เครื่องบินถ่ายภาพอัตโนมัติที่สามารถขึ้นบินและลงจอดได้เองโดยอัตโนมัติ

AUTOMATIC PHOTOGRAPHY CAPTURING UNMANNED AERIAL VEHICLE

โดย

นายบุรินทร์ ทรัพย์ศิริ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

เครื่องบินถ่ายภาพอัตโนมัติที่สามารถขึ้นบินและลงจอดได้เองโดยอัตโนมัติ

AUTOMATIC PHOTOGRAPHY CAPTURING UNMANNED AERIAL VEHICLE

โดย

นายบุรินทร์ ทรัพย์ศิริ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องบินถ่ายภาพอัตโนมัติที่สามารถขึ้นบินและลงจอดได้เองโดยอัตโนมัติ
AUTOMATIC PHOTOGRAPHY CAPTURING UNMANNED AERIAL VEHICLE

โดย

นายบุรินทร์ ทรัพย์ศิริ 53010881

อาจารย์ที่ปรึกษา

รศ.ดร. ปราโมทย์ วาดเขียน

รศ.ดร. จีรสุดา โกษียาภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ผ่านการตรวจรูปเล่มแล้ว

(ปราโมทย์)

อาจารย์ที่ปรึกษา

15/05/57

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

15/มิ.ย./57.

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารนี้ทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2556
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง เครื่องบินถ่ายภาพอัตโนมัติที่สามารถขึ้นบินและลงจอดได้เองโดยอัตโนมัติ

AUTOMATIC PHOTOGRAPHY CAPTURING UNMANNED AERIAL VEHICLE

ผู้จัดทำ
1. นายบุรินทร์ ทรัพย์ศิริ 53010881

.....ปราโมทย์..... อาจารย์ที่ปรึกษา
(รองศาสตราจารย์ ดร. ปราโมทย์ วาดเขียน)

.....จีรสุดา..... อาจารย์ที่ปรึกษา
(รองศาสตราจารย์ ดร. จีรสุดา โกษียาภรณ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบคุณ รศ.ดร. ปราโมทย์ วาดเขียน และ รศ.ดร. จีรสุดา โกษียาภรณ์ ที่ช่วยให้คำปรึกษา แนะนำ และคอยเอาใจใส่ในการทำปริญญานิพนธ์ และขอขอบคุณเพื่อนๆ และ น้องๆ ร่วมห้องโปรเจค ที่ช่วยแนะนำ และ ตอบคำถามที่สงสัยของผู้จัดทำ

นายบุรินทร์ ทรัพย์ศิริ
ผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องบินถ่ายภาพอัตโนมัติที่สามารถขึ้นบินและลงจอดได้เองโดยอัตโนมัติ
UNMANNED AERIAL VEHICLE PHOTOGRAPHY

โดย นายบุรินทร์ ทรัพย์ศิริ 53010881

อาจารย์ที่ปรึกษา รศ.ดร. ปราโมทย์ วาดเขียน
รศ.ดร. จีรสุดา โกษียาภรณ์

บทคัดย่อ

ปริญญานิพนธ์นี้ออกแบบและสร้างระแบบเครื่องบินถ่ายภาพอัตโนมัติที่สามารถขึ้นบินและลงจอดได้เองโดยอัตโนมัติ เพื่อเป็นทางเลือกสำหรับผู้ที่ต้องการเครื่องมือในการถ่ายภาพทางอากาศอัตโนมัติ โดยการกำหนดจุดเป้าหมายและความสูงของตำแหน่งที่ต้องการจะถ่ายภาพ หลังจากนั้นให้เริ่มสั่งงานให้ระบบทำงาน เครื่องบินจะทำการขึ้นบินเองและเคลื่อนที่ไปยังเป้าหมายและระดับความสูงตามที่กำหนด และระบบจะทำการถ่ายภาพ ณ ตำแหน่งและความสูงที่กำหนด แล้วหลังจากนั้นเครื่องบินจะทำการบินกลับมายังจุดที่ทำการปล่อยและลงจอดเอง โดยที่ไม่จำเป็นต้องมีผู้ควบคุม

ABSTRACT

This thesis is to design and build a system of automatic photograph capturing unmanned aerial vehicle. It aims for an alternative choice for those who need to automatic photograph capturing. It can be accomplished by defining the target point and the target height. After the system is started, the vehicle will automatically take off to the defined height, and then move to the defined target point. After the photograph is captured, it moves back and landing to the base automatically.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

		หน้า
กิตติกรรมประกาศ		I
บทคัดย่อ		II
สารบัญ		III
สารบัญรูป		V
บทที่ 1	**บทนำ**	1
	1.1 ความเป็นมาและความสำคัญของปัญหา	1
	1.2 วัตถุประสงค์	1
	1.3 ขอบเขตของปริญญานิพนธ์	1
บทที่ 2	**ทฤษฎีและหลักการที่เกี่ยวข้อง**	
	2.1 หลักการของเครื่องบิน 4 ใบพัด	2
	2.2 สัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ (RC Control signal)	4
	2.3 วงจรกรองสัญญาณแบบคอมพลีเมนทารี่ (Complementary Filter)	5
	2.4 กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ (Naza m lite Controller)	6
	2.5 มุมแบริ่ง (Bearing)	7
	2.6 ระยะทางระหว่างจุดสองจุดบนโลก (Distance between 2 point)	8
	2.7 ระบบควบคุมพีไอดี (PID Controller)	9
บทที่ 3	**การออกแบบและการจัดทำปริญญานิพนธ์**	
	3.1 การออกแบบ	12
	3.2 เครื่องมือที่ใช้ในการทดลอง	30

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

		หน้า
บทที่ 4	ผลการทดลอง	31
	4.1 การทดลองการทำงานของระบบตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด	31
	4.2 การทดลองการทำงานของระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ	32
	4.3 การทดลองการทำงานของระบบสร้างสัญญาณเพิ่มควบคุมกล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์	36
	4.4 การทดลองการทำงานของระบบการกรองสัญญาณแบบคอมพลีเมนทารี่	46
	4.5 การทดลองการทำงานของระบบหาค่ามุมแบริ่ง	47
	4.6 การทดลองการทำงานของระบบหาค่าระยะทางระหว่างจุดสองจุดบนโลก	49
	4.7 การทดลองการทำงานของระบบการบินแบบอัตโนมัติ	50
	4.8 การทดลองการทำงานของระบบการบินแบบเต็มรูปแบบ	54
บทที่ 5	**สรุปผลและข้อเสนอแนะ**	56
	5.1 สรุปผล	56
	5.2 ข้อเสนอแนะ	57
บรรณานุกรม		58
ภาคผนวก ก	โค๊ดโปรแกรมที่ใช้ในการตรวจจับสัญญาลักษณ์ที่ใช้ในการลงจอดของเครื่องบินถ่ายภาพไรคนขับ	59
ภาคผนวก ข	โค๊ดโปรแกรมที่ใช้ในระบบการควบคุมการบินอัตโนมัติทั้งหมด	66

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่		หน้า
2.1	เครื่องบินสี่ใบพัด	3
2.2	ลักษณะการหมุนของแต่ละใบพัดของเครื่องบิน 4 ใบพัด	3
2.3	รูปที่ 2.3 สัญญาณความกว้างพัลส์ในมาตรฐานของสัญญาณควบคุมที่ในอุปกรณ์บังคับวิทยุ	4
2.4	บล็อกไดอะแกรมของวงจรกรองคอมพลีเมนทารี่	5
2.5	บล็อกไดอะแกรมของวงจรกรองแบบคอมพลีเมนทารี่	6
2.6	รายละเอียดการต่อสายสัญญาณของกล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์	6
2.7	โมดูลเข็มทิศและจีพีเอสของ นาซ่าเอ็มไลท์	7
2.8	ตัวอย่างมุมแบริ่งจาก จุด A ไปยังจุด B และจากจุด B ไปยังจุด A	7
2.9	บล็อกไดอะแกรมของระบบควบคุมพีไอดี	9
3.1	บล็อกไดอะแกรมการติดต่อของอุปกรณ์ในระบบเครื่องบินถ่ายภาพไร้คนขับ	11
3.2	แสดงการเชื่อมต่อระหว่างแต่ละระบบของโปรแกรม	13
3.3	โฟลว์ชาร์ตของโปรแกรมที่ใช้ในการตรวจจับสัญญาณที่ใช้ในระบบบังคับวิทยุที่ได้รับมาจากตัวรับสัญญาณของรีโมตโฟลว์ชาร์ตของโปรแกรมที่ใช้	14
3.4	โฟลว์ชาร์ตของโปรแกรมที่ใช้สร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ	16
3.5	การเชื่อมต่อเพื่อทำการทดลองส่งสัญญาณที่ใช้ในระบบบังคับวิทยุจากบอร์ดอาดูไพล็อต ไปยังกล่องควบคุมสมดุล นาซ่า เอม ไลท์	18
3.6	โฟลว์ชาร์ตของโปรแกรมที่ใช้ในการทดลองควบคุมกล่องควบคุมสมดุล นาซ่า เอม ไลท์	19
3.7	โฟลว์ชาร์ตของโปรแกรมการกรองแบบคอมพลีเมนทารี่	20
3.8	สถานะไดอะแกรมสถานะของระบบควบคุมการบินอัตโนมัติ	21
3.9	โฟลว์ชาร์ตของโปรแกรมจัดการการทำงานแบบ MULTITASK	22
3.10	โฟลว์ชาร์ตของระบบตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด	28
4.1	ภาพเอาท์พุตที่ผ่านการตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด	31
4.2	ค่าเอาท์พุตของตำแหน่งสัญลักษณ์ที่ใช้ในการลงจอด	32
4.3	สัญญาณอินพุตที่เข้ามาในระบบ ที่ได้มาจากตัวรับสัญญาณของรีโมตเมื่อปรับจอยสติกของรีโมตไปยังค่าต่ำสุด	33

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4	เอาต์พุตที่เป็นค่าที่ระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ ตรวจจับได้ จากสัญญาณที่มีความกว้างพัลส์ 1 มิลลิวินาที	33
4.5	สัญญาณอินพุตที่เข้ามาในระบบ ที่ได้มาจากตัวรับสัญญาณของรีโมตเมื่อปรับจอยสติกของรีโมตไปยังค่ากึ่งกลาง	34
4.6	เอาต์พุตที่เป็นค่าที่ระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ ตรวจจับได้ จากสัญญาณที่มีความกว้างพัลส์ 1.5 มิลลิวินาที	34
4.7	สัญญาณอินพุตที่เข้ามาในระบบ ที่ได้มาจากตัวรับสัญญาณของรีโมตเมื่อปรับจอยสติกของรีโมตไปยังค่าสูงสุด	35
4.8	เอาต์พุตที่เป็นค่าที่ระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ ตรวจจับได้ จากสัญญาณที่มีความกว้างพัลส์ 2 มิลลิวินาที	35
4.9	สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 1 มิลลิวินาที	37
4.10	หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาญที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที	38
4.11	สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 เป็น 2 มิลลิวินาที ช่องสัญญาณที่ 2 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 1 มิลลิวินาที	39
4.12	หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาญที่เข้ามาช่องสัญญาณ เอ (A) เป็น 2 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที	40
4.13	สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 เป็น 2 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 1 มิลลิวินาที	41
4.14	หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาญที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 2 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที	42

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.15	สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 2 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 1 มิลลิวินาที	43
4.16	หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาญที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 2 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที	44
4.17	สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 2 มิลลิวินาที	45
4.18	หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาญที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 2 มิลลิวินาที	46
4.19	กราฟค่ามุมความเอียงที่ได้จากการคำนวณค่าในเซ็นเซอร์วัดความเร่งและเซ็นเซอร์วัดความเร็วเชิงมุม โดยกราฟสีแดงจะเป็นกราฟของค่ามุมความเอียงที่ไม่ได้ผ่านวงจรกรองแบบคอมพลีเมนทารี่ และกราฟสีน้ำเงินเป็นกราฟของค่ามุมความเอียงที่ผ่านวงจรกรองแบบคอมพลีเมนทารี่	47
4.20	ผลลัพธ์ที่เมื่อนำมาแสดงบนแผนที่ระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้	48
4.21	ผลลัพธ์ที่ตัวเครื่องบินคำนวณค่ามุมแบรริ่งระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้	48
4.22	ผลลัพธ์ที่เมื่อนำมาแสดงบนแผนที่ระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้	49
4.23	ผลลัพธ์ที่ตัวเครื่องบินคำนวณค่าระยะทางระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้	50
4.24	เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ แบบยังไม่มีการปรับแต่งค่า	51
4.25	เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 1	51
4.26	เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 2	52
4.27	เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 3	53

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.28 เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 4 53
4.29 เส้นทางการบินแบบเต็มรูปแบบของระบบการบินถ่ายภาพอัตโนมัติ ที่บิน จากจุดออกบินไปยังจุดเป้าหมายและถ่ายภาพ แล้วทำการบินกลับมายังจุด ออกบินและลงจอดได้เองโดยอัตโนมัติ 54
4.30 ภาพถ่ายทางอากาศที่ได้จากเครื่องบินถ่ายภาพไร้คนขับ 55

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากการจะถ่ายภาพมุมสูงในปัจจุบันนี้ จะอาศัยเครื่องบินไร้คนขับขนิดบังคับ เพื่อบังคับให้ไปอยู่เหนือในจุดตำแหน่งต้องการจะถ่ายภาพแล้วถ่ายรูปกลับมา ซึ่งก็ยังต้องใช้มนุษย์ในการบังคับ แต่ถ้าหากว่า สามารถใช้ระบบนำร่อง นำพาเครื่องบินไปถ่ายภาพ ณ จุดต่างๆได้โดยเพียงใช้การตั้งเวลา และ สถานที่ที่จะไป ทำแทนมนุษย์ได้ จะช่วยทำให้สะดวกขึ้น จึงเป็นเหตุผลให้เกิดปริญญานิพนธ์นี้ขึ้นมา

1.2 วัตถุประสงค์

1) เพื่อออกแบบและสร้างระบบที่ใช้กล้องในการตรวจจับสัญลักษณ์ที่ใช้ในการลงจอดของเครื่องบินไร้คนขับ
2) เพื่อนำระบบตรวจจับสัญลักษณ์มาควบคุมเครื่องบินไร้คนขับให้สามารถล็อคตำแหน่งเพื่อลงจอดได้ตรงตามเป้าหมาย
3) เพื่อออกแบบและสร้างเครื่องบินไร้คนขับที่สามารถบินไปถ่ายภาพ ณ ตำแหน่งที่กำหนดได้โดยอัตโนมัติ

