ระบบควบคุมอากาศยานด้วยคอมพิวเตอร์

AIRCRAFT CONTROL SYSTEM BY COMPUTER

คณนันท์ สิทธิพงศ์พิทยา
อภิสิทธิ์ วชิรจิรากร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมแมคคาทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2557

ระบบควบคุมอากาศยานด้วยคอมพิวเตอร์
AIRCRAFT CONTROL SYSTEM BY COMPUTER

คณนันท์ สิทธิพงศ์พิทยา
อภิสิทธิ์ วชิรจิรากร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมแมคคาทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2557

AIRCRAFT CONTROL SYSTEM BY COMPUTER

KANANAN SITTIPONGPITAYA
APISIT WACHIRAJIRAGONE

THE THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN MECHATRONICS ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2014

ปริญญานิพนธ์ปีการศึกษา 2557

ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมอากาศยานด้วยคอมพิวเตอร์

AIRCRAFT CONTROL SYSTEM BY COMPUTER

ผู้จัดทำ นายคณนันท์ สิทธิพงศ์พิทยา 54010148

นายอภิสิทธิ์ วชิรจิรากร 54011502

.....อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์)

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร.เกียรติศักดิ์ คมวัชระ)

ระบบควบคุมอากาศยานด้วยคอมพิวเตอร์

โดย
นายคณนันท์ สิทธิพงศ์พิทยา 54010148
นายอภิสิทธิ์ วชิรจิรากร 54011502

อาจารย์ที่ปรึกษา
ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์
รองศาสตราจารย์ ดร.เกียรติศักดิ์ คมวัชระ

ปีการศึกษา 2557

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการนำเสนอการออกแบบระบบควบคุมอากาศยานด้วยคอมพิวเตอร์ ซึ่งในปัจจุบันได้มีการเดินทางหรือการขนส่งด้วยอากาศยานกันเป็นจำนวนมากที่ต้องอาศัยความชำนาญ รวมถึงในกรณีนักบินไม่อยู่ในสภาพที่จะบังคับอากาศยานได้ เราจึงได้นำระบบควบคุมด้วยคำสั่งทางคอมพิวเตอร์มาใช้ควบคุม เพื่อให้การควบคุมอากาศยานมีประสิทธิภาพความแม่นยำและมีความถูกต้องมากยิ่งขึ้น โดยผู้ควบคุมสามารถควบคุมตัวอากาศยานผ่านคอมพิวเตอร์จากสถานที่ไหนก็ได้ ทำให้เกิดความปลอดภัยมากยิ่งขึ้นได้

ในการทดลองเราได้เริ่มจากการเลือกอากาศยานที่จะใช้ทำการทดลอง ศึกษาการทำงานของตัวอากาศยานหรือ Quadrotor การออกแบบระบบที่ใช้ในการควบคุม เลือกไมโครคอนโทรลเลอร์ที่เหมาะสม เพื่อให้ระบบสามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพ ทำการค้นคว้าก่อนปฏิบัติและทำการทดสอบเพื่อทำการบันทึกผล และสรุปผล

AIRCRAFT CONTROL SYSTEM BY COMPUTER

By
Mr.Kananan Sittipongpitaya 54010148
Mr.Apisit Washirajirakorn 54011502

Advisor
Asst.Prof.Sumit Panaudomsup
Assoc.Prof.Dr.Kiattisak Kumwachara

Academic Year 2014

ABSTRACT

This project presents The Aircraft Control System by Computer isbeingconducted bythepresentations is creating of the aircraft control system by computer. Currently, To travel or airliftthat requiresa lot ofexpertise.Includingthepilot was notcontrolling of theaircraft.We have thecomputer control used to the aircraftcontrol effectively,Precision and accuracy.Theoperatorcan control theaircraft by computer from anywhere.

In our experiment, we began with the selection of the sample aircraft, followed by the detailed study and research into the function of the aircraft - the quadrotor. when designing the control system, we chose an appropriate microcontroller for the system towork correctly and efficiently.Researched and testedbefore practiceto recordthe results andconclusions.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับความช่วยเหลือเป็นอย่างดี จาก ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์ และรองศาสตราจารย์.ดร.เกียรติศักดิ์ คมวัชระ ที่ได้ให้คำปรึกษาแนะนำที่ดีมาโดยตลอดตั้งแต่ต้น กลุ่มผู้จัดทำขอขอบพระคุณผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์ และรองศาสตราจารย์.ดร.เกียรติศักดิ์ คมวัชระ เป็นอย่างสูง

ขอบคุณเพื่อนๆ ทุกคนที่ให้กำลังใจ สนับสนุนอุปกรณ์ที่ขาดเหลือ และการทำงานกันเป็นทีมกระตุ้นเตือน รวมทั้งคอยถามไถ่ความคืบหน้าของโครงงานเสมอ

สุดท้ายนี้ผู้จัดทำขอกราบขอบพระคุณบิดา มารดา และครอบครัว ที่คอยเป็นกำลังใจที่ดีตลอดมารวมถึงการสนับสนุนในเรื่องของงบประมาณที่ขาดเหลือ ตลอดจนเป็นแรงบันดาลใจที่ดีที่สุดที่ทำให้โครงงานนี้สำเร็จสมบูรณ์ลงได้

ผู้จัดทำ
นายคณนันท์ สิทธิพงศ์พิทยา
นายอภิสิทธิ์ วชิรจิรากร

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในปัจจุบันการคมนาคมและการขนส่งต่างๆ สามารถทำได้อย่างรวดเร็ว สะดวก และมีประสิทธิภาพมากขึ้น เนื่องจากได้มีการนำเทคโนโลยีสมัยใหม่มากมายเข้ามาใช้งาน ทำให้การดำเนินงานทำได้ง่ายและมีประสิทธิภาพสูงกว่าในอดีต โดยเฉพาะการคมนาคมทางอากาศ เนื่องจากเป็นวิธีการที่รวดเร็วที่สุดในโลก เพราะไม่ต้องประสบปัญหาการจราจรต่างๆ และมีความปลอดภัยสูง แต่การขนส่งทางอากาศก็ยังคงมีความเสี่ยงอยู่บ้าง เนื่องจากสภาพอากาศหรือปัจจัยต่างๆ อย่างเช่น ในบางพื้นที่ที่การเดินทางเข้าไปด้วยอากาศยานเป็นไปด้วยความลำบากจากหลากหลายสาเหตุ อาทิ เช่น พายุ หรือสภาพอากาศอันเลวร้าย และอาจเกิดอุบัติเหตุได้ ทำให้นักบินและลูกเรือมีโอกาสประสบกับสถานการที่ก่อให้เกิดอันตรายถึงชีวิต และทรัพย์สิน ในโครงงานนี้จึงได้นำระบบควบคุมอากาศยานเข้ามาช่วยในการควบคุม โดยมีจุดประสงค์ที่จะใช้โปรแกรมคอมพิวเตอร์เป็นตัวช่วยในการควบคุมการทำงานแทนมนุษย์ โดยเราใช้ตัวอากาศยานเป็นเฮลิคอปเตอร์ 4 ใบพัด หรือ Quadrotor และใช้บอร์ดไมโครคอนโทรลเลอร์ คือ ArduinoUNOR3 เป็นส่วนของระบบควบคุมโดยรับสัญญาณจากรีโมทคอนโทรลที่ส่งสัญญาณอินพุตให้แก่บอร์ดไมโครคอนโทลเลอร์ หลังจากนั้นจะเกิดการประมวลผลเป็นคำสั่งโปรแกรมการควบคุมการทำงาน และการเคลื่อนที่เป็นสัญญาณค่าเอาต์พุตไปยังตัว Quadrotor เพื่อทำงานตามคำสั่งที่เราต้องการและยังมีการใช้กล้อง Pixy CMU Cam5 เพื่อทำการตรวจจับวัตถุและสามารถสั่งการให้ Quadrotor เคลื่อนที่ติดตามวัตถุนั้นๆ หรือหลบหลีกวัตถุนั้นๆได้ โดยโครงงานนี้ถูกจัดทำขึ้นเพื่อเป็นต้นแบบของระบบควบคุมอากาศยานเพื่อใช้ในการศึกษาและต่อยอดต่อไป

1.2 วัตถุประสงค์ของโครงงาน

1. ศึกษาการใช้ความรู้และทฤษฎีต่างๆ ทางวิศวกรรมเพื่อการประยุกต์ใช้งาน
2. ศึกษาการทำงานและการประยุกต์ใช้วัสดุและอุปกรณ์ทางอิเล็กทรอนิกส์ และแมคคานิกส์ในการออกแบบและเลือกใช้ไมโครคอนโทรลเลอร์ที่จะใช้ในการควบคุมระบบได้
3. ศึกษาการออกแบบระบบควบคุมในบอร์ดไมโครคอนโทรลเลอร์และเลือกโปรแกรมที่ใช้ในการเขียนไมโครคอนโทรลเลอร์ได้อย่างเหมาะสม
4. โครงงานนี้ถูกจัดทำขึ้นเป็นต้นแบบในการศึกษาของบุคคลทั่วไป

1.3 ขั้นตอนการดำเนินโครงงาน

1. ศึกษาและหาข้อมูลความรู้และทฤษฎีต่างๆ จากสื่อต่างๆ อาทิเช่น หนังสือ งานวิจัย และอินเตอร์เน็ต ที่จะใช้สำหรับออกแบบระบบควบคุมอากาศยาน
2. กำหนด วางแผน และออกแบบระบบควบคุมอากาศยาน
3. ทำการเลือกวัสดุและอุปกรณ์ทางอิเล็กทรอนิกส์ และแมคคานิกส์ที่เหมาะสมสำหรับการออกแบบ Quadrotor และการออกแบบระบบควบคุมอากาศยาน
4. ดำเนินงานสร้างชิ้นงานที่ได้ออกแบบไว้

5. ทดสอบระบบที่ได้ออกแบบว่าสามารถปฏิบัติการได้ถูกต้องตามคำสั่งหรือไม่
6. สรุปผลการทดลอง

1.4 กรอบแนวคิดในการศึกษาโครงงาน

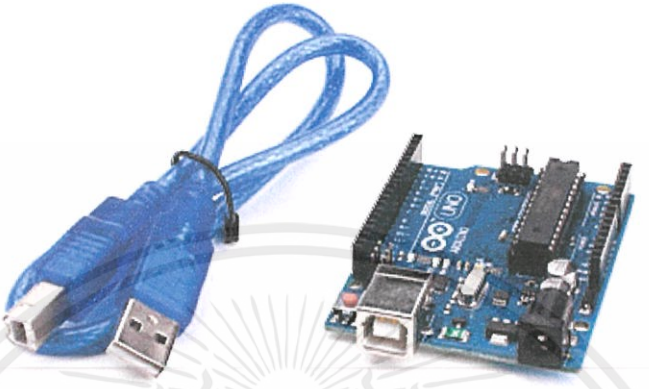
แนวคิดในการศึกษาโครงงานเรื่องนี้ มาจากการเขียนโค้ดโปรแกรมคำสั่งการควบคุมการทำงานและการเคลื่อนที่ของ Quadrotor โดยส่งอินพุตไปยังบอร์ดไมโครคอนโทลเลอร์เพื่อการประมวลผลคำสั่ง และคำนวณแปลค่าคำสั่งออกไปเป็นเอาต์พุตไปยัง Quadrotor ที่เราต้องการ และยังได้มีการเพิ่มฟังก์ชั่นของกล้องเซนเซอร์ Pixy CMU Cam5 เข้ามาเพื่อช่วยในการควบคุมการทำงานในส่วนของการตรวจจับวัตถุ ติดตาม และการหลบหลีกวัตถุที่เราไม่ต้องการได้อย่างประสิทธิภาพ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำความรู้และทฤษฎีต่างๆ ทางด้านวิศวกรรมมาประยุกต์ใช้งานได้จริง
2. สามารถนำความรู้มาทำการเลือกใช้วัสดุและอุปกรณ์ทางอิเล็กทรอนิกส์ และแมคคานิกส์เพื่อสร้างวงจรระบบควบคุมอื่นๆ ที่เหมาะสมได้
3. สามารถนำความรู้ทางด้านการเขียนโปรแกรมภาษาซี และอื่นๆ มาใช้ในการเขียนโปรแกรมในระบบควบคุมอื่นๆ ได้
4. เป็นต้นแบบที่สามารถใช้ต่อยอดในงานอื่นๆ ได้

บทที่ 2
อุปกรณ์ หลักการและทฤษฎี

2.1 ไมโครคอนโทรลเลอร์ Arduino UNO R3

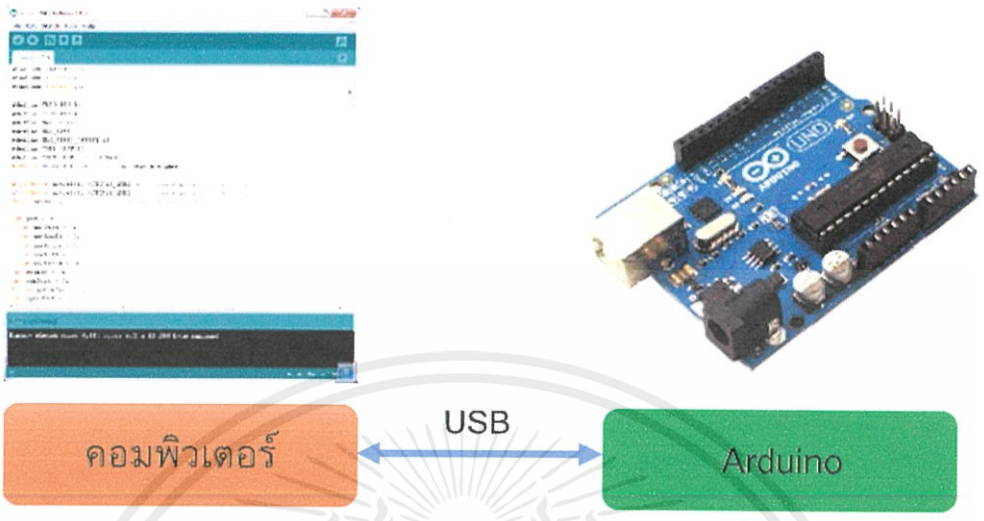


รูปที่ 2.1 Arduino UNO R3

2.1.1 Arduino คืออะไร

Arduino อ่านว่า (อา-ดู-อิ-โน่ หรือ อาดุยโน่) เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้อีกด้วย ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่างๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิคส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด หรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆ เช่น Arduino Xbee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino Wireless Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย

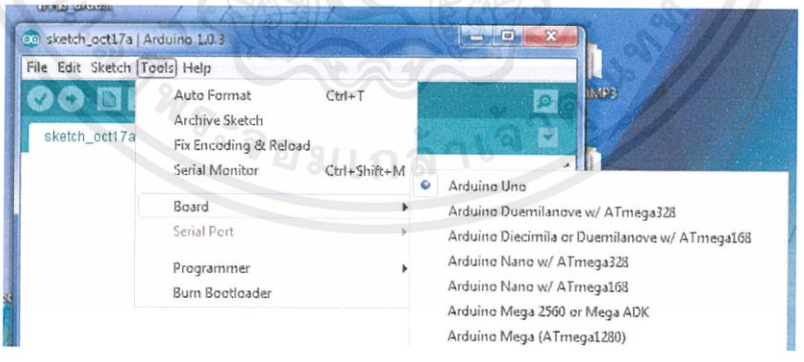
2.1.2 รูปแบบการเขียนโปรแกรมบน Arduino



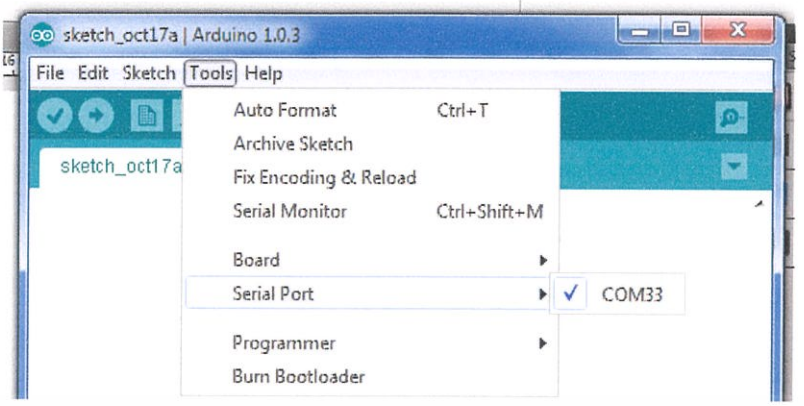
รูปที่ 2.2 การเขียนโปรแกรมบน Arduino

1. เขียนโปรแกรมบนคอมพิวเตอร์ ผ่านทางโปรแกรม ArduinoIDE ซึ่งสามารถดาวน์โหลดได้จาก Arduino.cc/en/main/software

2. หลังจากที่เขียนโค้ดโปรแกรมเรียบร้อยแล้ว ให้ผู้ใช้งานเลือกรุ่นบอร์ด Arduino ที่ใช้และหมายเลข Comport



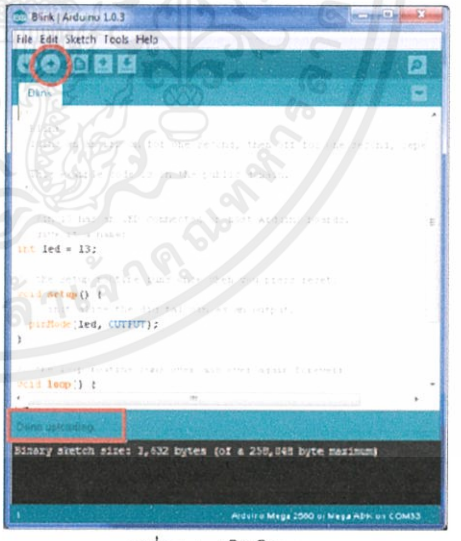
รูปที่ 2.3 เลือกรุ่นบอร์ด Arduino ที่ต้องการ Upload



รูปที่ 2.4 เลือกหมายเลข Comport ของบอร์ด

3. กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรมจากนั้นกดปุ่ม Upload โค้ดโปรแกรมไปยังบอร์ด Arduino ผ่านทางสาย USB เมื่ออัพโหลดเรียบร้อยแล้วจะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที

