เฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

Quad Copter

นิติภัทร เวียงเพ็ม

ปุณยนา ยาพันธ์

พงศกร สง่างาม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2556

เฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

Quad Copter

ปีติภัทร เวียงเพิ่ม
PITIPAT WIANGPERM

ปุณศยา ผาพันธ์
PUNSAYA PHAPHAN

พงศกร สง่างาม
PONGSAKORN SANGANGAM

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2556

เฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

Quad Copter

โดย

ปีติภัทร เวียงเพิ่ม

ปุณศยา ผาพันธ์

พงศกร สง่างาม

อาจารย์ที่ปรึกษา

ผศ.พลผดุง ผดุงกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2556

ปริญญานิพนธ์ปีการศึกษา 2556

สาขา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

Quad Copter

ผู้จัดทำ นายปีติภัทร เวียงเพิ่ม รหัสนักศึกษา 53011004

นางสาวปุณศยา ผาพันธ์ รหัสนักศึกษา 53011007

นายพงศกร สง่างาม รหัสนักศึกษา 53011024

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(ผศ.พลผดุง ผดุงกุล)

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์	เฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด		
นักศึกษา	นายปีติภัทร	เวียงเพิ่ม	รหัสนักศึกษา 53011004
	นางสาวปุณศยา	ผาพันธ์	รหัสนักศึกษา 53011007
	นายพงศกร	สง่างาม	รหัสนักศึกษา 53011024
ปริญญา	วิศวกรรมศาสตร์บัณฑิต		
สาขาวิชา	**วิศวกรรมอิเล็กทรอนิกส์**		
ปีการศึกษา	2556		
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.พลผดุง ผดุงกุล		

บทคัดย่อ

รายงานฉบับนี้เป็นการออกแบบและการสร้างเครื่องบินบังคับวิทยุ 4 ใบพัด ที่มีการควบคุมด้วยรีโมทคอนโทรลที่มีการแปลงสัญญาณก่อนส่ง ด้วยไมโครคอนโทรลเลอร์ โดยเครื่องบินบังคับวิทยุสามารถเคลื่อนที่ได้ 6 ทิศทาง โดยสัญญาณควบคุมจากรีโมทคอนโทรลจะไปเปลี่ยนแปลงการทำงานของมอเตอร์แบบไร้แปรงถ่าน ด้วยบอร์ดควบคุมชุดอาร์ดุยโน่

Thesis Title	Quad Copter			
Student	Mr. PITIPAT	WIANGPERM	Student ID	53011004
	Miss. PUNSAYA	PHAPHAN	Student ID	53011007
	Mr. PONGSAKORN	SANGANGAM	Student ID	53011024
Degree	Bachelor of Engineering			
Program	Electronics Engineering			
Year	2013			
Thesis Advisor	Asst.Prof. POLPHADUNG PHADUNGKUL			

Abstract

This report is Project to design and build Quad copter that control by remote control. It's received signal form micro controller and can fly 6 directions. Signal form remote control will convert Brushless Motor by Arduino Board

กิตติกรรมประกาศ

โครงงานเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัดนี้จะไม่สามารถดำเนินการให้สำเร็จได้ หากขาดความอนุเคราะห์และกรุณาจาก ผศ.พลผดุง ผดุงกุล ท่านอาจารย์ที่ปรึกษาโครงงาน ที่ให้ความรู้และเทคนิคต่างๆ ตลอดจนให้คำปรึกษา ชี้แนะในระหว่างการดำเนินโครงงาน กระทั่งงานสำเร็จลุล่วงไปได้ด้วยดี ขอกราบขอบพระคุณท่านอาจารย์เป็นอย่างสูง

ขอกราบขอบพระคุณ คุณพ่อคุณแม่ สำหรับกำลังใจที่มอบให้ อีกทั้งการอบรมสั่งสอนให้มีระเบียบวินัย จนสามารถนำคำสอนและข้อคิดมาใช้ในการดำเนินโครงงานจนประสบความสำเร็จ

ขอขอบคุณเพื่อนๆ ในภาควิชาอิเล็กทรอนิกส์ ที่ได้ให้การช่วยเหลือและอยู่เคียงข้างกันจนกระทั่งงานครั้งนี้สำเร็จลุล่วงไปได้

สุดท้ายนี้ขอขอบคุณโชคชะตาที่ทำให้เราได้เรียนคณะวิศวกรรมศาสตร์ ภาควิชาอิเล็กทรอนิกส์ และสถาบันแห่งนี้ จนได้มาพบกับท่านคณาจารย์ พี่ๆ เพื่อน และได้ทำโครงงานนี้ไปพร้อมกับเพื่อนๆ

ปีติภัทร เวียงเพิ่ม
ปุณศยา ผาพันธ์
พงศกร สง่างาม

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

โลกในยุคโลกาภิวัตน์ ยุคที่ข้อมูลข่าวสารไร้พรมแดน เทคโนโลยีต่างๆเข้ามามีบทบาทมากขึ้น โดยเฉพาะในด้านของการสื่อสาร และรับ-ส่งข้อมูล ซึ่งก่อเกิดแนวคิดในการถ่ายภาพและภาพเคลื่อนไหวจากมุมสูงและในพื้นที่ที่ไม่สามารถเข้าถึงได้ด้วยบุคคล ดังนั้นแนวคิดดังกล่าวจึงเป็นต้นเหตุในการพัฒนาเป็นเครื่องบินบังคับวิทยุขนาดเล็ก ซึ่งมีความเหมาะสมสำหรับการสำรวจถ่ายภาพ หรือถ่ายภาพยนตร์ สื่อโฆษณาต่างๆ แทนการใช้เครื่องบินหรือเฮลิคอปเตอร์ขนาดใหญ่ ที่ต้องมีคนบังคับอยู่ภายในห้องเครื่อง ที่ถือว่าเป็นความเสี่ยงต่ออันตรายอย่างยิ่ง และแน่นอนว่าการใช้เครื่องบินบังคับวิทยุ จะมีค่าใช้จ่ายที่ต่ำกว่า และสะดวกมากกว่าด้วยเทคโนโลยีทางอิเล็กทรอนิกส์ ซึ่งปัจจุบันมีแนวโน้มว่าจะมีขนาดเล็กลงเรื่อยๆ และราคาก็ยังถูกลงอีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาและออกแบบองค์ประกอบของเครื่องบินบังคับวิทยุ โดยคำนึงถึงสมดุลต่างๆ

1.2.2 เพื่อศึกษาหลักการทำงานและการควบคุมของเครื่องบินบังคับวิทยุ

1.2.3 เพื่อศึกษาและนำความรู้ทางด้านซอฟแวร์มาประยุกต์เข้ากับส่วนของฮาร์ดแวร์ในการแก้ไขปัญหาที่เกิดขึ้นระหว่างการทำโครงงาน

1.3 ขอบเขตของโครงงาน

ศึกษา ออกแบบ และจัดทำเครื่องบินบังคับวิทยุแบบ 4 ใบพัด ขนาด 450 mm. โดยมีฟังก์ชันการทำงานหลัก ได้แก่ การบินขึ้น-ลง บินเดินหน้า-ถอยหลัง การเลี้ยวซ้าย-ขวา

1.4 ขั้นตอนการดำเนินโครงงาน

1.4.1 ศึกษาหลักการเบื้องต้นของเฮลิคอปเตอร์ 4 ใบพัด

1.4.2 ศึกษาคุณสมบัติและการทำงานของ Brushless DC Motor

1.4.3 ทดสอบเขียนโปรแกรมไมโครคอนโทรลเลอร์ควบคุม Brushless Speed Controller

1.4.4 เขียนโปรแกรมสร้างสัญญาณ Pulse Width Modulation ส่งให้กับ Brushless Speed Controller ในการขับและควบคุมความเร็วของ Brushless DC Motor

1.4.5 ออกแบบและสร้าง flame ของตัวเฮลิคอปเตอร์

1.4.6 ประกอบ Brushless DC Motor, Brushless Speed Controller เข้ากับตัว flame

1.4.7 ศึกษาสัญญาณจากจอยสติก

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้เรียนรู้การออกแบบระบบควบคุมโดยใช้ไมโครคอนโทรลเลอร์

1.5.2 ได้ฝึกทักษะการแก้ปัญหา การทำงานเป็นกลุ่ม และการบริหารเวลา

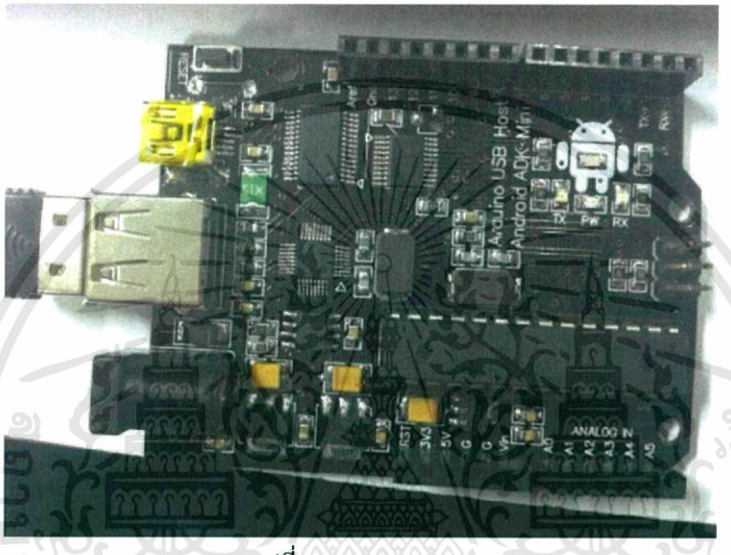
1.5.3 ได้ฝึกการค้นคว้าและหาความรู้ด้วยตนเอง

1.5.4 เพื่อเป็นต้นแบบในการพัฒนาต่อยอดต่อไป

1.5.5 เพื่อศึกษาและประยุกต์ใช้งานไจโรเซนเซอร์

บทที่ 2
ทฤษฎี

2.1 บอร์ดอาร์ดุยโน่ (Arduino Board)



รูปที่ 2.1 Arduino Board

Arduino เป็น platform ของ I/O บอร์ดอย่างง่ายๆ ที่มี I/O ขั้นพื้นฐานที่พอเพียงกับการใช้งาน และ การเรียนรู้ โดยตัวบอร์ดจะมาพร้อมกับชุดคาสั่งที่ใช้ควบคุม port I/O ไม่ว่าจะเป็น port digital, port analog, PWM และ Serial port ซึ่ง Arduino นั้นเป็นเครื่องมือที่จะทาให้คอมพิวเตอร์สามารถรับสัญญาณจากภายนอกและส่งสัญญาณไป ควบคุมอุปกรณ์ภายนอกได้อย่างมีประสิทธิภาพมากกว่าใช้เครื่องพีซีตั้งโต๊ะ ตัวบอร์ดออกแบบจากไมโครคอมพิวเตอร์ชิพเดี่ยว, และมีโปรแกรมพัฒนาสาหรับเขียนโปรแกรมให้บอร์ดทางาน Arduino สามารถประยุกต์ทาเครื่องใช้อัจฉริยะ รับสัญญาณจากสวิทช์ หรือ เซนเซอร์, และควบคุม หลอดไฟ, มอเตอร์, หรืออุปกรณ์อื่นๆ โปรเจค Arduino เป็นได้ทั้งแบบทางานอิสระ หรือทางานติดต่อกับโปรแกรมที่ทางานบนเครื่องพีซี ตัวบอร์ดสามารถประกอบขึ้นใช้เอง หรือจะซื้อสำเร็จที่มีขาย ส่วนโปรแกรมพัฒนา Arduino สามารถดาวน์โหลดได้ฟรี