1.3 ขอบเขตของปริญญานิพนธ์

1) สร้างเครื่องบินไร้คนขับแบบ 4 ใบพัด ขนาด 56 เซนติเมตร x 56 เซนติเมตร และมีขนาดเส้นผ่านศูนย์กลางของใบพัด 25.4 เซนติเมตร
2) สร้างระบบรักษาสมดุลการบินของเครื่องบินไร้คนขับแบบ 4 ใบพัดโดยอาศัยเซ็นเซอร์วัดความเอียง
3) สามารถตรวจจับสัญลักษณ์ที่เป็นรูปวงกลม ที่มีพื้นที่ภายในเป็นสีขาวครึ่งหนึ่ง และสีดำครึ่งหนึ่ง
4) สามารถขึ้นบินและลงจอด ณ ตำแหน่งที่มีสัญลักษณ์สำหรับลงจอดได้
5) มีรัศมีการบินไม่เกิน 250 เมตร สูงไม่เกิน 50 เมตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

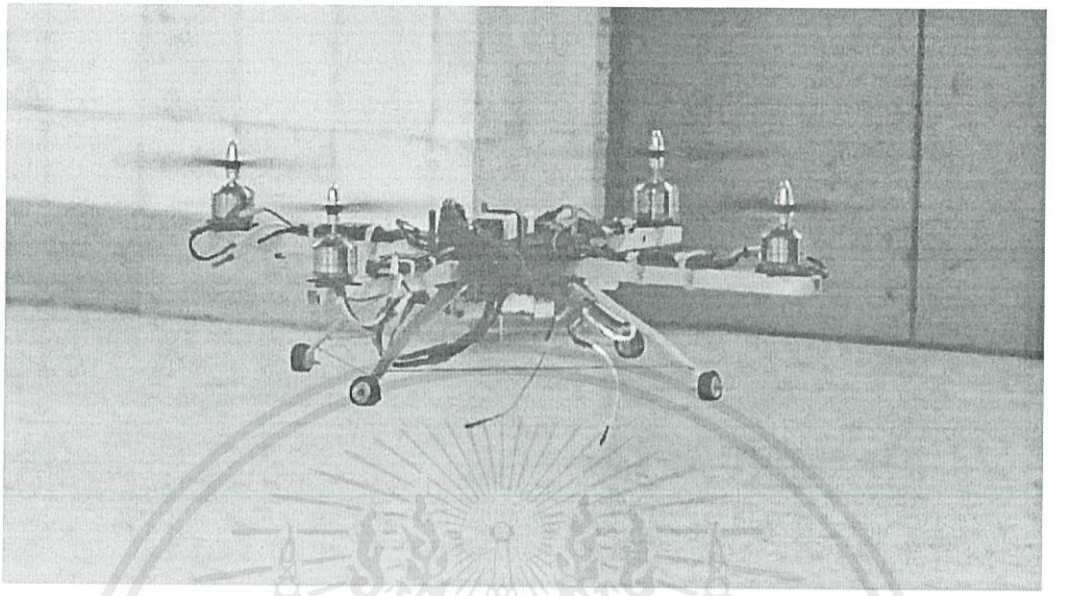
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 หลักการของเครื่องบิน 4 ใบพัด

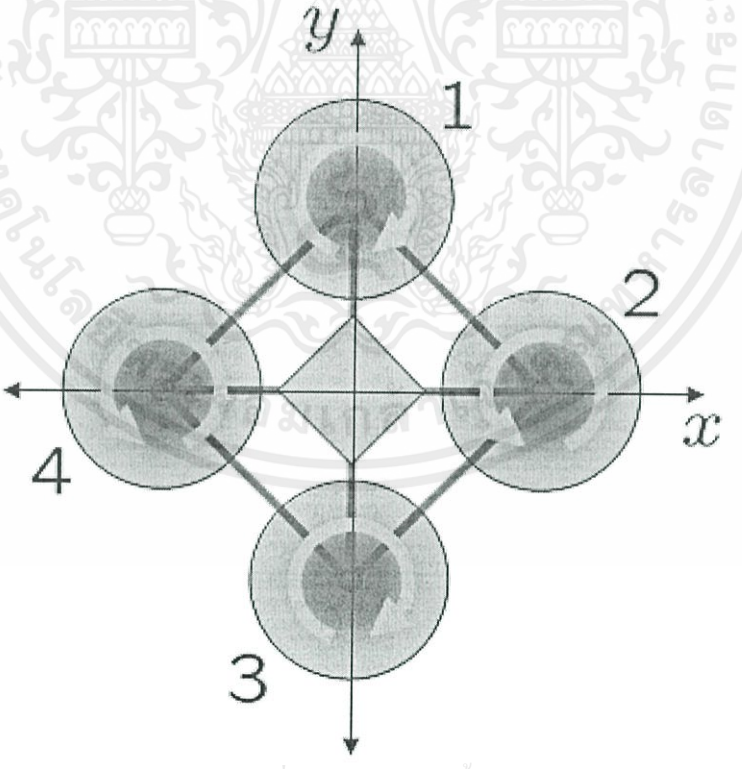
เครื่องบิน 4 ใบพัดเป็นเครื่องบินที่ใช้ใบพัดเพื่อขับเคลื่อนจำนวน 4 ใบ ดังรูปที่ 2.1 และรูปที่ 2.2 ซึ่งสามารถเคลื่อนที่ได้ทุกรูปแบบ ข้อดีของเครื่องบิน 4 ใบพัดคือ สามารถเคลื่อนที่ได้ทุกรูปแบบโดยที่ใบพัดทั้ง 4 ไม่จำเป็นต้องเปลี่ยนองศาหรือทิศทางการหมุน สามารถแบ่งใบพัดเป็น 2 คู่คือ คู่ที่หมุนตามเข็มนาฬิกา และคู่ที่หมุนทวนเข็มนาฬิกาตามลำดับ โดยใบพัดที่คู่กันนั้นอยู่ในตำแหน่งที่ตรงกันข้ามกันตามรูปที่ 2.2 สังเกตได้ว่าใบพัดที่ 1 และใบพัดที่ 3 ซึ่งอยู่ตรงข้ามกันนั้นหมุนตามเข็มนาฬิกาเช่นกัน และสังเกตได้ว่าใบพัดที่ 2 และ ใบพัดที่ 4 ที่อยู่ตรงข้ามกันหมุนทวนเข็มนาฬิกาเช่นกัน คู่ใบพัดที่หมุนตามเข็มนาฬิกาทำให้ตัวเครื่องบินเกิดโมเมนต์ในทางตรงกันข้ามกับใบพัดคือโมเมนต์ทวนเข็มนาฬิกา และคู่ใบพัดที่หมุนตามเข็มนาฬิกา ส่งผลให้ตัวเครื่องบินเกิดโมเมนต์ในทิศทางตรงกันข้ามกันคือโมเมนต์ตามเข็มนาฬิกา ที่สภาวะสมดุลโมเมนต์ตามเข็มนาฬิกาและโมเมนต์ทวนเข็มนาฬิกาจะเท่ากัน ทำให้เกิดการหักล้างโมเมนต์กันหมด ส่งผลให้เครื่องบินไม่เกิดการหมุน หากต้องการให้เครื่องบิน 4 ใบพัดเคลื่อนที่ ต้องอาศัยการเปลี่ยนแปลงความเร็วของใบพัดแต่ละใบ เพื่อให้เกิดแรงที่แตกต่างกัน หากต้องการให้เครื่องบินขึ้น-ลง ต้องอาศัยการเพิ่ม-ลดความเร็วของใบพัดทุกใบพัดในปริมาณที่เท่ากัน เพื่อให้เกิดสมดุลทางโมเมนต์คงเดิม ดังนั้นตัวเครื่องบินจึงไม่เกิดการหมุน

ในกรณีที่ตัวเครื่องบินเคลื่อนที่ไปทางข้างหน้าต้องลดความเร็วใบพัดที่ 1 และเพิ่มความเร็วใบพัดที่ 3 โดยอ้างอิงจากรูปที่ 2.2 ซึ่งทำให้ตัวเครื่องบินเอียงตัวไปข้างหน้า เป็นผลให้เครื่องบินเคลื่อนที่ไปข้างหน้า การลดความเร็วใบพัดที่ 1 และเพิ่มความเร็วใบพัดที่ 3 ต้องเพิ่ม-ลดเท่ากัน เพื่อให้เกิดสมดุลทางโมเมนต์ ในทางกลับกัน ถ้าต้องการให้ตัวเครื่องบินเคลื่อนที่ไปข้างหลังต้องเพิ่มความเร็วใบพัดที่ 1 และลดความเร็วใบพัดที่ 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 เครื่องบินสี่ใบพัด



รูปที่ 2.2 ลักษณะการหมุนของแต่ละใบพัดของเครื่องบิน 4 ใบพัด

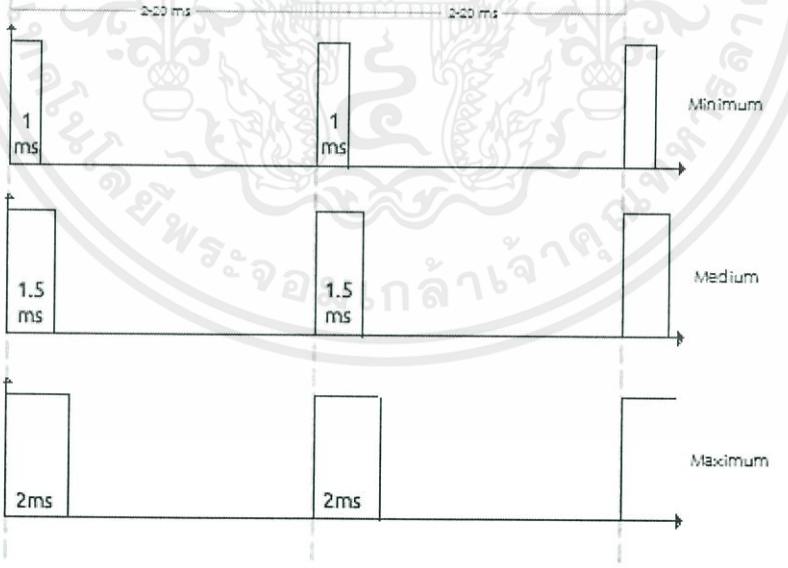
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อต้องการให้ตัวเครื่องบินเคลื่อนที่ไปทางซ้าย จำเป็นต้องลดความเร็วใบพัดที่ 4 และเพิ่มความเร็วใบพัดที่ 2 โดยอ้างอิงจากรูปที่ 2.2 เพื่อให้ตัวเครื่องบินเอียงไปทางซ้ายส่งผลให้เครื่องบินเคลื่อนที่ไปทางซ้าย การลดความเร็วใบพัด 4 และเพิ่มความเร็วใบพัดที่ 2 จะต้องเพิ่ม-ลดเท่ากัน เพื่อรักษาสมดุลทางโมเมนต์ ในทางกลับกัน หากต้องการให้ตัวเครื่องบินเคลื่อนไปทางขวาต้องเพิ่มความเร็วใบพัดที่ 4 และลดความเร็วใบพัดที่ 2

สำหรับการให้ตัวเครื่องบินหมุนทวนเข็มนาฬิกา ต้องเพิ่มโมเมนต์ทวนเข็มนาฬิกาและลดโมเมนต์ตามเข็มนาฬิกาของตัวเครื่องบิน โดยการเพิ่มความเร็วคู่ใบพัดที่ 1 กับใบพัดที่ 3 และลดความเร็วคู่ใบพัดที่ 2 กับ ใบพัดที่ 4 โดยการเพิ่ม-ลดเท่ากัน เพื่อให้แรงยกรวมของทั้ง 4 ใบพัดเท่าเดิม ในทางกลับกันสำหรับการจะทำให้ตัวเครื่องบินหมุนตามเข็มนาฬิกา เราต้องเพิ่มโมเมนต์ตามเข็มนาฬิกาและลดโมเมนต์ทวนเข็มนาฬิกาของตัวเครื่องบินโดยการลดความเร็วคู่ใบพัดที่ 1 กับ ใบพัดที่ 3 และเพิ่มความเร็วคู่ใบพัดที่ 2 กับใบพัดที่ 4

2.2 สัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ (RC Control Signal)

สัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุเพื่อควบคุมมอเตอร์และอุปกรณ์ในระบบ ใช้สัญญาณความกว้างพัลส์ (Pulse Width Modulation) ที่มีความกว้างพัลส์ตั้งแต่ 1 มิลลิวินาที ถึง 2 มิลลิวินาที จากคาบเวลาที่ตั้งแต่ 2 มิลลิวินาที ถึง 20 มิลลิวินาที โดยค่าความกว้างพัลส์ที่ 1 มิลลิวินาที หมายถึงค่าต่ำสุด และค่าความกว้างพัลส์ที่ 2 มิลลิวินาที หมายถึงค่าสูงสุด ดังรูปที่ 2.3



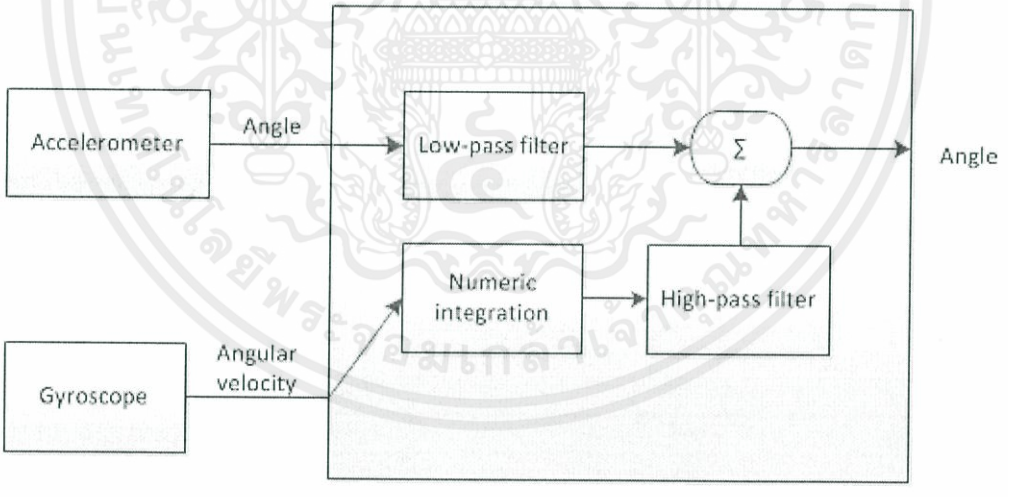
รูปที่ 2.3 สัญญาณความกว้างพัลส์ในมาตรฐานของสัญญาณควบคุมที่ในอุปกรณ์บังคับวิทยุ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 วงจรกรองสัญญาณแบบคอมพลีเมนทารี่ (Complementary Filter)