รูปที่5 กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม



รูปที่6 Upload โค้ดโปรแกรม

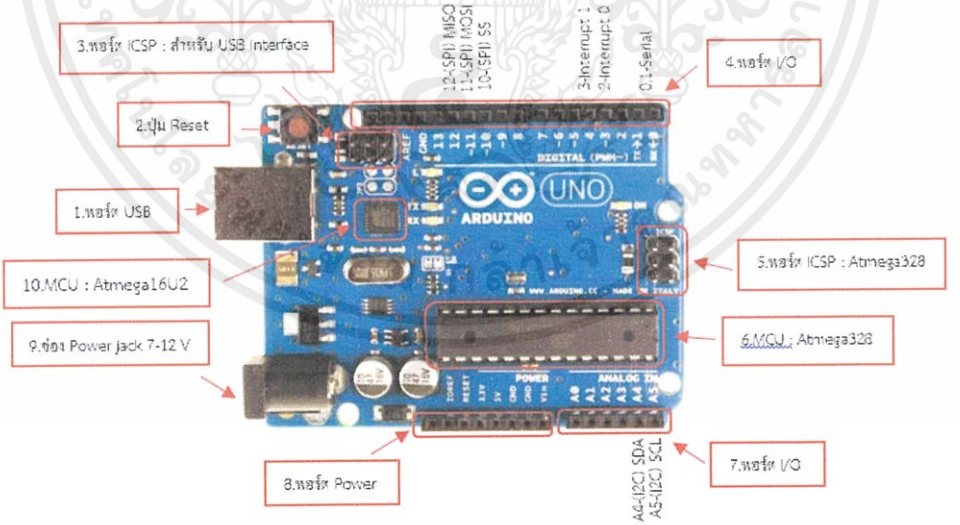
รูปที่ 2.5 การตรวจสอบ และ การอัพโหลด

2.1.3 ตัวอย่าง Code ของ Arduino Uno R3

โปรแกรมควบคุมการกระพริบของหลอดไฟ LED สีแดง โดยให้หลอดไฟกระพริบติด และหน่วงเวลา 1 วินาที หรือ 1000 มิลลิวินาที และดับ 1 วินาที หรือ 1000 มิลลิวินาที และให้หลอดไฟติดอีกครั้งใน1 วินาที หรือ 1000 มิลลิวินาที และวนการทำงานดังกล่าวต่อไปเรื่อยๆ จนจบการทำงานของโปรแกรม

```
int led = 13;
voidsetup()
{
    pinMode(led, OUTPUT);
}
   voidloop()
{
     digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
   delay(1000);
}
```

2.1.4 Layout & Pin out Arduino Board



รูปที่ 2.6 Layout & Pin out Arduino Board

1. **USBPort :** ใช้สำหรับต่อกับ Computer เพื่ออัพโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด

2. **Reset Button :** เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่

3. **CSP Port :** ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Comport บน Atmega16U2

4. **I/Oport :** Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0, 1 เป็นขา Tx, Rx Serial, Pin 3, 5, 6, 9, 10 และ 11 เป็นขา PWM

5. **ICSP Port :** Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader

6. **MCU :** Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino

7. **I/Oport :** นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็นช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0 - A5

8. **Power Port :** ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, V_{in}

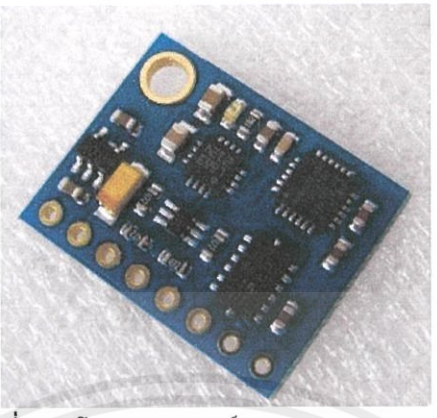
9. **Power Jack :** รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7 - 12 V

10. **MCU** ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U2

2.1.5 Arduino Uno R3

Arduino Uno R3 บอร์ดไมโครคอนโทรลเลอร์แบบ Open-source บนแพลตฟอร์มออกแบบมาให้ใช้งานได้ง่าย ใช้ชิพ ATmega328 รันที่ความถี่ 16 MHz หน่วยความจำแฟลช 32 KB แรม 2 KB บอร์ดใช้ไฟเลี้ยง 7 ถึง 12 V มีระดับแรงดันไฟฟ้าในการทำงานและขาสัญญาณอยู่ที่ 5 V (TTL) มี Digital Input/Output 14 ขา (เป็น PWM ได้ 6 ขา) มี Analog Input 6 ขา Serial UART 1 ชุด I2C 1 ชุด SPI 1 ชุด เขียนโปรแกรมบนซอฟท์แวร์ Arduino IDE และโปรแกรมผ่านพอร์ต USB เหมาะสำหรับผู้ที่สนใจเริ่มต้นเรียนรู้การพัฒนาไมโครคอนโทรลเลอร์ หรือแม้แต่ผู้ที่ไม่เคยเรียนรู้ด้านอิเล็กทรอนิกส์มาก่อนก็สามารถนำมาสร้างต้นแบบที่เกี่ยวกับอิเล็กทรอนิกส์ได้

2.2 โมดูลเซนเซอร์ GY-85 IMU/9DOF (ITG3205 ADXL345 HMC5883L)



รูปที่ 2.7 โมดูลเซนเซอร์ GY-85 IMU/9DOF

2.2.1 รายละเอียดโมดูลเซนเซอร์ GY-85 IMU/9DOF

ข้อมูลเบื้องต้น (Introduction/Overview)

GY-85 เป็นโมดูลเซนเซอร์ Accelerometers & Gyroscope & Compass ซึ่งสามารถทางานได้ทั้ง 3 อย่างในเวลาเดียวกัน ใช้ในการตรวจสอบทิศการเคลื่อนที่และสามารถใช้ในการตรวจสอบความเร็วในการเปลี่ยนแปลงทิศทางของแกน X Y Z ได้ยกตัวอย่าง ถ้าวัตถุเกิดการเคลื่อนที่หรือเอียง Output ของ Accelerometer จะบอกค่าของการเอียงว่าสถานะปัจุบันค่าของ X Y Z อยู่ที่เท่าไร แต่ Gyroscope จะวัดค่าได้ตอนที่กาลังเอียงหรือตอนกาลังเคลื่อนไหวเท่านั้น เมื่อวัตถุหยุดนิ่งค่างของ Gyroscope จะวัดไม่ได้เพราะไม่มีการเคลื่อนไหวและสามารถเป็นเข็มทิศได้ ในเวลาเดียวกันบนโมดูลประกอบด้วยชิป ADXL345 HMC5883L ITG3205 ส่งข้อมูลผ่าน Bus I2C

คุณสมบัติ (Features)

- ใช้ไฟเลี้ยง +3.3 ถึง+5 V
- ชิป ADXL345 HMC5883L ITG3205
- เชื่อมต่อผ่านบัส I2C

การนำไปประยุกต์ใช้งาน (Application Ideas)

- ตรวจสอบทิศทางการเคลื่อนที่เคลื่อนไหวต่างๆ ของวัตถุข้อควรระวังในการใช้งาน (Caution / Warning)
- ควรหลีกเหลี่ยงการต่อวงจรให้เกิดการลัดวงจร
- ควรอ่านเอกสารก่อนการต่อวงจรจริง
- ไม่ควรใช้ไฟเกินตามที่เอกสารกำหนด

คุณลักษณะ (Specification)

- อุณหภูมิที่รองรับ -40 to +85 °C
- รองรับแรงดัน 3.3 – 5 V
- ทดสอบการตกกระแทกที่ 1.8 เมตร

- ขนาด : 17 mm * 22 mm

2.3 เซนเซอร์ Ultrasonic Module HC-SR04 Distance Measuring Transducer Sensor

รูปที่ 2.8 เซนเซอร์ Ultrasonic Sensor

เซนเซอร์ Ultrasonic ใช้เสียงสะท้อนกลับในการคำนวณวัดระยะทาง ความแม่นยำที่ 2 – 400 cm ใช้ง่ายและมีไลบารี่มาตรฐานพร้อมใช้งาน

การใช้งาน โมดูลวัดระยะทาง Ultrasonic Module Distance Measuring Transducer Sensor กับ Arduino

การวัดระยะทางโดยใช้ โมดูล Ultrasonic ร่วมกับ Arduino สามารถทำได้ง่าย อุปกรณ์โมดูล Ultrasonic มีความแม่นยำในการวัดระยะทาง การทำงานเป็นแบบคลื่นสะท้อนกลับแล้วนำมาคำนวณระยะทาง จึงเหมาะสำหรับมาใช้ในการหลบหลีกสิ่งกีดขวาง ตรวจจับวัตถุที่อยู่ในรัศมีที่ต้องการ

วิธีการต่อขาใช้งานโมดูลวัดระยะทาง Ultrasonic Module Distance Measuring Transducer Sensor กับ Arduino

- Vcc – 5v
- Gnd – Gnd
- Trig – 13
- Echo – 12

2.4 รีโมทคอนโทรล

เราเลือกใช้ RadioLink T6EHP 2.4Ghz 6CH เครื่องส่งสัญญาณวิทยุและ Receiver (สำหรับ 6CH 3D Helicpter บิน) ดิจิตอล 2.4GHz 6 ช่องทาง RC ระบบ T6EHP นำเทคโนโลยี 2.4 GHz FHSS มาใช้สำหรับเฮลิคอปเตอร์ 3D เนื่องจากการทำงานที่ทันสมัยที่สุดวิทยุ 2.4 GHz เทคโนโลยีการควบคุมเป็นสิ่งจำเป็นเพื่อให้เครื่องส่งและเครื่องรับสัญญาณระบบ RC เครื่องบินบังคับดีที่สุด ในขณะเดียวกันสามารถหลีกเลี่ยงปัญหาการทำงานที่ผิดพลาดได้ดี เพราะมีกลไกที่ทันสมัยในรหัสที่ตรงกัน ดังรูปที่ 2.9

รูปที่ 2.9 รีโมทคอนโทรล Radio Link

ข้อมูลรายละเอียด

ชื่อโมเดล	2.4G 6Ch
ยี่ห้อ	RadioLink
โมเดล	T6EHP-E
ประเภทโวลต์	Standard Voltage
หมายเลขโมเดล	T6EHP-E
ปริมาณ	Standard
ช่องรับสัญญาณ	6 Channel
น้ำหนัก (กรัม)	640
ช่อง	6 Channel
เข้ากันกับ	FUTABA 6EX TREX T-REX 450 500
ปลอก	2.4 กรัม
ประเภทระบบ	FHSS 2.4GHz
ประเภทเครื่องส่งสัญญาณ	Panel Style
ตัวรับสัญญาณ	R7EH 2.4G
โวลต์ปฏิบัติการ	9V
ระยะทาง	900 m
น้ำหนัก	50 กรัม
ตัวเลือกการส่ง	โหมด 2
ระบบการควบคุม	RC System T6EHP Adopted FHSS 2.4GHz
ควบคุม	Panel Style
Power Adapter Option	อะแดปเตอร์ UK
ตามการใช้งาน	เฮลิคอปเตอร์

2.4.1 การใช้งานของรีโมทคอนโทรล

รูปที่ 2.10 ตำแหน่งการใช้งานของรีโมท Radio Link

รายละเอียดของตำแหน่งการใช้งานของรีโมทคอนโทรล มีดังนี้

1. เสาอากาศ (Antenna)
   - สำหรับส่งสัญญาณวิทยุไปยังตัวรับสัญญาณ Receiver
2. สวิตซ์ไจโรสโคป (Gyro Switch)
   - สำหรับปรับการทำงานของไจโรสโคป
3. คันโยกควบคุมการยกระดับ (Elevator and Rudder Control Stick)
   - สำหรับควบคุมการเคลื่อนที่ในแนวระดับ
4. ตัวปรับการควบคุมคันโยก (Trimmer)
   - สำหรับการเบี่ยงเบนการควบคุมของคันบังคับ
5. จอมอนิเตอร์ (LCD Screen)
   - สำหรับอ่านค่าระดับการทำงานของ Quadrotor
6. สวิตซ์ล็อคความเร่ง (Throttle Hold Switch)
   - สำหรับควบคุมความเร่งคงที่ของ Quadrotor
7. คันโยกควบคุมทิศทาง (Aileron and Throttle Control Stick)
   - สำหรับควบคุมทิศทางการเคลื่อนที่
8. ไฟเตือนการทำงาน (Power LED)
   - สำหรับเตือนการใช้งาน และหยุดใช้งานของรีโมทคอนโทรล

9. ปุ่มเปิด/ปิดสวิตช์ (On/Off Switch)
   - สำหรับการเปิดและปิดสวิตช์ที่เราต้องการ
10. ปุ่มปรับโหมดการทำงาน (Mode Button)
   - สำหรับปรับเลือกการใช้งานของรีโมทคอนโทรล
11. ปุ่มรับข้อมูลอินพุต (Data Input Button)
   - สำหรับปุ่มกำหนดค่าอินพุตสำหรับการใช้งานรีโมทคอนโทรล
12. ปุ่มเลือกการทำงาน (Select Button)
   - สำหรับเลือกการทำงานของข้อมูลต่างๆ ของรีโมทคอนโทรล

2.5 รีซีฟเวอร์ (Receiver)

Receiver คือ อุปกรณ์ที่ใช้สำหรับรับสัญญาณความถี่ 2.4 GHz จากเครื่องส่ง แล้วทำการแปลงสัญญาณเพื่อไปควบคุมการทำงานของ Speed Control ของแต่ละช่องสัญญาณประกอบด้วยเสาอากาศและแผงวงจรไฟฟ้าเพื่อคอยรับสัญญาณจากเครื่องส่ง และควบคุมให้มอเตอร์ของเฮลิคอปเตอร์ทำงานโดยผ่าน ESC ดังรูปที่ 2.11

รูปที่ 2.11 เครื่องรับสัญญาณ (Receiver)

2.6 มอเตอร์กระแสตรงแบบไร้แปรงถ่าน (Brushless DC Motor)

Brushless Motor คือ มอเตอร์ชนิดที่ไม่แปรงถ่านหรือมอเตอร์ซิงโครนัส 3 เฟสที่ทำงานโดยอาศัยอุปกรณ์อิเล็กทรอนิกส์กำลังเป็นสวิตช์ในการตัดต่อกระแสที่จ่ายให้กับขดลวดมอเตอร์ โดยที่ชนิดของมอเตอร์จะพิจารณาตามลักษณะรูปคลื่นกระแส และคุณสมบัติของแรงบิดหรือทอร์คโดยจะนิยมเรียกว่า Brushless DC Motor ในกรณีที่รูปแบบของกระแสและทอร์คของมอเตอร์ที่ใช้มีลักษณะเป็นแบบสี่เหลี่ยม (Trapezoidal Current/Torque) Brushless DC Motor เป็นมอเตอร์ไฟฟ้าที่ใช้พลังงานไฟฟ้ากระแสตรง และอาศัยระบบอิเล็กทรอนิกส์คอมพิวเตอร์ในการหมุน โดยภายในจะมีขดลวดวางทำมุมห่างกัน 120 องศาในโครงงานนี้เลือกใช้มอเตอร์บรัชเลสขนาด 1000 KV กระแส 30 แอมป์

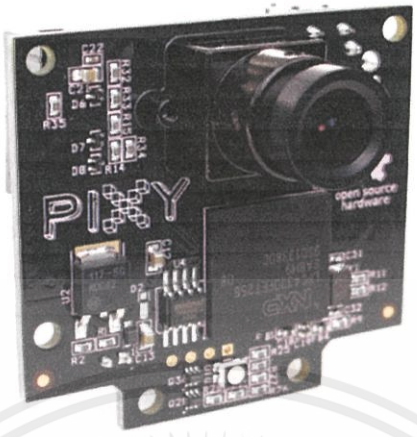
รูปที่ 2.12 มอเตอร์กระแสตรงแบบไร้แปรงถ่าน (Brushless DC Motor)

2.7 ESC (Electronic Speed Controller)

Speed Control เป็นอุปกรณ์อิเล็กทรอนิกส์ประเภทหนึ่งที่ประกอบด้วยวงจรในการควบคุมความเร็วของมอเตอร์ ซึ่งสำหรับเฮลิคอปเตอร์ 4 ใบพัดนั้นจำเป็นต้องมีอุปกรณ์นี้เพื่อใช้ในการควบคุมมอเตอร์ โดยรับสัญญาณการควบคุมมาจาก Receiver ซึ่งจะสัมพันธ์กับตำแหน่งของคันโยกที่เราขยับ เช่น ถ้าตำแหน่งคันโยกอยู่ตำแหน่งต่ำสุด ESC จะไม่จ่ายกำลังไฟฟ้าให้กับมอเตอร์ ถ้าตำแหน่งคันโยกอยู่ที่กึ่งกลาง ESC จะจ่ายกำลังไฟฟ้าให้กับมอเตอร์ 50% เป็นต้น

รูปที่ 2.13 ESC (Electronic Speed Controller)

2.8 กล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5)

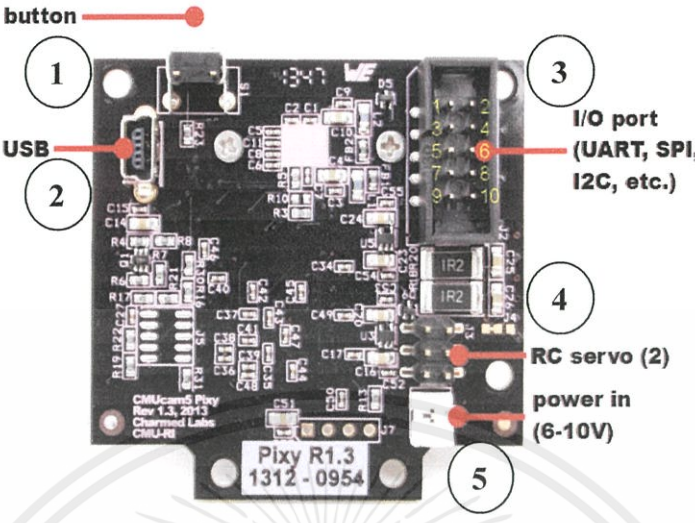


รูปที่ 2.14 กล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5)

กล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5) คือ กล้องที่มีบอร์ดเซนเซอร์ติดตั้งมาด้วย สามารถอ่านค่าติดตามวัตถุที่มีสีฉูดฉาด หรือสีเข็มสว่างได้ โดยแสดงออกผ่านจอคอมพิวเตอร์ด้วยโปรแกรม Pixy Mon จะสามารถอ่านค่าของสี และกำหนดชื่อและลักษณะของวัตถุได้ ซึ่งหลักการนี้สามารถใช้กล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5) เชื่อมต่อกับบอร์ด Arduino UNO R3 เพื่อให้ Quadrotor สามารถบินติดตามวัตถุได้ โดยการเขียนโปรแกรมคำสั่งให้กล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5) ติดตามวัตถุที่ต้องการ และสั่งระบบควบคุมให้เซอร์โวมอเตอร์เคลื่อนที่ตามวัตถุนั้นๆ โดยกำหนดค่าที่เซอร์โวมอเตอร์แต่ละตัว ระบบควบคุมกล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5) เป็นอีกฟังก์ชั่นหนึ่งที่จัดทำขึ้นเพื่อตรวจจับวัตถุในระยะใกล้ เพื่อสามารถติดตามหรือหลบหลีกวัตถุนั้น และทำงานตามคำสั่งโปรแกรมได้อัตโนมัติได้ ซึ่งแยกจากการควบคุมด้วยรีโมทคอนโทรลกล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5) จึงเป็นฟังก์ชั่นเสริมการทำงานอีกฟังก์ชั่นหนึ่งเพื่อสร้าง Quadrotor ที่มีประสิทธิภาพสูงในทุกรูปแบบการใช้งาน