Arduino เป็นภาษาอิตาลี ซึ่งใช้เป็นชื่อของโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open Source ที่ได้รับการปรับปรุงมาจากโครงการพัฒนา Open Source ของ AVR อีกโครงการหนึ่งที่ชื่อว่า "Wiring" แต่เรื่องจากโครงการของ "Wiring" เลือกใช้ AVR เบอร์ ATmega128 ซึ่งเป็นไมโครคอนโทรลเลอร์ ที่มีจำนวนของหน่วยความจำ และ I/O ค่อนข้างมากและที่สำคัญ ATmega128 เป็นชิพที่มีตัวถังแบบ SMD จึงทาให้เป็นอุปสรรคสาหรับผู้เริ่มต้นในการสร้างบอร์ดและต่อวงจรขึ้นมาใช้งานกันเอง และบอร์ดจะมีขนาดค่อนข้างใหญ่ ซึ่งอาจดูว่าเกินความจำเป็นสาหรับผู้เริ่มต้น จึงไม่ค่อยได้รับความนิยมเท่าที่ควร แต่หลังจากที่ทีมงาน Arduino นา Source Code ของ "Wiring" มาพัฒนาปรับปรุงใหม่โดยให้สามารถใช้งานกับไมโครคอนโทรลเลอร์ขนาดเล็ก เช่น Mega8, Mega168, Mega 328 ได้ จึงทาให้ระบบวงจรของบอร์ดมีขนาดเล็กลงกว่า "Wiring" มากและยังใช้อุปกรณ์น้อยชิ้น ทาให้ง่ายต่อการต่อวงจรใช้งานกันเอง และยังประหยัดต้นทุนในการสร้างบอร์ดไปได้มาก ด้วยเหตุนี้เองที่ทาให้ "Arduino" ได้รับความนิยมจากผู้ใช้งานทั่วโลกเป็นอย่างมากในระยะเวลาอันรวดเร็ว

Arduino มีจุดเด่นในเรื่องของความง่ายต่อการเรียนรู้และใช้งาน เนื่องจากมีการออกแบบคาสั่งต่างๆขึ้นมาสนับสนุนการใช้งาน ด้วยรูปแบบที่ง่ายไม่ซับซ้อน ในตลาดไมโครคอนโทรลเลอร์มีตัวเลือกมากมาย เช่น Parallax Basic Stamp, Netmedia's BX-24, Pidgets, MIT's Handyboard, และอีกหลายเจ้าที่มีคุณสมบัติใกล้เคียงกัน คือทาโปรเจคให้ใช้งานง่าย และเน้นการโปรแกรมไมโครคอนโทรลเลอร์เป็นหลัก Arduino ก็เช่นเดียวกันแต่มีข้อแตกต่างที่เห็นได้ชัดคือ

- มีราคาไม่แพง เนื่องจากมี Source Code และวงจรแจกฟรี สามารถต่อวงจรขึ้นมาใช้งานได้เอง
- ทางานได้หลายแพลตฟอร์ม -โปรแกรมพัฒนา Arduino ทางานได้ทั้งบนวินโดวส์ Macintosh OSX และ บนลีนุกซ์ ในขณะที่บอร์ดอื่นทางานได้เฉพาะบนวินโดวส์
- ใช้งานง่าย มีโปรแกรมพัฒนาที่ไม่ซับซ้อน มีโปรแกรมพัฒนา Arduino ใช้งานง่ายสาหรับมือใหม่และมีความสามารถครบความต้องการของนักพัฒนามืออาชีพ
- เปิดเผยซอร์สโค้ด และ นาไปพัฒนาต่อยอดได้ โปรแกรม Arduino ตีพิมพ์แบบเปิดเผย ซอร์สโค้ดและสามารถเพิ่มเติมความสามารถผ่าน C++ library, ถ้าต้องการศึกษาให้ลึกซึ้งสามารถข้ามไปเล่น AVR C ซึ่งเป็นต้นแบบของ Arduino, และสามารถเพิ่มเติม AVR - C โค้ดได้โดยตรง
- เปิดเผยวงจร และนาไปพัฒนาขยาย Hardware ได้ Arduino ใช้โมโครคอนโทรลเลอร์ของ Atmel วงจรของบอร์ดตีพิมพ์แบบเปิดเผยวงจรภายใต้ Creative Commons License สามารถนาไปดัดแปลงต่อขยายและเพิ่มประสิทธิภาพ เพื่อศึกษาการทางานของได้ฟรี

Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์โดยใช้ AVR ขนาดเล็กเป็นตัวประมวลผลและสั่งงาน เหมาะสาหรับนาไปใช้ในการศึกษาเรียนรู้ระบบไมโครคอนโทรลเลอร์ และนาไปประยุกต์ใช้งานเกี่ยวกับการควบคุมอุปกรณ์ Input / Output ต่างๆได้มากมาย ทั้งในแบบที่เป็นการทางานตัวเดียวอิสระ หรือเชื่อมต่อสั่งงานร่วมกับอุปกรณ์อื่นๆ เช่น คอมพิวเตอร์ PC ทั้งนี้ก็เนื่องมาจากว่า Arduino สนับสนุนการเชื่อมต่อกับอุปกรณ์ Input / Output ต่างๆได้มากมาย ทั้งแบบ Digital และ

Analog เช่น การรับค่าจากสวิตช์ หรืออุปกรณ์ตรวจจับ (Sensor) แบบต่างๆ รวมไปถึง การควบคุมอุปกรณ์ Output ต่างๆ ตั้งแต่ LED, หลอดไฟ, มอเตอร์, รีเลย์ ฯลฯ โดยระบบฮาร์ดแวร์ของ Arduino สามารถสร้างและประกอบขึ้นใช้งานได้เอง ในกรณีที่ผู้ใช้พอมีความรู้ด้านอิเล็กทรอนิกส์อยู่บ้าง หรือสามารถซื้อแผงวงจรสำเร็จรูปที่มีการผลิตออกจำหน่ายกันในราคาที่ไม่แพง อีกทั้งยังเผยแพร่ Source Code และตัวอย่างต่างๆ ให้ผู้ใช้นาไปใช้งาน หรือพัฒนาดัดแปลงต่อยอดได้โดยไม่เสียค่าใช้จ่ายใดๆ

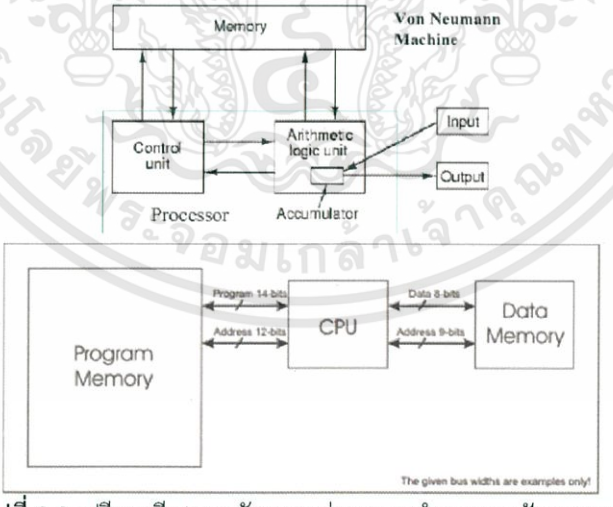
ส่วนภาษาในการเขียนโปรแกรมลงบน Arduino นั้นจะใช้ภาษา C++ ซึ่งเป็นรูปแบบของโปรแกรม ภาษาซีประยุกต์แบบหนึ่ง ที่มีโครงสร้างของตัวภาษาโดยรวมใกล้เคียงกันกับภาษาซีมาตรฐาน (ANSI-C) อื่นๆ เพียงแต่ได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจาก ANSI-C เล็กน้อย เพื่อช่วยลดความยุ่งยากในการเขียนโปรแกรมและให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรม ได้ง่ายและสะดวกมากขึ้นกว่าการเขียนภาษาซีตามแบบมาตรฐานของ ANSI-C โดยตรง ซึ่งจากการที่ได้ทาการศึกษาค้นคว้าทดลองการใช้งานภาษาซีของ Arduino มาในระยะเวลาหนึ่งจะพบว่าในความเป็นจริงแล้ว Arduino นั้นไม่ใช่ C-Compiler โดยตรง แต่ Arduino จะมีลักษณะการ ทางานเช่นเดียวกันกับ Text Editor เป็นฉากหน้าในการติดต่อสื่อสารกับผู้ใช้เท่านั้น ส่วนเบื้องหลังจริงๆ นั้น Arduino จะไปเรียกใช้ตัวแปลภาษาซีและ Utility อื่นๆ ที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของ ไมโครคอนโทรลเลอร์ตระกูล AVR อีกทีหนึ่ง โดย Arduino จะเลือกใช้ C-Compiler ของ “GNU AVR-GCC Toolchain” ร่วมกับ Library Function ของ “avr-libc” ส่วน Utility ที่ใช้ในการ Upload Code ให้กับ AVR นั้นก็จะใช้ของ “AVRDude” ดังนั้นผู้ที่เขียนภาษาซีของ AVR เป็นอยู่แล้ว และต้องการประยุกต์ใช้ งาน Arduino ให้ได้ประสิทธิภาพการทางานมายิ่งขึ้นไปอีก ก็สามารถศึกษาข้อกำหนด และหน้าที่ใน การใช้งาน Library และคา สั่งอื่นๆที่บรรจุ ไว้ใน Library ต่างๆ ทั้งจากของ “GNU AVR-GCC Toolchain” และ “avr-libc” เพิ่มเติมอีก เพื่อใช้เป็นแนวทางในการปรับปรุงและประยุกต์ใช้งาน Arduino ในรูปแบบที่สลับซับซ้อนมากๆ ขึ้นไปได้อีก

2.1.1 ชิพ และ ไอซี ภายในบอร์ดอาร์ดุยโน่ที่สำคัญ

- ATmega328P-PU

รูปที่2.2 ซีพียู ATMEGA328P ขนาด 28 ขา

ตัวบอร์ด Arduino ที่ใช้ในโปรเจคนี้จะกล่าวถึงสถาปัตยกรรมของ AVR ขนาด 8 บิต โดยในสถาปัตยกรรม AVR ซึ่งออกแบบโดย ATMEL เมื่อปี 1996 เป็นซีพียูแบบ RISC (Reduced Instruction Set Computer) มีสถาปัตยกรรมการต่อหน่วยความจำแบบ Harvard ซึ่งแยกหน่วยความจำโปรแกรม และหน่วยความจำ ข้อมูลออกจากกันโดยเด็ดขาด ดังแสดงในรูปที่2.2 โดยใช้หน่วยความจำแบบ Flash สาหรับเป็นหน่วยความจำโปรแกรม และใช้หน่วยความจำแบบ SRAM สำหรับหน่วยความจำข้อมูล และ นอกจากนี้ยังมีหน่วยความจำ แบบ EEPROM ซึ่งสามารถเก็บข้อมูลเอาไว้ได้โดยไม่จำเป็นต้องมีไฟเลี้ยง อีกด้วย

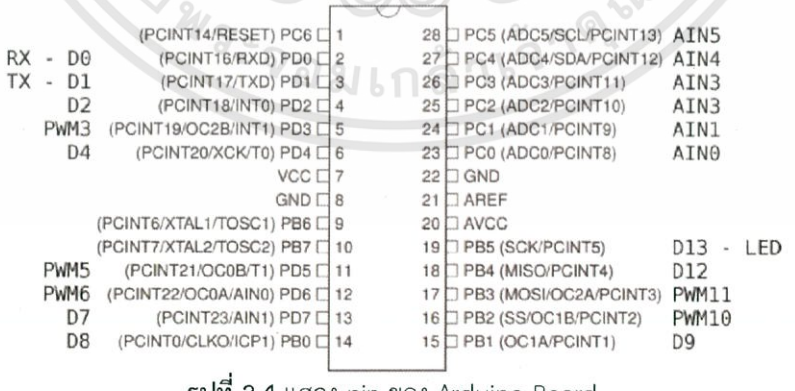


รูปที่ 2.3 เปรียบเทียบการจัดการหน่วยความจำของสถาปัตยกรรมแบบ Von-Neumann และ Harvard