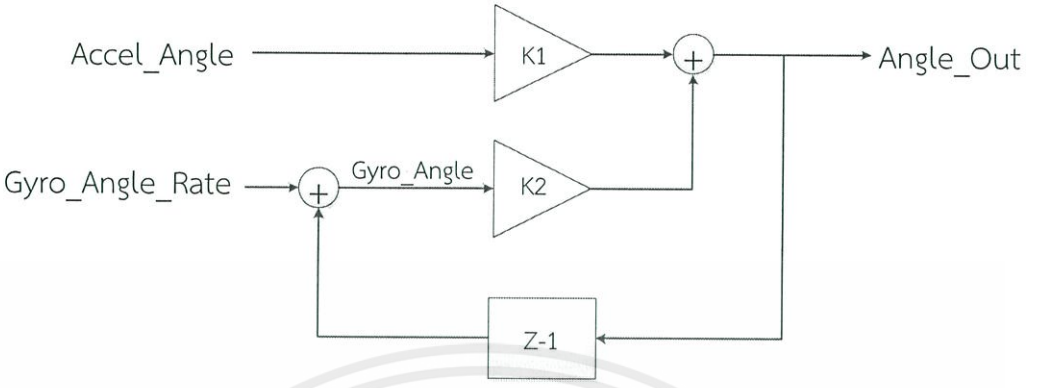
โดยทั่วไปแล้วในการวัดความเอียงจากระดับพื้นโลก นิยมใช้วิธีการรับค่าความเร่งที่ได้จากเซ็นเซอร์วัดความเร่งมาคำนวณหาค่ามุม โดยการคำนวณค่าอาร์คแทนจากค่าความเร่งระหว่างแกนสองแกน ในกรณีที่เซ็นเซอร์ได้รับแรงสั่นสะเทือน ส่งผลให้ค่าความเร่งที่ได้ ไม่ใช่ค่าความเร่งที่เกิดจากผลของแรงโน้มถ่วงโลกเพียงอย่างเดียว ส่งผลให้มุมที่ได้จากการคำนวณค่าอาร์คแทนจากค่าความเร่งระหว่างแกนสองแกนนั้นเกิดความผิดพลาดขึ้น ดังนั้นจึงมีการนำวงจรกรองสัญญาณแบบคอมพลีเมนทารี่เข้ามาช่วยในการกรองค่ามุมความเอียงที่เกิดความผิดพลาด ให้มีค่าความผิดพลาดน้อยลง

วงจรกรองสัญญาณแบบคอมพลีเมนทารี่ นำค่าความเร็วเชิงมุมที่ได้จากเซ็นเซอร์วัดความเร็วเชิงมุมเข้ามาช่วยในการกรองค่ามุมความเอียงให้มีค่าความผิดพลาดน้อยลง โดยมีหลักการคือ นำค่ามุมความเอียงที่ได้จากการคำนวณค่าอาร์คแทนระหว่างแกนสองแกน มาผ่านวงจรกรองความถี่ต่ำผ่าน และมารวมกับค่ามุมความเอียงที่ได้จากการคำนวณอินทิเกรตของค่าความเร็วเชิงมุมที่ผ่านวงจรกรองความถี่สูงผ่าน ดังบล็อกไดอะแกรมดังรูปที่ 2.4 สามารถเขียนโปรแกรมได้ดังโฟลว์ชาร์ตดังรูปที่ 2.5 โดยค่า K1 < K2 และ K1 + K2 = 1 จึงทำให้ค่าอัตราขยายของระบบรวมมีค่าเท่ากับ 1



รูปที่ 2.4 บล็อกไดอะแกรมของวงจรกรองคอมพลีเมนทารี่

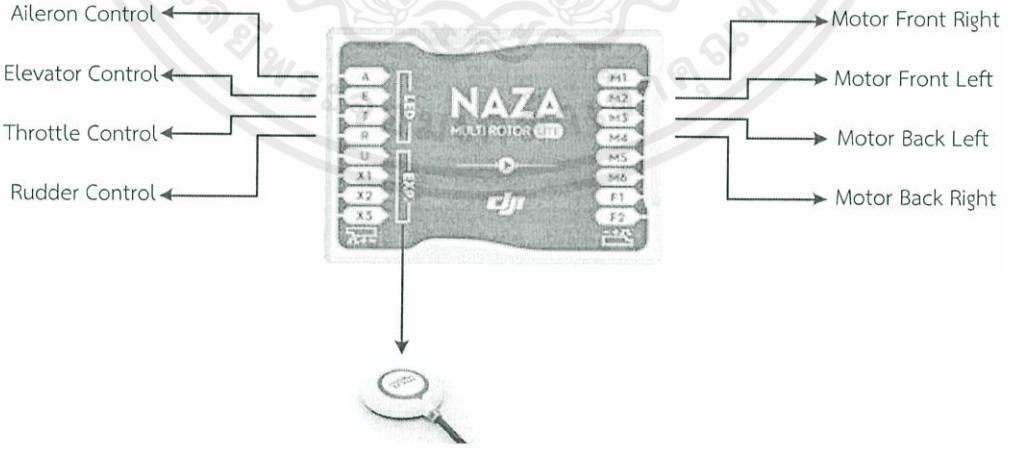
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.5 บล๊อกไดอะแกรมของวงจรกรองแบบคอมพลีเมนทารี่

2.4 กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ (Naza m lite Controller)

กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เป็นกล่องควบคุมที่ทำหน้าที่ควบคุมมอเตอร์ทั้งสี่ตัวให้สามารถรักษาการทรงตัวของตัวเครื่องบิน ให้อยู่ในแนวระดับที่ขนานกับพื้นโลก ให้อยู่ที่ระดับความสูงเท่าเดิม และล็อกอยู่กับที่โดยอาศัยตำแหน่งที่ได้จากจีพีเอสอยู่ตลอดเวลาเมื่อไม่มีการสั่งการให้เคลื่อนที่ โดยกล่องควบคุมสุมดุล นาซ่า เอ็ม ไลท์ รับอินพุตที่ใช้ในการควบคุมการเคลื่อนที่เข้ามาในรูปแบบของสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ โดยรับสัญญาณทั้งหมด 4 ชนิด คือ สัญญาณควบคุมการขึ้น-ลง สัญญาณควบคุมการหมุนตัว สัญญาณควบคุมการเอียงหน้า-หลัง และสัญญาณควบคุมการเอียงซ้าย-ขวา แสดงรายละเอียดช่องเสียบสายสัญญาณต่างๆ ได้ดังรูปที่ 2.6 ซึ่งกล่องควบคุม น่าซ่า เอ็ม ไลท์ จะมีโมดูลที่ติดมาด้วยที่ใช้ร่วมในการควบคุมการบินคือ โมดูลเข็มทิศและโมดูลจีพีเอส ซึ่งแสดงดังรูปที่ 2.7



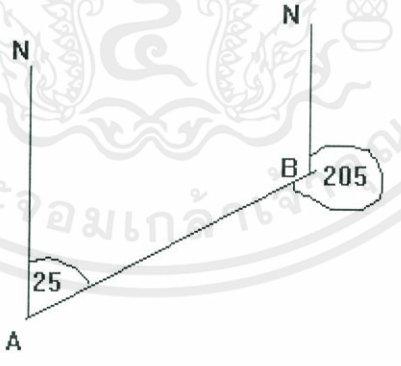
รูปที่ 2.6 รายละเอียดการต่อสายสัญญาณของกล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์

เอกสารนี้เป็นเอกสาร ... ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.7 โมดูลเข็มทิศและจีพีเอสของ นาซ่าเอ็มไลท์

2.5 มุมแบริ่ง (Bearing)

มุมแบริ่ง คือ มุมของเส้นจากจุดหนึ่งๆ ไปยังจุดหนึ่งๆ บนพิกัดของโลก เมื่อเทียบกับทิศเหนือ ซึ่งเป็นสิ่งที่สำคัญมากในระบบนำทาง เนื่องจากเป็นเครื่องมือที่ใช้บอกทิศทางที่ต้องเดินทางไปยังเป้าหมาย โดยแสดงตัวอย่างของมุมแบริ่งได้ ดังรูปที่ 2.8



รูปที่ 2.8 ตัวอย่างมุมแบริ่งจาก จุด A ไปยังจุด B และจากจุด B ไปยังจุด A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.8 เมื่อหามุมแบริ่งจากจุด A ไปยังจุด B จะได้มุมแบริ่งเท่ากับ 25 องศา แต่ถ้าหามุมแบริ่งจากจุด B ไปยังจุด A จะได้มุมแบริ่งเท่ากับ 205 องศา ซึ่งมุมแบริ่งหาได้จาก

$$\theta = \tan^{-1}(\sin(\Delta\lambda).\cos(\varphi_2).\cos(\varphi_1).\sin(\varphi_2) - \sin(\varphi_1).\cos(\varphi_2).\cos(\Delta\lambda)) \qquad (2.1)$$

โดย

θ คือค่ามุมที่ได้จากการคำนวณ ซึ่งมุมจะมีค่าเพิ่มขึ้นเมื่อมีการหมุนจาก ทิศเหนือไปยังทิศตะวันออก

φ_1 คือค่าองศาเส้นรุ้งของตำแหน่งปัจจุบัน

φ_2 คือค่าองศาเส้นรุ้งของตำแหน่งเป้าหมาย

$\Delta\lambda$ คือค่าความต่างระหว่างองศาเส้นแวงของตำแหน่งปัจจุบันกับของตำแหน่งเป้าหมาย

2.6 ระยะทางระหว่างจุดสองจุดบนโลก

ระยะทางระหว่างจุดสองจุดบนโลก คือ ระยะทางที่ได้จากการคำนวณค่าระยะทางระหว่างจุดสองจุดบนโลกโดยสามารถคำนวณได้ดังสมการที่ 2.2 - 2.4

$$a = \sin^2(\Delta\varphi/2) + \cos(\varphi_1)\cos(\varphi_2)\sin^2(\Delta\lambda/2) \qquad (2.2)$$

$$c = \tan^{-1}\left(\sqrt{a},\ \sqrt{(1-a)}\right) \qquad (2.3)$$

$$d = R \times c \qquad (2.4)$$

โดย

d คือค่าระยะห่างระหว่างจุดสองสุดที่ได้จากการคำนวณ

φ_1 คือค่าองศาเส้นรุ้งของตำแหน่งปัจจุบัน

φ_2 คือค่าองศาเส้นรุ้งของตำแหน่งเป้าหมาย

$\Delta\varphi$ คือค่าความต่างระหว่างองศาเส้นรุ้งของจุดปัจจุบันกับเป้าหมาย

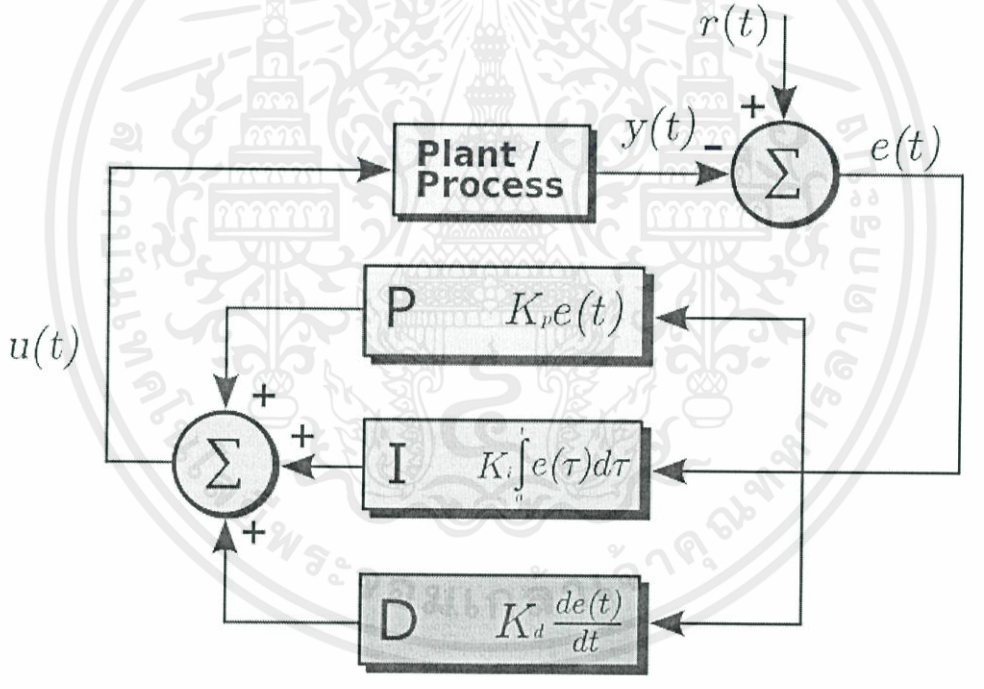
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$\Delta\lambda$ คือค่าความต่างระหว่างองศาเส้นแวงของจุดปัจจุบันกับเป้าหมาย

R คือค่ารัศมีของโลก

2.7 ระบบควบคุมพีไอดี (PID Controller)

ระบบควบคุมพีไอดี เป็นระบบควบคุมที่ถูกใช้งานอย่างเป็นวงกว้างในอุตสาหกรรม โดยระบบความคุมพีไอดีนั้น จะทำการคำนวณค่าความผิดพลาดที่เกิดขึ้นระหว่างค่าเอาต์พุตที่เราต้องการ กับค่าเอาต์พุตที่เป็นอยู่ในปัจจุบัน โดยจะมีการแบ่งการคำนวณออกเป็น 3 ส่วนคือ ส่วนการคำนวณแบบสัดส่วน ส่วนการคำนวณแบบปริพันธ์ และส่วนการคำนวณแบบอนุพันธ์ แสดงบล็อกไดอะแกรมดังรูปที่ 2.9



รูปที่ 2.9 บล็อกไดอะแกรมของระบบควบคุมพีไอดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.1 ส่วนการคำนวณแบบสัดส่วน