รูปที่ 2.15 โปรแกรม Pixymon แสดงภาพในการตรวจจับวัตถุ

2.8.1 องค์ประกอบของบอร์ดกล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5)



รูปที่ 2.16 องค์ประกอบของกล้องเซนเซอร์ติดตามวัตถุ (PIXY CMU Cam5)

รายละเอียดขององค์ประกอบของบอร์ดกล้องเซนเซอร์ (PIXY CMU Cam5)

1. Button
   - ปุ่มกดเพื่อโฟกัสภาพ ในการตรวจจับ และจดจำวัตถุที่ต้องการ
2. USB
   - ช่องต่อ USB เพื่อแสดงภาพผ่านคอมพิวเตอร์ด้วยโปรแกรมPixyMon
3. พอร์ต UART
   - พอร์ตที่สามารถเชื่อมต่อกับบอร์ด Arduino UNO R3
4. RC Servo(2)
   - ช่องพอร์ตสำหรับต่อกับเซอร์โวมอเตอร์ที่ใช้ในระบบควบคุม
5. Power In(6-10V)
   - ช่องพอร์ตสำหรับจ่ายไฟเลี้ยง 6-10 V ของกล้องเซ็นเซอร์ติดตามวัตถุ

2.9 USB to UART Cable (PL2303 HX)

รูปที่ 2.17 USB to UART Cable (PL2303 HX)

USB to UART Cable (PL2303 HX) เป็นสายที่เชื่อมต่อระหว่างบอร์ด Arduio ผ่านทางพอร์ต Rx และ Tx เชื่อมต่อกับคอมพิวเตอร์ด้วยพอร์ต USB ใช้ในการรับส่งข้อมูลโปรแกรม เพื่อรับค่าและส่งค่าต่างๆ เข้าหา บอร์ด Arduino โดยตรง โดยมีสายต่อกับบอร์ด Arduino 4 เส้นและมีการต่อสายแต่ละเส้นดังนี้

Black	- GND (0V)
Red	- +5V
Green	- Tx
White	- Rx

2.10 แบตเตอรี่ (Lithinum Polymer Battery3000 mAh)

รูปที่ 2.18 แบตเตอรี่ (Lithinum Polymer Battery *3000* mAh)

แบตเตอรี่ (Lithinum Polymer Battery 3000 mAh) มีน้ำหนักเบาโดยมีความจุ 3000 mAh สามารถจ่ายกระแสได้ปริมาณมากกว่าความจุ ทำให้มีความเร็วเพิ่มขึ้น และแรงดันคงที่ตลอดเวลา จะทำให้ Quadrotor สามารถใช้งานได้เป็นเวลายาวนานและไม่เกิดการกระตุกระหว่างการใช้งาน ทำให้การเคลื่อนที่ของ Quadrotor มีประสิทธิภาพมากยิ่งขึ้น โดยแบตเตอรี่นี้สามารถชาร์ตความจุได้ด้วยเครื่องชาร์ต จึงเหมาะแก่การใช้งานกับ Quadrotor

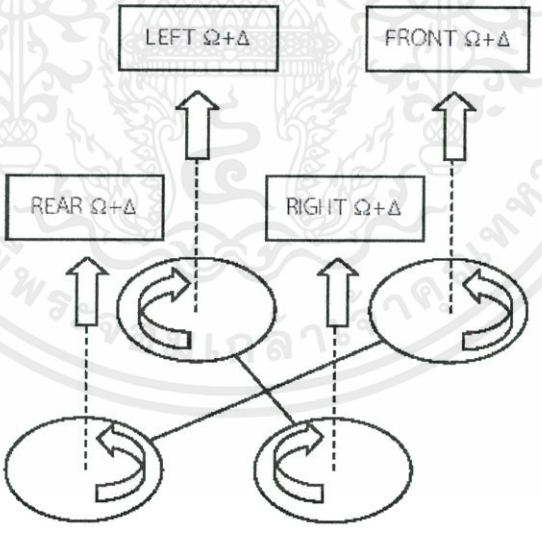
2.11 หลักการควบคุมเฮลิคอปเตอร์ 4 ใบพัด

การควบคุมเฮลิคอปเตอร์ 4 ใบพัดทำได้โดยการเปลี่ยนความเร็วของใบพัด ซึ่งทำให้แรงบิดและแรงยกของแต่ละใบพัดเปลี่ยนไป หลักการควบคุมเฮลิคอปเตอร์ 4 ใบพัดในแบบต่างๆ ทำได้ดังนี้ กำหนดสัญลักษณ์และความหมายดังต่อไปนี้

1. Ω หมายถึง ความเร็วขณะนั้น
2. Δ หมายถึง ความเร็วที่เปลี่ยนไป
3. Ö หมายถึง การเร่งความเร็วในแนวดิ่ง (Throttle)
4. R หมายถึง การเอียงตัวไปทางขวา (Roll)
5. P หมายถึง การเอียงตัวไปข้างหน้า (Pitch)

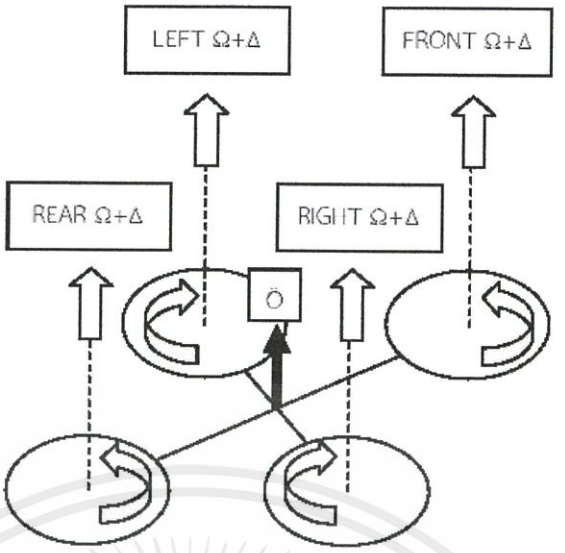
รูปที่ 2.19 Quadrotor Body

การลอยตัวนิ่ง (Hovering) ทำได้โดยควบคุมความเร็วของ ESC โดยสั่งให้เซอร์โวมอเตอร์ขับเคลื่อนใบพัดทั้งสี่ตัวหมุน เพื่อสร้างแรงยกในปริมาณความเร็วที่เท่ากัน โดยใบพัดแต่ละคู่จะหมุนในทิศทางตรงข้ามกันคือใบพัดหน้าและหลังจะหมุนทวนเข็มนาฬิกา ใบพัดซ้ายและขวาจะหมุนตามเข็มนาฬิกาจะก่อให้เกิดแรงบิดจากใบพัดแต่ละคู่จะหักล้างกันทำให้ Quadcopter ลอยตัวนิ่งได้ดังรูปที่ 2.20



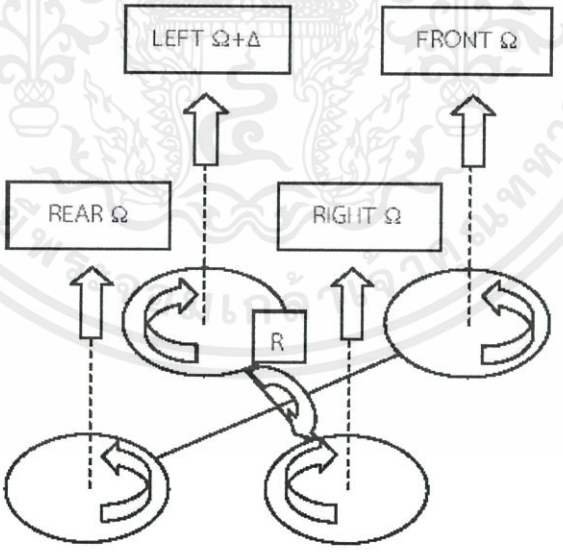
รูปที่ 2.20 การลอยตัวนิ่ง (Hovering)

การเร่ง – ลดความเร็วในแนวดิ่ง (Throttle) ทำได้โดยควบคุมความเร็วของ ESC โดยสั่งให้เซอร์โวมอเตอร์ขับเคลื่อนใบพัดทั้งสี่มีปริมาณความเร่งที่เท่าๆ กันทั้งหมดจะก่อให้เกิดแรงยก หรือแรงต้าน เพื่อการยกระดับขึ้นหรือลงเพื่อเร่งความเร็ว Quadropter บินขึ้น–ลง ได้อย่างสมดุลดังรูปที่ 2.21



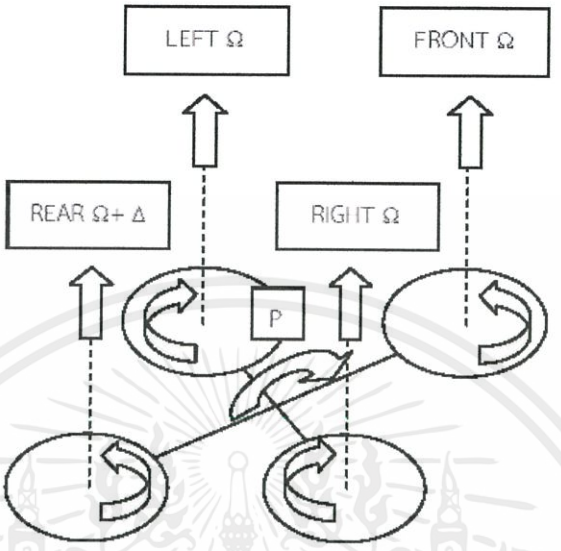
รูปที่ 2.21 การเร่ง – ลดความเร็วในแนวดิ่ง (Throttle)

การเอียงตัวซ้าย-ขวา (Roll) ได้โดยควบคุมความเร็วของ ESC โดยสั่งให้เซอร์โวมอเตอร์ขับเคลื่อนใบพัดหน้า (Front) และหลัง (Rear) จะมีความเร็วเท่าเดิม แต่ความเร็วใบพัดซ้าย (Left) จะหมุนเร็วขึ้น ทำให้มีความเร็วเพิ่มขึ้นทำให้ทิศทางนี้ยกตัวขึ้นและในส่วนของใบพัดขวา (Right) จะช้าลง ทำให้ทิศทางนี้มีความเร็วตกลงทำให้เกิดการเอียงตัวไปทางขวาส่วนเอียงตัวซ้ายก็ใช้วิธีตรงกันข้ามดังรูปที่ 2.22



รูปที่ 2.22 การเอียงตัวซ้าย-ขวา (Roll)

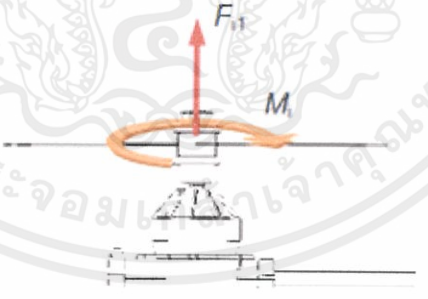
การเอียงตัวหน้า-หลัง (Pitch) วิธีนี้คล้ายกับการเอียงตัวซ้าย-ขวา แต่เปลี่ยนเป็นให้ใบพัดซ้าย–ขวาและข้างหน้าให้มีความเร็วคงที่ แต่ความเร็วใบพัดหลังจะหมุนเร็วขึ้นทำให้ทิศทางนี้ยกตัวทำให้บินเอียงไปข้างหน้าดังรูปที่ 2.23 ส่วนการเอียงตัวทางด้านหลังก็ใช้วิธีตรงกันข้าม



รูปที่ 2.23 การเอียงตัวหน้า-หลัง (Pitch)

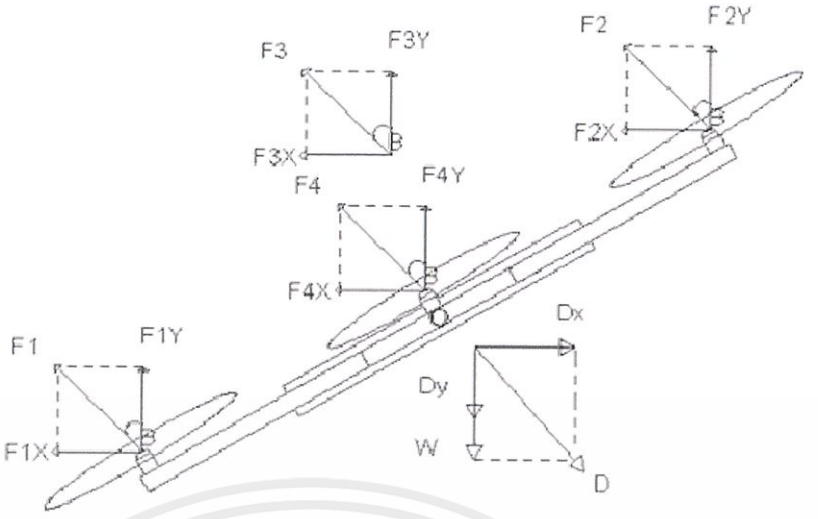
2.12 พลศาสตร์การเคลื่อนที่

เมื่อมอเตอร์หมุนจะทำให้เกิดแรงยกที่มีทิศทางตามแรง F ซึ่งจะอธิบายได้ดังต่อไปนี้



รูปที่ 2.24 การหมุนของมอเตอร์

ในการเคลื่อนที่ของเฮลิคอปเตอร์จะมีทั้งหมดอยู่ 3 แกนคือ แกน X Y และ Z แต่ถ้าพิจารณาในเรื่องสมดุลแล้วแรงยกจะถูกแบ่งออกเป็นในแนวแกน X และแกน Y



รูปที่ 2.25 การคำนวณแรงตามแนวแกน X Y และ Z

จากกฎของนิวตัน $\Sigma F = ma$ (2.1)

ดังนั้นจะได้ $F1 + F2 + F3 + F4 - D - W = ma$ (2.2)

รวมแรงในแนวแกน X จะได้ $F1X + F2X + F3X + F4X - DX = max$ (2.3)

เมื่อแตกแรงเข้าแกน X จะได้ $F1\sin B + F2\sin eB + F3\sin eB + F4\sin eB - Dx = max$ (2.4)

หาความเร่งในแนวแกน X ได้เป็น $ax = \frac{(F1+ F2 + F3B+ F4)\sin eB - Dx}{m}$ (2.5)

รวมแรงในแนวแกน Y จะได้ $F1Y + F2Y + F3Y + F4Y - W - DY = may$ (2.6)

เมื่อแตกแรงเข้าแกน Y จะได้ $F1\cos B + F2\cos B + F3\cos B + F4\cos B - W - Dy = may$ (2.7)

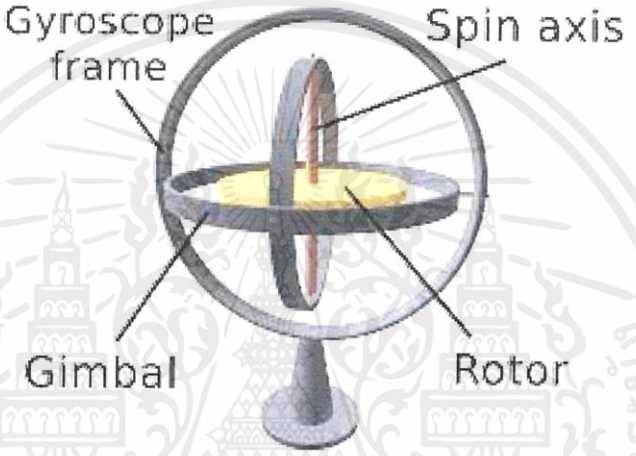
หาความเร่งในแนวแกน Y ได้เป็น $ay = \frac{(F1+ F2 + F3 + F4)\cos B - W - Dy}{m}$ (2.8)

2.13 Inertial Measurement Unit (IMU)

เซนเซอร์หรืออุปกรณ์ที่ใช้ใน INS (Inertial Navigation System) ถูกเรียกว่า Inertial Measurement Units (IMU) ซึ่งเป็นส่วนประกอบหลักของ INSs ที่ใช้ในเครื่องบิน, ยานอวกาศ และเรือ เช่นเดียวกับจรวดขีปนาวุธ IMU ประกอบด้วย 2 ส่วนหลักคือ Accelerometers 3 ทิศทาง และ Gyroscopes 3 ทิศทาง ซึ่งรับความเร่งยานพาหนะและความเร็วเชิงมุมตามลำดับ

2.13.1 Gyroscope

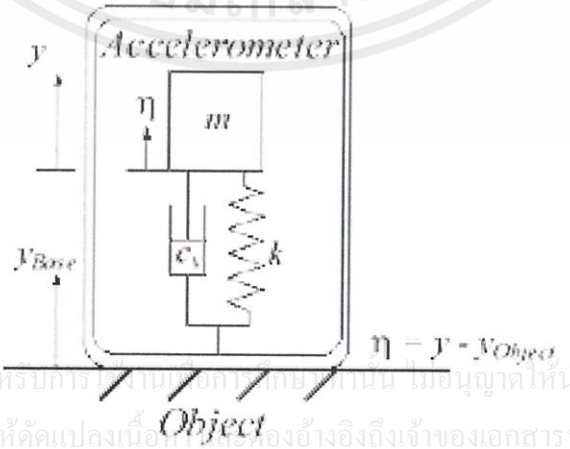
- เป็นอุปกรณ์สำหรับการวัด หรือการรักษาการปรับทิศทาง ขึ้นอยู่กับหลักการของการอนุรักษ์โมเมนตัมเชิงมุมที่อาศัยแรงเฉื่อยของล้อหมุน เพื่อช่วยรักษาระดับทิศทางของแกนหมุน ประกอบด้วยล้อหมุนเร็วบรรจุอยู่ในกรอบอีกทีหนึ่ง ทำให้เอียงในทิศทางต่างๆ ได้โดยอิสระ นั่นคือหมุนในแกนใดๆ ก็ได้ โมเมนตัมเชิงมุมของล้อดังกล่าวทำให้มันคงรักษาตำแหน่งของมันไว้แม้กรอบล้อจะเอียง จากคุณสมบัติดังกล่าวทำให้สามารถนำหลักการนี้ไปประยุกต์ใช้เพื่อประโยชน์ต่างๆ มากมาย เช่น เข็มทิศ และนักบินอัตโนมัติของเครื่องบิน เรือ กลไกบังคับหางเสือของตอร์ปิโด อุปกรณ์ป้องกันการกลิ้งบนเรือใหญ่ และระบบนำร่องเฉื่อย (Inertial Guidance) รวมถึงระบบในยานอวกาศ และสถานีอวกาศดังรูปที่ 2.26