จากรูปที่ 2.3 จะเห็นว่าโพรเซสเซอร์ที่ใช้สถาปัตยกรรมแบบ Harvard จะแยกหน่วยความจำสำหรับเก็บข้อมูลออกจากโปรแกรมอย่างชัดเจน สถาปัตยกรรม AVR และ MCS-51 จะใช้รูปแบบนี้ในการจัดการหน่วยความจำ ส่วนสถาปัตยกรรมแบบ Von-neumann การตัดสินใจว่าจะเก็บโปรแกรมหรือข้อมูลจะแบ่งเก็บอย่างไรจะทำได้อย่างอิสระ โดยขึ้นอยู่กับโปรแกรมเมอร์ หรืออาจจะเป็นระบบ

ปฏิบัติการเป็นผู้ดำเนินการให้ลักษณะเด่นของสถาปัตยกรรม AVR คือ คำสั่งส่วนใหญ่สามารถทำงานได้เสร็จภายใน 1 clock cycle ตัวซีพียู AVR ขนาด 8 บิตจะแบ่งออกเป็นประเภทการใช้งานได้ 5 กลุ่ม ได้แก่

- tinyAVR เป็นซีพียูในรุ่นเล็ก ซึ่งต้องการความเล็กกะทัดรัดของวงจร โดยเหมาะกับระบบควบคุมขนาดเล็กๆ ที่ต้องการหน่วยความจำและวงจรสนับสนุนไม่มากนัก ซีพียูในรุ่นนี้จะมีราคาถูกกว่ากลุ่มอื่นๆ
- megaAVR จะมีชื่ออีกอย่างว่า ATmega โดยมีวงจรสนับสนุนภายในเพิ่มเติมตลอดจนเพิ่มขนาดของหน่วยความจำให้ใช้งานมากกว่าตระกูล Tiny เหมาะกับงานควบคุมทั่วๆ ไป
- XMEGA เพิ่มความละเอียดของวงจร A/D จากปกติมีความละเอียด 10 บิตในรุ่นเล็กกว่าเป็น 12 บิต และวงจร DMA controller ซึ่งช่วยลดภาระของซีพียูในการควบคุมการรับส่งข้อมูลระหว่างอุปกรณ์ I/O กับหน่วยความจำ
- FPSLIC (AVR core with FPGA) สำหรับงานที่ต้องการควบคุมที่ต้องการความยืดหยุ่นในขั้นตอนการออกแบบและพัฒนา โดยผู้ออกแบบสามารถออกแบบวงจรในระดับฮาร์ดแวร์เพิ่มเติมด้วยภาษาบรรยายฮาร์ดแวร์ (HDL: Hardware Description Language) เช่น ภาษา VHDL หรือภาษา Verilog และให้วงจรที่ออกแบบทางานร่วมกับซีพียู AVR core
- Application Specific AVR เป็นซีพียูที่ออกแบบมาโดยเพิ่มวงจรควบคุมเฉพาะด้านเข้าไปซึ่งไม่พบในซีพียูกลุ่มอื่นๆ เช่น วงจร USB controller หรือ CAN Bus เป็นต้น



รูปที่ 2.4 แสดง pin ของ Arduino Board

ซีพียู AVR มีให้เลือกใช้งานหลายเบอร์ แต่ละเบอร์จะมีขนาด ราคา ความสามารถ และขนาดหน่วยความจำตลอดจนถึงวงจรสนับสนุนภายในที่แตกต่างกันออกไป ในโครงงานนี้จะเลือกใช้ซีพียูรุ่น ATmega328P ซึ่งมีคุณสมบัติดังนี้

- หน่วยความจำโปรแกรมแบบ FLASH ขนาด 32 กิโลไบต์
- หน่วยความจำข้อมูลแบบ SRAM ขนาด 2 กิโลไบต์
- หน่วยความจำข้อมูลแบบ EEPROM ขนาด 1 กิโลไบต์
- สนับสนุนการเชื่อมต่อแบบ I2C bus
- พอร์ตอินพุตเอาต์พุตจำนวน 23 บิต
- วงจรสื่อสารอนุกรม
- วงจรนับ/จับเวลาขนาด 8 บิต จำนวน 2 ตัว และu3586 ขนาด 16 บิตจำนวน 1 ตัว
- สนับสนุนช่องสัญญาณสาหรับสร้าง Pulse Width Modulation (PWM) จำนวน 6 ช่อง
- วงจรแปลงอนาลอกเป็นดิจิตอลขนาด 10 บิตในตัวจำนวน 8 ช่อง
- ทางานได้ตั้งแต่ย่านแรงดัน 1.8-5.5 Volts
- ความถี่ใช้งานสูงสุด 20 MHz

2.2 มอเตอร์ (Motor)

มอเตอร์ไฟฟ้าเป็นอุปกรณ์ไฟฟ้าที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังกล มอเตอร์ที่ใช้งานในปัจจุบัน แต่ละชนิดก็จะมีคุณสมบัติที่แตกต่างออกไป ขึ้นอยู่กับผู้ออกแบบว่าต้องการจะนำไปใช้ในลักษณะงานใด เช่นมอเตอร์ในงานอุตสาหกรรมต้องการแรงบิดมากมอเตอร์ในของเล่นต่างๆ ต้องการความเร็ว รอบหรือกำลังงานที่แตกต่างกันซึ่งมอเตอร์แต่ละชนิด จะมีคุณสมบัติที่แตกต่างกันมากมาย แต่สามารถแบ่งตามการใช้กระแสไฟฟ้า ก็จะแบ่งได้เป็น 2 ชนิด

2.2.1 มอเตอร์กระแสตรง (DC: Direct Current Motor)

มอเตอร์ชนิดนี้ต้องการแรงดันไฟฟ้ากระแสตรง ส่วนมากใช้งานกับเครื่องเล่นเด็ก อุปกรณ์ที่ใช้มอเตอร์กระแสตรง เช่น มอเตอร์ในรถจักรยานไฟฟ้า ที่ปัดน้ำฝน เป็นต้น มอเตอร์กระแสตรงยังสามารถแบ่งได้เป็น 2 ชนิด

2.2.1.1 มอเตอร์กระแสตรง (DC Motor)

เป็นมอเตอร์ที่หมุนได้ทันทีเมื่อมีการป้อนแรงดันที่เหมาะสม ทิศทางการหมุนจะขึ้นอยู่กับขั้วแรงดันที่ป้อน หากต้องการให้มอเตอร์หมุนกับทางก็เพียงแคร่กับขั้วของแหล่งจ่าย เท่านั้น มอเตอร์ก็จะเปลี่ยนทิศทางทันที มอเตอร์ชนิดนี้ทำงานเพียง 3 แบบ คือหมุนตามเข็มนาฬิกา หมุนทวนเข็มนาฬิกา และหยุดหมุน ซึ่งอัตราความสำเร็จที่หมุนขึ้นอยู่กับกระแสและแรงดันที่จ่ายให้

มอเตอร์ ถ้าหากแรงบิดไม่เพียงพอสามารถทดรอบของการหมุนได้ มอเตอร์ต้องแรงดันไฟฟ้าในระดับต่างกัน ก่อนไปใช้ควรดูว่ามอเตอร์นี้ต้องการแรงเท่าไร

2.2.1.2 สเต็ปเปอร์มอเตอร์ (Stepper motor)

เป็นมอเตอร์ที่มีการหมุนเป็นสเต็ป สามารถกำหนดตำแหน่งการหมุนได้อย่างแม่นยำ ข้อดีของสเต็ปเปอร์มอเตอร์เทียบกับ DC มอเตอร์

- สามารถควบตำแหน่งในการหมุนได้แม่นยำโดยอาศัยการนับจำนวนพัลล์ที่ส่งไปควบคุมการหมุน

- ไม่มีส่วนของแปรงถ่านที่จะสึกหรอและไม่เกิดการสปาร์คที่แปรงถ่านซึ่งอาจก่อ ให้เกิดสัญญาณรบกวน

2.2.2 มอเตอร์ไฟฟ้ากระแสสลับ (AC: Alternating current Motor) หรือเอซี มอเตอร์

มอเตอร์กระแสสลับเป็นมอเตอร์ที่ใช้กันแพร่หลายในอาคารบ้านเรือน หรือโรงงานอุตสาหกรรมเช่น เครื่องไฟฟ้าที่ต้องใช้แรงขับ จำพวกพัดลม แอร์ ตู้เย็น เครื่องดูดฝุ่น และอื่นๆ ซึ่งมอเตอร์ไฟฟ้ากระแสสลับยังสามารถแบ่งชนิดได้อีกดังนี้

2.2.2.1 มอเตอร์ไฟฟ้ากระแสสลับชนิด 1 เฟส จะใช้กับแรงดันไฟฟ้า 220 โวลต์มีสายไฟ เข้า 2 สาย มีแรงม้าไม่สูง ส่วนใหญ่ตามบ้านเรือน

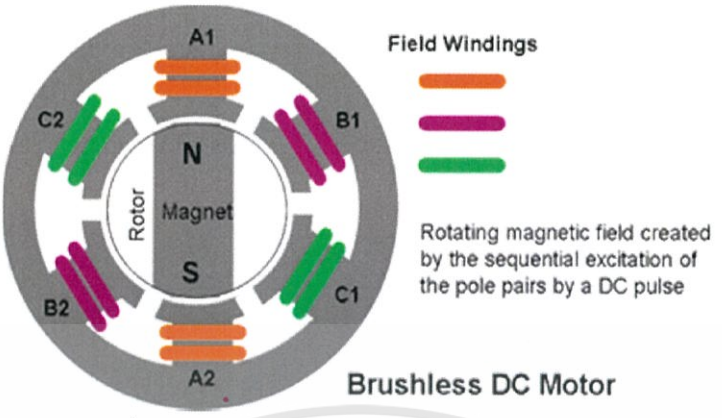
2.2.2.2 มอเตอร์ไฟฟ้ากระแสสลับชนิด 3 เฟส เป็นมอเตอร์ที่ใช้ในงานอุตสาหกรรม ต้องใช้ระบบไฟฟ้า 3 เฟส ใช้แรงดัน 380 โวลต์ มีสายไฟเข้ามอเตอร์ 3 สาย

2.2.3 มอเตอร์ไร้แปรงถ่าน (Brushless Motor)

มอเตอร์แบบไร้การแปรงถ่านเป็นซิงโครนัสมอเตอร์ มีความเหมาะสมในการนำมาควบคุมการไดร์ฟแบบถาวร เช่น ปั๊มน้ำมันเชื้อเพลิง, พัดลมทำความเย็น มอเตอร์ที่มีการแปลงถ่านแบบ 3, 4 และ 5 เฟส มีการเพิ่มเฟสขึ้นเพื่อมาแทนที่ของมอเตอร์ที่มีการแปรงถ่าน

ประโยชน์ของ มอเตอร์แบบไร้การแปรงถ่าน คือความแข็งแรงที่มากกว่าแบบมีการแปรงถ่าน และสิ่งที่สำคัญอีกอย่างคือ มีประสิทธิภาพ และมีแรงบิดสูง

การหมุนของมอเตอร์แบบไร้แปรงถ่าน เกิดจากแม่เหล็กที่สามรถปรับได้ตั้งแต่ 2-8 ขั้ว ด้วยการปรับเปลี่ยน ขั้วเหนือ(N) และ ขั้วใต้(S) โดยการหมุนยังอยู่บนพื้นฐานของทฤษฎีความหนาแน่นของสนามแม่เหล็ก