การคำนวณจะเป็นการนำค่าผิดพลาดระหว่างเอาต์พุตที่ต้องการ กับเอาต์พุตที่เป็นอยู่ในปัจจุบัน ไปคูณค่าคงที่ตัวหนึ่ง เรียกว่า k_p แล้วนำค่าที่ได้จากการคูณไปจ่ายเป็นอินพุตให้ระบบ การคำนวณแบบสัดส่วนแสดงได้ดังนี้

$$P = k_p \times e(t) \tag{2.5}$$

2.7.2 ส่วนการคำนวณแบบปริพันธ์

การคำนวณจะเป็นการนำค่าผิดพลาดระหว่างเอาต์พุตที่ต้องการ กับเอาต์พุตที่เป็นอยู่ในปัจจุบัน ไปอินทิเกรต แล้วนำค่าที่ได้จากการอินทิเกรตไปคูณค่าคงที่ตัวหนึ่ง เรียกว่า k_i แล้วนำค่าที่ได้จากการคูณไปจ่ายเป็นอินพุตให้ระบบ สมการในส่วนการคำนวณแบบปริพันธ์แสดงดังนี้

$$I = k_i \times \int e(t)dt \tag{2.6}$$

2.7.3 ส่วนการคำนวณแบบอนุพันธ์

การคำนวณจะเป็นการนำค่าผิดพลาดระหว่างเอาต์พุตที่ต้องการ กับเอาต์พุตที่เป็นอยู่ในปัจจุบัน ไปดิฟเฟอเรนชิเอต แล้วนำค่าที่ได้จากการดิฟเฟอเรนชิเอตไปคูณค่าคงที่ตัวหนึ่ง เรียกว่า k_d แล้วนำค่าที่ได้จากการคูณไปจ่ายเป็นอินพุตให้ระบบ สมการในส่วนการคำนวณแบบอนุพันธ์แสดงดังนี้

$$D = k_d \times \frac{d}{dt} e(t) \tag{2.7}$$

เมื่อนำการคำนวณทั้งสามรูปแบบมารวมกัน จะได้เป็น

$$u(t) = k_p \times e(t) + k_i \times \int e(t)dt + k_d \times \frac{d}{dt} e(t) \tag{2.8}$$

ซึ่งค่า k_p , k_i และ k_d จะต้องทำการจูนเพื่อที่จะทำให้ระบบเกิดเสถียรภาพที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

สำหรับปริญญานิพนธ์นี้เป็นการออกแบบระบบเครื่องบินถ่ายภาพไร้คนขับที่มีความสามารถในการขึ้นบินจากฐานออกบินและลงจอดสู่ฐานออกบินได้อย่างแม่นยำโดยมีการใช้ระบบจีพีเอสในการระบุตำแหน่งโดยหยาบ และใช้ระบบประมวลผลทางภาพตรวจจับสัญลักษณ์ที่ใช้ในการลงจอดของเครื่องบินไร้คนขับในการระบุตำแหน่งโดยละเอียด

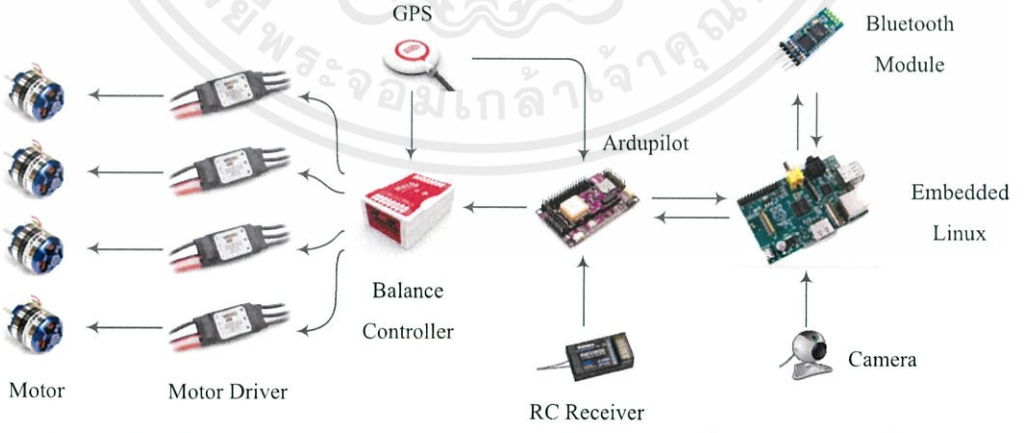
ในระบบประกอบไปด้วย 3 ส่วนหลัก คือ ส่วนควบคุมการบิน (Flight Control) ส่วนการเชื่อมต่อ (Interface) และส่วนการวางแผน (Planner)

ส่วนของควบคุมการบิน ประกอบไปด้วย กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์, จีพีเอส , วงจรขับมอเตอร์ และมอเตอร์ ทำหน้าที่ในการจัดการเรื่องการควบคุมการบินทั้งหมด ไม่ว่าเป็นการรักษาการทรงตัว การรักษาระดับความสูง การควบคุมทิศทางเครื่องบิน และการระบุตำแหน่งกับจีพีเอส โดยรับคำสั่งรูปแบบเคลื่อนที่จากส่วนการเชื่อมต่อ

ส่วนที่ทำหน้าที่เชื่อมระหว่างอุปกรณ์ เป็นส่วนที่ทำหน้าที่เชื่อมต่ออุปกรณ์แต่ละส่วนเข้าด้วยกัน เป็นเสมือนตัวแปลงรูปแบบการติดต่อของอุปกรณ์แต่ละตัวซึ่งมีรูปแบบการติดต่อสื่อสารที่ต่างรูปแบบกันให้สามารถต่อติดสื่อสารกันได้

ส่วนการวางแผน เป็นส่วนที่ทำหน้าที่ในการกำหนดเส้นทางการบินให้กับเครื่องบิน ทำหน้าที่ติดต่อคอยรับคำสั่งและส่งข้อมูลสถานะของเครื่องบินกับส่วนควบคุมกลาง และทำหน้าที่ในการประมวลผลภาพ เพื่อตรวจจับสัญลักษณ์ที่ใช้ในการลงจอดของเครื่องบินถ่ายภาพไร้คนขับ

บล็อกไดอะแกรมการติดต่อของอุปกรณ์แต่ละตัวของระบบแสดงได้ดังรูปที่ 3.1



รูปที่ 3.1 บล็อกไดอะแกรมการติดต่อของอุปกรณ์ในระบบเครื่องบินถ่ายภาพไร้คนขับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1 การออกแบบ

3.1.1 การเตรียมทรัพยากรณ์ให้พร้อมต่อการเขียนโปรแกรม

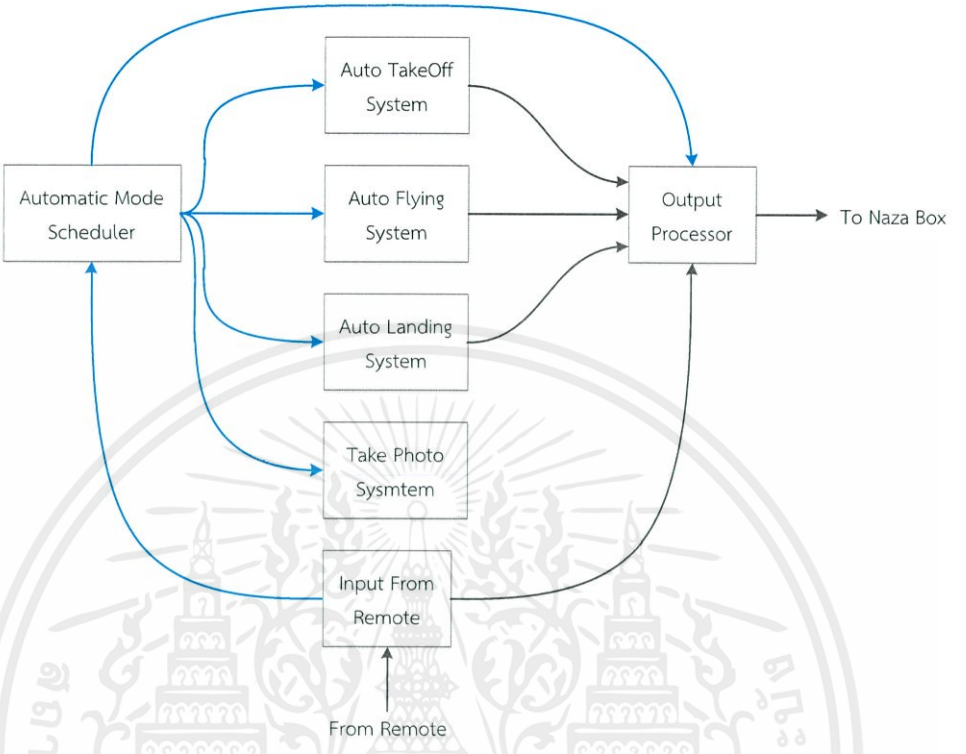
ขั้นตอนการดาวน์โหลดและติดตั้งโปรแกรม ArduPilot-Arduino

1) ดาวน์โหลดโปรแกรมที่ https://code.google.com/p/ardupilot-mega/downloads/list โดยไฟล์โปรแกรมจะมีชื่อว่า ArduPilot-Arduino-1.0.3-windows.zip
2) แตกไฟล์ ArduPilot-Arduino-1.0.3-windows.zip ออกมาจะได้โฟล์เดอร์ชื่อ ArduPilot-Arduino-1.0.3-windows ซึ่งบรรจุโปรแกรมทั้งหมดอยู่ข้างใน สามารถใช้งานได้เลย
3) ดาวน์โหลดไลบรารี่ ที่ใช้ในการเขียนโปรแกรมที่ https://github.com/diydrones/ardupilot
4) แตกไฟล์ที่ดาวน์โหลดออกมาจะได้โฟล์เดอร์ชื่อ ardupilot- master
5) เข้าไปข้างในโฟล์เดอร์ ardupilot-master จะเจอโฟล์เดอร์ชื่อ libraries เข้าไปในโฟล์เดอร์ libraries แล้วทำการก๊อปปี้ไฟล์และโฟล์เดอร์ทั้งหมดที่อยู่ในโฟล์เดอร์ libraries ไปไว้ในโฟล์เดอร์ ArduPilot-Arduino-1.0.3-windows/libraries
6) รอจนกว่าจะก๊อปปี้เสร็จ เป็นอันเสร็จสิ้นการติดตั้งโปรแกรม ArduPilot-Arduino เพื่อที่ใช้ในการพัฒนาโปรแกรม

3.1.2 การออกแบบโครงสร้างโปรแกรมโดยรวม

โปรแกรมในระบบจะถูกแบ่งออกเป็น 7 ส่วนหลักๆ ได้แก่ ระบบรับสัญญาณจากรีโมต ระบบควบคุมและกำหนดขอบเขตในการส่งสัญญาณควบคุมไปยังกล่องนาซ่า ระบบควบคุมการขึ้นบิน ระบบควบคุมการลงจอด ระบบการบินอัตโนมัติ ระบบการถ่ายภาพ และระบบ ควบคุมกำหนดการของการบิน แสดงการเชื่อมต่อระหว่างแต่ละระบบแสดงดังรูปที่ 3.2 โดยเส้นสีดำหมายถึง สัญญาณที่เป็นสัญญาณเพื่อที่จะส่งออกไปควบคุมกล่องนาซ่า และเส้นสีฟ้า หมายถึง สัญญาณควบคุมในระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

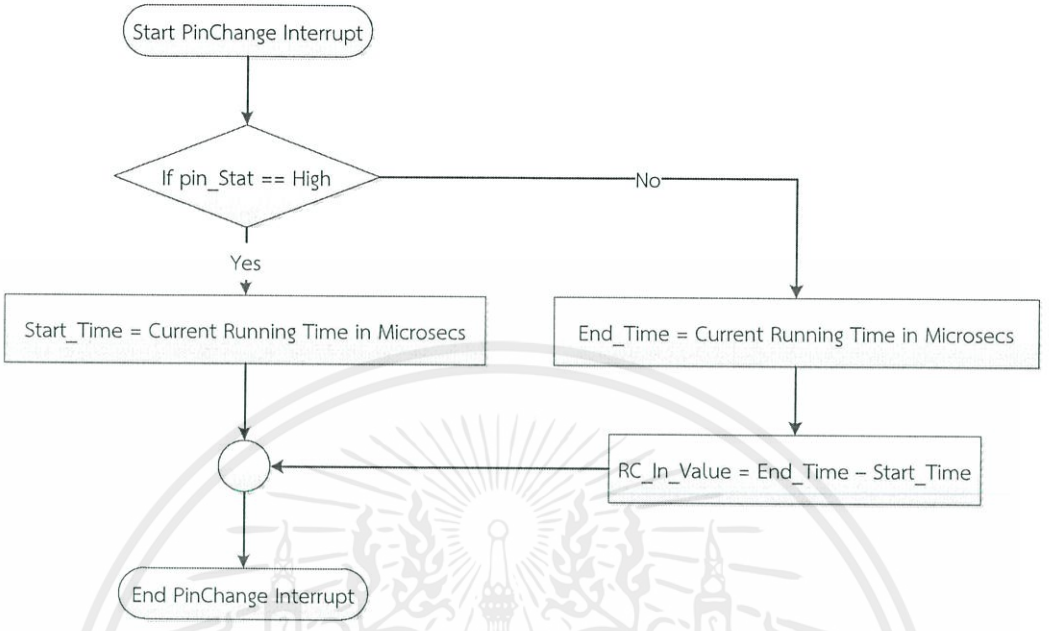


รูปที่ 3.2 แสดงการเชื่อมต่อระหว่างแต่ละระบบของโปรแกรม

3.1.3 การออกแบบโปรแกรมที่ใช้ในการตรวจจับสัญญาณควบคุมที่ได้รับมาจากตัวรับสัญญาณของรีโมต

ในการที่จะตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุนั้น สามารถทำได้หลายวิธี ปริญญานิพนธ์นี้ได้เลือกใช้วิธีการตรวจจับด้วย อินเทอร์รัพท์ภายนอกชนิดการเปลี่ยนแปลงของลอจิก (Pin-Change External Interrupt) และระบบการเดินเวลาหลักของระบบ ในการตรวจจับสัญญาณที่ใช้ระบบบังคับวิทยุที่ได้รับมาจากตัวรับสัญญาณของรีโมต โดยแสดงโฟลว์ชาร์ตของโปรแกรมได้ดังรูปที่ 3.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.3 โฟลว์ชาร์ตของโปรแกรมที่ใช้ในการตรวจจับสัญญาณที่ใช้ในระบบบังคับวิทยุที่ได้รับมาจากตัวรับสัญญาณของรีโมต

แสดงโค๊ดการเขียนโปรแกรมดังนี้

โปรแกรมในส่วนของการเตรียมการ	
sei();	- เปิดการใช้ Interrupt โดยรวมของไมโครคอนโทรลเลอร์
EICRA = 0b00000001;	- ใช้งาน External Interrupt ในโหมด Pin Change ที่ขา INT0
EIMSK = 0b00000001;	- เปิดการใช้งาน Interrupt ที่ขา INT0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