รูปที่ 2.26 Mechanic Gyroscope ซึ่งมี Two-degree of Freedom (TDF)

2.13.2 Accelerometer

- เป็นอุปกรณ์ที่ใช้วัดความเร่งตามแนวแกนที่เฉพาะเจาะจง ตั้งข้อสังเกตได้ว่า Accelerometers ใน IMU รับเพียง Specific Forces ดังรูปที่ 2.27



รูปที่ 2.27 โครงสร้างของ Accelerometer

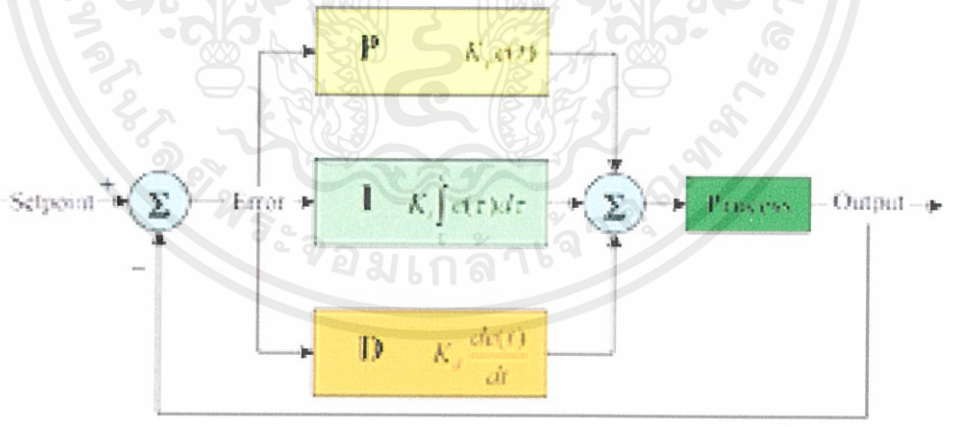
2.14 ระบบควบคุมพีไอดี

ระบบควบคุมแบบสัดส่วน-ปริพันธ์-อนุพันธ์ (อังกฤษ : PID Controller) เป็นระบบควบคุมแบบป้อนกลับที่ใช้กันอย่างกว้างขวาง ซึ่งค่าที่นำไปใช้ในการคำนวณเป็นค่าความผิดพลาดที่หามาจากความแตกต่างของตัวแปรในกระบวนการและค่าที่ต้องการ ตัวควบคุมจะพยายามลดค่าผิดพลาดให้เหลือน้อยที่สุดด้วยการปรับค่าสัญญาณขาเข้าของกระบวนการ ค่าตัวแปรของ PID ที่ใช้จะปรับเปลี่ยนตามธรรมชาติของระบบ

วิธีคำนวณของ PID ขึ้นอยู่กับสามตัวแปรคือ ค่าสัดส่วน, ปริพันธ์ และอนุพันธ์ ค่าสัดส่วนกำหนดจากผลของความผิดพลาดในปัจจุบัน, ค่าปริพันธ์กำหนดจากผลบนพื้นฐานของผลรวมความผิดพลาดที่ซึ่งพึ่งผ่านพ้นไป, และค่าอนุพันธ์กำหนดจากผลบนพื้นฐานของอัตราการเปลี่ยนแปลงของค่าความผิดพลาด น้ำหนักที่เกิดจากการรวมกันของทั้งสามนี้จะใช้ในการปรับกระบวนการ

โดยการปรับค่าคงที่ใน PID ตัวควบคุมสามารถปรับรูปแบบการควบคุมให้เหมาะกับที่กระบวนการต้องการได้ การตอบสนองของตัวควบคุมจะอยู่ในรูปของการไหวตัวของตัวควบคุมจนถึงค่าความผิดพลาด ค่าโอเวอร์ชูต (Overshoots) และค่าแกว่งของระบบ (Oscillation) วิธี PID ไม่รับประกันได้ว่าจะเป็นระบบควบคุมที่เหมาะสมที่สุดหรือสามารถทำให้กระบวนการมีความเสถียรแน่นอน

การประยุกต์ใช้งานบางครั้งอาจใช้เพียงหนึ่งถึงสองรูปแบบ ขึ้นอยู่กับกระบวนการเป็นสำคัญ พีไอดีบางครั้งจะถูกเรียกว่าการควบคุมแบบ PI, PD, P หรือ I ขึ้นอยู่กับว่าใช้รูปแบบใดบ้าง



รูปที่ 2.28 แผนภาพบล็อกของการควบคุมแบบพีไอดี

ทฤษฎี

การควบคุมแบบ PID ได้ชื่อตามการรวมกันของเทอมของตัวแปรทั้งสามตามสมการ :

$$\mathrm{MV(t)} = P_{\mathrm{out}} + I_{\mathrm{out}} + D_{\mathrm{out}}$$

เมื่อ

$P_{\mathrm{out}}, I_{\mathrm{out}},$ และ D_{out} เป็นผลของสัญญาณขาออกจากระบบควบคุม PID จากแต่ละเทอมซึ่งนิยามตามรายละเอียดด้านล่าง

สัดส่วน

เทอมของสัดส่วน (บางครั้งเรียก อัตราขยาย) จะเปลี่ยนแปลงเป็นสัดส่วนของค่าความผิดพลาด การตอบสนองของสัดส่วนสามารถทำได้โดยการคูณค่าความผิดพลาดด้วยค่าคงที่ K_p, หรือที่เรียกว่าอัตราขยายสัดส่วน

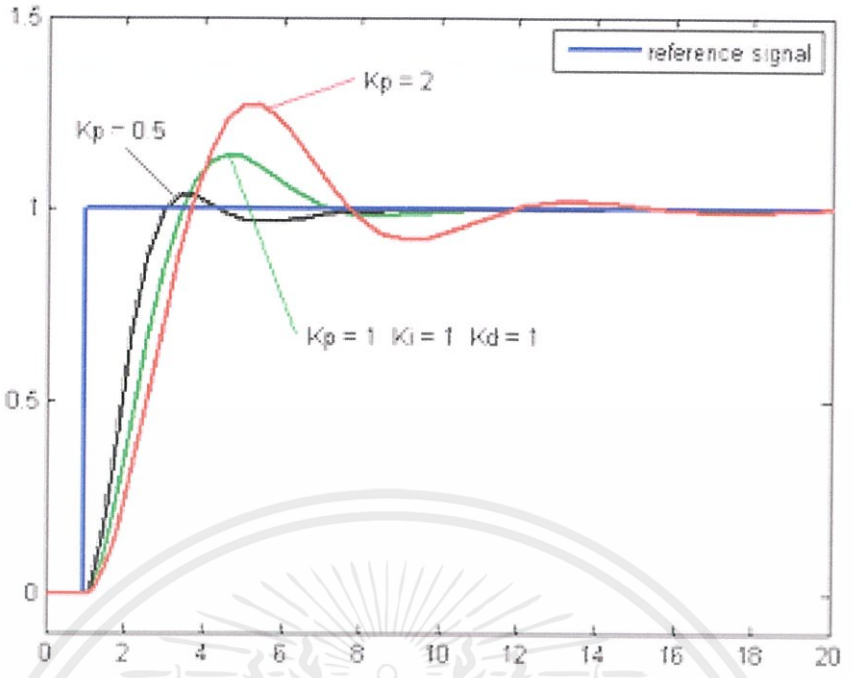
เทอมของสัดส่วนจะเป็นไปตามสมการ :

$$P_{\mathrm{out}} = K_p e(t)$$

เมื่อ

P_{out} : สัญญาณขาออกของเทอมสัดส่วน
K_p : อัตราขยายสัดส่วน, ตัวแปรปรับค่าได้
e : ค่าความผิดพลาด = SP - PV
t : เวลา

ผลอัตราขยายสัดส่วนที่สูงค่าความผิดพลาดก็จะเปลี่ยนแปลงมากเช่นกัน แต่ถ้าสูงเกินไประบบจะไม่เสถียรได้ ในทางตรงกันข้ามผลอัตราขยายสัดส่วนที่ต่ำ ระบบควบคุมจะมีผลตอบสนองต่อกระบวนการน้อยตามไปด้วย



รูปที่ 2.29 กราฟ PV ต่อเวลา, K_p กำหนดเป็น 3 ค่า(K_i และ K_d คงที่)

ปริพันธ์

ผลจากเทอมปริพันธ์ (บางครั้งเรียก Reset) เป็นสัดส่วนของขนาดความผิดพลาด และระยะเวลาของความผิดพลาด ผลรวมของความผิดพลาดในทุกช่วงเวลา (ปริพันธ์ของความผิดพลาด) จะให้ออฟเซตสะสมที่ควรจะเป็นในก่อนหน้า ความผิดพลาดสะสมจะถูกคูณโดยอัตราขยายปริพันธ์ ขนาดของผลของเทอมปริพันธ์จะกำหนดโดยอัตราขยายปริพันธ์, K_i.

เทอมปริพันธ์จะเป็นไปตามสมการ :

$$I_{\mathrm{out}} = K_i \int_0^t e(\tau)\, d\tau$$

เมื่อ

I_{out} : สัญญาณขาออกของเทอมปริพันธ์

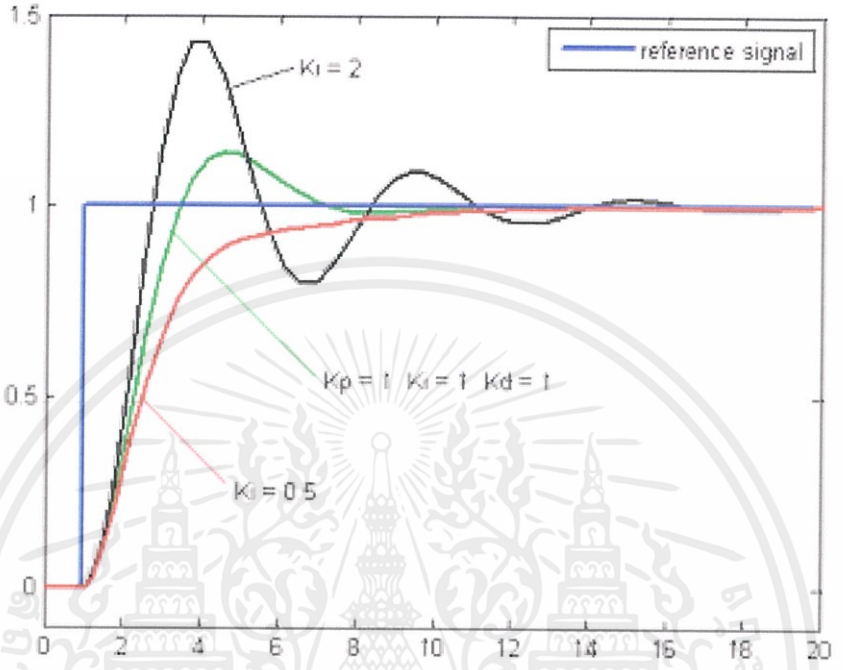
K_i : อัตราขยายปริพันธ์, ตัวแปรปรับค่าได้

e : ความผิดพลาด = SP - PV

t : เวลา

τ : ตัวแปรปริพันธ์หุ่น

เทอมปริพันธ์ (เมื่อรวมกับเทอมสัดส่วน) จะเร่งกระบวนการให้เข้าสู่จุดที่ต้องการและขจัดความผิดพลาดที่เหลืออยู่ที่เกิดจากการใช้เพียงเทอมสัดส่วน แต่อย่างไรก็ตามเทอมปริพันธ์เป็นการตอบสนองต่อความผิดพลาดสะสมในอดีต จึงสามารถทำให้เกิดโอเวอร์ชูตได้ (ข้ามจุดที่ต้องการและเกิดการหันเหไปทางทิศทางอื่น)



รูปที่ 2.30 กราฟ PV ต่อเวลา, K_i กำหนดเป็นสามค่า (K_p และ K_d คงที่)

อนุพันธ์

อัตราการเปลี่ยนแปลงของความผิดพลาดจากกระบวนการนั้น คำนวณหาจากความชันของความผิดพลาดทุกๆ เวลา (นั่นคือ เป็นอนุพันธ์อันดับหนึ่งสัมพันธ์กับเวลา) และคูณด้วยอัตราขยายอนุพันธ์ K_d ขนาดของผลของเทอมอนุพันธ์ (บางครั้งเรียก อัตรา) ขึ้นกับ อัตราขยายอนุพันธ์ K_d

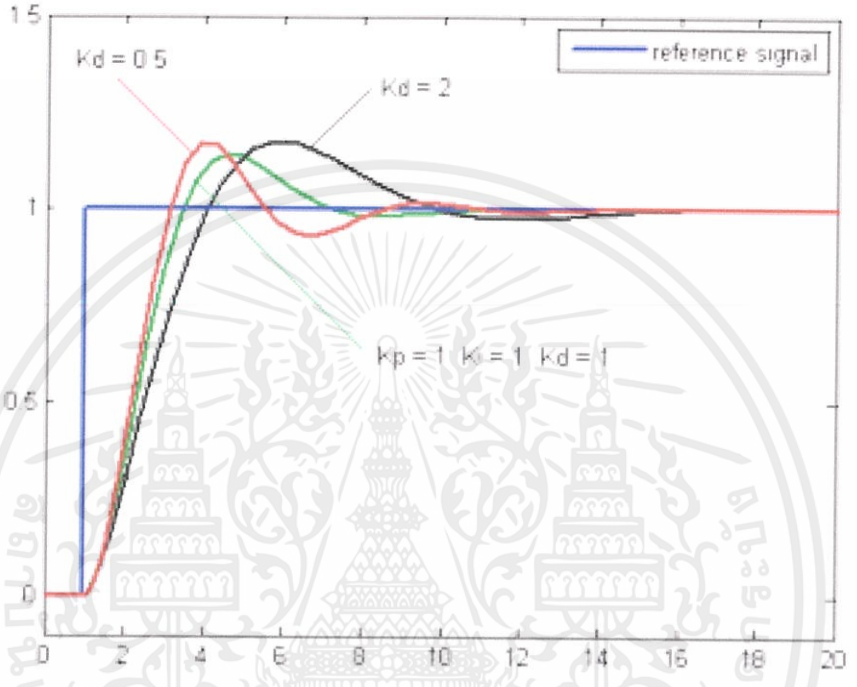
เทอมอนุพันธ์เป็นไปตามสมการ :

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

เมื่อ

D_{out}: สัญญาณขาออกของเทอมอนุพันธ์
K_d: อัตราขยายอนุพันธ์, ตัวแปรปรับค่าได้
e: ความผิดพลาด = SP - PV
t: เวลา

เทอมอนุพันธ์จะชะลออัตราการเปลี่ยนแปลงของสัญญาณขาออกของระบบควบคุม และด้วยผลนี้จะช่วยให้ระบบควบคุมเข้าสู่จุดที่ต้องการ ดังนั้นเทอมอนุพันธ์จะใช้ในการลดขนาดของโอเวอร์ชูตที่เกิดจาเทอมปริพันธ์ และทำให้เสถียรภาพของการรวมกันของระบบควบคุมดีขึ้น แต่อย่างไรก็ตามอนุพันธ์ของสัญญาณรบกวนที่ถูกขยายในระบบควบคุมจะไวมากต่อการรบกวนในเทอมของความผิดพลาด และสามารถทำให้กระบวนการไม่เสถียรได้ถ้าสัญญาณรบกวน และอัตราขยายอนุพันธ์มีขนาดใหญ่เพียงพอ



รูปที่ 2.31 กราฟ PV ต่อเวลา, สำหรับ K_d 3 ค่า (K_p และ K_i คงที่)

ผลรวม

เทอมสัดส่วน, ปริพันธ์, และอนุพันธ์จะนำมารวมกันเป็นสัญญาณขาออกของการควบคุมแบบ PID กำหนดให้ $u(t)$ เป็นสัญญาณขาออก สมการสุดท้ายของวิธี PID คือ :

$$\mathrm{u(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$$

รหัสเทียม

รหัสเทียม (อังกฤษ: Pseudocode) ของ ขั้นตอนวิธีระบบควบคุมพีไอดี โดยอยู่บนสมมติฐานว่าตัวประมวลผลแบบขนานอย่างสมบูรณ์แบบ เป็นดังต่อไปนี้

```
previous_error = setpoint – actual_position
integral = 0
start:
error = setpoint – actual_position
integral = integral + (error*dt)
derivative = (error – previous_error)/dt
output = (Kp*error) + (Ki*integral) + (Kd*derivative)
previous_error = error
wait(dt)
goto start
```

การปรับจูน

การปรับจูนด้วยมือ

ถ้าระบบยังคงทำงาน ขั้นแรกให้ตั้งค่า K_i และ K_d เป็นศูนย์ เพิ่มค่า K_P จนกระทั่งสัญญาณขาออกเกิดการแกว่ง (Oscillate) แล้วตั้งค่า K_P ให้เหลือครึ่งหนึ่งของค่าที่ทำให้เกิดการแกว่ง สำหรับการตอบสนองชนิด “Quarter Amplitude Decay”แล้วเพิ่ม K_i จนกระทั่งออฟเซตถูกต้องในเวลาที่พอเพียงของกระบวนการ แต่ถ้า K_i มากไปจะทำให้ไม่เสถียร สุดท้ายถ้าต้องการให้เพิ่มค่า K_d จนกระทั่งลูปอยู่ในระดับที่ยอมรับได้ แต่ถ้า K_d มากเกินไปจะเป็นเหตุให้การตอบสนองและโอเวอร์ชูตเกินยอมรับได้ ปกติการปรับจูน PID ถ้าเกิดโอเวอร์ชูตเล็กน้อยจะช่วยให้เข้าสู่จุดที่ต้องการเร็วขึ้น แต่ในบางระบบไม่สามารถยอมให้เกิดโอเวอร์ชูตได้ และถ้าค่า K_P น้อยเกินไปก็จะทำให้เกิดการแกว่ง

ตารางที่ 2.1 การปรับจูนด้วยมือ

ผลของการเพิ่มค่าตัวแปรอย่างอิสระ

ตัวแปร	ช่วงเวลาขึ้น (Rise Time)	โอเวอร์ชูต (Overshoot)	เวลาสู่สมดุล (Settling Time)	ความผิดพลาดสถานะคงตัว (Steady-state Error)	เสถียรภาพ
K_p	ลด	เพิ่ม	เปลี่ยนแปลงเล็กน้อย	ลด	ลด
K_i	ลด	เพิ่ม	เพิ่ม	ลดลงอย่างมีนัยสำคัญ	ลด
K_d	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ตามทฤษฎีไม่มีผล	ดีขึ้นถ้า K_d มีค่าน้อย

วิธีการ Ziegler–Nichols

วิธีการนี้นำเสนอโดย John G. Ziegler และ Nathaniel B. Nichols ในคริสต์ทศวรรษที่ 1940 ขั้นแรกให้ตั้งค่า K_i และ K_d เป็นศูนย์ เพิ่มอัตราขยาย P สูงที่สุด, K_u, จนกระทั่งเริ่มเกิดการแกว่ง นำค่า K_u และค่าช่วงการแกว่ง P_u มาหาค่าตัวแปรที่เหลือดังตาราง :