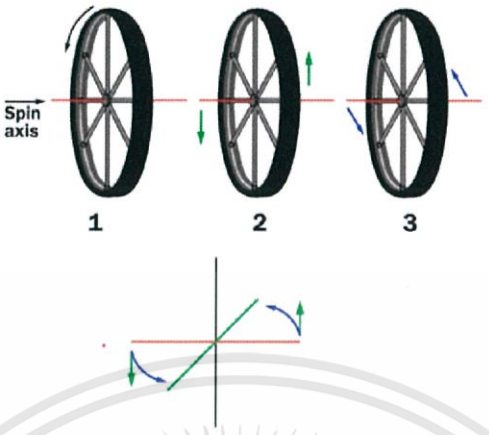


รูปที่ 2.5 ขดแม่ลวดภายในตัวมอเตอร์แบบไร้แปลงถ่าน

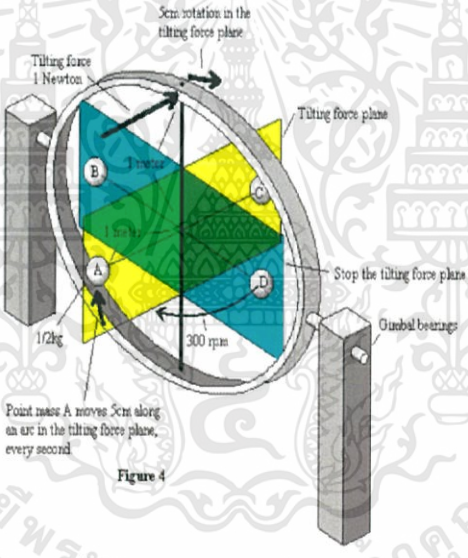
2.3 ไจโรสโคป (Gyroscope)

การทำงานของไจโรสโคปนั้น จะเป็นไปตามกฎของนิวตันคือ มวลจะเคลื่อนที่เป็นเส้นตรงด้วยความเร็วคงที่ ถ้าไม่มีแรงภายนอกมากระทำ เมื่อตัวไจโรหมุนไป 90 องศา จุดบนจะหมุนเปลี่ยนตำแหน่งไป 90 องศา และยังเคลื่อนที่ไปทางซ้าย เช่นเดียวกับจุดล่าง เมื่อหมุนขึ้นมา 90 องศา มันยังคงเคลื่อนที่ไปทางขวา ทำให้ล้อเกิดการหมุนควง ขณะที่จุดบนและจุดล่างเปลี่ยนตำแหน่งไป 90 องศา การเคลื่อนที่ในครั้งแรก จะถูกยกเลิกไป ไม่เกิดการพลิกของล้อ ดังนั้นแกนหมุนของไจโรจะเหมือนกับห้อยอยู่กับที่ตลอดเวลาหรือนั้นก็คือ เมื่อวัตถุเกิดการเคลื่อนที่ มวล (proof mass) ของ ไจโรสโคป จะเคลื่อนที่เช่นกันแต่เคลื่อนที่ช้ากว่าเนื่องจากความเฉื่อยและทาให้ ตำแหน่งของปลายทั้งสองด้านของมวลเปลี่ยนแปลงไป อนึ่งตำแหน่งที่เปลี่ยนไปนี้สามารถวัดได้ด้วยเซ็นเซอร์แบบตัวเก็บประจุไฟฟ้า (capacitive sensing) ทาให้รู้ค่าความเร่งเชิงเส้นและความเร่งเชิงมุมของการเคลื่อนที่นั้นได้

ข้อดีของไจโรสโคป คือมีขนาดเล็กทำให้มีความไวในการตอบสนองสูงและสามารถขยายช่วงความถี่ในการทำงานให้ครอบคลุมความถี่ที่สูงขึ้นได้



รูปที่ 2.6 ทิศทางการควบคุมของไจโรสโคป

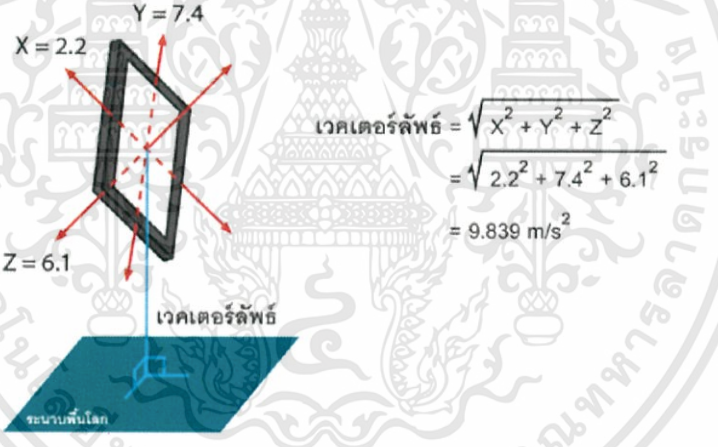


รูปที่ 2.7 หลักการทำงานของไจโรสโคป

2.4 ตัววัดความเร่ง (Accelerometer)

Accelerometer คือ เครื่องตัววัดความเร่ง ของการเคลื่อนที่ของวัตถุ หรือ เซ็นเซอร์ที่ใช้ตรวจวัดความเอียงของเครื่องในทั้ง 3 แกน คือแกน x แกน y และแกน z เป็นที่คุ้นเคยดีในมือถือสมาร์ทโฟนทั่วไป เช่น iPhone ตัวอย่างการใช้งานเช่น การเขย่าเพื่อเปลี่ยนเพลง หรือการเขย่าตัวเครื่องเพื่อใช้ในการควบคุมการเล่นเกม ล้วนเป็นคุณสมบัติของ accelerometer ที่ติดมาในเครื่อง

Accelerometer จะวัดความเร่งในแต่ละแกนง่ายๆก็คือ เวลาที่เครื่องอยู่นิ่งๆไม่มีการขยับค่าแต่ละแกนก็เป็น 0 แต่ในความเป็นจริง อย่าลืมว่ายังมีแรงโน้มถ่วงของโลกอยู่ด้วยดังนั้นค่าจาก Accelerometer จึงไม่ได้เป็น 0 ทั้งหมด เวลาไม่เคลื่อนที่ถ้าเราตั้งเครื่องให้แกน Z ตั้งฉากกับพื้นโลก แกน X และ Y จะเป็น 0 แต่ว่าแกน Z จะไม่เป็น 0 เพราะมีแรงโน้มถ่วงของโลกกระทำอยู่ดังนั้นค่าที่ได้จากแกน Z จะเป็น 9.8 m/s^2 แต่มันเป็นค่าในอุดมคติในความเป็นจริงเป็นไปไม่ได้อยู่แล้วที่จะได้ค่าเป็น 9.8 ตลอดเวลา

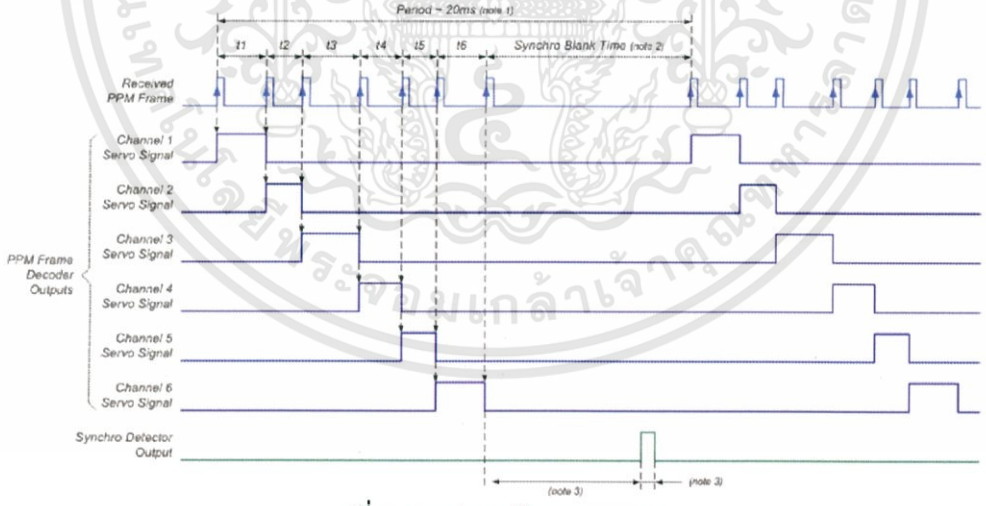


รูป 2.8 รูปการคำนวณเวคเตอร์จาก Accelerometer

2.5 รีโมท คอนโทรล (Remote Control)

รูป 2.9 รูปของ Remote Control

Remote Control ทำหน้าที่ในการส่งสัญญาณ PPM เข้าสู่ตัวรับหรือ Rx Module เพื่อรับสัญญาณไปใช้ในการทำงาน โดย Channel ที่ใช้ในการควบคุมคอปเตอร์ 4 ใบพัด ใช้ทั้งหมด 4 channel คือ Throttle, Low, Pitch, Yaw



รูปที่ 2.10 รูปแบบสัญญาณ PPM

2.6 แบตเตอรี่ลิเทียมพอลิเมอร์ (Lithium Polymer Battery)

- ทำจากธาตุลิเทียม (li น้ำหนักต่อ 1 ลบ.ซม.) จึงมีน้ำหนักเบา (170to200wh/kg) เพียงหนึ่งในสามของถ่านนิแคด หรือ เมทัลไฮไดร์แต่มีพลังไฟหนาแน่นสูง เมื่อเทียบกับน้ำหนัก (310-350wh/l) ถ่านเป็นทรงสี่เหลี่ยมแบน ประมาณ0.3-0.5 ซ.ม. มีเปลือกเป็นถุงอลูมิเนียมเบา ไม่มีปัญหาเรื่องความจำเหมือนถ่านนิแคด และยังเป็นมิตรกับสิ่งแวดล้อม
- ห้ามใช้งานให้โวลต์ตกต่ำกว่า 3 โวลต์ หรือ 2.5 โวลต์ เมื่อใช้งานต่อกับอุปกรณ์ไฟฟ้าหรือโหลด (มอเตอร์ไฟฟ้า หลอดไฟ อุปกรณ์ไฟฟ้า ฯลฯ) ถ้าเป็นถ่านที่ต่อกันอยู่แบบอนุกรมก็ให้นำ 3 โวลต์ คูณด้วยจำนวนก้อน ก็จะได้ค่าโวลต์ต่ำสุดที่ต้องหยุดใช้งาน และนำกลับไปใช้ใหม่
- อัตราการดึงกระแสของโหลด (Discharge) กับถ่านชนิดนี้ อย่างเต็มประสิทธิภาพคือปลอดภัย โดยทั่วไปอยู่ที่ 2c (2เท่าของความจุของถ่าน เช่นสามารถนำมอเตอร์ที่กินกระแส 1600มิลลิแอมป์ต่อกับถ่าน 800มิลลิแอมป์ ได้อย่างปลอดภัยและมอเตอร์ยังทำงานได้ดี)

รูปที่ 2.11 Lithium Polymer Battery

2.7 ตัวควบคุมความเร็วของมอเตอร์ (Speed Control)

รูปที่ 2.12 speed control หรือ ESC

สปีดคอนโทรล หรือ ESC (Electronic Speed Control) ทำหน้าที่ ควบคุมความเร็วมอเตอร์ ให้หมุนช้า-เร็วเดินหน้า-ถอยหลัง หรือเบรค ตามสัญญาณควบคุมที่ได้รับจาก กล่องภาครับ (Receiver)

สปีด สำหรับรถไฟฟ้า มีหลายชนิด ถ้าแบ่งโดยใช้มอเตอร์เป็นหลัก จะแบ่งได้ 2 ประเภทใหญ่ๆ คือ สปีดสำหรับมอเตอร์ใช้แปรงถ่าน (Brushed Motor) และ สปีดสำหรับมอเตอร์ไร้แปรงถ่าน (Brushless Motor)

2.8 ชุดตัวรับสัญญาณ (Receiver)

มีขนาด 64 mm x 48 mm x 21mm เป็น ปลั๊กแบบสามหมุด แบบดิจิตอลมีตัวรับสัญญาณแบบดิจิตอล 9 ช่อง ใช้รับสัญญาณความถี่ 2.4 GHz

รูปที่ 2.13 ชุดตัวรับสัญญาณ (Receiver)

