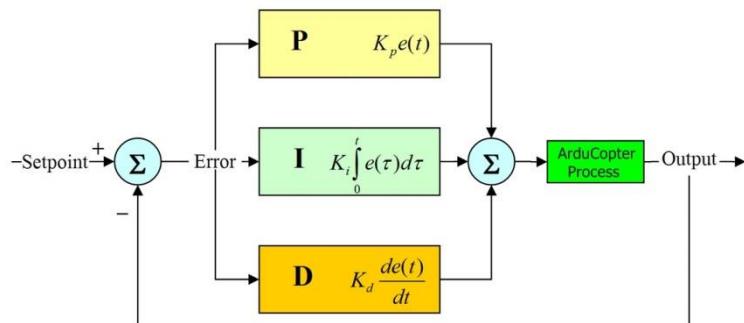
อุปกรณ์

- 1.บอร์ด ArduPilotMaga 2.8
 - 2.สาย Micro USB
 - 3.โปรแกรม Arduino IDE สำหรับ ArduPilot
 - 4.โปรแกรม Missionplanner

วิธีการทดลอง

1. ทำการปรับค่า K_p , K_i หรือ K_d เริ่มต้นจาก 0 จากนั้นทำการเขียนโปรแกรมแสดง output เป็น text file และทำการ plot กราฟ

2. ทำการเพิ่มหรือเปลี่ยนแปลงค่า K_p , K_i หรือ K_d ไปเรื่อยๆ จนกว่าจะพบจุดที่เป็น Stable หรือจุดที่เกิด overshoot



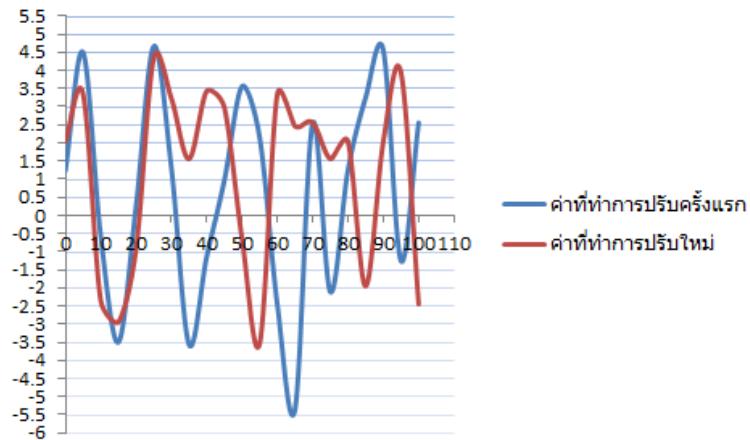
การทำงานของระบบควบคุมแบบพิโอดี (Proportional + Integral + Derivative)

ค่าที่สามารถนำไปใช้จริงได้

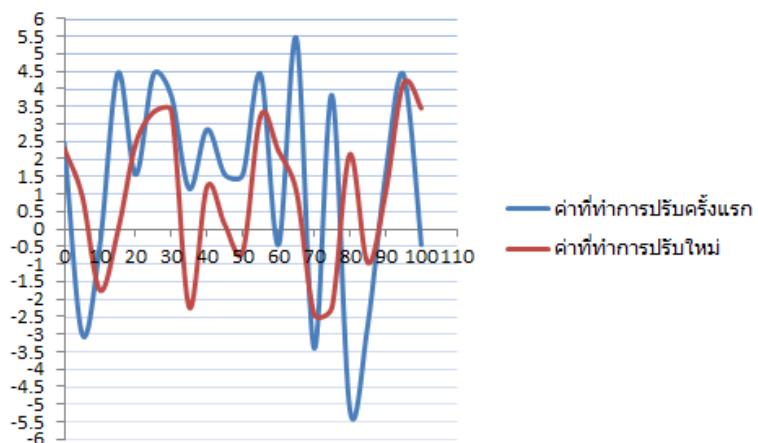
การทำงาน	K_p	K_i	K_d
ค่าเดิม (Roll)	0.15	0.1	0.004
ค่าใหม่ (Roll)	0.145	0.095	0.0045
ค่าเดิม (Pitch)	0.15	0.6	0.0125
ค่าใหม่ (Pitch)	0.145	0.605	0.01255