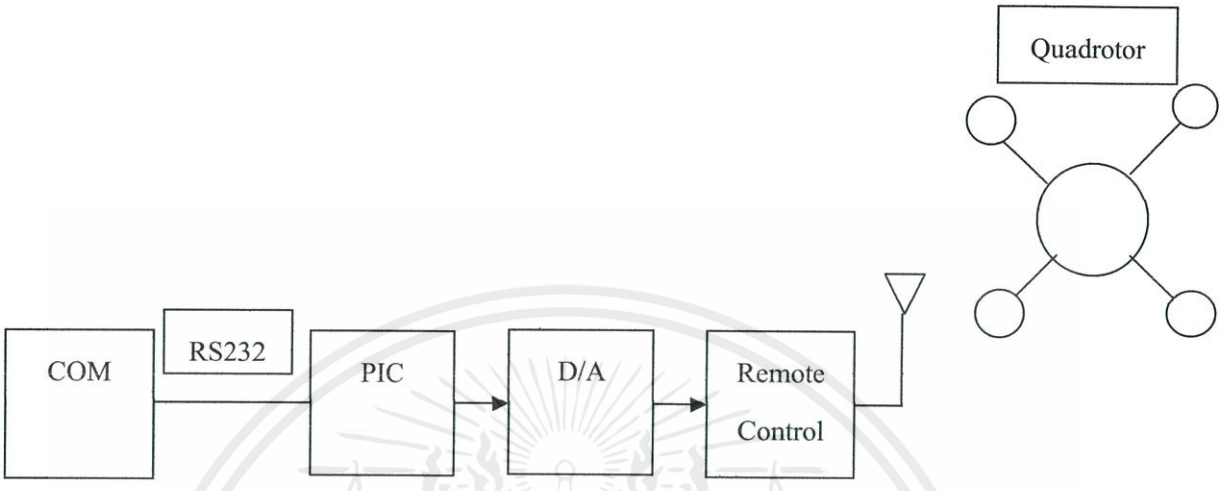
ตารางที่ 2.2 วิธีการ Zigler–Nichols

Ziegler–Nichols Method

Control Type	K_p	K_i	K_d
P	$0.50K_u$	-	-
PI	$0.45K_u$	$1.2K_p/P_u$	-
PID	$0.60K_u$	$2K_p/P_u$	$K_pP_u/8$

บทที่ 3
การออกแบบและควบคุม

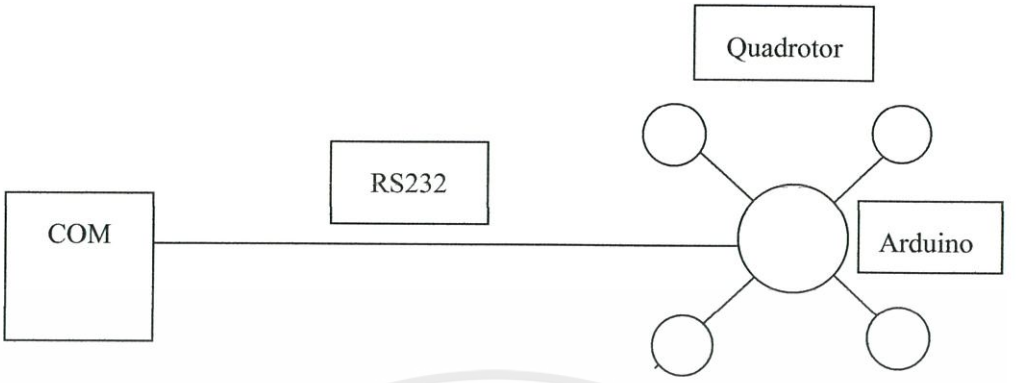
3.1 การออกแบบระบบควบคุม



รูปที่ 3.1 การควบคุมผ่านสาย RS232

การควบคุมจะใช้การส่งข้อมูลโดยการป้อนคำสั่งจากคอมพิวเตอร์ผ่านสาย RS232 ไปยัง PIC เมื่อตัว PIC ได้รับสัญญาณคำสั่งอินพุตจากคอมพิวเตอร์ แล้วจะดำเนินการส่งสัญญาณเอาต์พุตจากคำสั่งที่ได้รับ โดยผ่านการแปลงสัญญาณดิจิตอลเป็นอนาล็อก (DAC) ก่อน เพื่อจะได้สัญญาณที่เป็นอนาล็อกตามที่ต้องการ เพราะสัญญาณอนาล็อกจะให้กราฟที่มีความราบลื่นและต่อเนื่องมากกว่ากราฟแบบดิจิตอลที่เป็นค่าขึ้นลง 0-1 เป็นขั้นบันได เพราะจะทำให้ Quadrotor เคลื่อนที่ไม่ราบลื่นตามต้องการ หลังจากนั้นสัญญาณจะถูกส่งไปควบคุมรีโมทคอนโทรล รีโมทคอนโทรลก็จะส่งสัญญาณเป็นคลื่นความถี่ที่กำหนดไปยัง Quadrotor เพื่อควบคุมตัวอากาศยานตามโค้ดข้อมูลคำสั่ง แต่เราพบปัญหาเกิดขึ้นจากการทดลอง พบว่าไม่สามารถควบคุมการทำงานได้อย่างที่ต้องการ เพราะสัญญาณที่ถูกส่งไปยัง Quadrotor ไม่มีการป้อนค่ากลับ ทำให้ไม่ทราบตัว Quadrotor ว่ามันบินขึ้นได้ตรงหรือไม่ ทำมุมอย่างไร ไม่สามารถตรวจสอบได้ จึงเป็นปัญหาในการควบคุม ทำให้การควบคุมการทำงานไม่มีประสิทธิภาพเพียงพอ และอีกทั้งไม่ทราบว่าในขณะที่ทำการใช้งานอยู่ ตัว Quadrotor อยู่ในตำแหน่งที่ต้องการหรือไม่ เพราะไม่มีเซนเซอร์ในการตรวจสอบพิกัดตำแหน่ง จากการทดลองนี้จึงได้ออกแบบตัวควบคุมใหม่ โดยออกแบบบอร์ดควบคุมใหม่ โดยเพิ่มเซนเซอร์เพื่อวัดตำแหน่งในที่ที่ Quadrotor เคลื่อนที่ไป และต่อสาย USB To UART Cable pl2303 HX ผ่านช่อง Tx และ Rx เพื่อรับส่งข้อมูล ทำให้แก้ปัญหาการตรวจสอบพิกัดตำแหน่งและการไม่มีข้อมูลป้อนกลับได้

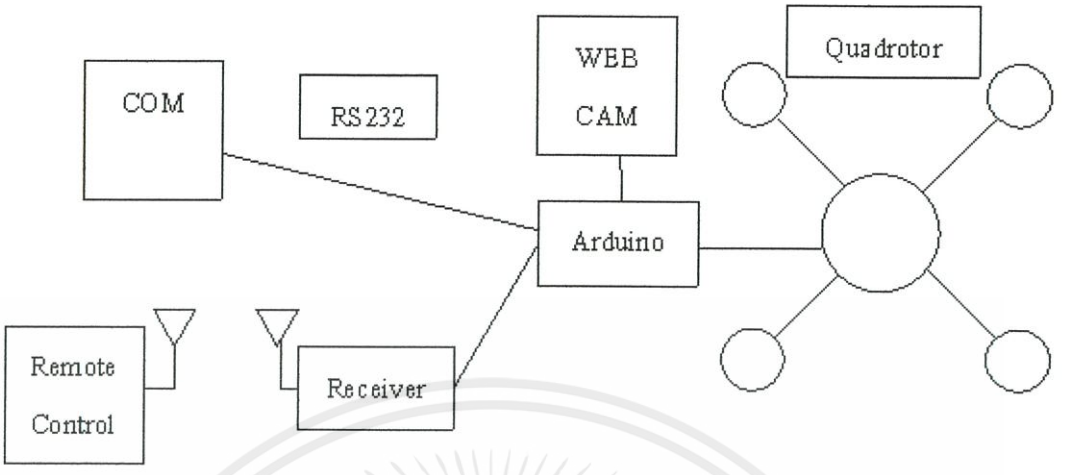
3.2 การแก้ไขการออกแบบระบบควบคุม (ครั้งที่ 1)



รูปที่ 3.2 การควบคุมผ่านสาย RS232 ไปยัง Quadrotor โดยตรง

การออกแบบแก้ไขระบบควบคุม ทำการแก้ไขระบบควบคุมโดยส่งข้อมูลคำสั่งจากคอมพิวเตอร์ผ่าน RS232 ที่พอร์ต Tx และ Rx บนบอร์ด Arduino UNO R3 เพื่อทำการรับและส่งข้อมูลผ่านได้โดยตรง โดยใช้สาย USB To UART Cable pl2303 HX ทำให้มีการส่งข้อมูลไปยังบอร์ด Arduino UNO R3 ที่ควบคุมตัวอากาศยาน นอกจากนี้ยังเพิ่ม Ultrasonic Sensor เพื่อใช้ในการวัดระยะความสูงของ Quadrotor และระบุพิกัดตำแหน่งของ Quadrotor อีกด้วย จากการป้อนข้อมูลคำสั่งผ่านสาย USB To UART Cable pl2303 HX ผ่านพอร์ต Tx และ Rx บนบอร์ด Arduino UNO R3 จะทำการควบคุมความเร็วของมอเตอร์แต่ละตัวให้มีความเร็วเท่ากันในการยกตัว Quadrotor ขึ้น และปรับค่าความเร็วของมอเตอร์แต่ละตัวให้เหมาะสมกับการเคลื่อนที่ เช่น การเคลื่อนที่ไปทางซ้าย ขวา หน้าและหลัง รวมทั้งควบคุมความเร่งของ Quadrotor ผ่านโค้ดข้อมูลคำสั่งของโปรแกรม และ Ultrasonic Sensor จะคอยตรวจสอบระยะของการเคลื่อนที่ และตำแหน่งของ Quadrotor เพื่อให้ทราบตำแหน่งที่แน่นอน ทำให้สามารถควบคุมได้อย่างมีประสิทธิภาพมากขึ้น

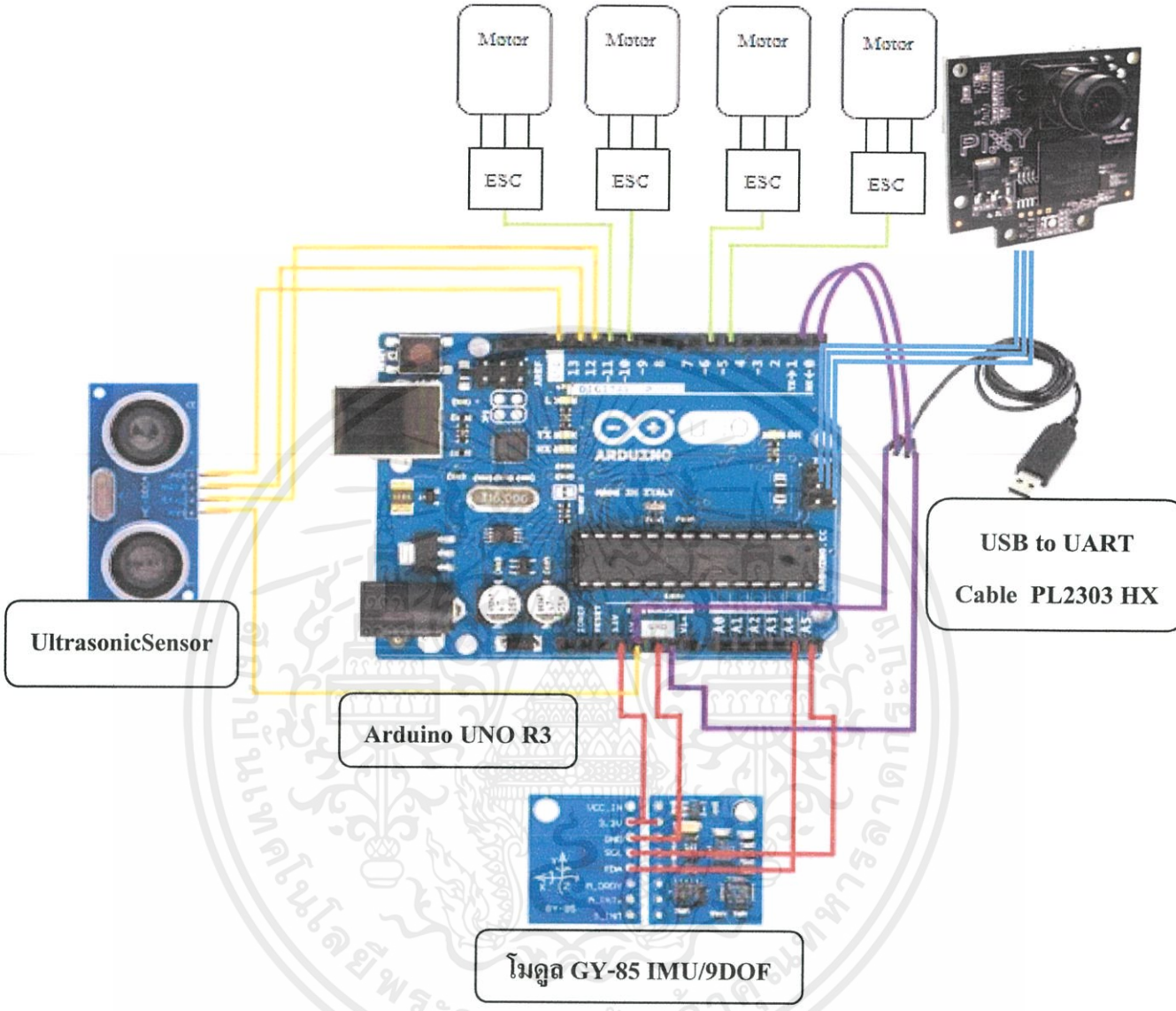
3.3 การแก้ไขออกแบบระบบควบคุม (ครั้งที่ 2)



รูปที่ 3.3 แก้ไขการควบคุม Quadrotor

การออกแบบนี้ทำการแก้ไขแบบอีกครั้งโดยการออกแบบนี้ จะเปลี่ยนจากการใช้คอมพิวเตอร์เป็นตัวควบคุมอากาศยานมาใช้รีโมทเป็นตัวควบคุมตัวอากาศยาน ส่วนคอมพิวเตอร์จะใช้เป็นตัวช่วยในการควบคุม ทำการติดตั้งกล้อง Web Cam เพิ่มที่ตัวอากาศยาน โดยกล้องตัวนี้จะทำหน้าที่รับภาพที่ตัวอากาศยานเคลื่อนที่ผ่านและจะส่งข้อมูลผ่าน RS232 มาแสดงที่คอมพิวเตอร์ทำให้สามารถทราบตำแหน่งของตัวอากาศยานได้ ในการบังคับใช้รีโมทเป็นตัวบังคับอากาศยาน โดยรีโมทจะส่งสัญญาณไปยังตัวรับ Receiver ตัว Receiver เมื่อได้รับสัญญาณก็จะทำการส่งสัญญาณไปควบคุมมอเตอร์โดยผ่านบอร์ด Arduino อีกที แบบที่ได้รับการแก้ไขนี้ทำให้สามารถควบคุมตัวอากาศยานได้ง่ายขึ้น

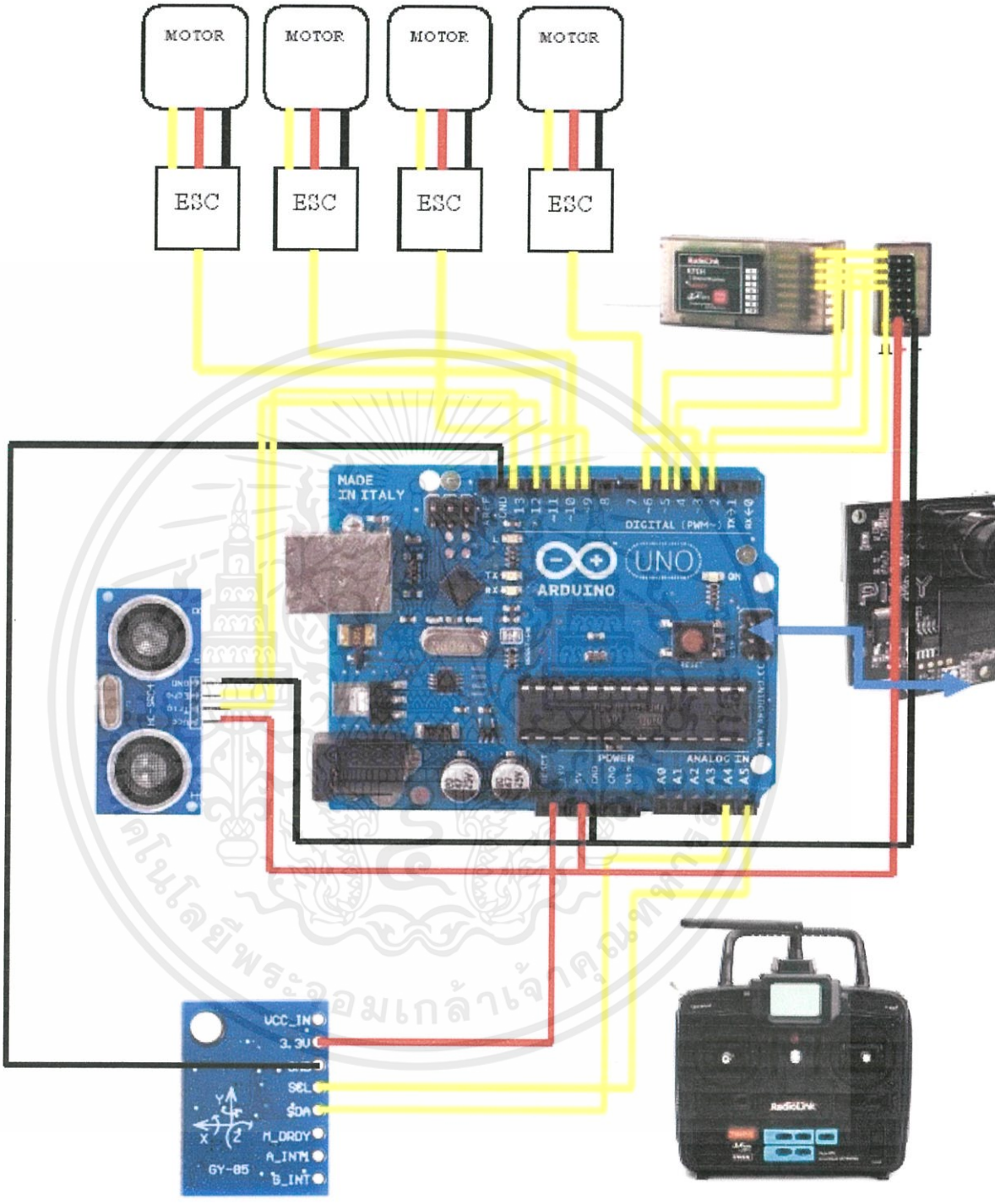
3.4 การต่ออุปกรณ์ควบคุมตัว Quadrotor กับบอร์ด Arduino UNO R3