บทที่ 3
การออกแบบเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

3.1 คุณสมบัติของเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

3.1.1 การบินขึ้น – ลง

3.1.2 การบินเลี้ยวซ้าย – ขวา

3.1.3 การบินเดินหน้า – ถอยหลัง

3.2 หลักการควบคุมเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

การควบคุมเฮลิคอปเตอร์ 4 ใบพัด ทำได้โดยการเปลี่ยนแปลงความเร็วของใบพัดซึ่งทำให้แรงบิดและแรงยกของแต่ละใบพัดเปลี่ยนไปโดยมีสัญลักษณ์และความหมายดังต่อไปนี้

Ω หมายถึง ความเร็วขณะนั้น

Δ หมายถึง ความเร็วที่เปลี่ยนไป

$\ddot{Z}$ หมายถึง การเร่งความเร็วในแนวดิ่ง (Throttle)

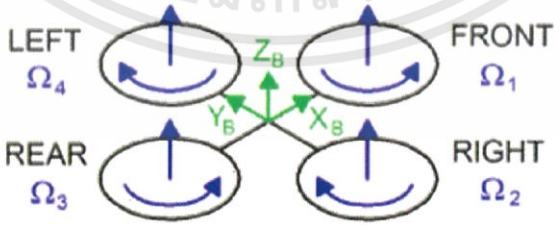
$\ddot{\phi}$ หมายถึง การเอียงตัวไปทางขวา (Roll)

$\ddot{\vartheta}$ หมายถึง การเอียงตัวไปข้างหน้า (Pitch)

$\ddot{\psi}$ หมายถึง การหมุนทวนเข็มนาฬิกา (Yaw)

3.2.1 หลักการควบคุมการบิน

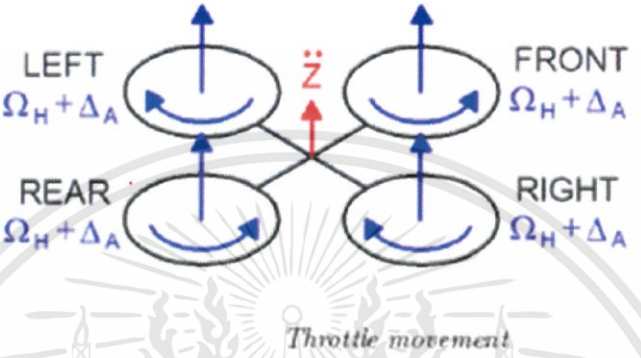
การลอยตัวนิ่ง (Hovering) ทำได้โดยควบคุมความเร็วใบพัดทั้งสี่ตัวให้มีความเร็วเท่ากันเพื่อสร้างแรงบิด (Torque) จากรูปที่ 3.1 ใบพัดแต่ละคู่จะหมุนในทิศตรงข้ามกันโดยใบพัดหน้าและหลังจะหมุนทวนเข็มนาฬิกาใบพัดซ้ายและขวาจะหมุนตามเข็มนาฬิกา แรงบิดจากใบพัดแต่ละคู่จะหักล้างกันทำให้เครื่องบินไม่เกิดการหมุนตัว



Simplified quadrotor motor in hovering

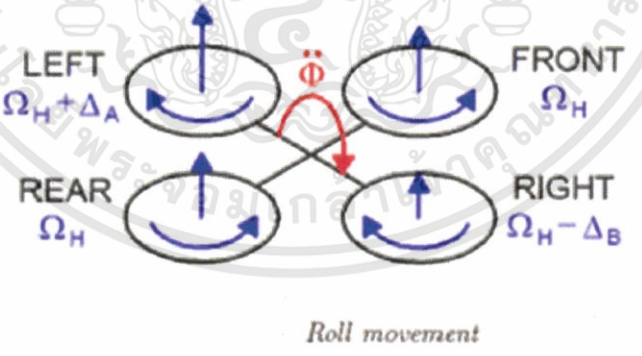
รูปที่ 3.1 หลักการลอยตัวนิ่ง (Hovering)

การเร่ง-ลดความเร็วในแนวดิ่ง(Throlle) ใบพัดทั้งสี่จะต้องเพิ่มหรือลดความเร็วทุกใบพัดเท่าๆกัน ทำให้เครื่องบิน บินขึ้น-ลง จากรูปที 3.2 เป็นการเร่งความเร็วในแนวดิ่งโดยเพิ่มความเร็วทั้งสี่ใบพัดเท่ากันทำให้เครื่องบินสามารถลอยขึ้นได้



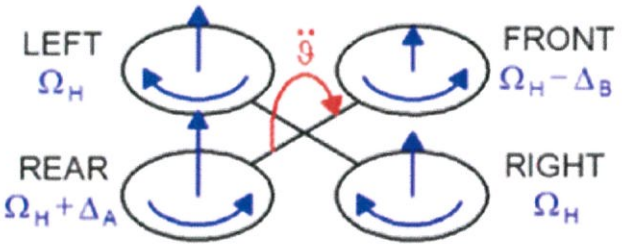
รูปที่ 3.2 หลักการเร่งความเร็วในแนวดิ่ง (Throttle)

การเอียงตัวจากซ้าย–ขวา (Roll) จากรูปที่ 3.3 ใบพัดหน้า (Front) และหลัง (Rear) จะมีความเร็วเท่าเดิมแต่ความเร็วใบพัดซ้าย (Left) จะหมุนเร็วขึ้นในทิศทางนี้จะยกตัวขึ้น ส่วนใบพัดขวา (Right) จะช้าลงทำให้ทิศทางนี้ตกลงทำให้เกิดการเอียงตัวไปทางขวา และการเอียงตัวไปทางซ้ายก็ใช้วิธีตรงกันข้าม



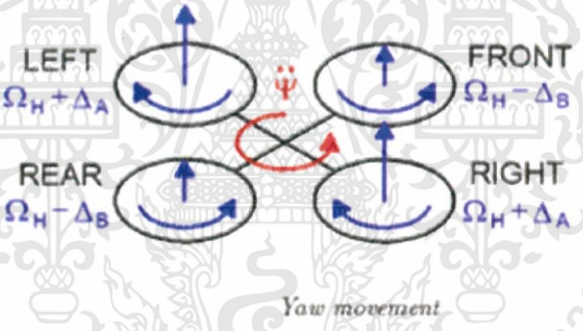
รูปที่ 3.3 หลักการเอียงตัวไปทางขวา (Roll)

การเอียงตัวหน้า-หลัง (Pitch) วิธีนี้คล้ายกับการเอียงซ้ายและขวา เพียงแต่มีการเปลี่ยนค่าความเร็วของใบพัดซ้ายและขวาให้มีความเร็วคงที่ และเพิ่มความเร็วของใบพัดหลังมากขึ้น ทำให้ยกตัวใบพัดหน้าจะหมุนช้ากว่าทำให้ทิศทางนี้ตกลง เครื่องบินจะเอียงไปข้างหน้าดังรูปที่ 3.4 ส่วนการเอียงตัวด้านหลังก็ใช้หลักการตรงกันข้าม



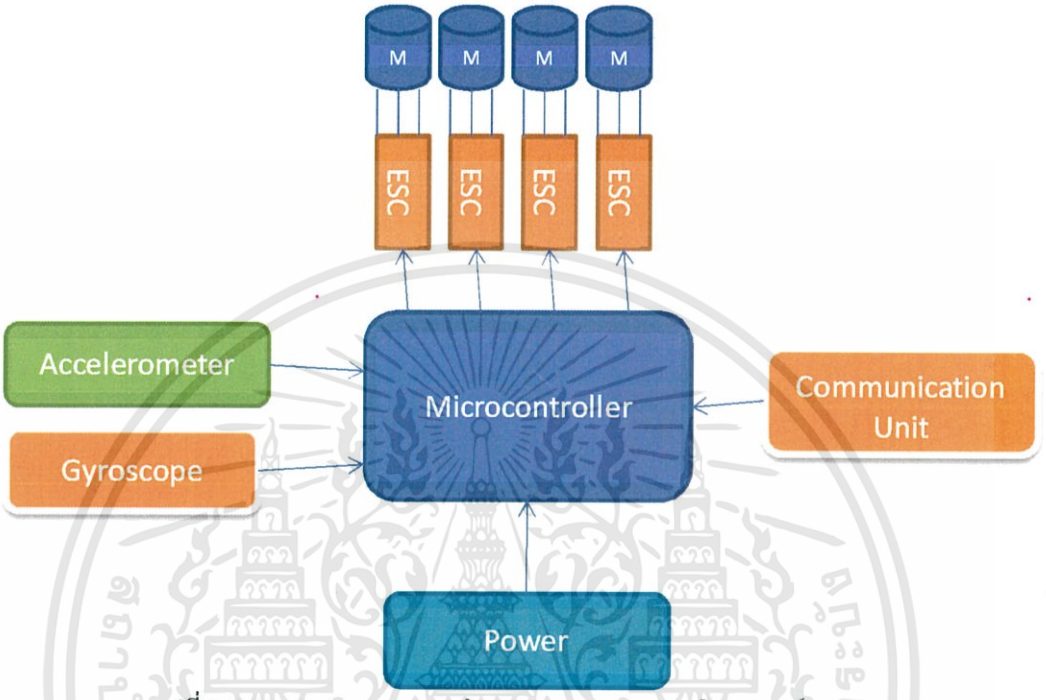
รูปที่ 3.4 หลักการเอียงตัวข้างหน้า-หลัง (Pitch)

การหมุนตัวทวน-ตามเข็มนาฬิกา (Yaw) ให้ความเร็วใบพัดหน้า-หลังมากกว่าความเร็วใบพัดซ้าย-ขวา เพื่อให้แรงบิดที่เกิดจากการหมุนของใบพัดด้านนี้มากกว่าทำให้เครื่องบินหมุนตัวทวนเข็มนาฬิกา ดังรูปที่ 3.5 ส่วนการหมุนตัวตามเข็มนาฬิกาก็มีหลักการตรงกันข้าม



รูปที่ 3.5 หลักการเอียงตัวทวน-ตามเข็มนาฬิกา (Yaw)

3.3 ระบบการทำงานและการควบคุม



รูปที่ 3.6 Block diagram การทำงานและควบคุมของเฮลิคอปเตอร์

3.4 การเลือกใช้มอเตอร์

3.4.1 การเลือกใช้มอเตอร์

เลือกใช้มอเตอร์ชนิด Brushless DC motor 800 KV เนื่องจากให้แรงบิดที่สูงและไม่มีแปรงถ่าน ทำให้มีอายุการใช้งานมากกว่า DC motor ธรรมดา

3.4.2 การคำนวณความเร็วรอบของมอเตอร์

ความเร็วรอบมอเตอร์ (รอบต่อนาที) = KV motor x Volt x efficiency

KV motor เป็นค่าที่ระบุมาสำหรับมอเตอร์แต่ละตัว มีหน่วยเป็น รอบต่อนาทีต่อโวลต์

Volt เป็นค่าแรงดันที่ป้อนให้กับมอเตอร์

Efficiency เป็นค่าความสามารถ ที่มอเตอร์จะใช้ งานจริงๆ โดยปรกติจะมีการสูญเสียไปในรูปแบบ พลังงานความร้อนด้วย มอเตอร์แบบไร้แปรงถ่านทั่วๆ ไป จะมีค่าประสิทธิภาพอยู่ที่ 80% หรือ 0.80

จะได้ ความเร็วรอบสูงสุด = KV motor x Volt x Efficiency

= 800 x (3.7 x 3) x 0.8

= 7104 รอบต่อนาที (RPM)

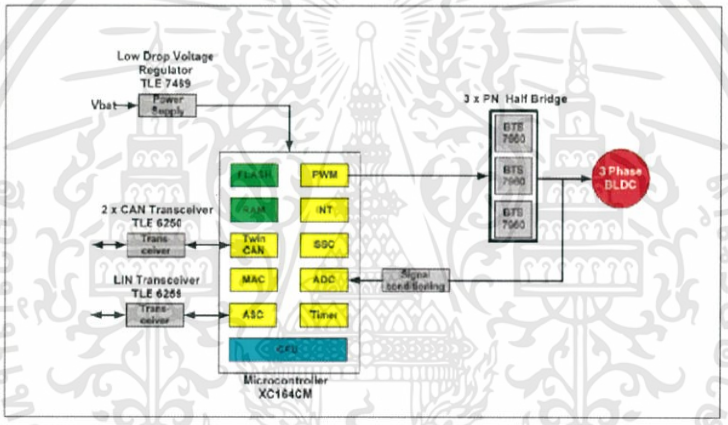
3.5 การใช้ตัวควบคุมความเร็วมอเตอร์แบบไร้การแปรงถ่าน (ESC)