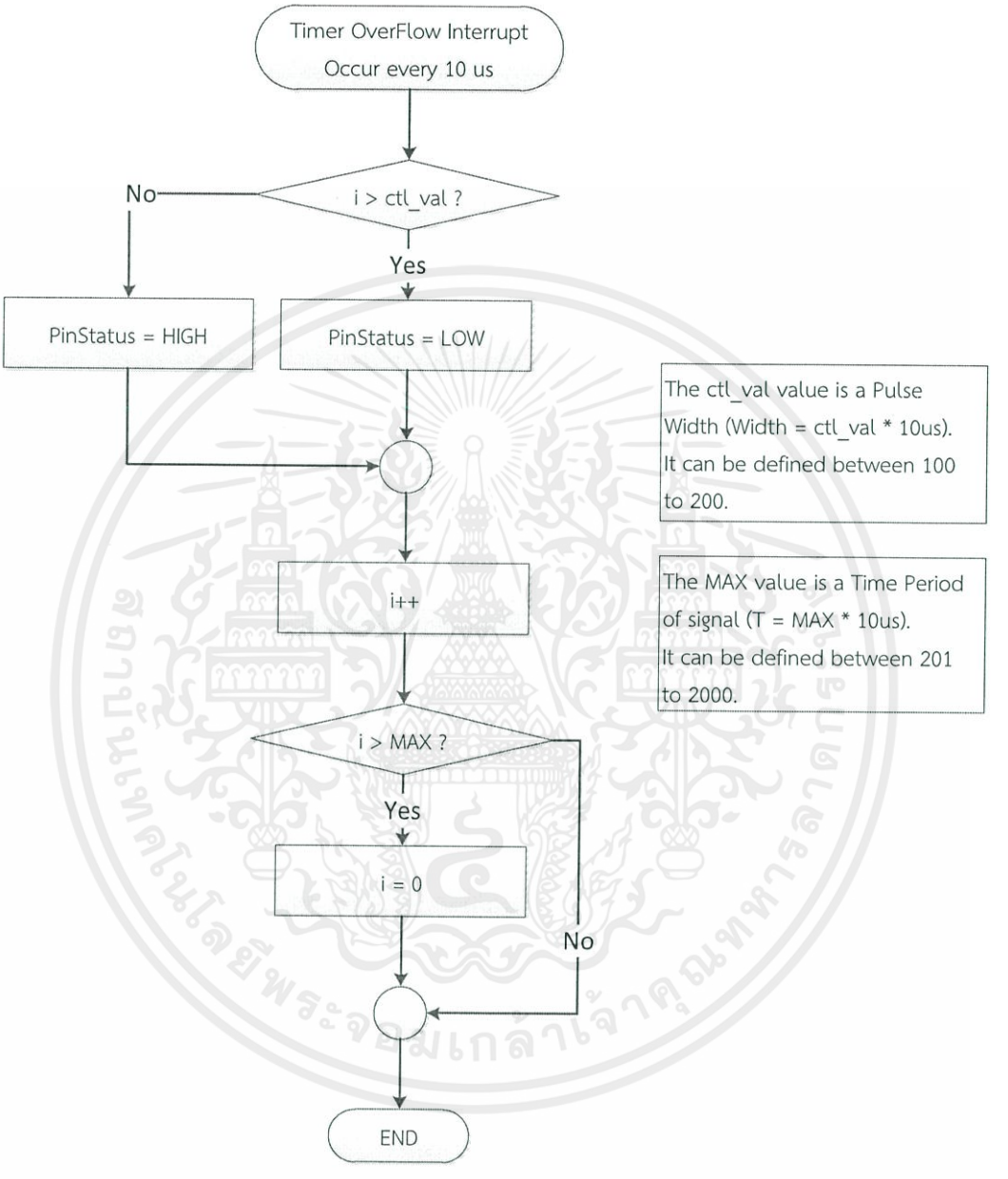
โปรแกรมในส่วนของการทำงาน	
ISR (INT0_vect) {	- ฟังชั่น ISR (INT0_vect) เป็นฟังชั่น Interrupt Routine ที่จะถูกเรียกเมื่อเกิด Interrupt จากขา INT0
static uint32_t start_t,stop_t;	- ประกาศตัวแปล start_t , stop_t ให้ชนิดจำนวนเต็มแบบถาวร ขนาด 32 bit
if(pinStatus() == HIGH)	- ถ้าสัญญาณที่เข้ามา เป็นขอบขาขึ้น
{	
start_t = timer_now();	- ให้เริ่มนับเวลา
}	
else	- ถ้าไม่ใช่ขอบขาขึ้น (เป็นขอบขาลง)
{	
stop_t = timer_now();	- ให้หยุดการนับเวลา
width = stop_t – start_t;	- ความกว้างของสัญญาณเกิดจากเวลาเริ่มต้นลบด้วย เวลาสิ้นสุด
}	
}	

3.1.4 การออกแบบโปรแกรมที่ใช้ในการสร้างสัญญาณที่ใช้ในระบบบังคับวิทยุเพื่อที่จะนำไปสั่งการกล่องควบคุมการทรงตัว

ในการที่จะสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุนั้น สามารถทำได้หลายวิธี สามารถทำได้ทั้งโดยฮาร์ดแวร์และโดยซอฟต์แวร์ ในปริญญานิพนธ์นี้ได้เลือกใช้การสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุด้วยซอฟต์แวร์ เนื่องจากถึงแม้ว่าการสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุด้วยฮาร์ดแวร์นั้น จะลดภาระการประมวลผลของไมโครคอนโทรลเลอร์ลงได้มากก็ตาม แต่ก็มีข้อจำกัดตรงจำนวนช่องสัญญาณที่สามารถใช้ได้เพียง 2 ช่องเท่านั้น แต่ระบบที่ใช้นั้นจำเป็นต้องใช้สัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุขั้นต่ำ 5 ช่อง เพราะฉะนั้นการสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุด้วยฮาร์ดแวร์นั้นจึงไม่เพียงพอสำหรับระบบในปริญญา-นิพนธ์นี้ จึงต้องสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุด้วยซอฟต์แวร์แทน

การสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุด้วยซอฟต์แวร์นั้นก็สามารถทำได้หลายวิธี สำหรับปริญญานิพนธ์นี้ได้ใช้การเกิดอินเทอร์รัพท์จากการเกิดการโอเวอร์โฟล (Over Flow) ของไทม์เมอร์ เพื่อเป็นการสร้างการอ้างอิงทางเวลาสำหรับการสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ โดยแสดงโฟลว์ชาร์ตของโปรแกรมดังรูปที่ 3.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.4 โฟลว์ชาร์ตของโปรแกรมที่ใช้สร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมในส่วนของการเตรียมการ	
sei();	- เปิดการใช้ Interrupt โดยรวมของไมโครคอนโทรลเลอร์
TIMSK1 = 0b00000001;	- เปิดการใช้งาน Interrupt จาก Timer1 Overflow
TCCR1B = 0b00000001;	- เปิดการใช้งาน Timer ชนิดมีค่าการหารความถี่เท่ากับ 1/8
TCNT1 = 65536 – 20;	- ตั้งตัวนับให้มีค่าเท่ากับค่าสูงสุดลบด้วย 20 ซึ่งเป็นจำนวนครั้งที่เราต้องนับ เพื่อให้เกิดการ Interrupt ด้วยคาบเวลา 10 us

โปรแกรมในส่วนของการทำงาน	
ISR (TIM1_OVF_vect)	- ฟังชั่น ISR (TIM1_OVF_vect) เป็นฟังชั่น Interrupt Routine ที่จะถูกเรียกเมื่อเกิด Interrupt จากการ Overflow ของ Timer1
{	
static uint16 i;	- ประกาศตัวแปรชื่อ i ชนิดจำนวนเต็มขนาด 16 bit เอาไว้เป็นตัวนับ
if(i > ctl_val)	- ถ้า i มีค่ามากกว่า ค่า Pulse Width ที่ต้องการ
{	
setPinStatus(LOW);	- ให้สถานะของขาเอาต์พุตเป็น LOW
}	
else	- ถ้า i มีค่าไม่มากกว่า ค่า Pulse Width ที่ต้องการ
{	
setPinStatus(HIGH);	- ให้สถานะของขาเอาต์พุตเป็น HIGH
}	
i++;	- เพิ่มค่า i ขึ้น 1 ค่า
if(i > MAX)	- ถ้า i มีค่ามากกว่าค่า MAX
{	
i = 0;	- รีเซ็ตให้ค่า i มีค่าเท่ากับ 0
}	
}	

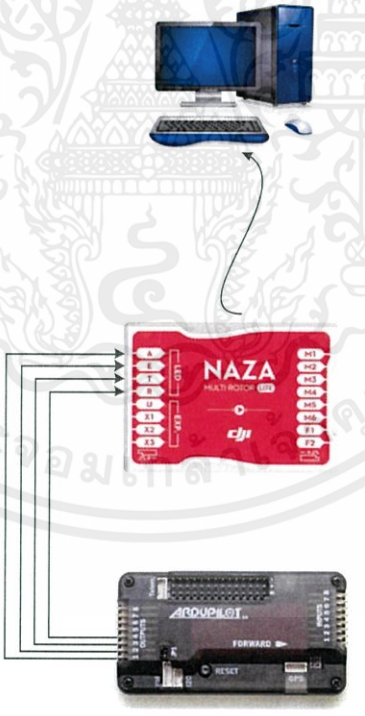
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าหากต้องการเปลี่ยนค่าความกว้างพัลส์ จะทำได้โดยการเปลี่ยนค่าตัวแปร ctl_val ซึ่งเป็นตัวแปรที่กำหนดความกว้างพัลส์โดยตรง

ถ้าหากต้องการเปลี่ยนค่าคาบเวลาของสัญญาณ จะทำได้โดยการเปลี่ยนค่าตัวแปร MAX ซึ่งเป็นตัวแปรที่กำหนดค่าคาบเวลาของสัญญาณ

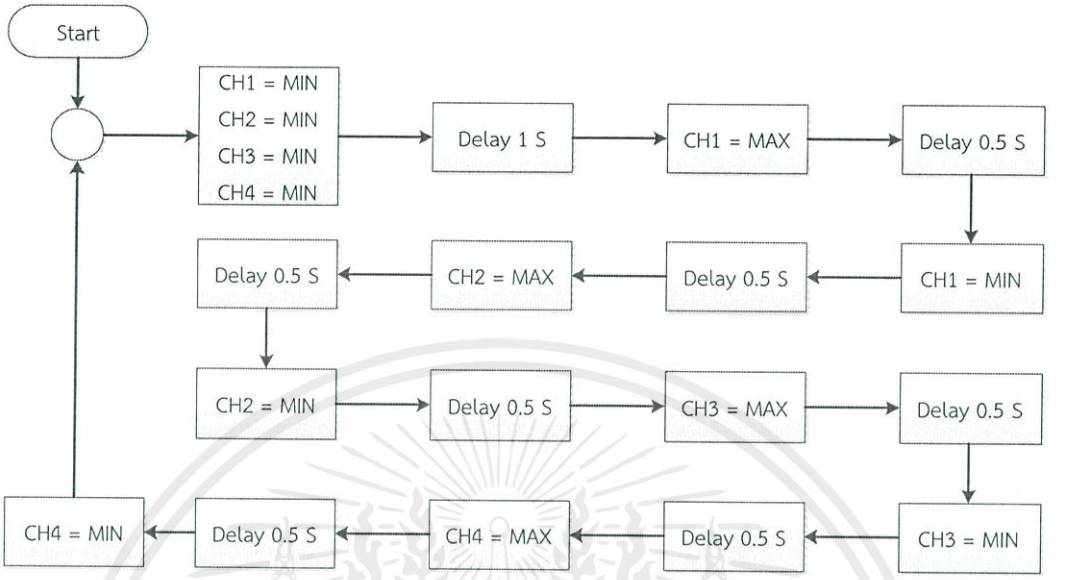
3.1.5 การออกแบบการทดลองใช้สัญญาณที่ใช้ในระบบบังคับวิทยุที่ถูกสร้างขึ้นจากโปรแกรมที่ออกแบบ ไปต่อเพื่อควบคุมกล่องควบคุมสมดุล นาซ่า เอม ไลท์

ในการออกแบบนั้นจะเขียนโปรแกรมลงบอร์ดอาดูไพล็อต ให้ทำการส่งสัญญาณที่ใช้ในระบบบังคับวิทยุไปยังกล่องควบคุมสมดุล นาซ่า เอม ไลท์ และดูค่าที่กล่องควบคุมสมดุล นาซ่า เอม ไลท์ ได้รับสัญญาณเข้ามาผ่านโปรแกรมนาซ่า เอม ไลท์ แอสซิสแทนท์ (Naza m lite Assistant) ว่า เป็นไปตามที่ต้องการหรือไม่ โดยแสดงการเชื่อมต่อดังรูปที่ 3.5 และเขียนโปรแกรมทดสอบตามโฟลว์ชาร์ตดังรูปที่ 3.6



รูปที่ 3.5 การเชื่อมต่อเพื่อทำการทดลองส่งสัญญาณที่ใช้ในระบบบังคับวิทยุจากบอร์ดอาดูไพล็อต ไปยังกล่องควบคุมสมดุล นาซ่า เอม ไลท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