รูปที่ 3.4 วงจรควบคุมตัว Quadrotor

1. ต่อ Motor และ ESC เข้ากับบอร์ด Arduino UNO ที่พอร์ต PWM ที่ขา 5, 6, 10, 11
2. ต่อ Ultrasonic Sensor เข้ากับบอร์ด Arduino UNO ที่ขา 12, 13, 5V, GND
3. ต่อโมดูล GY-85 IMU/9DOF เข้ากับบอร์ด Arduino UNO ที่ขา 3.3V, GND, A4, A5
4. ต่อ USB To UART Cable PL2303 HX เข้ากับบอร์ด Arduino UNO ที่ขา Tx, Rx, 5V, GND
5. ต่อกล้องเซนเซอร์ตรวจจับวัตถุ (PIXY CMU Cam5) ต่อกับพอร์ต ICSP

3.5 แก้ไขแบบวงจร Quadrotor

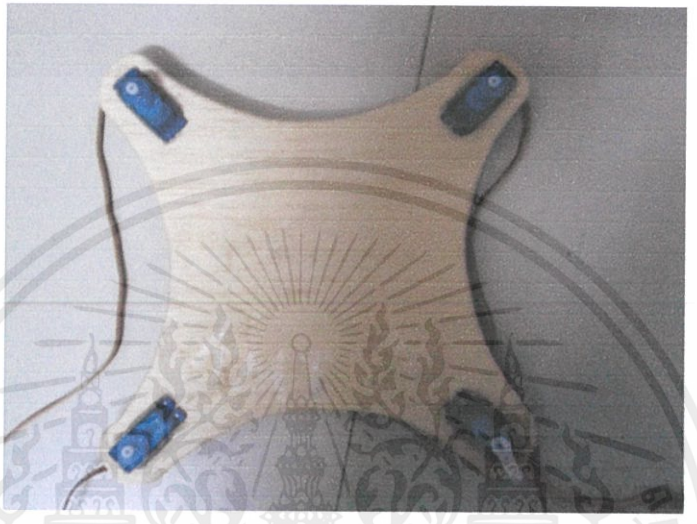


รูปที่ 3.5 วงจรที่ถูกแก้ไข

บทที่ 4
ผลการทดลอง

4.1 การควบคุม Servo Motor

ความเร็วรอบของ Brushless Motor จะเป็นการกำหนดทิศทางการเคลื่อนที่ของ Quadrotor โดยเริ่มแรกทำการจำลองทิศทางการเคลื่อนที่โดยใช้ Servo Motor ตามรูปที่ 4.1



รูปที่ 4.1 Servo Motor ที่ใช้ทดสอบ

โดยในการควบคุมจะอาศัยค่าสัญญาณพัลส์ ที่ส่งออกมาจากรีซีฟเวอร์เป็นตัวควบคุมการหมุนของ Servo Motor โดยเซอร์โวที่ใช้มีองศาในการหมุน 0 ถึง 180 องศา

ในการควบคุมการหมุนของเซอร์โวจะขึ้นอยู่กับค่าสัญญาณพัลส์ที่เพิ่มขึ้นหรือลดลง โดยจะกำหนดทิศทางการหมุนของเซอร์โว ดังตารางที่ 4.1

ตารางที่ 4.1 ทิศทางหมุนของเซอร์โวตามค่าสัญญาณพัลส์

เซอร์โวมอเตอร์	ทิศการหมุนตามค่าสัญญาณพัลส์	
	เพิ่มขึ้น	ลดลง
ซ้ายบน	180 ไป 0 องศา	0 ไป 180 องศา
ขวาบน	0 ไป 180 องศา	180 ไป 0 องศา
ซ้ายล่าง	0 ไป 180 องศา	180 ไป 0 องศา
ขวาล่าง	180 ไป 0 องศา	0 ไป 180 องศา

เมื่อทำการโยกคันบังคับ THROTTLE, ROLL, PITCH และ YAW ค่าสัญญาณพัลส์ของเซอร์โวแต่ละตัวจะมีค่าเพิ่มขึ้น (+) หรือลดลง (-) ตามตารางที่ 4.2

ตารางที่ 4.2 ค่ามอเตอร์ตามการควบคุม

	ซ้ายบน	ขวาบน	ซ้ายล่าง	ขวาล่าง
THROTTLE UP	+	+	+	+
THROTTLE DOWN	-	-	-	-
ROLL RIGHT	+	-	+	-
ROLL LEFT	-	+	-	+
PITCH UP	-	-	+	+
PITCH DOWN	+	+	-	-
YAW RIGHT	-	+	+	-
YAW LEFT	+	-	-	+

ค่าการหมุนของ Gyro กับค่า Accel จะมีค่าเพิ่มขึ้น (+) หรือลดลง (-) เกิดจากการเอียงตัวของตัวอากาศยานที่เคลื่อนที่จากการควบคุม ตามตารางที่ 4.3

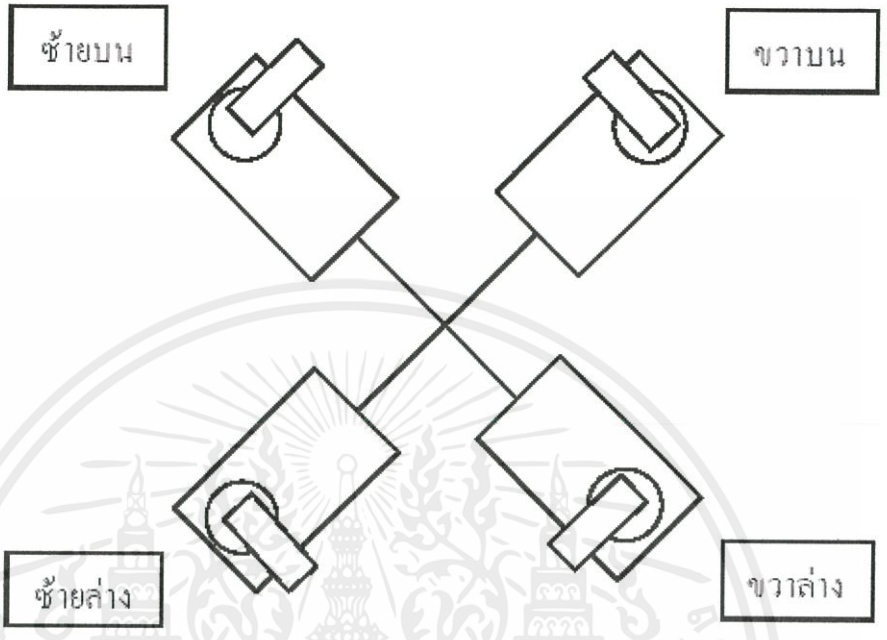
ตารางที่ 4.3 ค่าเซนเซอร์ตามการควบคุม

เซนเซอร์	ROLL		PITCH		YAW	
	RIGHT	LEFT	UP	DOWN	RIGHT	LEFT
GyroX	คงที่	คงที่	-	+	คงที่	คงที่
GyroY	+	-	คงที่	คงที่	คงที่	คงที่
GyroZ	คงที่	คงที่	คงที่	คงที่	-	+
AccX	+	-	คงที่	คงที่	คงที่	คงที่
AccY	คงที่	คงที่	-	+	คงที่	คงที่

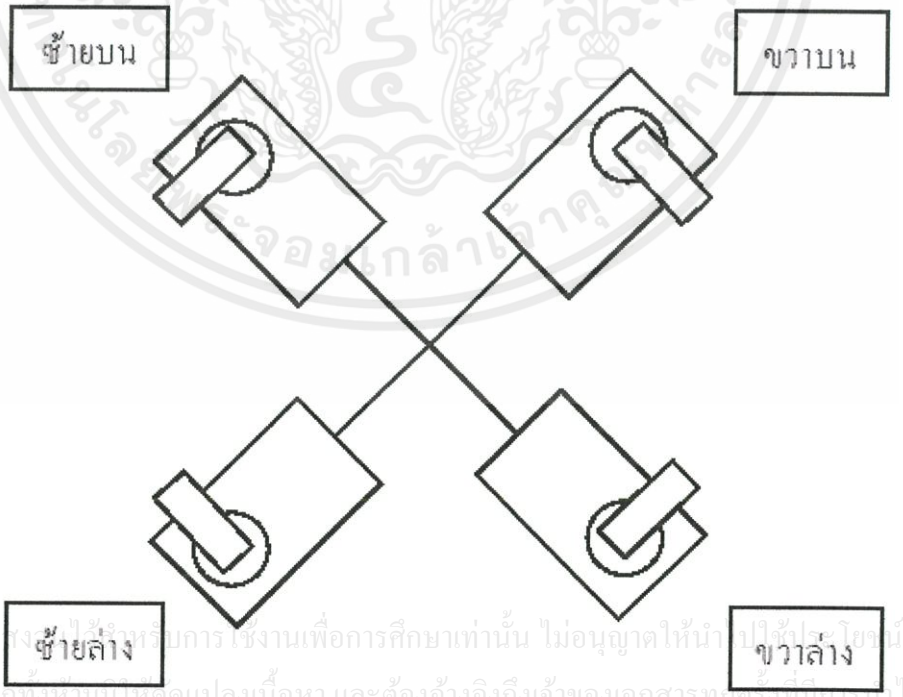
ซึ่งเซอร์โวแต่ละตัวจะทำงานตามตารางที่ 4.2 ดังนี้