3.5.1 การเลือกใช้ตัวควบคุมความเร็วมอเตอร์แบบไร้แปรงถ่าน

เฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด จะต้องใช้มอเตอร์ทั้งหมด 4 ตัว ดังนั้น จะต้องมีวงจรควบคุมความเร็วทั้งหมด 4 ชุด และทุกชุดจะต้องมีคุณสมบัติเหมือนกัน โดยเลือกใช้ชุดควบคุมความเร็วที่มีการจ่ายกระแสได้สูงสุด 30 A

3.5.2 การป้อนสัญญาณ PWM ให้กับ ESC

การป้อนสัญญาณ PWM ให้กับ ESC เพื่อควบคุมการหมุนของมอเตอร์ โดยความเร็วรอบของมอเตอร์นั้นจะขึ้นอยู่กับค่าของความกว้างของ Pulse Width แต่การจะให้มอเตอร์ทำงานนั้น ก่อนอื่นจะต้องป้อนสัญญาณที่มีค่า Pulse Width ขนาด 1000 us เพื่อหาตำแหน่งของโรเตอร์



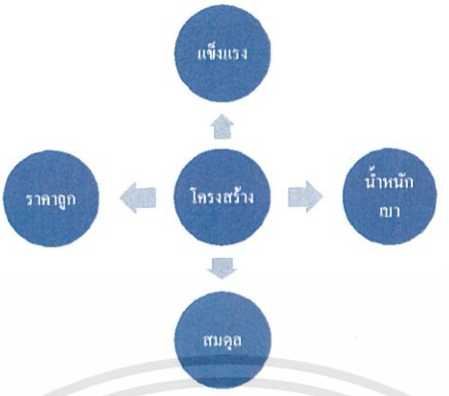
รูป 3.7 Block diagram of block commutation with B-EMF detection

3.6 การออกแบบโครงสร้าง

3.6.1 ปัจจัยสำคัญของโครงสร้างทางกลของเฮลิคอปเตอร์

การออกแบบโครงสร้าง flame ของเฮลิคอปเตอร์ที่ดีและให้มีประสิทธิภาพในการใช้งานนั้น ต้องคำนึงถึงคุณสมบัติ ดังนี้

1) แข็งแรง
2) น้ำหนักเบา
3) สมดุล
4) ราคาถูก



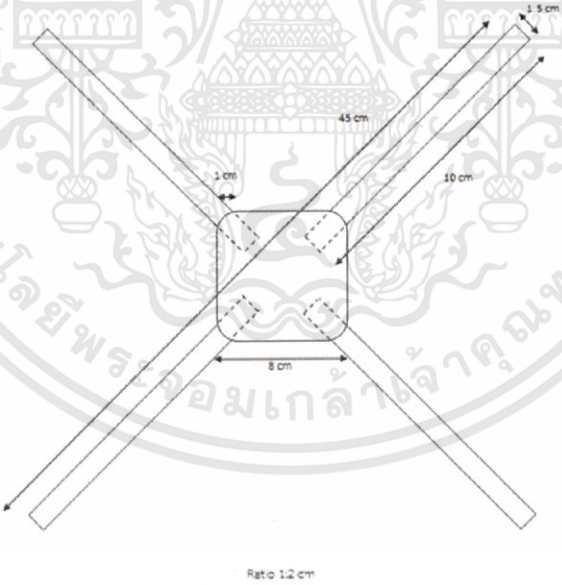
รูปที่ 3.8 คุณสมบัติของเฟรมเฮลิคอปเตอร์

3.6.2 การออกแบบ flame ขนาด 450 มิลลิเมตร

1) เลือกใช้โลหะอลูมิเนียมเป็นแขนของโครงสร้าง เนื่องจากมีความแข็งแรง เหนียว แต่มีน้ำหนักที่เบา

2) เลือกใช้อะคริลิค สำหรับเป็นฐานกลางของโครงสร้างตัวเฮลิคอปเตอร์

3.6.3 แบบที่ใช้ในการทำเฟรมของเฮลิคอปเตอร์

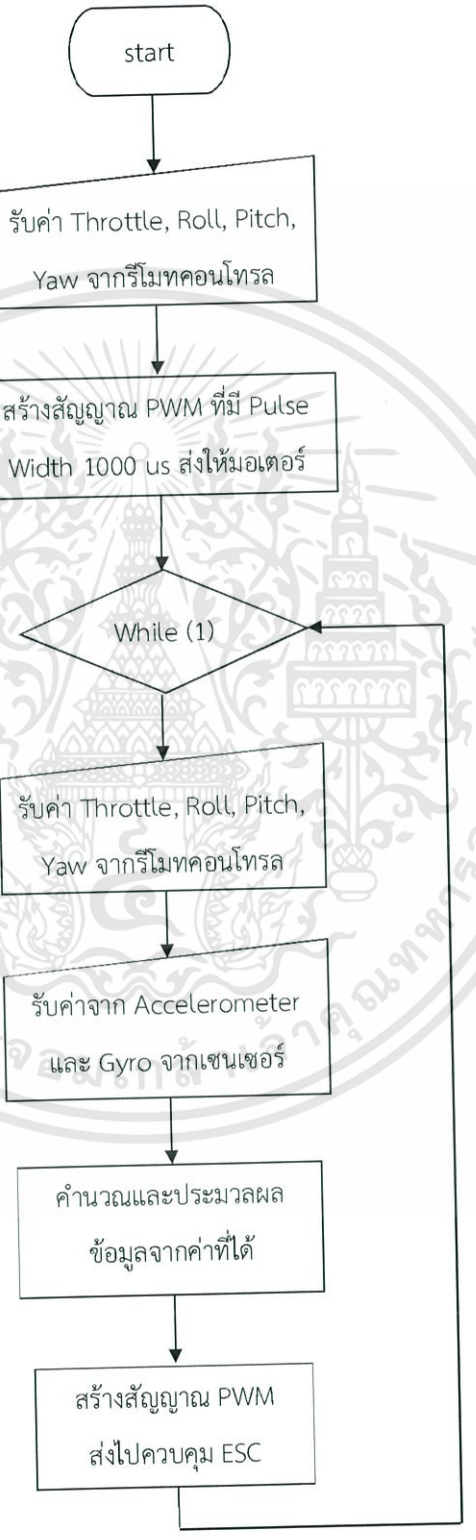


รูปที่ 3.9 แบบที่ใช้ในการทำเฟรมของเฮลิคอปเตอร์

3.6.4 ตัวเฮลิคอปเตอร์ที่สำเร็จแล้ว

รูปที่ 3.10 เฮลิคอปเตอร์ 4 ใบพัดแบบ X

3.7 แผนภาพการทำงานของโปรแกรม



บทที่ 4
ผลการทดลอง

4.1 การทดลองการเปลี่ยนแปลงค่าความกว้างพัลส์ (Pulse Width)

4.1.1 วิธีการทดลอง

4.1.1.1 ต่อมอเตอร์เข้ากับชุดควบคุม ESC และต่อเข้ากับบอร์ด Arduino

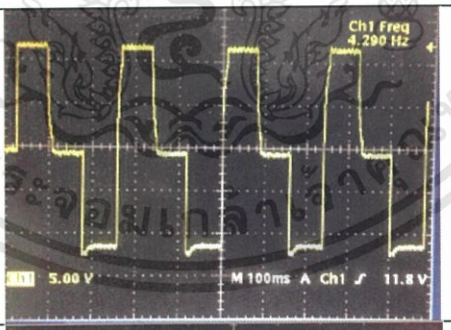
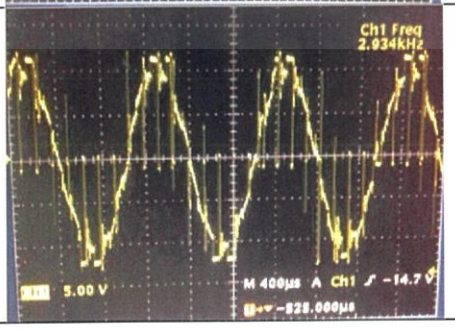
4.1.1.2 เขียนโปรแกรมควบคุมด้วยขนาดของ Pulse Width โดยใช้ค่าความถี่ 50 Hz

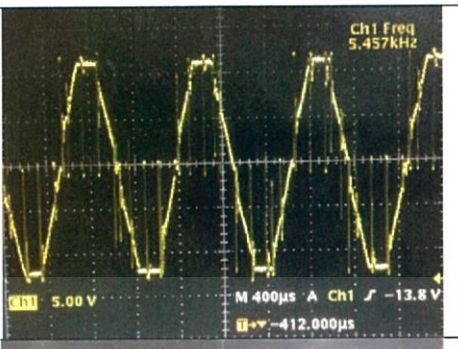
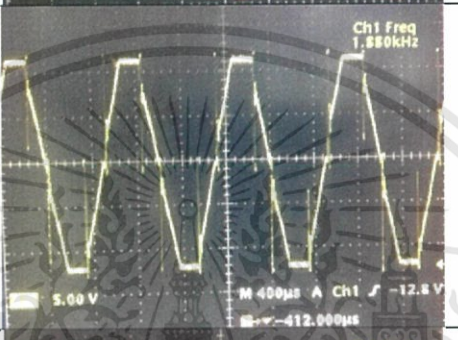
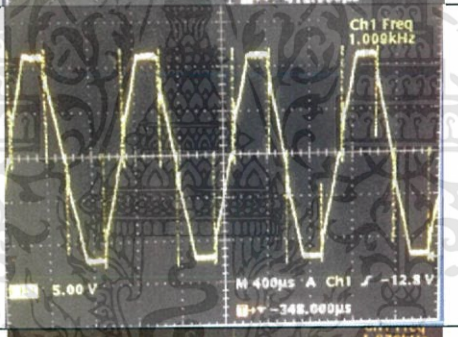
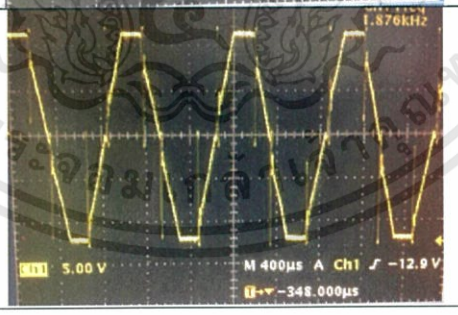
4.1.1.3 วัดรูปสัญญาณและความถี่จากเอาท์พุทของตัวควบคุมความเร็วมอเตอร์ที่ความกว้างพัลส์ 1200us, 1400us, 1600us, 1800us, 2000 us สังเกตและบันทึกผลการทดลองในตารางที่ 4.1

4.1.1.4 ทำการวัดความเร็วรอบของมอเตอร์โดยการเปลี่ยนค่า Pulse Width ที่มีขนาด 1200us, 1400us, 1600us, 1800us โดยใช้เครื่องวัดความเร็วรอบสังเกตและบันทึกผลการทดลองในตารางที่ 4.2

4.1.2 ผลการทดลอง

ตารางที่ 4.1 แสดงกราฟและความถี่จากเอาท์พุทของตัวควบคุมความเร็วมอเตอร์

ค่าความกว้าง Pulse	สัญญาณจากชุดควบคุม	Frequency(Hz)
ขณะที่ยังไม่ใส่มอเตอร์		4.17
1200 us		862.07

1400 us		925.93
1600 us		961.54
1800 us		1000.00
2000 us		1000.00

ตารางที่ 4.2 แสดงความเร็วรอบของมอเตอร์แต่ละตัวเมื่อทำการเปลี่ยนค่าความกว้างของสัญญาณ