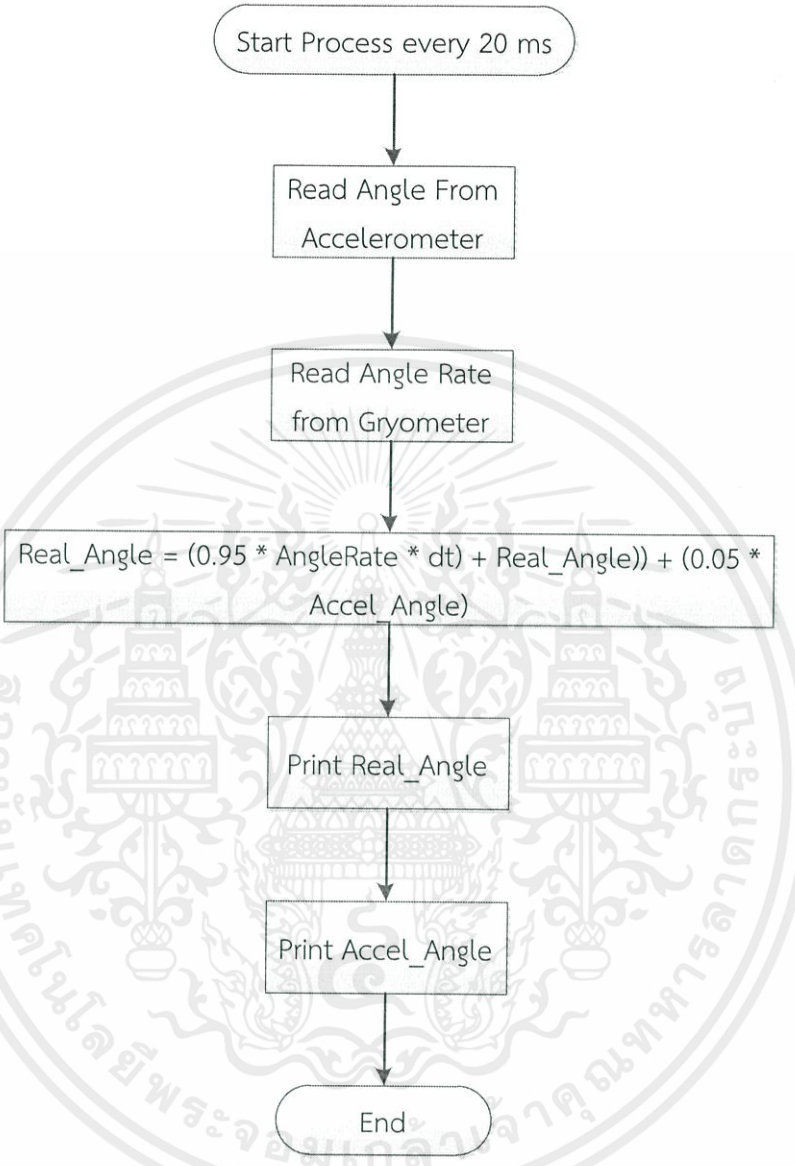


รูปที่ 3.6 โฟลว์ชาร์ตของโปรแกรมที่ใช้ในการทดลองควบคุมกล่องควบคุมสมดุล นาซ่า เอม ไลท์

3.1.6 การออกแบบการทดลองรับสัญญาณที่ได้จากเซ็นเซอร์วัดความเร่งและเซ็นเซอร์วัดความเร็วเชิงมุม คำนวณผ่านวงจรกรองสัญญาณแบบคอมพลีเมนทารี่

ในการทดลองนี้ จะเป็นการเขียนโปรแกรมอ่านค่าจากเซ็นเซอร์วัดความเร่ง และเซ็นเซอร์วัดความเร็วเชิงมุม แล้วนำค่าที่ได้มากรองผ่านวงจรกรองแบบคอมพีเมนทารี่ แล้วส่งค่าออกมาแสดงค่าออกมาดูในรูปของกราฟบนคอมพิวเตอร์ ซึ่งมีโฟลว์ชาร์ตของการเขียนโปรแกรมได้ดังรูปที่ 3.7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

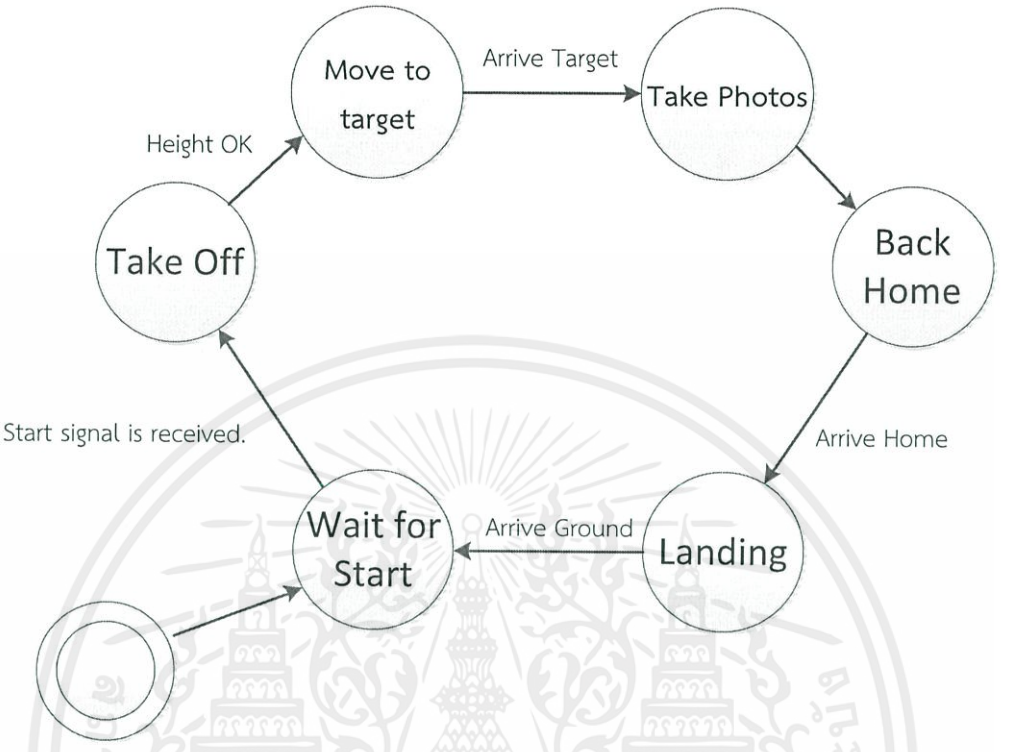


รูปที่ 3.7 โฟลว์ชาร์ตของโปรแกรมการกรองแบบคอมพลีเมนทารี่

3.1.7 การออกแบบระบบการบินอัตโนมัติ

ในระบบการบินอัตโนมัติ จะเป็นรูปแบบการทำงานแบบเป็นกำหนดการ คือมีลำดับกำหนดการที่เก็บไว้ในหน่วยความจำ การทำงานจะไม่มีการเปลี่ยนรูปแบบกำหนดการ แต่สิ่งที่เปลี่ยนแปลงได้คือ ค่าพารามิเตอร์ของระบบ เช่น ค่าพิกัดเป้าหมาย ค่าความสูงในการบิน เป็นต้น แสดงสถานะต่างๆดังรูปที่ 3.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.8 สถานะไดอะแกรมของระบบควบคุมการบินอัตโนมัติ

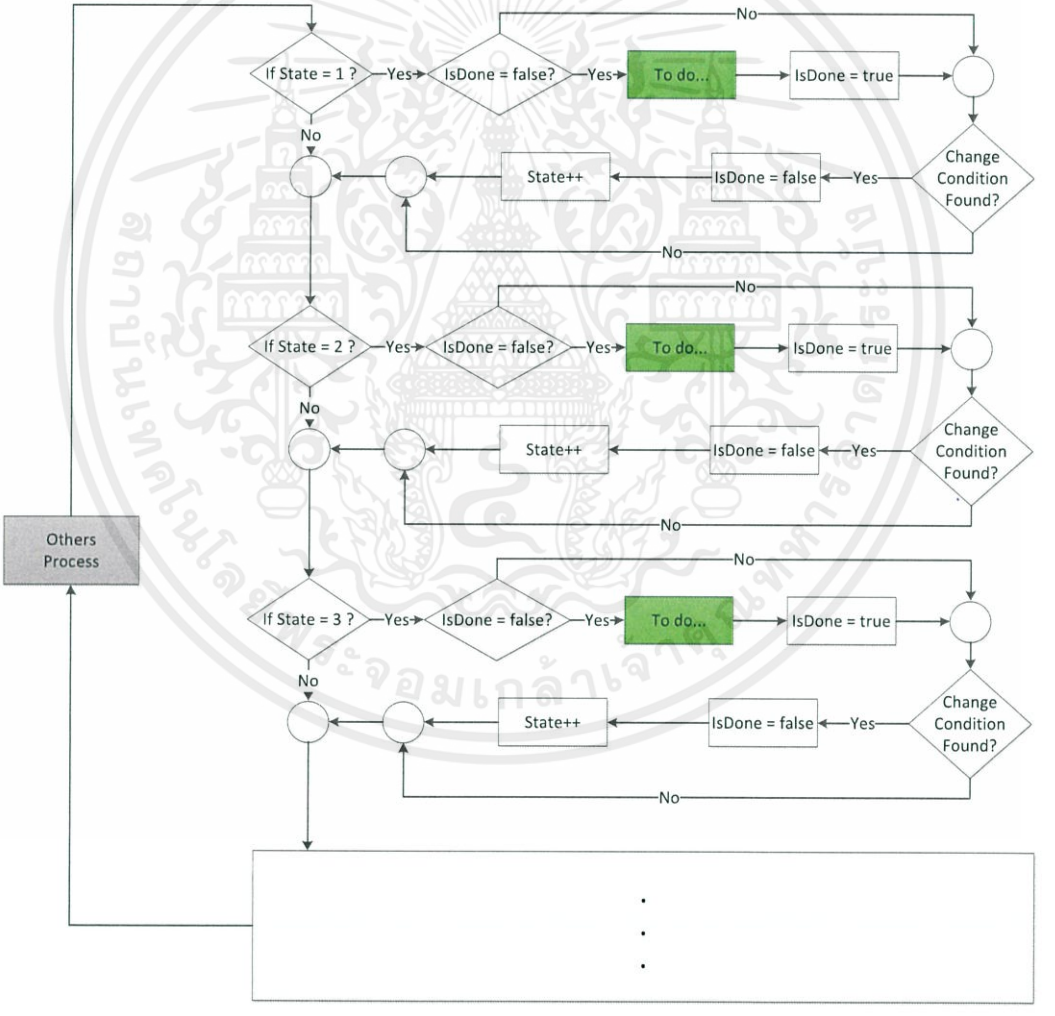
โปรแกรมจะเริ่มที่การรอรับคำสั่งการเริ่มงาน เมื่อได้รับคำสั่ง จะเปลี่ยนสถานะไปยังสถานะการออกบิน สถานะการออกบินจะทำการสั่งการให้เครื่องบินขึ้นบินในแนวดิ่ง เมื่อถึงระดับความสูงที่กำหนดไว้ ก็จะทำการเปลี่ยนสถานะไปยังสถานะการเคลื่อนที่ไปยังเป้าหมาย สถานะการเคลื่อนที่ไปยังเป้าหมาย จะทำการหันหัวเครื่องบินไปยังเป้าหมายและเคลื่อนที่ไปข้างหน้า เมื่อเครื่องบินเคลื่อนที่ไปถึงเป้าหมาย จะทำการเปลี่ยนสถานะไปยังสถานะการถ่ายภาพ เมื่อถ่ายภาพเสร็จจะทำการเปลี่ยนสถานะไปยังสถานะการบินกลับ สถานะการบินกลับจะทำการหันหัวเครื่องบินไปยังฐานออกบินและเคลื่อนที่ไปข้างหน้า เมื่อเครื่องบินกลับถึงฐานออกบินจะทำการเปลี่ยนสถานะไปยังสถานะการลงจอด สถานะการลงจอดจะลดระดับของเครื่องบินลงเรื่อยๆ และเมื่อเครื่องบินลดระดับลงถึงพื้น จะทำการเปลี่ยนสถานะไปยังสถานะการรอคำสั่งเริ่มการทำงานอีกครั้ง ลักษณะของโปรแกรมที่เขียนจะเป็นในลักษณะการเปิดสถาณะของโปรแกรมไปเรื่อยๆ แต่จะไม่ได้เป็นรูปแบบอนุกรม เนื่องจากว่าโปรแกรมจะต้องทำงานในลักษณะของ การทำงานแบบขนาน จึงไม่สามารถเขียนโปรแกรมแบบอนุกรมได้ โดยข้างล่างนี้จะเป็นโค๊ดโปรแกรมในส่วนของ ลำดับการบินอัตโนมัติ

ในการเขียนโปรแกรมควบคุมการบินอัตโนมัติจะสามารถแบ่งการเขียนโปรแกรมได้เป็น 2 ชั้น คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) **ชั้นการจัดการ การเปลี่ยนสถาณะของลำดับการทำงานของระบบบินอัตโนมัติ (Multitask Management Layer)**

ในระบบที่ต้องการความสามารถในการทำงานหลายๆอย่างพร้อมกัน หรือที่เรียกกันว่า Multitask นั้น จะต้องออกแบบโปรแกรมให้ทำงานในลักษณะของการแบ่งช่วงเวลาในการทำงานให้แต่ละงาน โดยจะสลับกันไปมา ฉะนั้นรูปแบบการเขียนโปรแกรม จะต้องอยู่ในลักษณะของการ วิ่งผ่านไปเรื่อยๆ ห้ามมีรูปแบบของการหยุดอยู่ที่กระบวนการใดกระบวนการหนึ่ง ฉะนั้นทางผู้จัดทำจึงได้ออกแบบตัวจัดการ การทำงานแบบลำดับ ให้สามารถทำงานในรูปแบบ Multitask ได้ขึ้นมา ดังโฟลวชาร์ตรูปที่ 3.9



รูปที่ 3.9 โฟลว์ชาร์ตของโปรแกรมจัดการการทำงานแบบ Multitask

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยมีโค๊ดโปรแกรมดังนี้

โปรแกรมในส่วนของการจัดการให้ทำงานแบบ Multitask	

if(State == 1)
{
  if(IsDone == 0)
  {
  //Do Something here
  IsDone = 1;
  }
  if(<ChangeStateCondition>)
  {
  IsDone = 0;
  State++;
  }
}
.
.
.
	- ถ้า State อยู่ในลำดับที่เราต้องการ
	
	- ถ้ายังไม่ได้กระทำกระบวนการ
	
	- ทำกระบวนการที่ต้องการทำ
	- เปลี่ยนค่าตัวแปร IsDone =1 เพื่อบอกว่าทำกระบวนการใน State ไปแล้ว
	- ถ้าเกิดเหตุการณ์ที่ทำให้เปลี่ยน State
	
	- รีเซ็ตตัวแปร IsDone เพื่อใช้ใน State ถัดไป
	- เปลี่ยนตัวแปร State ให้มีค่าเพิ่มขึ้น เพื่อ ให้ไปทำ State ถัดไป

2) ชั้นกระบวนการ (Process Layer)

เป็นชั้นการทำงานที่เป็นคำสั่ง สั่งงานให้เครื่องบินทำงานต่างๆ ทางผู้จัดทำได้สร้างฟังชั่นในการบังคับการบินไว้ทั้งหมด 30 ฟังชั่นเพื่อใช้งานได้แก่