4.1.1 THROTTLE

เมื่อทำการโยก THROTTLE เซอร์โวจะมีทิศทางการหมุน ดังรูปที่ 4.2 และรูปที่ 4.3



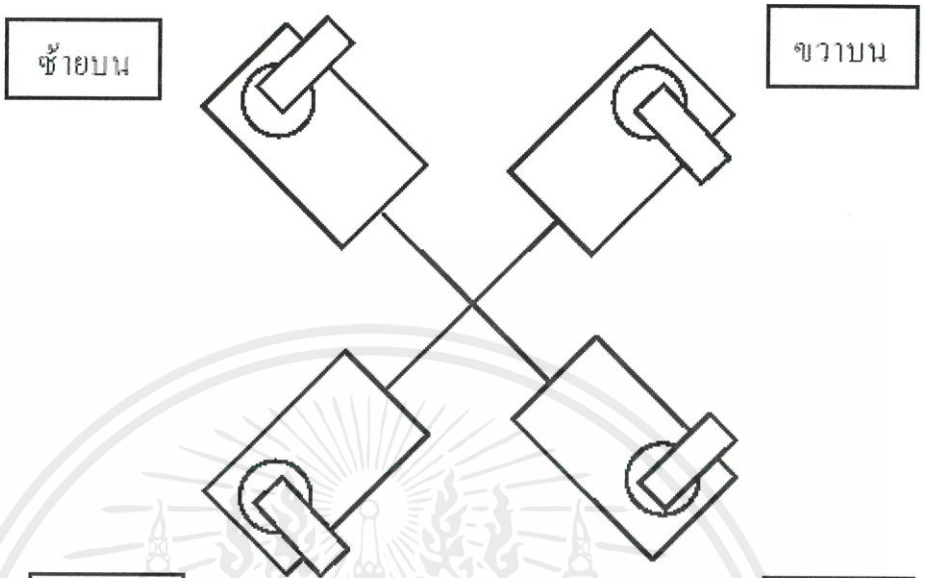
รูปที่ 4.2 เมื่อโยก THROTTLE UP



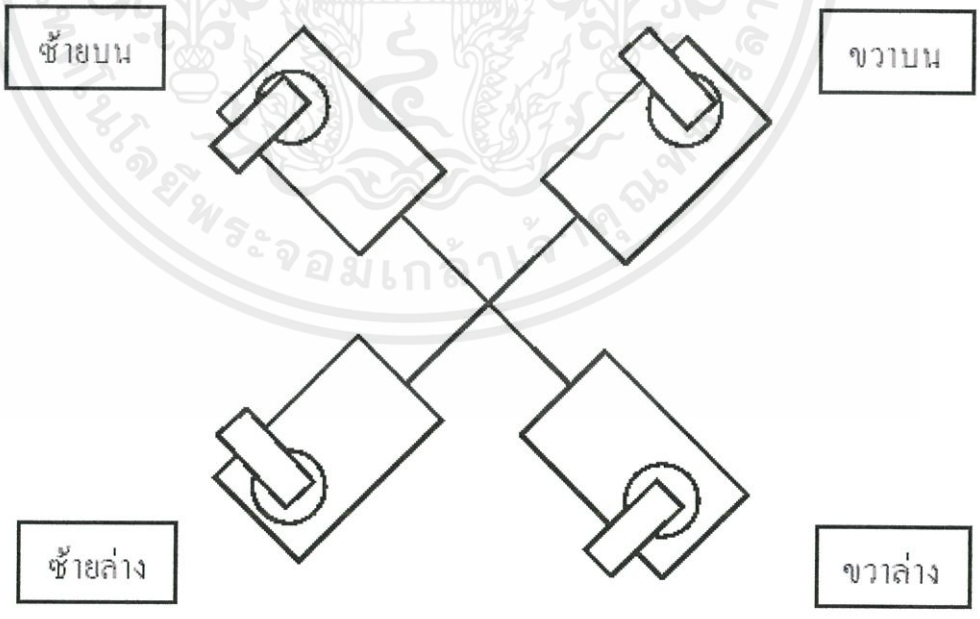
รูปที่ 4.3 เมื่อโยก THROTTLE DOWN

4.1.2 ROLL

เมื่อทำการโยกคันโยก ROLL เซอร์โวจะมีทิศทางหมุน ดังรูปที่ 4.4 และรูปที่ 4.5



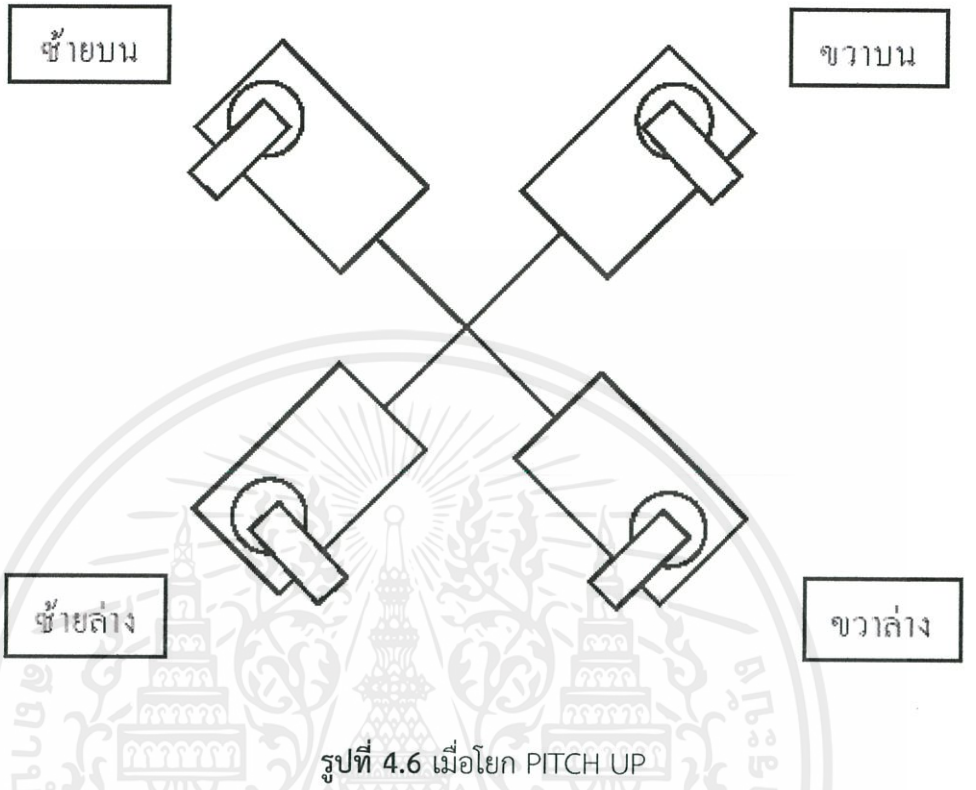
รูปที่ 4.4 เมื่อโยก ROLL RIGHT



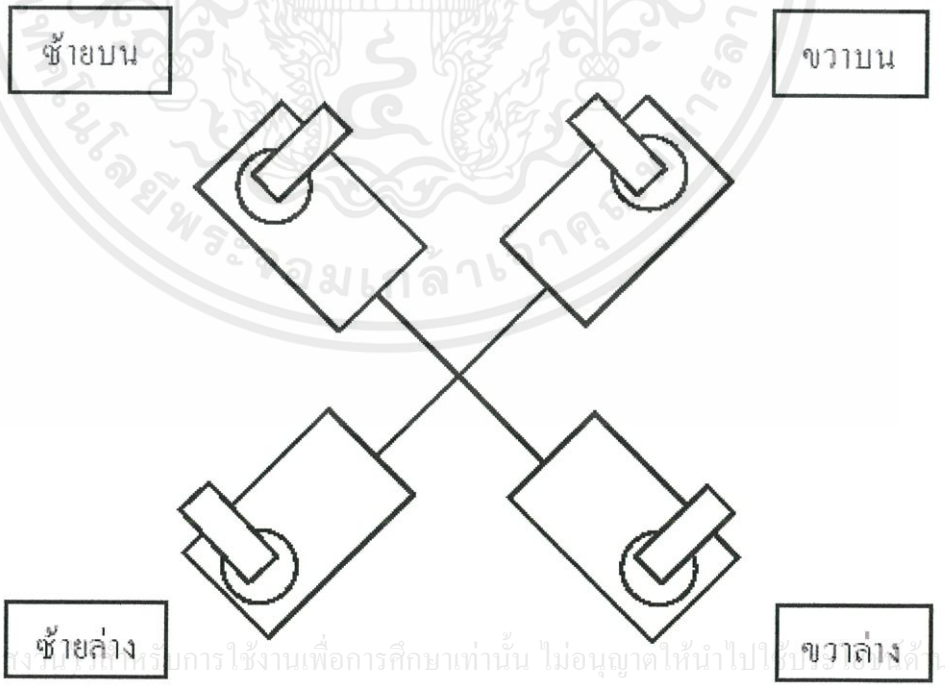
รูปที่ 4.5 เมื่อโยก ROLL LEFT

4.1.3 PITCH

เมื่อทำการโยกคันโยก PITCH เซอร์โวจะมีทิศหมุน ดังรูปที่ 4.6 และรูปที่ 4.7



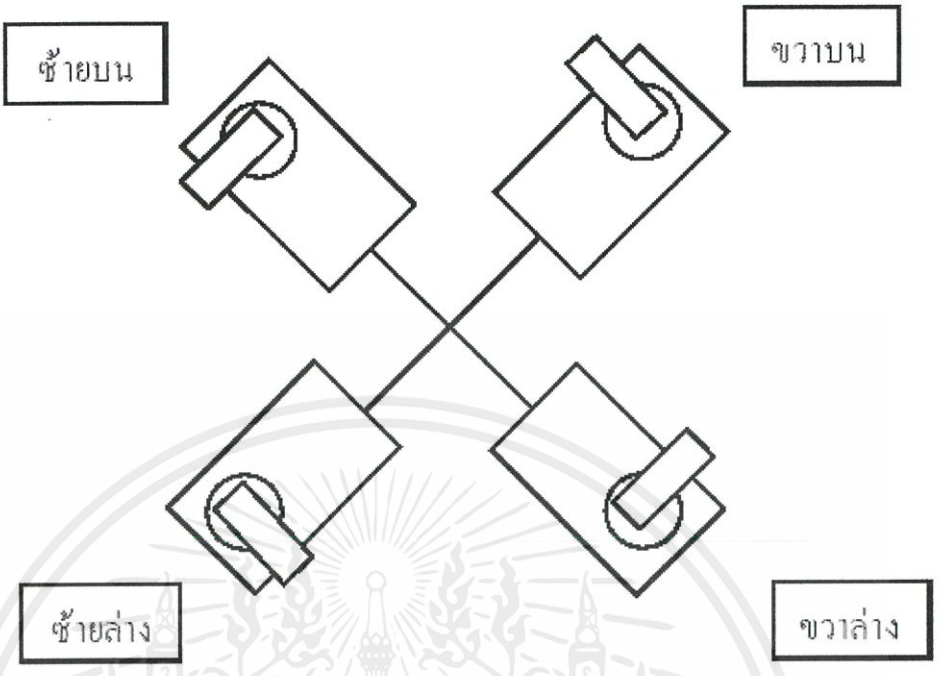
รูปที่ 4.6 เมื่อโยก PITCH UP



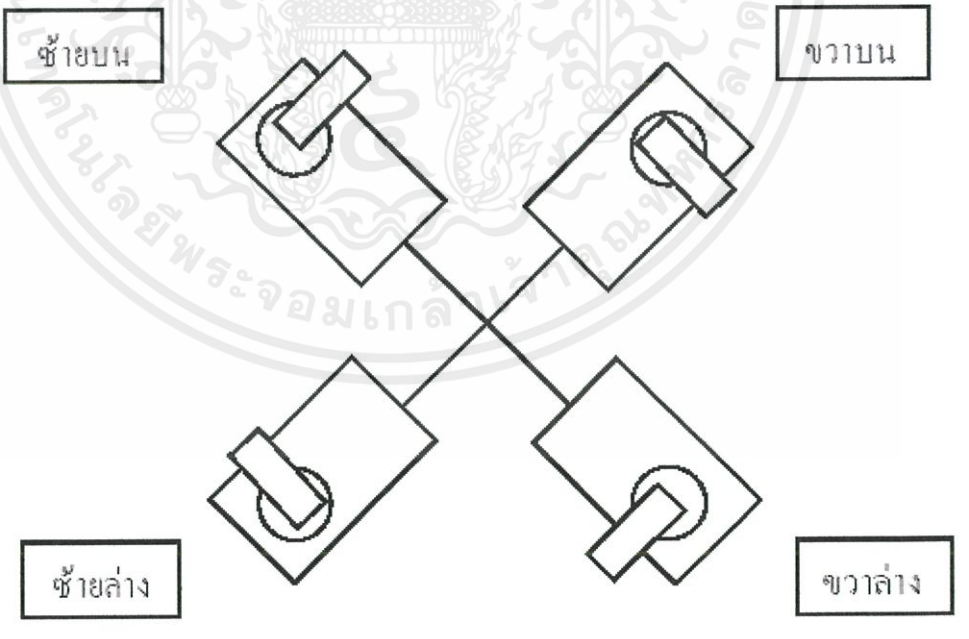
รูปที่ 4.7 เมื่อโยก PITCH DOWN

4.1.4 YAW

เมื่อโยกคันโยก YAW เซอร์โวจะมีทิศหมุน ดังรูปที่ 4.8 และรูปที่ 4.9



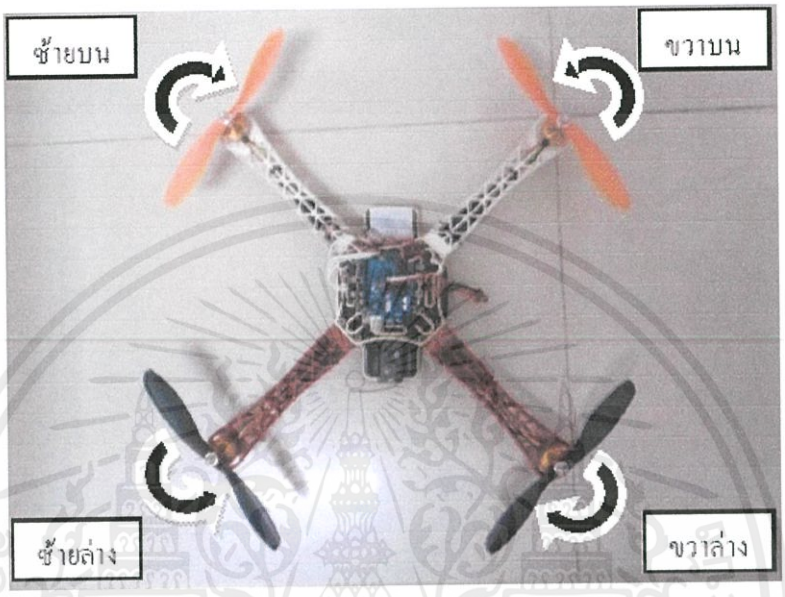
รูปที่ 4.8 เมื่อโยก YAW RIGHT



รูปที่ 4.9 เมื่อโยก YAW LEFT

4.2 การควบคุมบัสเลสมอเตอร์

หลังจากที่ได้ทำการจำลองการทำงานด้วยเซอร์โวแล้ว จะทำการนำโค๊ดมาปรับใช้ในการควบคุมบัสเลสมอเตอร์ ซึ่งในการควบคุมจะใช้รีซิฟเวอร์ที่ส่งค่าสัญญาณพัลล์ออกมาใช้ในการควบคุมโดยความเร็วของมอเตอร์แต่ละตัวที่จะเป็นตัวกำหนดการเคลื่อนที่ของตัวคอปเตอร์ จะขึ้นอยู่กับการโยกคันบังคับของรีโมทคอนโทรลการหมุนของมอเตอร์ทั้ง 4 ตัว จะแสดงดังรูปที่ 4.10



รูปที่ 4.10 การทำงานของบัสเลสมอเตอร์

4.3 ตรวจสอบค่า Gyro และ Accelerometer และหาค่า Offset เริ่มต้น

ทำการตรวจสอบเซนเซอร์หาทิศทางการหมุนของ Gyro กับ Accel ดังรูปที่ 4.11 และรูปที่ 4.12

```
Gyro(degree/s): 0.00, 0.00, -0.07
Temperature: 29.00
Gyro(degree/s): 0.00, 0.07, 0.00
Temperature: 29.00
Gyro(degree/s): 0.00, 0.07, 0.00
Temperature: 29.00
Gyro(degree/s): 0.00, 0.07, 0.00
Temperature: 29.00
Gyro(degree/s): 0.00, 0.00, 0.00
Temperature: 29.00
Gyro(degree/s): 0.00, 0.00, -0.07
Temperature: 29.00
Gyro(degree/s): 0.00, 0.00, 0.00
Temperature: 29.00
Gyro(degree/s): 0.00, 0.07, -0.07
Temperature: 29.00
```


รูปที่ 4.11 การทดสอบ Gyro

```
                Connected to ADXL345.
Raw:    8   2   247     Scaled: 0.03G   0.01G   0.96G
Raw:    9   3   247     Scaled: 0.04G   0.01G   0.96G
Raw:    8   2   245     Scaled: 0.03G   0.01G   0.96G
Raw:    9   2   248     Scaled: 0.04G   0.01G   0.97G
Raw:    9   2   247     Scaled: 0.04G   0.01G   0.96G
Raw:    8   3   246     Scaled: 0.03G   0.02G   0.96G
Raw:    8   2   246     Scaled: 0.03G   0.01G   0.96G
Raw:    8   2   246     Scaled: 0.03G   0.01G   0.96G
Raw:    8   2   248     Scaled: 0.03G   0.01G   0.97G
Raw:    8   2   246     Scaled: 0.03G   0.01G   0.96G
Raw:    8   2   246     Scaled: 0.03G   0.01G   0.96G
Raw:    9   2   248     Scaled: 0.04G   0.01G   0.97G
Raw:    9   2   249     Scaled: 0.04G   0.01G   0.97G
Raw:    10   3   248    Scaled: 0.04G   0.01G   0.97G
Raw:    8   2   248     Scaled: 0.03G   0.01G   0.96G
Raw:    9   2   247     Scaled: 0.04G   0.01G   0.96G
Raw:    9   2   246     Scaled: 0.04G   0.01G   0.96G
Raw:    9   2   246     Scaled: 0.04G   0.01G   0.96G
```

รูปที่ 4.12 การทดสอบ Accel

4.3.1 การทดสอบ Gyro

เมื่อทำการทดสอบเมื่อหมุนแกน X แกน Y และแกน Z ค่าของไจโรจะเพิ่มขึ้นหรือลดลง แล้วแต่ทิศทางการหมุน เมื่อหยุดหมุนค่าไจโรจะกลับไปยังค่าศูนย์

4.3.2 การทดสอบ Accelerometer

ค่าของ Accelerometer จะมีค่าเปลี่ยนแปลงตามองศาของแกน X แกน Y และ แกน Z โดยค่าที่มากที่สุดเท่ากับ 1 g โดย 1 g เท่ากับ 9.81 m/s^2

หลังจากทำการทดสอบทิศทางการหมุนของ Gyro กับ Accel ทำการหาค่า Offset เริ่มต้นของทั้ง 2 โดยไจโรเมื่อไม่เกิดการหมุนค่า Offset เริ่มต้นเท่ากับ 0 ส่วน Accel ค่าจะเปลี่ยนแปลงตามองศาของแกน

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

การทดลองนี้ทำการออกแบบระบบควบคุมอากาศยาน โดยโปรแกรมที่ใช้เขียนภาษาซี คือ โปรแกรมอาดูโนว์ ซึ่งการดำเนินงานมีดังนี้

1. ทำการออกแบบระบบการทำงานของตัวอากาศยาน โดยในการออกแบบต้องทำการออกแบบระบบที่เป็นตัวควบคุมอากาศยานก่อน โดยจะทำการเลือกบอร์ดอาดูโนว์และเซนเซอร์ที่เหมาะสมกับแบบที่ได้ออกแบบ

2. ทำการต่อวงจรตามแบบที่ได้ออกแบบและเขียนโค๊ดที่ใช้ในการควบคุม การเขียนโค๊ดควบคุมตัวอากาศยานในเริ่มต้นจะต้องเขียนเพื่อให้สามารถควบคุมตัวมอเตอร์ทั้ง 4 ตัวได้ก่อน แล้วจึงทำการเขียนโค๊ดเซนเซอร์ IMU

3. ขั้นต่อมาทำการติดตั้งกล้อง PIXY โดยกล้องตัวนี้จะทำการตรวจจับวัตถุสีเข้ม ซึ่งจะช่วยควบคุม Quadrotor ให้เคลื่อนที่ไปถึงวัตถุนั้นได้ และติดตั้ง Ultrasonic Sensor เพื่อช่วยในการกำหนดวัดระยะความสูงซึ่งจะช่วยให้ Quadrotor บินในระดับความสูงที่กำหนดได้

4. ทำการทดสอบและบันทึกผลการทดลอง

5.2 ปัญหาที่พบในการทดลอง

ปัญหาที่พบในการทดลองมีดังนี้

1. เมื่อทำการทดลองมอเตอร์ไม่หมุนตามที่สั่งการด้วยรีซิฟเวอร์
2. เมื่อโยกคันโยกของรีโมทค่าสัญญาณพัลส์ที่เพิ่มขึ้น และลดลงของมอเตอร์แต่ละตัวสลับค่ากัน
3. เซนเซอร์ไม่ตอบสนองต่อคำสั่งในการควบคุมได้ทัน

5.3 ข้อเสนอแนะ

ในการดำเนินงานต้องทำการออกแบบตัวอากาศยานให้เป็นที่เรียบร้อยก่อน แล้วจึงทำการต่อวงจรและเขียนโค๊ดเพื่อควบคุม โดยเริ่มแรกเขียนโค๊ดควบคุมมอเตอร์ของตัวอากาศยานก่อน เมื่อมอเตอร์ทำงานแล้วทำการเขียนเซนเซอร์ IMU เพื่อใช้ในการการปรับสมดุลของ Quadrotor ซึ่งจากการทดลองอากาศยานพลิกคว่ำ เนื่องจากเซนเซอร์ไม่สามารถตอบสนองต่อการสั่งงานได้ทัน จึงต้องตรวจสอบเซนเซอร์และทำการเขียนโปรแกรมควบคุมใหม่อีกครั้ง ในการเขียนต้องกลับไปศึกษาการเขียนระบบควบคุมเซนเซอร์และระบบควบคุมอากาศยานให้มีประสิทธิภาพมากขึ้น

เอกสารอ้างอิง

[1] GitHub. **"QuadTinnakon"**[Online].
Available : https://github.com/QuadTinnakon?tab=repositories

[2] Slideshare. **"Inertial Measurement Unit"** [Online].
Available : http://www.slideshare.net/hunchxx/inertial-measurement-unit

[3] Data_sheets. **"ADXL345.pdf"** [Online].
Available : http://www.analog.com/static/importedfiles/data_sheets/ADXL345.pdf

[4] InvenSense. **"ITG-3205.pdf.html "** [Online].
Available : http://www.datasheet-pdf.com/datasheet/InvenSense/731954/ITG-3205.pdf.html

[5] Devices. **"atmega328p.aspx"** [Online].
Available : http://www.atmel.com/devices/atmega328p.aspx

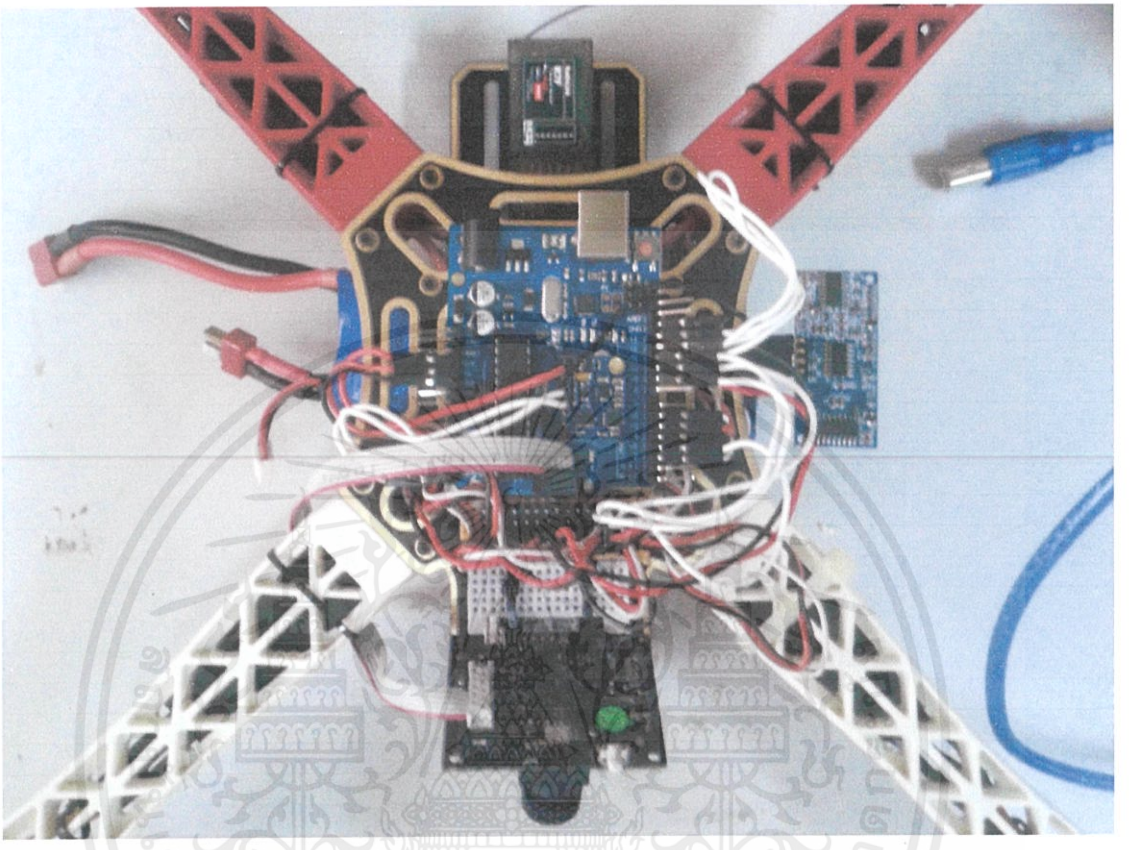
[6] Wikipedia. **"Gyroscope"** [Online].
Available : *th.wikipedia.org/wiki/ไจโรสโคป*

[7] Wikipedia. **"PID Control"** [Online].
Available : *th.wikipedia.org/wiki/ระบบควบคุมพีไอดี*

[8] Control. **"รูปแบบปริญญานิพนธ์ (PDF)"**[Online].
Available : http://www.kmitl.ac.th/control/news.html