ความกว้างพัลส์ (µs)	ความเร็วรอบของมอเตอร์ (rpm)			
	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
1200	4678	4406	2657	2627
1400	7819	7613	7305	7402
1600	8452	8062	8340	8269
1800	8426	8236	8622	8199

4.2 การทดลองเรื่องการอ่านค่าของตัวนับเวลา

4.2.1 วิธีการทดลอง

4.2.1.1 เขียนโปรแกรมอ่านค่า Pulse Width ทั้ง 4 channel จากรีโมทคอนโทรลด้วย Timerและอ่านค่าจากตัวรับสัญญาณโดยแสดงผลผ่านทางมิเตอร์ในโปรแกรม

4.2.1.2 ทำการเลือก channel ต่างๆ สังเกตและบันทึกผลลงในตารางที่ 4.3

4.2.2 ผลการทดลอง

ตารางที่ 4.3 แสดงผลการทดลองจากการอ่านค่าความกว้างพัลส์ของสัญญาณจากรีซีพเวอร์ที่ช่องสัญญาณต่างๆ

Channel	เริ่มต้น		โยกสวิตช์ทางซ้าย/ขึ้น		โยกสวิตช์ทางขวา/ลง	
	ค่าที่ควรเป็น (µs)	ค่าที่อ่านได้ (µs)	ค่าที่ควรเป็น (µs)	ค่าที่อ่านได้ (µs)	ค่าที่ควรเป็น (µs)	ค่าที่อ่านได้ (µs)
Channel 1 (Aileron)	1500	1520	1000	ซ้าย 1128	2000	ขวา 1952
Channel 2 (Elevator)	1500	1528	1000	ขึ้น 1104	2000	ลง 1944
Channel 3 (Throttle)	1500	-	1000	ขึ้น 1108	2000	ลง 1932
Channel 4 (Rudder)	1500	1544	1000	ซ้าย 1112	2000	ขวา 1940

4.3 การอ่านค่าจากตัววัดความเร่ง

4.3.1 วิธีการทดลอง

4.3.1.1 ป้อนโปรแกรมการอ่านค่าของตัววัดความเร่ง โดยให้แสดงผลผ่านทางมิเตอร์ในตัวโปรแกรม

4.3.1.2 ทำการพลิกตัวบอร์ดตามมุมต่างๆ โดยทำการเทียบมุมกับไม้ครึ่งวงกลม

4.3.1.3 สังเกตและบันทึกผลลงในตารางที่ 4.4

4.3.2 ผลการทดลอง

ตารางที่ 4.4 ค่าที่อ่านได้จากเซนเซอร์ที่มุมต่างๆ ในแกน x และแกน y

sensor	ค่าที่อ่านได้จาก sensor ที่มุมต่างๆ								
	0°	$+30^{\circ}$	$+45^{\circ}$	$+60^{\circ}$	$+90^{\circ}$	-30°	-45°	-60°	-90°
X-Axis	8	-8012	-11284	-14088	-16200	8296	11620	14260	16656
Y-Axis	160	-7068	-11500	-13800	-16076	8624	11900	14300	16664

4.4 การทดสอบการบินและการทรงตัวของเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด

4.4.1 วิธีการทดลอง

4.4.1.1 ทำการป้อนโปรแกรมที่สมบูรณ์แล้วลงในบอร์ด

4.4.1.2 นำเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด วางไว้บนแท่นทดสอบการทรงตัว

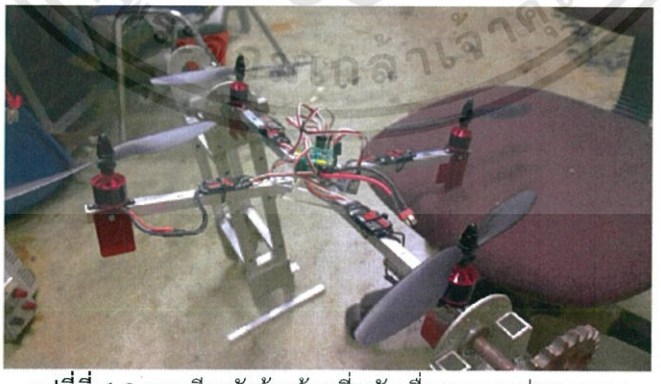
4.4.1.3 ทดสอบการบินและการทรงตัวของเฮลิคอปเตอร์

4.4.1.4 สังเกตและบันทึกผลการทดลอง

4.4.2 ผลการทดลอง

จากการทดลองที่ 4.4 ผลปรากฏว่า เริ่มแรกของการทดสอบในขณะที่ใบพัดยังไม่มีการหมุนของมอเตอร์ เฮลิคอปเตอร์นั้นมีการเอียงเนื่องด้วยน้ำหนักที่ไม่สมดุล เมื่อมีการเพิ่มขนาดความเร็วของมอเตอร์แล้วตัวเฮลิคอปเตอร์มีการปรับสมดุลโดยยกด้านที่เอียงลงขึ้นมา แต่มีการเปลี่ยนแปลงที่รวดเร็วมาก ส่งผลให้เกิดการพลิกเอียงไปอีกด้านหนึ่ง และเมื่ออีกฝั่งเอียง ก็จะยกด้านนั้นหลับมาอีกเช่นกัน

รูปที่ 4.1 แท่นทดสอบการทรงตัว



รูปที่ที่ 4.2 การเอียงตัวข้างด้านที่หนักเมื่อวางบนแท่นทดสอบ

4.5 การทดสอบการบินและการทรงตัวของเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด โดยใช้ไจโรเซ็นเซอร์ร่วมกับเซนเซอร์วัดความเร่ง

4.5.1 วิธีการทดลอง

4.5.1.1 ทำการป้อนโปรแกรมที่สมบูรณ์แล้วลงในบอร์ด

4.5.1.2 นำเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด วางไว้บนแท่นทดสอบการทรงตัว

4.5.1.3 ทดสอบการบินและการทรงตัวของเฮลิคอปเตอร์

4.5.1.4 สังเกตและบันทึกผลการทดลอง

4.5.1.5 ป้อนแรงให้กับปลายเฮลิคอปเตอร์ด้านใดด้านหนึ่ง สังเกตและบันทึกผลการทดลอง

4.5.2 ผลการทดลอง

จากการทดลองที่ 4.5 โดยทำการวัดบนแท่นทดสอบ โดยเริ่มแรกปล่อยให้เฮลิคอปเตอร์เอียงไปด้านใดด้านหนึ่ง ผลปรากฏว่าเมื่อเริ่มเพิ่มความเร็วรอบของมอเตอร์ ตัวเฮลิคอปเตอร์เริ่มมีการปรับสมดุลและเบนกลับมาสู่จุดสมดุล และจากการทดลองการป้อนแรงให้กับเฮลิคอปเตอร์ด้านใดด้านหนึ่ง ผลปรากฏว่าตัวเฮลิคอปเตอร์จะพยายามปรับสมดุลโดยการเบนกลับเข้าที่ตำแหน่งที่สมดุล

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

โครงการนี้เป็นการทดลองสร้าง เฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด เพื่อที่จะทำการศึกษาระบบโครงสร้าง และการทำงาน และเพื่อนำมาพัฒนาความสามารถในการบินของเฮลิคอปเตอร์บังคับวิทยุ 4 ใบพัด ในปัจจุบันให้ดีขึ้น โดยศึกษาระบบการควบคุมสัญญาณวิทยุ ซึ่งจะใช้งานร่วมกับส่วนของ ไจโร ตัววัดความเร่ง และไมโครคอนโทรลเลอร์ โดยตัวรับจะทำหน้าที่รับสัญญาณควบคุมจากรีโมทคอนโทรลเลอร์ เพื่อที่จะส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ เพื่อที่จะทำการคำนวณและประมวลผล ร่วมกับค่าที่รับมาจากไจโรเซนเซอร์ที่ทำหน้าที่คำนวณมุม และเซนเซอร์วัดความเร่ง โดยหลักการทำงานคือการกำหนดความกว้างของพัลส์ ที่จะส่งไปยังชุดควบคุมความเร็ว(ESC) เพื่อที่จะไปควบคุมความเร็วของตัวมอเตอร์แต่ละตัว และในการทำงานร่วมกันของอุปกรณ์ทั้งหมดนี้สามารถควบคุมการเคลื่อนที่ของเฮลิคอปเตอร์ได้

5.1 สรุปผลการวิจัย

จากการศึกษาการทำงานของตัวส่งสัญญาณวิทยุจากรีโมทวิทยุทำให้ทราบว่าเป็นสัญญาณพัลส์ โดยมีความกว้างพัลส์เปลี่ยนแปลงไปตามโยกสวิตช์ปุ่มของรีโมทวิทยุซึ่งมีขนาดความกว้างพัลส์อยู่ระหว่าง 1000ไมโครวินาที ถึง2000 ไมโครวินาที เมื่อตัวรับสัญญาณได้รับสัญญาณจากรีโมทบังคับแล้วจะทำการส่งสัญญาณไปยังตัวบอร์ดควบคุมโดยสัญญาณจะถูกแบ่งออกเป็น 4ช่องสัญญาณ ได้แก่ ช่องสัญญาณที่1 คือการเอียงตัวซ้าย-ขวา, ช่องสัญญาณที่2 คือควบคุมการเดินหน้า-ถอยหลัง, ช่องสัญญาณที่3 คือควบคุมการลอยตัวในแนวดิ่งขึ้น-ลงโดยตัวบอร์ดควบคุมจะทำการสร้างสัญญาณพัลส์ให้กับตัวควบคุมความเร็วมอเตอร์แต่ละตัว ในการออกแบบเฮลิคอปเตอร์บังคับวิทยุสี่ใบพัดเพื่อที่จะทำให้บินได้สมดุลนั้น จำเป็นต้องใช้การทำงานของเซนเซอร์ที่ช่วยในการปรับสมดุลและมุมเอียง นั้นคือไจโรเซนเซอร์และตัววัดความเร่ง ในส่วนการใช้งานของไจโรเซนเซอร์จะทำหน้าที่วัดความเร็วเชิงมุมเมื่อเฮลิคอปเตอร์มีการเอียง ถ้าความเร็วในการเอียงมากค่าที่อ่านได้จากเซนเซอร์จะมีค่ามาก ส่วนตัววัดความเร่งจะทำหน้าที่บอกตำแหน่งของมุมเอียงได้ โดยเมื่อค่าทั้งสองถูกส่งเข้าไปที่บอร์ดควบคุมโดยตัวบอร์ดจะนำค่าที่ได้ไปคำนวณร่วมกับสัญญาณจากรีโมทคอนโทรล และนำไปปรับความกว้างพัลส์ที่สร้างให้ตัวควบคุมความเร็วมอเตอร์ทั้งสี่ตัว เพื่อให้เฮลิคอปเตอร์ทำการปรับเข้าสู่สมดุล

5.2 ปัญหาและการแก้ไขปัญหา

ในการสร้างเฮลิคอปเตอร์บังคับวิทยุสี่ใบพัด สิ่งที่สำคัญประการหนึ่งคือเรื่องของน้ำหนักในการออกแบบตัวโครงสร้างซึ่งมีผลต่อสมดุล ดังนั้นการตัดอลูมิเนียมและการวางอุปกรณ์ที่ตำแหน่งต่างๆ นั้นมีความสำคัญอย่างยิ่ง จากการปฏิบัติจริงพบว่าเกิดการไม่สมดุล มีการเอียงเล็กน้อยจากนั้นได้ทดสอบการบิน ผลที่ได้คือเฮลิคอปเตอร์เอียงทำให้ไม่สามารถขึ้นได้จึงต้องทำแท่นสำหรับการทดสอบการบินขึ้นมา เพื่อทดสอบความสมดุลเพื่อลดความเสียหายที่อาจเกิดขึ้นกับเฮลิคอปเตอร์ จากการทดลองบินจริง