- Movement_Set_Flying_Height(height)
  มีหน้าที่ใช้สำหรับกำหนดค่าความสูงในการบิน
- Movement_ConstForward_Set_Value(ForwardSpeed)
  มีหน้าที่ใช้สำหรับกำหนดค่าความเร็วในการเคลื่อนที่ไปข้างหน้า
- Movement_ConstForward_Start(ForwardSpeed)
  มีหน้าที่ใช้สำหรับสั่งเริ่มต้นการบินไปข้างหน้าด้วยความเร็วคงที่
- Movement_ConstForward_End()
  มีหน้าที่ใช้สำหรับสั่งหยุดการบินไปข้างหน้าด้วยความเร็วคงที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Movement_LockHeading_Start(angle_from_north)
  มีหน้าที่ใช้สำหรับสั่งเริ่มต้นให้ล็อกทิศทางการหันหัวของเครื่องบิน
- Movement_LockHeading_End()
  มีหน้าที่ใช้สำหรับสั่งหยุดการล็อกทิศทางการหันหัวของเครื่องบิน
- Movement_LockHeading_Set_Degree(angle_from_north)
  มีหน้าที่ใช้สำหรับกำหนดมุมที่ใช้ในการล็อกทิศทางการหันหัวของเครื่องบิน
- Movement_LockHeading2Target_Start()
  มีหน้าที่ใช้สำหรับสั่งเริ่มการล็อกทิศทางการหันหัวของเครื่องบินไปยังตำแหน่งเป้าหมาย
- Movement_LockHeading2Target_End()
  มีหน้าที่ใช้สำหรับสั่งหยุดการล็อกทิศทางการหันหัวของเครื่องบินไปยังตำแหน่งเป้าหมาย
- Movement_LockHeading2Target_Set_Target(target_lat,target_lon)
  มีหน้าที่ใช้สำหรับกำหนดเป้าหมายที่ต้องการจะหันหัวของเครื่องบินไปหา
- Movement_Baro_LockAlt_Start(height)
  มีหน้าที่สั่งเริ่มต้นการล็อกระดับความสูง โดยอ้างอิงจากเซ็นเซอร์วัดความสูงแบบวัดความดัน
- Movement_Baro_LockAlt_End()
  มีหน้าที่สั่งหยุดการล็อกระดับความสูง โดยอ้างอิงจากเซ็นเซอร์วัดความสูงแบบวัดความดัน
- Movement_Baro_LockAlt_Set_Meter(height)
  มีหน้าที่กำหนดระดับความสูงที่ต้องการให้ล็อก โดยอ้างอิงจากเซ็นเซอร์วัดความสูงแบบวัดความดัน
- Movement_GPS_LockAlt_Start(height)
  มีหน้าที่สั่งเริ่มต้นการล็อกระดับความสูง โดยอ้างอิงจากความสูงที่ได้จากจีพีเอส
- Movement_GPS_LockAlt_End()
  มีหน้าที่สั่งหยุดการล็อกระดับความสูง โดยอ้างอิงจากความสูงที่ได้จากจีพีเอส
- Movement_GPS_LockAlt_Set_Meter(height)
  มีหน้าที่กำหนดระดับความสูงที่ต้องการให้ล็อก โดยอ้างอิงจากความสูงที่ได้จากจีพีเอส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Movement_Go2Target_Set_Target(target_lat,target_lon)
  มีหน้าที่กำหนดเป้าหมายที่จะให้เคลื่อนที่ไป
- Movement_Go2Target_Start(target_lat,target_lon)
  มีหน้าที่สั่งเริ่มต้นการบินไปยังเป้าหมายที่กำหนดไว้
- Movement_Go2Target_End()
  มีหน้าที่สั่งหยุดการบินไปยังเป้าหมายที่กำหนดไว้
- Movement_Set_Home(home_lat,home_lon)
  มีหน้าที่กำหนดตำแหน่งของ ฐานออกบิน
- Movement_Go2Home_Start()
  มีหน้าที่สั่งเริ่มต้นการบินกลับฐานออกบิน
- Movement_Go2Home_End()
  มีหน้าที่สั่งหยุดการบินกลับฐานออกบิน
- Movement_TakeOff_Start()
  มีหน้าที่สั่งเริ่มต้นการขึ้นบิน
- Movement_TakeOff_End()
  มีหน้าที่สั่งหยุดการขึ้นบิน
- Movement_Landing_Start()
  มีหน้าที่สั่งเริ่มต้นการลงจอด
- Movement_Landing_End()
  มีหน้าที่สั่งหยุดการลงจอด
- Movement_Adj2Target_Start(target_lat,target_lon)
  มีหน้าที่สั่งเริ่มต้นการปรับแบบละเอียดเพื่อเข้าสู้เป้าหมาย
- Movement_Adj2Target_End()
  มีหน้าที่สั่งหยุดการปรับแบบละเอียดเพื่อเข้าสู้เป้าหมาย
- Movement_Lock2Symbol_Start()
  มีหน้าที่สั่งเริ่มต้นการล็อกตำแหน่งกับสัญลักษณ์ที่ใช้ในการลงจอด
- Movement_Lock2Symbol_End()
  มีหน้าที่สั่งหยุดการล็อกตำแหน่งกับสัญลักษณ์ที่ใช้ในการลงจอด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เงื่อนไขการเปลี่ยนสถานะทั้งหมด 7 เงื่อนไข เพื่อไว้ใช้เป็นเงื่อนไขในการเปลี่ยนสถานะ

- Movement_LockHeading_Arrive_Stat()
  เป็นสถานะเพื่อตรวจสอบการล็อกทิศทางการหันหัวของเครื่องบิน
- Movement_LockHeading2Target_Arrive_Stat()
  เป็นสถานะเพื่อตรวจสอบการล็อกทิศทางการหันหัวของเครื่องบินไปยังเป้าหมาย
- Movement_Go2Target_Arrive_Stat()
  เป็นสถานะเพื่อตรวจสอบการเคลื่อนที่ถึงเป้าหมาย
- Movement_Go2Home_Arrive_Stat()
  เป็นสถานะเพื่อตรวจสอบการเคลื่อนที่ถึงฐานออกบิน
- Movement_TakeOff_IsFinish()
  เป็นสถานะเพื่อตรวจสอบการขึ้นบิน
- Movement_Adj2Target_Arrive_Stat()
  เป็นสถานะเพื่อตรวจสอบการเคลื่อนที่แบบละเอียดไปยังเป้าหมาย
- Movement_Lock2Symbol_Arrive_Stat()
  เป็นสถานะเพื่อตรวจสอบการล็อกตำแหน่งกับสัญลักษณ์ลงจอด

โดยสามารถนำฟังชั่นและเงื่อนไขต่างๆเหล่านี้ มาประกอบกันเป็นโปรแกรมที่ต้องการ ผ่านตัวจัดการการเปลี่ยนสถานะ

3.1.8 การออกแบบการควบคุมให้เครื่องบินหันหน้าไปยังเป้าหมายตลอดเวลา

ในการที่ทำให้เครื่องบินสามารถหันหน้าไปยังเป้าหมายตลอดเวลานั้น จะนำมุมที่เครื่องบินหันอยู่ปัจจุบันนั้น ลบด้วยค่ามุมแบริ่งระหว่างที่อยู่ปัจจุบันและของเป้าหมาย จะได้ส่วนต่างมุมที่ต้องหันออกมา แล้วนำไปใช้เป็นค่าผิดพลาดป้อนกลับให้กับสมการควบคุม (P Controller) ดังสมการ

$$output = k_p \times (heading - bearing) \tag{3.1}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดย

output คือ ค่าเอาต์พุต ที่จะนำไปใช้ในการขับมอเตอร์ในพารามิเตอร์การหมุนของเครื่องบิน

k_p คือ ค่าคงที่ของการลู่เข้าของสมการควบคุม

heading คือ ค่ามุมที่เครื่องบินหันหน้าอยู่ปัจจุบัน

bearing คือ ค่ามุมแบริ่งระหว่างที่อยู่เครื่องบินปัจจุบันไปยังเป้าหมาย

3.1.9 การออกแบบการควบคุมให้เครื่องบิน บินไปยังเป้าหมายได้โดยอัตโนมัติ

ในการที่ทำให้เครื่องบินสามารถบินจากจุดปัจจุบันไปยังเป้าหมายได้โดยอัตโนมัตินั้น มีขั้นตอนอยู่ทั้งหมด 4 ขั้นตอนคือ

1) ให้เครื่องบินเคลื่อนที่ไปข้างหน้าด้วยความเร็วสูงสุด

2) ให้เครื่องบินพยายามหันหน้าไปยังเป้าหมายตลอดเวลา

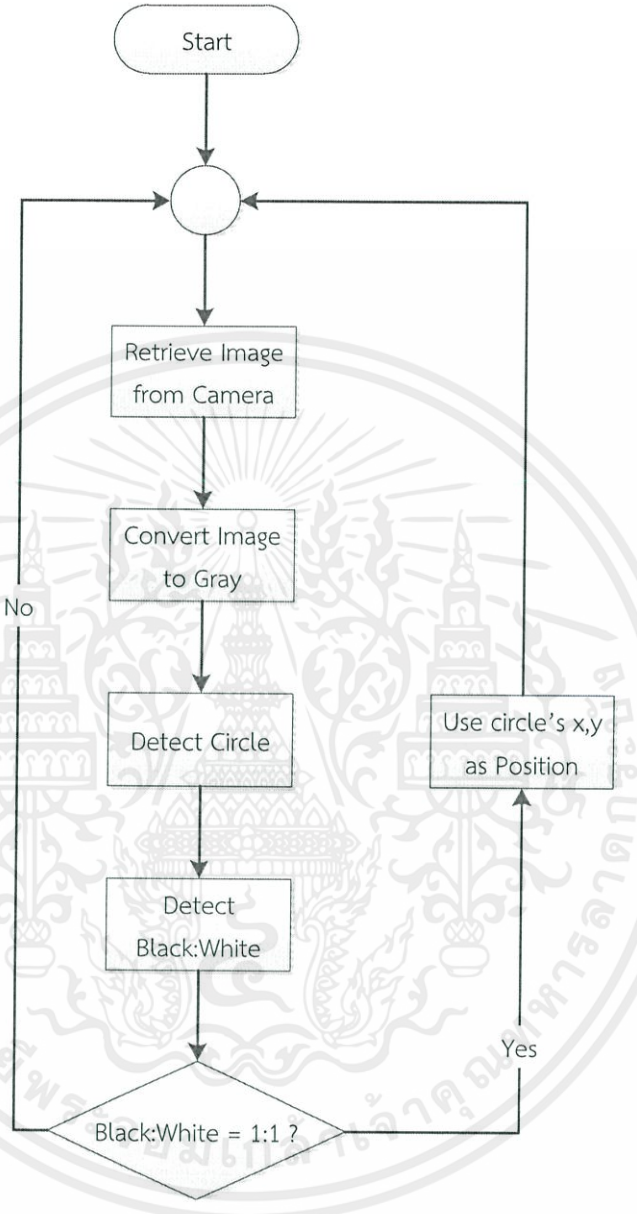
3) เมื่อเข้าใกล้เป้าหมายให้ลดความเร็วลง และค่อยๆบินเข้าสู่เป้าหมายอย่างช้าๆ

4) เมื่อเข้าใกล้เป้าหมายมากเพียงพอที่จะถือว่าถึงเป้าหมายแล้ว ให้เครื่องบินทำการหยุดการเคลื่อนที่ไปข้างหน้า

3.1.10 การออกแบบการตรวจจับสัญญาณที่ใช้ในการลงจอด

สัญลักษณ์ที่ใช้ในการลงจอด จะมีลักษณ์เป็นรูปวงกลมที่ข้างในวงกลมจะมีสีขาวครึ่งหนึ่ง และสีดำครึ่งหนึ่ง การตรวจจับจะใช้การตรวจหาวงกลมในภาพ แล้วนำวงกลมแต่ละวงที่ตรวจพบไปตรวจสอบว่า ข้างในวงกลมนั้น มีอัตราส่วนสีขาวต่อสีดำเท่ากันหรือไม่ ถ้าใช่ก็ถือว่าวงกลมวงนั้นเป็นสัญลักษณ์ที่ระบบต้องการ แต่ถ้าหากมีอัตราส่วนสีดำต่อสีขาวไม่เท่ากัน ก็แสดงว่าวงกลมวงนั้นที่ตรวจพบ ไม่ใช่สัญลักษณ์ที่ระบบต้องการ ก็จะไม่สนใจวงกลมวงนั้น แสดงโฟลว์ชาร์ตดังรูปที่ 3.10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.10 โฟลว์ชาร์ตของระบบตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมส่วนประมวลผลทางภาพเพื่อตรวจจับสัญลักษณ์ในการลงจอด	
src = cvQueryFrame(capture);	- รับภาพมาจากกล้อง
cvtColor(src, src_gray, CV_BGR2GRAY);	- แปลงโมเดลสีจาก BGR เป็น GrayScale
GaussianBlur(src_gray, src_gray, Size(3, 3), 2, 2);	- ทำการเบลอ เพื่อตัดสัญญาณรบกวนความถี่สูงออก
vector<Vec3f> circles;	- สร้างตัวแปรสำหรับเก็บข้อมูลของวงกลมที่ตรวจพบ
HoughCircles(src_gray, circles, CV_HOUGH_GRADIENT, 1, src_gray.rows/8, 200, 50, 0, 0);	- ทำการตรวจหาวงกลมในภาพด้วยวิธี Hough Transform
for(size_t i = 0; i < circles.size(); i++) {	- วนลูปเพื่อรับค่าพารามิเตอร์ของวงกลมทั้งหมดจากผลของ HoughCircles
Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));	- สร้างตัวแปรชนิดจุด เพื่อเก็บจุดศูนย์กลางของวงกลมแต่ละวง
int radius = cvRound(circles[i][2]);	- สร้างตัวแปรชนิดจำนวนเต็ม เพื่อเก็บรัศมีของวงกลมแต่ละวง
circle(src, center, 3, Scalar(0,255,0), -1, 8, 0);	- วาดวงกลมสีเขียวเป็นจุดเล็กๆ เพื่อบอกถึงจุดศูนย์กลางของวงกลม
circle(src, center, radius, Scalar(0,255,0), 3, 8, 0);	- วาดวงกลมสีเขียว ล้อมรอบวงกลมที่ตรวจพบทุกวง
if(IsHalf(src_gray,center,radius,0.35,0.65)) {	- ถ้าวงกลมใดมีอัตราส่วนระหว่างสีขาวต่อสีดำประมาณ 1 : 1
circle(src, center, radius, Scalar(0,0,255), 3, 8, 0);	- วาดวงกลมสีแดงล้อมรอบวงกลมที่มีอัตรส่วนระหว่างสีขาวต่อสีดำประมาณ 1 : 1
cout << "X: " << center.x << " Y: " << center.y << "\r\n"; }	- แสดงค่าตำแหน่งของจุดศูนย์กลางของวงกลมที่มีอัตราส่วนระหว่างสีขาวต่อสีดำประมาณ 1 : 1
imshow("Output",src);	- แสดงผลภาพที่ผ่านการระบุตำแหน่งวงกลมแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 เครื่องมือที่ใช้ในการทดลอง