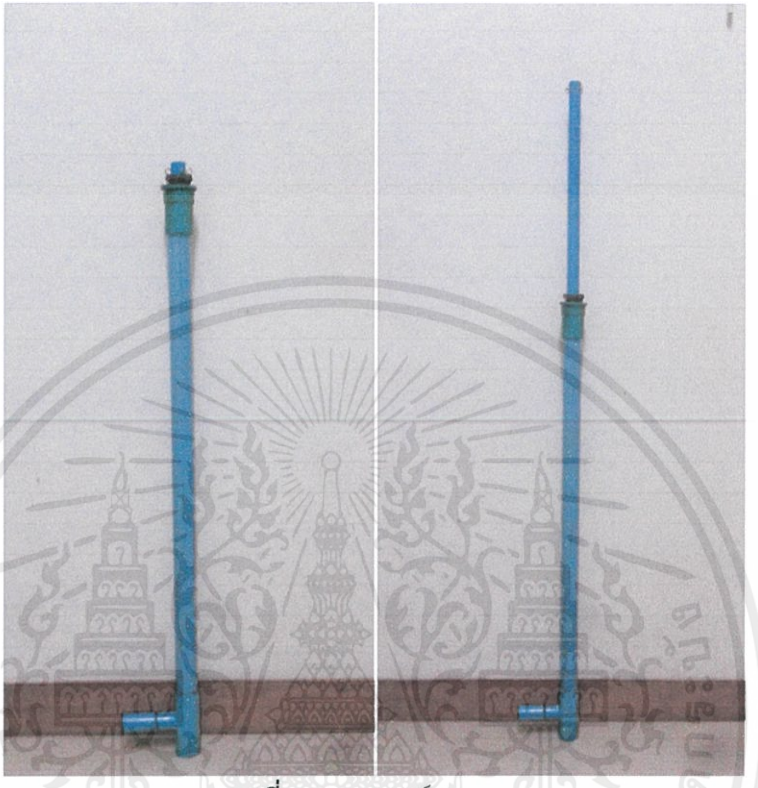
ภาคผนวก

ภาคผนวก ก
วงจรที่ประกอบ



รูปที่ ก.1 วงจรที่ประกอบตามแบบ

ภาคผนวก ข
อุปกรณ์ที่ใช้ในการทดสอบตัวอากาศยาน



รูปที่ ข.1 อุปกรณ์ทดสอบ

ในการทดสอบจะใช้อุปกรณ์ตามรูปที่ ข.1 ในการทดสอบจะยึดตัวอากาศยานไว้บนสุดของอุปกรณ์ที่ทำจากท่อพีวีซี ซึ่งท่อที่ใช้สามารถเลื่อนขึ้นเลื่อนลงได้ทำให้ตรวจสอบได้ว่าอากาศยานสามารถทำงานได้ถูกต้องหรือไม่

ภาคผนวก ค
โค๊ดคำสั่งที่ใช้ควบคุม

```
#include <Servo.h>
#include <Wire.h>

#define THROTTLETIMEOUT 25000
#define PULSETIMEOUT 20000

int throttle;
int CH_THR;
int roll;
int CH_AIL;
int pitch;
int CH_ELE;
int yaw;
int CH_RUD;
int uAlititude = 1060;
#define THROTTLEPIN 2
#define ROLLPIN 4
#define PITCHPIN 5
#define YAWPIN 6
#define AUX1PIN 7

#define FRONTMOTOR_L_PIN 3
#define FRONTMOTOR_R_PIN 10
#define REARMOTOR_L_PIN 11
#define REARMOTOR_R_PIN 9

#define ADXL345_Address 0x53
#define ITG3205_Address 0x68
double accXangle = 0;
double accYangle = 0;
double GyroTemp,GyroX,GyroY,GyroZ,AccX,AccY,AccZ;
double GyroX2, GyroY2, GyroZ2, GyroXf, GyroYf, GyroZf;
double AccXf, AccYf, AccZf;
double gyroXtrim = 0,gyroYtrim = 0,gyroZtrim = 0;
double xtrim = 0,ytrim = 0,ztrim = 0;
double gyro_offsetX = 0;
double gyro_offsetY = 0;
double gyro_offsetZ = 0;
double acc_offsetX = 0;
```

```
double acc_offsetY = 0;
double acc_offsetZ = 0;
double xAngle = 0,yAngle = 0;
#define RAD_TO_DEG 57.295779513082320876798154814105

#define tar 0.01
#define MINTHROTTLE 1060
#define MAXTHROTTLE 1900
#define MINCHECK 1100
#define MAXCHECK 1900
#define MINCOMMAND 1000
#define MAXCOMMAND 1900

double K = 1;
double Kp = 1;
double Ki = 1;
double Kd = 1;

double exInt,eyInt,ezInt;
double q0,q1,q2,q3;
double ahrs_p,ahrs_r,ahrs_y;

double DCM00 = 0;
double DCM01 = 1;
double DCM02 = 0;
double DCM10 = -1;
double DCM11 = 0;
double DCM12 = 0;
double DCM20 = 0;
double DCM21 = 0;
double DCM22 = 1;
unsigned long sensorPreviousTime = 0;
unsigned long previousTime = 0;
double G_Dt = 0.01;
double Dt_sensor = 1000;
double Dt_roop = 10000;
double sensorsample = 0;

double roll_I_rate = 0;
double roll_D_rate = 0;
double err_roll_rate = 0;
double err_roll_ant_rate = 0;
double pitch_I_rate = 0;
```

```
double pitch_D_rate = 0;
double err_pitch_rate = 0;
double err_pitch_ant_rate = 0;
double yaw_I_rate = 0;
double yaw_D_rate = 0;
double err_yaw_rate = 0;
double err_yaw_ant_rate = 0;
double u2_roll = 0;
double u3_pitch = 0;
double u4_yaw = 0;
double accrX_cutg = 0;
double accrY_cutg = 0;
double accrZ_cutg = 0;

Servo FRONTMOTOR_L;
Servo FRONTMOTOR_R;
Servo REARMOTOR_L;
Servo REARMOTOR_R;

int motorFONT_L = MINCOMMAND;
int motorFONT_R = MINCOMMAND;
int motorREAR_L = MINCOMMAND;
int motorREAR_R = MINCOMMAND;

void setup()
{
  Serial.begin(115200);
  RC_Receiver();
  motor_initialize();
  Wire.begin();
  configureReceiver();
  Accel_init();
  delay(10);
  Gyro_init();
  delay(10);
  for(int i=0;i<10;i++)
  {
    Accel_Read();
    Gyro_Read();
    delay(10);
  }
  sensor_Calibrate();
  sensorPreviousTime = micros();
```

```
  previousTime = micros();
}

void loop()
{
  Dt_sensor = micros() - sensorPreviousTime;
  if(Dt_sensor <= 0)
  {
    Dt_sensor = 1001;
  }
  if(Dt_sensor >= 1000 && sensorsample < 3)
  {
    sensorPreviousTime = micros();
    sensor_Read();
  }
  Dt_roop = micros() - previousTime;
  if(Dt_roop <= 0)
  {
    Dt_roop = 10001;
  }
  if(Dt_roop >= 10000)
  {
    previousTime = micros();
    G_Dt = Dt_roop/1000000;
    sensor_Get();

    GyroXf = (GyroX2 + GyroX)/2;
    GyroYf = (GyroY2 + GyroY)/2;
    GyroZf = (GyroZ2 + GyroZ)/2;
    GyroX2 = GyroX;
    GyroY2 = GyroY;
    GyroZ2 = GyroZ;
    AccXf = AccX;
    AccYf = AccY;
    AccZf = AccZ;

    ahrs_update(GyroXf, GyroYf, GyroZf, AccXf, AccYf, AccZf, G_Dt);
    ahrs_toEuler();

    Control_PIDRate();
    motorFONT_L = uAlititude + u3_pitch + u2_roll - u4_yaw;
    motorFONT_R = uAlititude + u3_pitch - u2_roll + u4_yaw;
```

```
    motorREAR_L = uAlititude - u3_pitch + u2_roll + u4_yaw;
    motorREAR_R = uAlititude - u3_pitch - u2_roll - u4_yaw;

    motorFONT_L = map(motorFONT_L, MINCOMMAND, MAXCOMMAND, MINTHROTTLE,
MAXCOMMAND);
    motorFONT_R = map(motorFONT_R, MINCOMMAND, MAXCOMMAND, MINTHROTTLE,
MAXCOMMAND);
    motorREAR_L = map(motorREAR_L, MINCOMMAND, MAXCOMMAND, MINTHROTTLE,
MAXCOMMAND);
    motorREAR_R = map(motorREAR_R, MINCOMMAND, MAXCOMMAND, MINTHROTTLE,
MAXCOMMAND);
    motorCommand();
  }
}
void RC_Receiver()
{
  pinMode(THROTTLEPIN, INPUT);
  pinMode(ROLLPIN, INPUT);
  pinMode(PITCHPIN, INPUT);
  pinMode(YAWPIN, INPUT);
  pinMode(AUX1PIN, INPUT);
}
void motor_initialize()
{
  FRONTMOTOR_L.attach(FRONTMOTOR_L_PIN);
  FRONTMOTOR_R.attach(FRONTMOTOR_R_PIN);
  REARMOTOR_L.attach(REARMOTOR_L_PIN);
  REARMOTOR_R.attach(REARMOTOR_R_PIN);

  FRONTMOTOR_L.write(MINCOMMAND);
  FRONTMOTOR_R.write(MINCOMMAND);
  REARMOTOR_L.write(MINCOMMAND);
  REARMOTOR_R.write(MINCOMMAND);
}
void motorCommand()
{
  FRONTMOTOR_L.write(motorFONT_L);
  FRONTMOTOR_R.write(motorFONT_R);
  REARMOTOR_L.write(motorREAR_L);
  REARMOTOR_R.write(motorREAR_R);
}
```

```
void Control_PIDRate()
{
  roll = pulseIn(ROLLPIN, HIGH, PULSETIMEOUT);
  CH_AIL = map(roll, 1050, 1950, -200, 200);
  double Set_roll = (CH_AIL*K);
  err_roll_rate = (Set_roll - GyroYf*RAD_TO_DEG);
  roll_I_rate += err_roll_rate*G_Dt;
  roll_D_rate += ((err_roll_rate - err_roll_ant_rate)/G_Dt);
  err_roll_ant_rate = err_roll_rate;
  u2_roll = (Kp*err_roll_rate)+ (Ki*roll_I_rate) + (Kd*roll_D_rate);

  pitch = pulseIn(PITCHPIN, HIGH, PULSETIMEOUT);
  CH_ELE = map(pitch, 1050, 1950, -200, 200);
  double Set_pitch = (CH_ELE*K);
  err_pitch_rate = (Set_pitch - GyroXf*RAD_TO_DEG);
  pitch_I_rate += err_pitch_rate*G_Dt;
  pitch_D_rate += ((err_pitch_rate - err_pitch_ant_rate)/G_Dt);
  err_pitch_ant_rate = err_pitch_rate;
  u3_pitch = (Kp*err_pitch_rate)+ (Ki*pitch_I_rate) + (Kd*pitch_D_rate);

  yaw = pulseIn(YAWPIN, HIGH, PULSETIMEOUT);
  CH_RUD = map(yaw, 1050, 1950, -200, 200);
  double Set_yaw = (CH_RUD*K);
  err_yaw_rate = (Set_yaw - GyroZf*RAD_TO_DEG);
  yaw_I_rate += err_yaw_rate*G_Dt;
  yaw_D_rate += ((err_yaw_rate - err_yaw_ant_rate)/G_Dt);
  err_yaw_ant_rate = err_yaw_rate;
  u4_yaw = (Kp*err_yaw_rate)+ (Ki*yaw_I_rate) + (Kd*yaw_D_rate);

  throttle = pulseIn(THROTTLEPIN, HIGH, THROTTLETIMEOUT);
  CH_THR = map(throttle, 1050, 1950, MINTHROTTLE, MAXTHROTTLE);
  uAlititude = uAlititude + ((CH_THR - uAlititude)*K);
}
void Accel_init()
{
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x38);
  Wire.write(0x00);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x31);
  Wire.write(0x0B);
```

```
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x2D);
  Wire.write(0x00);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x1E);
  Wire.write(0x7E);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x1F);
  Wire.write(0x7E);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x20);
  Wire.write(0x7E);
  Wire.endTransmission();
  delay(5);
}
void Gyro_init()
{
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x3E);
  Wire.write(0x80);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x16);
  Wire.write(0x1A);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x15);
  Wire.write(0x09);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x3E);
  Wire.write(0x01);
```

```
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x3E);
  Wire.write(0x02);
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x3E);
  Wire.write(0x03);
  Wire.endTransmission();
  delay(5);
}
void accX()
{
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x32);
  Wire.endTransmission();
  Wire.requestFrom(ADXL345_Address, 2);
  int i = 0;
  byte result[2];
  while(Wire.available())
  {
    result[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  AccX = (result[0] | (result[1] << 8))*9.81;
}
void accY()
{
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x34);
  Wire.endTransmission();
  Wire.requestFrom(ADXL345_Address, 2);
  int i = 0;
  byte result[2];
  while(Wire.available())
  {
    result[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
```

```
  AccY = (result[0] | (result[1] << 8))*-9.81;
}
void accZ()
{
  Wire.beginTransmission(ADXL345_Address);
  Wire.write(0x36);
  Wire.endTransmission();
  Wire.requestFrom(ADXL345_Address, 2);
  int i = 0;
  byte result[2];
  while(Wire.available())
  {
    result[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  AccZ = (result[0] | (result[1] << 8))*9.81;
}
void gyroX()
{
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x1D);
  Wire.endTransmission();
  Wire.requestFrom(ITG3205_Address, 2);
  int i = 0;
  byte buffer[2];
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  GyroX = (((buffer[0] << 8) | buffer[1])*-1/14.375)/RAD_TO_DEG;
}
void gyroY()
{
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x1F);
  Wire.endTransmission();
  Wire.requestFrom(ITG3205_Address, 2);
  int i = 0;
  byte buffer[2];
  while(Wire.available())
```

```
  {
    buffer[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  GyroY = (((buffer[0] << 8) | buffer[1])*-1/14.375)/RAD_TO_DEG;
}
void gyroZ()
{
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x21);
  Wire.endTransmission();
  Wire.requestFrom(ITG3205_Address, 2);
  int i = 0;
  byte buffer[2];
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  GyroZ = (((buffer[0] << 8) | buffer[1])*-1/14.375)/RAD_TO_DEG;
}
void gyroTemp()
{
  Wire.beginTransmission(ITG3205_Address);
  Wire.write(0x1B);
  Wire.endTransmission();
  Wire.requestFrom(ITG3205_Address, 2);
  int i = 0;
  byte buffer[2];
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  GyroTemp = (((buffer[0] << 8) | buffer[1])+13200)/280.0 + 35.0;
}
void Accel_Read()
{
  accX();
  accY();
```

```
  accZ();
}
void Gyro_Read()
{
  gyroX();
  gyroY();
  gyroZ();
  gyroTemp();
}
void sensor_Read()
{
  Accel_Read();
  Gyro_Read();
  gyroXtrim += GyroX;
  gyroYtrim += GyroY;
  gyroZtrim += GyroZ;
  xtrim += AccX;
  ytrim += AccY;
  ztrim += AccZ;
  sensorsample ++;
}
void sensor_Get()
{
  GyroX = (gyroXtrim/sensorsample) - gyro_offsetX;
  GyroY = (gyroYtrim/sensorsample) - gyro_offsetY;
  GyroZ = (gyroZtrim/sensorsample) - gyro_offsetZ;

  AccX = (xtrim/sensorsample) - acc_offsetX;
  AccY = (ytrim/sensorsample) - acc_offsetY;
  AccZ = (ztrim/sensorsample) - acc_offsetZ;

  gyroXtrim = 0;
  gyroYtrim = 0;
  gyroZtrim = 0;
  xtrim = 0;
  ytrim = 0;
  ztrim = 0;
  sensorsample = 0;
}
void sensor_Calibrate()
{
  for(uint8_t i=0;i<20;i++)
  {
```

```
    sensor_Read();
    delay(10);
  }
  gyro_offsetX = gyroXtrim/sensorsample;
  gyro_offsetY = gyroYtrim/sensorsample;
  gyro_offsetZ = gyroZtrim/sensorsample;
  acc_offsetX = xtrim/sensorsample;
  acc_offsetY = ytrim/sensorsample;
  acc_offsetZ = ztrim/sensorsample;
  sensorsample = 0;

  acc_offsetX = 0.49;
  acc_offsetY = 0.10;
  acc_offsetZ = -0.39;
}
void configureReceiver()
{
  PORTC |= ((1<<PORTC4) | (1<<PORTC5));
  TWBR = ((F_CPU/400000L)-16)/2;

  TWDR = 0xFF;
  TWCR = (1<<TWEN)|(0<<TWIE)|(0<<TWINT)|(0<<TWEA)|(0<<TWSTA)|(0<<TWSTO)|(0<<TWWC);
}
void ahrs_initialize()
{
  exInt = 0;
  eyInt = 0;
  ezInt = 0;
  q0 = 1;
  q1 = 0;
  q2 = 0;
  q3 = 0;
}
void ahrs_update(double gx, double gy, double gz, double ax, double ay, double az, double
G_Dt)
{
  double norm;
  double vx, vy, vz;
  double ex, ey, ez;
```



```
  norm = sqrt(ax*ax + ay*ay + az*az);
  ax = ax/norm;
  ay = ay/norm;
```

```
  az = az/norm;

  vx = 2*(q1*q3 - q0*q2);
  vy = 2*(q0*q1 + q2*q3);
  vz = q0*q0 - q1*q1 - q2*q2 + q3*q3;

  ex = (ay*vz - az*vy);
  ey = (az*vx - ax*vz);
  ez = (ax*vy - ay*vx);

  gx += Kp*ex + exInt;
  gy += Kp*ey + eyInt;
  gz += Kp*ez + ezInt;

  q0 = q0 + (-q1*gx - q2*gy - q3*gz)*G_Dt/2;
  q1 = q1 + (q0*gx + q2*gz - q3*gy)*G_Dt/2;
  q2 = q2 + (q0*gy - q1*gz + q3*gx)*G_Dt/2;
  q3 = q3 + (q0*gz + q1*gy - q2*gx)*G_Dt/2;

  norm = sqrt(q0*q0 + q1*q1 + q2*q2 + q3*q3);
  q0 = q0/norm;
  q1 = q1/norm;
  q2 = q2/norm;
  q3 = q3/norm;

  DCM00 = 2*(0.5 - q2*q2 - q3*q3);
  DCM01 = 2*(q1*q2 - q0*q3);
  DCM02 = 2*(q1*q3 + q0*q2);
  DCM10 = 2*(q1*q2 + q0*q3);
  DCM11 = 2*(0.5 - q1*q1 - q3*q3);
  DCM12 = 2*(q2*q3 - q0*q1);
  DCM20 = 2*(q1*q3 - q0*q2);
  DCM21 = 2*(q2*q3 + q0*q1);
  DCM22 = 2*(0.5 - q1*q1 - q2*q2);

  ahrs_y = atan2(DCM10, DCM00);
  ahrs_p = -asin(DCM20);
  ahrs_r = atan2(DCM21, DCM22);

  accrX_cutg = (-AccXf*DCM00 - AccYf*DCM01 + AccZf*DCM02);
  accrY_cutg = (-AccXf*DCM10 - AccYf*DCM11 + AccZf*DCM12);
  accrZ_cutg = (-AccXf*DCM20 - AccYf*DCM21 + AccZf*DCM22) - 9.42;
}
```

```
void ahrs_toEuler()
{
  ahrs_y = atan2(2*q1*q2 + 2*q0*q3, 2*q0*q0 + 2*q1*q1 - 1);
  ahrs_p = -asin(2*q1*q3 - 2*q0*q2);
  ahrs_r = atan2(2*q2*q3 + 2*q0*q1, 2*q0*q0 + 2*q3*q3 - 1);
}
```