กราฟที่ได้จากการปรับค่า

ที่แกน Roll



ที่แกน Pitch



4.19 การทดลองที่ 19 การทดสอบการบินของโดรนโดยใช้ค่าในกรณีต่าง ๆ

จุดประสงค์

เพื่อทำการทดสอบการบินของตัวโดรนโดยใช้ค่าที่แตกต่างกัน

อุปกรณ์

1. ตัวโดรนที่ทำการประกอบเสร็จเรียบร้อยแล้ว

วิธีการทดลอง

1. ทำการตรวจสอบคุณภาพของตัวโดรนให้เรียบร้อย

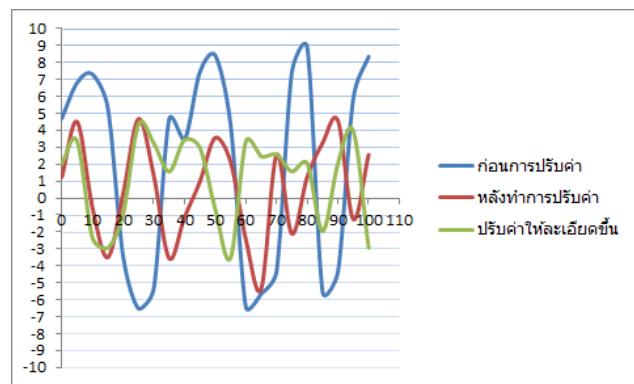
2. ทำการ ARM เพื่อเป็นการปลดล็อกระบบความปลอดภัยของโดรนเพื่อป้องกันการเกิดอุบัติเหตุ

3. เริ่มทำการบินโดยทำการเพิ่มแรงลงเข้ามากระแทกตัวโดรนทั้งด้านบนไปพัดและด้านใต้ไปพัด

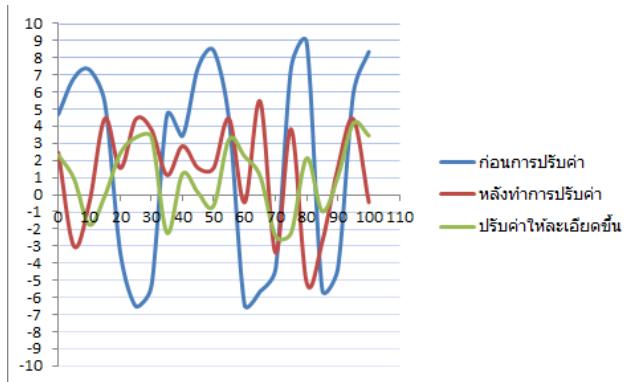
4. ทำการเก็บค่า Roll กับ Pitch ผ่านตัว Telemetry

ผลการทดลอง

ที่แกน Roll



ที่แกน Pitch



บทที่ 5 สรุปผลการดำเนินงาน

ในบทนี้จะเป็นการสรุปผลการดำเนินงานของโครงการ ปัญหา อุปสรรค และข้อเสนอแนะ เพื่อเป็นแนวทางในการนำผลงานที่ได้ไปพัฒนาต่อเพื่อให้เกิดศักยภาพในการใช้งานที่ดีขึ้นในอนาคต

5.1 สรุปการทำงาน

ในส่วนของระบบควบคุมแบบ PID ตัวโดรนสามารถบินได้นิ่งกว่าเดิมมากขึ้นหลังจากการปรับแต่งค่า PID