นอกจากนี้ยังพบปัญหาในด้านการควบคุมสมดุล จากการทดลองโดยใช้เพียงแค่ตัววัดความเร่งเพียงอย่างเดียว ผลปรากฏว่าไม่สามารถทำให้เฮลิคอปเตอร์บินได้อย่างสมดุล จึงแก้ไขโดยการใช้ไจโรเซนเซอร์ร่วมทำงานกับตัววัดความเร่ง ทำให้เฮลิคอปเตอร์มีการทรงตัวที่ดีขึ้นและมีการปรับตัวให้เข้าสมดุลได้รวดเร็วยิ่งขึ้น แม้มีแรงภายนอกมากระทำ

5.3 ข้อเสนอแนะ

การสร้างเฮลิคอปเตอร์นี้ต้องคำนึงถึงเรื่องน้ำหนักของเฮลิคอปเตอร์และความสมดุล ซึ่งในเรื่องความสมดุลนั้นหากต้องการให้มีความคลาดเคลื่อนน้อยที่สุดควรคำนวณขนาดความกว้าง ความยาว และนำมาเขียนแบบให้เรียบร้อยก่อนจะทำการสร้างจริง ส่วนในเรื่องของน้ำหนักก็สำคัญเช่นเดียวกัน เนื่องจากวัสดุที่จะนำมาสร้างโครงของตัวเฮลิคอปเตอร์นั้นจะต้องเลือกที่มีน้ำหนักเบา แต่มีความแข็งแรงและทนทาน

เอกสารอ้างอิง

[1] HobbyKing MultiWii SE V2.0 Flight Controller.
http://www.rcgroups.com/forums/showthread.php?t=1723857

[2] Arduino Your Home & Environment
http://arduinotronics.blogspot.com/2012/11/arduino-ir-receiver-part-2.html

[3] Brushless DC Motor.
http://www.infineon.com/dgdl/p1609010_Brushless_DC_Motor.pdf?folderId=db3a304412b407950112b409d4b00386&fileId=db3a304412b407950112b409f53703f0

[4] ชนิดของมอเตอร์.
http://www.t4rbm.ac.th/~electronichunsa/motor%20%E0%B8%8A%E0%B8%99%E0%B8%B4%E0%B8%94%E0%B8%95%E0%B9%88%E0%B8%B2%E0%B8%87%E0%B9%86.html

[5] Sensor Gyro
http://bme231metrology.blogspot.com/2011/07/sensor-gyro-gyroscope.html

ภาคผนวก

โปรแกรมควบคุม

```
#include <PinChangeInt.h>
#include <PinChangeIntConfig.h>
#include <Servo.h>

unsigned int t0, i=0, ch[4]={0,0,0,0}, m1=0, m2=0, m3=0, m4=0, Roll=0, Pitch=0;

Servo a,b,c,d;

#include <Wire.h>
#include "Kalman.h" // Source: https://github.com/TKJElectronics/KalmanFilter

Kalman kalmanX; // Create the Kalman instances
Kalman kalmanY;

/* IMU Data */
int16_t accX, accY, accZ;
int16_t tempRaw;
int16_t gyroX, gyroY, gyroZ;

double accXangle, accYangle; // Angle calculate using the accelerometer
double temp; // Temperature
double gyroXangle, gyroYangle; // Angle calculate using the gyro
double compAngleX, compAngleY; // Calculate the angle using a complementary
filter
double kalAngleX, kalAngleY; // Calculate the angle using a Kalman filter

uint32_t timer;
```

```
uint8_t i2cData[14]; // Buffer for I2C data

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  TWBR = ((F_CPU / 400000L) - 16) / 2; // Set I2C frequency to 400kHz

  i2cData[0] = 7; // Set the sample rate to 1000Hz - 8kHz/(7+1) = 1000Hz
  i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc filtering, 256 Hz Gyro
filtering, 8 KHz sampling
  i2cData[2] = 0x00; // Set Gyro Full Scale Range to ±250deg/s
  i2cData[3] = 0x00; // Set Accelerometer Full Scale Range to ±2g
  while (i2cWrite(0x19, i2cData, 4, false)); // Write to all four registers at once
  while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis gyroscope reference and disable
sleep mode

  while (i2cRead(0x75, i2cData, 1));
  if (i2cData[0] != 0x68) { // Read "WHO_AM_I" register
    Serial.print(F("Error reading sensor"));
    while (1);
  }

  delay(100); // Wait for sensor to stabilize

  /* Set kalman and gyro starting angle */
  while (i2cRead(0x3B, i2cData, 6));
  accX = ((i2cData[0] << 8) | i2cData[1]);
  accY = ((i2cData[2] << 8) | i2cData[3]);
```

```
  accZ = ((i2cData[4] << 8) | i2cData[5]);
  // atan2 outputs the value of -ΠΠ to ΠΠ (radians) - see
http://en.wikipedia.org/wiki/Atan2
  // We then convert it to 0 to 2ΠΠ and then from radians to degrees
  accYangle = (atan2(accX, accZ) + PI) * RAD_TO_DEG;
  accXangle = (atan2(accY, accZ) + PI) * RAD_TO_DEG;

  kalmanX.setAngle(accXangle); // Set starting angle
  kalmanY.setAngle(accYangle);
  gyroXangle = accXangle;
  gyroYangle = accYangle;
  compAngleX = accXangle;
  compAngleY = accYangle;

  pinMode(2,INPUT);
  pinMode(3,INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  attachInterrupt(0,ch1,CHANGE);
  attachInterrupt(1,ch2,FALLING);
  PCintPort::attachInterrupt(4, ch3, FALLING);
  PCintPort::attachInterrupt(5, ch4, FALLING);
  a.attach(6);
     b.attach(9);
     c.attach(10);
     d.attach(11);
     a.writeMicroseconds(1000);
     b.writeMicroseconds(1000);
```

```
    c.writeMicroseconds(1000);
    d.writeMicroseconds(1000);
    delay(100);
}

void loop()
{
  while (i2cRead(0x3B, i2cData, 14));
  accX = ((i2cData[0] << 8) | i2cData[1]);
  accY = ((i2cData[2] << 8) | i2cData[3]);
  accZ = ((i2cData[4] << 8) | i2cData[5]);
  tempRaw = ((i2cData[6] << 8) | i2cData[7]);
  gyroX = ((i2cData[8] << 8) | i2cData[9]);
  gyroY = ((i2cData[10] << 8) | i2cData[11]);
  gyroZ = ((i2cData[12] << 8) | i2cData[13]);

  // atan2 outputs the value of -π to π (radians) - see
http://en.wikipedia.org/wiki/Atan2
  // We then convert it to 0 to 2π and then from radians to degrees
  accXangle = (atan2(accY, accZ) + PI) * RAD_TO_DEG;
  accYangle = (atan2(accX, accZ) + PI) * RAD_TO_DEG;

  double gyroXrate = (double)gyroX / 131.0;
  double gyroYrate = -((double)gyroY / 131.0);
  double gyroZrate = (double)gyroZ / 131.0;
  gyroXangle += gyroXrate * ((double)(micros() - timer) / 1000000); // Calculate gyro
angle without any filter
  gyroYangle += gyroYrate * ((double)(micros() - timer) / 1000000);
```

```
  //gyroXangle += kalmanX.getRate()*((double)(micros()-timer)/1000000); // Calculate
gyro angle using the unbiased rate
  //gyroYangle += kalmanY.getRate()*((double)(micros()-timer)/1000000);

  compAngleX = (0.93 * (compAngleX + (gyroXrate * (double)(micros() - timer) /
1000000))) + (0.07 * accXangle); // Calculate the angle using a Complimentary filter
  compAngleY = (0.93 * (compAngleY + (gyroYrate * (double)(micros() - timer) /
1000000))) + (0.07 * accYangle);

  kalAngleX = kalmanX.getAngle(accXangle, gyroXrate, (double)(micros() - timer) /
1000000); // Calculate the angle using a Kalman filter
  kalAngleY = kalmanY.getAngle(accYangle, gyroYrate, (double)(micros() - timer) /
1000000);
  timer = micros();

  temp = ((double)tempRaw + 12412.0) / 340.0;

    RollPitch();
    m1 = ch[2] + (181-kalAngleX)*1.5 - (179-kalAngleY)*1.5 - (gyroXrate-1) + (gyroYrate-
0.8) - (gyroZrate+1.4) + Roll + Pitch;
    m2 = ch[2] + (181-kalAngleX)*1.5 + (179-kalAngleY)*1.5 - (gyroXrate-1) - (gyroYrate-
0.8) + (gyroZrate+1.4) - Roll + Pitch;
    m3 = ch[2] - (181-kalAngleX)*1.5 - (179-kalAngleY)*1.5 + (gyroXrate-1) + (gyroYrate-
0.8) + (gyroZrate+1.4) + Roll - Pitch -33;
    m4 = ch[2] - (181-kalAngleX)*1.5 + (179-kalAngleY)*1.5 + (gyroXrate-1) - (gyroYrate-
0.8) - (gyroZrate+1.4) - Roll - Pitch -68;
    updateMotors();
  }
```

```
void ch1()
{
  if(digitalRead(2)==HIGH)
  {
    t0 = micros();
  }
  else if(digitalRead(2)==LOW)
  {
    ch[0] = 3000 - (micros() - t0);
  }
}
void ch2()
{
  ch[1] = 3000 - (micros() - t0);
}
void ch3()
{
  ch[2] = 3000 - (micros() - t0);
}
void ch4()
{
  ch[3] = 3000 - (micros() - t0);
}

void updateMotors()
{
  a.writeMicroseconds(m1);
  b.writeMicroseconds(m2);
  c.writeMicroseconds(m3);
```

```
  d.writeMicroseconds(m4);
}
void RollPitch()
{
  if(ch[0] < 1250)
  {
    Roll = -20;
  }
  if(ch[0] > 1750)
  {
    Roll = 20;
  }
  if(ch[0] > 1250 && ch[0] < 1750)
  {
    Roll = 0;
  }
  if(ch[1] < 1250)
  {
    Roll = -20;
  }
  if(ch[1] > 1750)
  {
    Roll = 20;
  }
  if(ch[1] > 1250 && ch[1] < 1750)
  {
    Roll = 0;
  }
}
```

```
const uint8_t IMUAddress = 0x68; // AD0 is logic low on the PCB
const uint16_t I2C_TIMEOUT = 1000; // Used to check for errors in I2C
communication

uint8_t i2cWrite(uint8_t registerAddress, uint8_t data, bool sendStop) {
  return i2cWrite(registerAddress, &data, 1, sendStop); // Returns 0 on success
}

uint8_t i2cWrite(uint8_t registerAddress, uint8_t *data, uint8_t length, bool sendStop)
{
  Wire.beginTransmission(IMUAddress);
  Wire.write(registerAddress);
  Wire.write(data, length);
  uint8_t rcode = Wire.endTransmission(sendStop); // Returns 0 on success
  if (rcode) {
    Serial.print(F("i2cWrite failed: "));
    Serial.println(rcode);
  }
  return rcode;
}

uint8_t i2cRead(uint8_t registerAddress, uint8_t *data, uint8_t nbytes) {
  uint32_t timeOutTimer;
  Wire.beginTransmission(IMUAddress);
  Wire.write(registerAddress);
  uint8_t rcode = Wire.endTransmission(false); // Don't release the bus
  if (rcode) {
    Serial.print(F("i2cRead failed: "));
    Serial.println(rcode);
```

```
    return rcode;
  }
  Wire.requestFrom(IMUAddress, nbytes, (uint8_t)true); // Send a repeated start and
then release the bus after reading
  for (uint8_t i = 0; i < nbytes; i++) {
    if (Wire.available())
      data[i] = Wire.read();
    else {
      timeOutTimer = micros();
      while (((micros() - timeOutTimer) < I2C_TIMEOUT) && !Wire.available());
      if (Wire.available())
        data[i] = Wire.read();
      else {
        Serial.println(F("i2cRead timeout"));
        return 5; // This error value is not already taken by endTransmission
      }
    }
  }
  return 0; // Success
}
```