5.2 ปัญหาและอุปสรรค

- บางครั้งมีการเกิดการผิดพลาดขึ้นทำให้ตัวโดรนตกมีการเสียหาย ต้องรอสักวันสองวันเพื่อมาซ่อมแซมจึงสามารถกลับมาบินได้ปกติ
- ระบบ Autopilot พึงการทำงานของ Sensor เป็นหลัก บางครั้ง Sensor ทำงานผิดปกติทำให้เกิดอันตรายขึ้นกับโดรน เช่น ตกโดยควบคุมไม่ได้
- บางครั้งการทำงานของ Motor กับ ESC มีประสิทธิภาพไม่เท่ากันเนื่องการสาเหตุหลายปัจจัย เช่น เมื่อโดรนต้องจะมีมอเตอร์บางตัวได้รับแรงกระแทกในขณะที่ต้องจึงเกิดความเสียหายต่อตัวมอเตอร์ถึงแม้จะ calibrate และรักษาตาม

บรรณานุกรม

bob. (18 เมษายน 2017). *Mission Planner.* (bob RC) เรียกใช้เมื่อ 25 เมษายน 2017 จาก <http://bob-lopburi.blogspot.com/2014/03/apm-setup-compass.html>

Chang. (3 April 2010). ควบรวมข้อมูลเครื่องบินปีก – Quadrotor กันดีกว่า. (Ayarafun) เรียกใช้เมื่อ 27 February 2017 จาก <http://www.ayarafun.com/2010/04/what-is-quadrotor/>

DMCA. (13 March 2013). ทฤษฎีของโครน. (DMCA) เรียกใช้เมื่อ 27 February 2017 จาก <http://readgur.com/doc/2158007/>

Mantra. (May 31 2015). โครน (*drone*) ความรู้เบื้องต้นและราคา. (WordPress.) เรียกใช้เมื่อ 27 February 2017 จาก <http://www.thairc.com/>

Mindphp dot com. (2 March 2017). *Arduino IDE.* (Mindphp) เรียกใช้เมื่อ 25 April 2017 จาก <http://www.mindphp.com/>

Phantom Thailand. (24 May 2014). เจาะลึกเรื่องใบพัด อุปกรณ์ชิ้นเด็ก แต่มีผลใหญ่! (Phantom Thailand) เรียกใช้เมื่อ 27 February 2017 จาก <http://www.phantomthailand.com/>

SWIFTLET Thailand. (31 January 2017). *SWIFTLET.* เรียกใช้เมื่อ 16 November 2017 จาก SWIFTLET: <https://swiftlet.co.th/>

thaicontrol01. (27 November 2011). *PID Tuning.* เรียกใช้เมื่อ 16 November 2017 จาก Thaicontrol's Blog: <https://thaicontrol.wordpress.com/2011/11/27/pid-tuning/>

กุลธิดา เด่นวิทยานันท์. (31 October 2015). “Drone” เทคโนโลยีอากาศยานไร้คนขับ. (PwC) เรียกใช้เมื่อ 27 February 2017 จาก <http://www.pwc.com/th/en/press-room/pwc-thailand-blog/2016/20161031-blog.html>

วิกิพีเดีย. (21 April 2017). ภาษาซีพลัสพลัส. (วิกิพีเดีย) เรียกใช้เมื่อ 25 April 2017 จาก <https://th.wikipedia.org/wiki/ภาษาซีพลัสพลัส>

วิกิพีเดีย สารานุกรมเสรี. (23 September 2014). ระบบควบคุมพีไอดี. เรียกใช้เมื่อ 16 November 2017 จาก วิกิพีเดีย สารานุกรมเสรี: <https://th.wikipedia.org/wiki/ระบบควบคุมพีไอดี>

อาจารย์ชนม์รัตน์ ตติยะวนันท์. (14 June 2015). ฝึกปรับแต่ง PID. เรียกใช้เมื่อ 16 November 2017 จาก คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีมหานคร: http://www.eng.mut.ac.th/article_detail.php?id=97