- คอมพิวเตอร์
- ออสซิลโลสโคป
- กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์
- บอร์ดอาร์ดูไพล็อต
- รีโมตพร้อมตัวรับ ฟุตาบ้า รุ่น T6J
- บอร์ดมัลติวีเอสอี เวอร์ชั่น 2.5

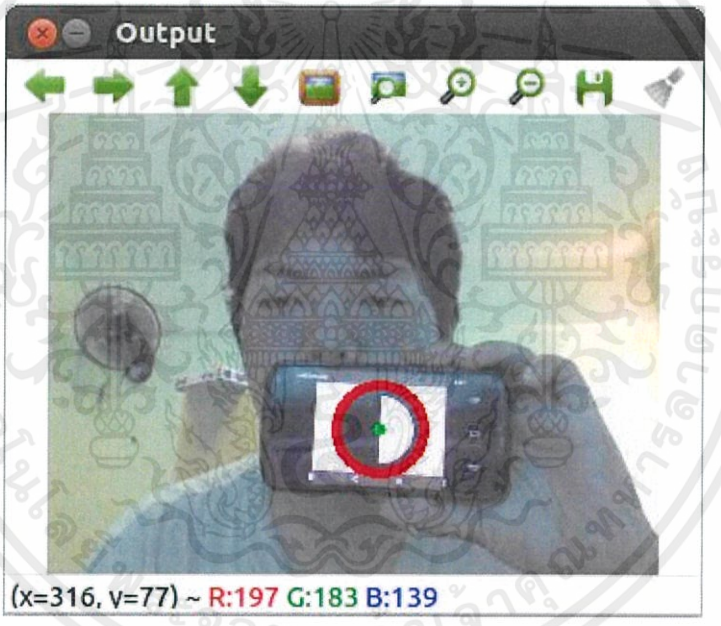
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

ผลการทดลอง

4.1 การทดลองการทำงานของระบบตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด

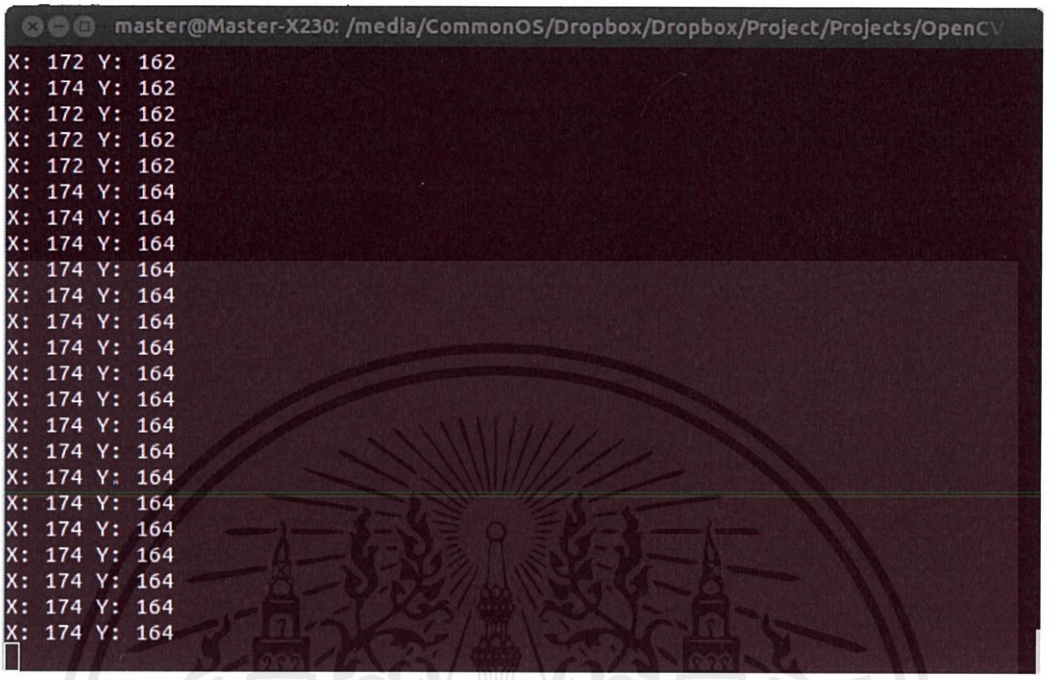
จากการออกแบบโปรแกรมดังโฟลว์ชาร์ตดังรูปที่ 3.9 นั้น ผลลัพธ์จากการทดลองสามารถแสดงได้ดังรูปที่ 4.1 และรูปที่ 4.2



รูปที่ 4.1 ภาพเอาต์พุตที่ผ่านการตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด

จากรูปที่ 4.1 เป็นผลลัพธ์ของโปรแกรมตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด โดยจะเห็นได้ว่า จะเกิดวงกลมสีแดงล้อมรอบสัญลักษณ์ที่ตรวจจับได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



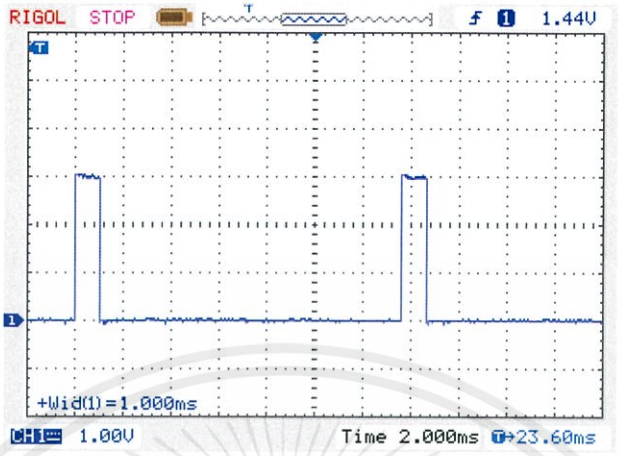
รูปที่ 4.2 ค่าเอาท์พุตของตำแหน่งสัญลักษณ์ที่ใช้ในการลงจอด

จากรูปที่ 4.2 เป็นค่าตำแหน่งในระนาบ X-Yของจุดศูนย์กลางของสัญลักณ์ที่ตรวจพบ

4.2 การทดลองการทำงานของระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุที่

ในการตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุตามบล็อกไดอะแกรมดังรูปที่ 3.3 โดยสัญญาณอินพุตของระบบจะเป็นดังรูปที่ 4.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



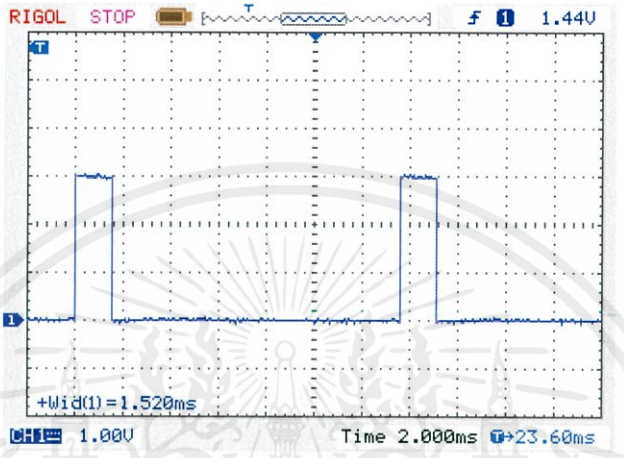
รูปที่ 4.3 สัญญาณอินพุตที่เข้ามาในระบบ ที่ได้มาจากตัวรับสัญญาณของรีโมตเมื่อปรับจอยสติกของรีโมตไปยังค่าต่ำสุด

จากสัญญาณที่เข้ามาดังรูปที่ 4.3 จะได้เอาต์พุตเป็นค่าที่ระบบตรวจจับของเราอ่านได้ในหน่วยของ ไมโครวินาที ดังรูปที่ 4.4

รูปที่ 4.4 เอาต์พุตที่เป็นค่าที่ระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ ตรวจจับได้จากสัญญาณที่มีความกว้างพัลส์ 1 มิลลิวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากนั้นทดลองป้อนสัญญาณที่ความกว้างพัลส์เท่ากับ 1.5 มิลลิวินาที ดังรูปที่ 4.5 จะได้เอาต์พุตเป็นค่าที่ระบบตรวจจับของเราอ่านได้ในหน่วยของ ไมโครวินาที ดังรูปที่ 4.6

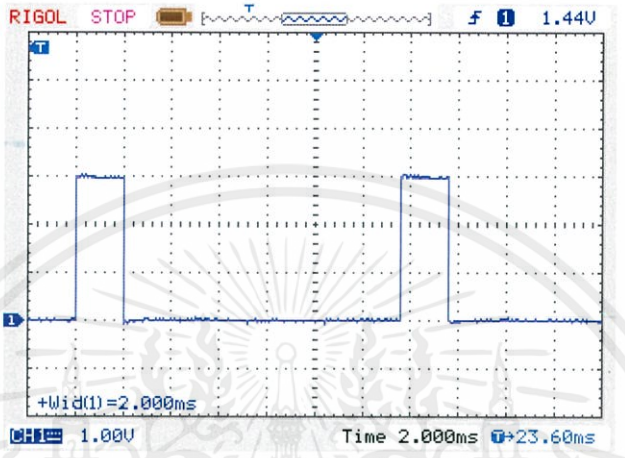


รูปที่ 4.5 สัญญาณอินพุตที่เข้ามาในระบบ ที่ได้มาจากตัวรับสัญญาณของรีโมตเมื่อปรับจอยสติกของรีโมตไปยังค่ากึ่งกลาง

รูปที่ 4.6 เอาต์พุตที่เป็นค่าที่ระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุตรวจจับได้ จากสัญญาณที่มีความกว้างพัลส์ 1.5 มิลลิวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังกจากนั้นทดลองป้อนสัญญาณที่ความกว้างพัลส์เท่ากับ 2 มิลลิวินาที ดังรูปที่ 4.7 จะได้เอาต์พุตเป็นค่าที่ระบบตรวจจับของเราอ่านได้ในหน่วยของ ไมโครวินาที ดังรูปที่ 4.8



รูปที่ 4.7 สัญญาณอินพุตที่เข้ามาในระบบ ที่ได้มาจากตัวรับสัญญาณของรีโมตเมื่อปรับจอยสติกของรีโมตไปยังค่าสูงสุด

ปที่ 4.8 เอาต์พุตที่เป็นค่าที่ระบบตรวจจับสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ ตรวจจับได้ จากสัญญาณที่มีความกว้างพัลส์ 2 มิลลิวินาที

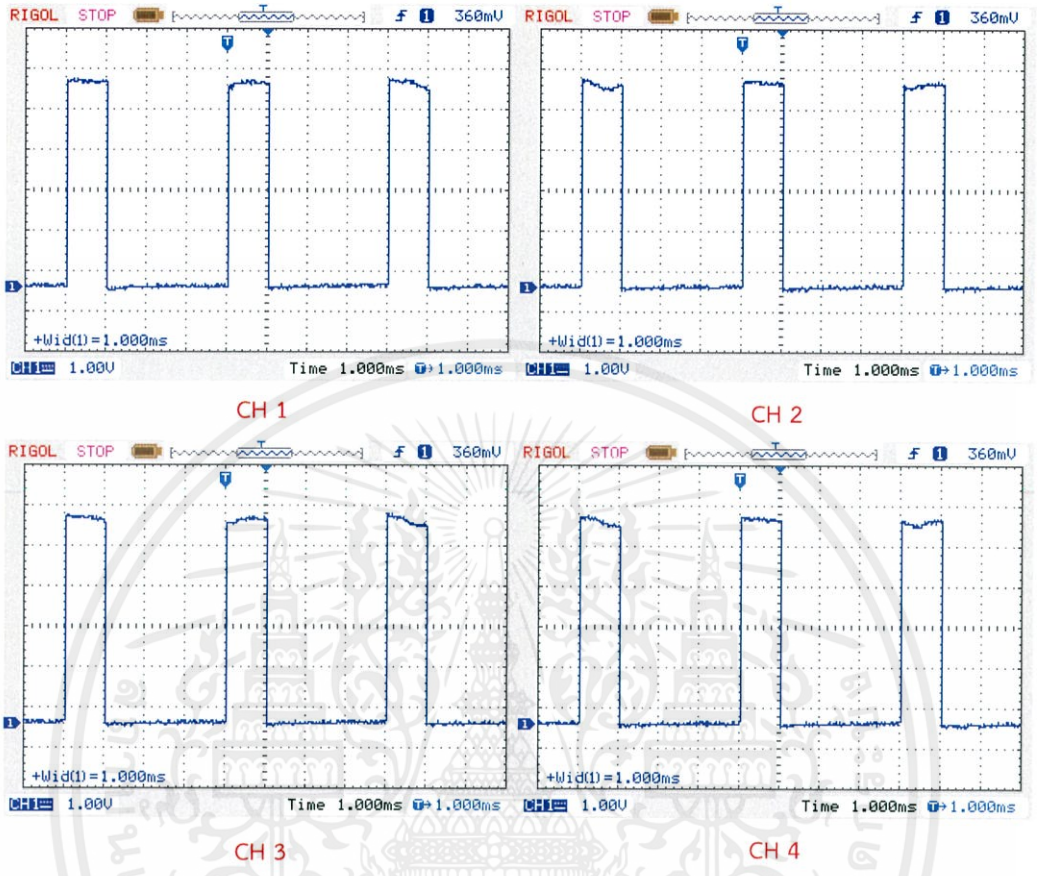
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 การทดลองการทำงานของระบบสร้างสัญญาณเพื่อควบคุมกล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์

ในการสร้างสัญญาณควบคุมที่ใช้ในอุปกรณ์บังคับวิทยุ โดยการเขียนโปรแกรมตามบล็อกไดอะแกรมดังรูปที่ 3.4 นำไปทดลองป้อนเป็นสัญญาณควบคุมของกล่องควบคุม นาซ่า เอ็ม ไลท์ จำนวน 4 ช่องสัญญาณ โดยป้อนจากช่องสัญญาณที่ 1 ถึงช่องสัญญาณที่ 4 เข้าไปที่ช่องสัญญาณควบคุมการเอียงทางซ้าย-ขวา ช่องสัญญาณควบคุมการเอียงข้างหน้า-หลัง ช่องสัญญาณควบคุมการขึ้น-ลง และช่องสัญญาณควบคุมการหมุนตัว ตามลำดับ

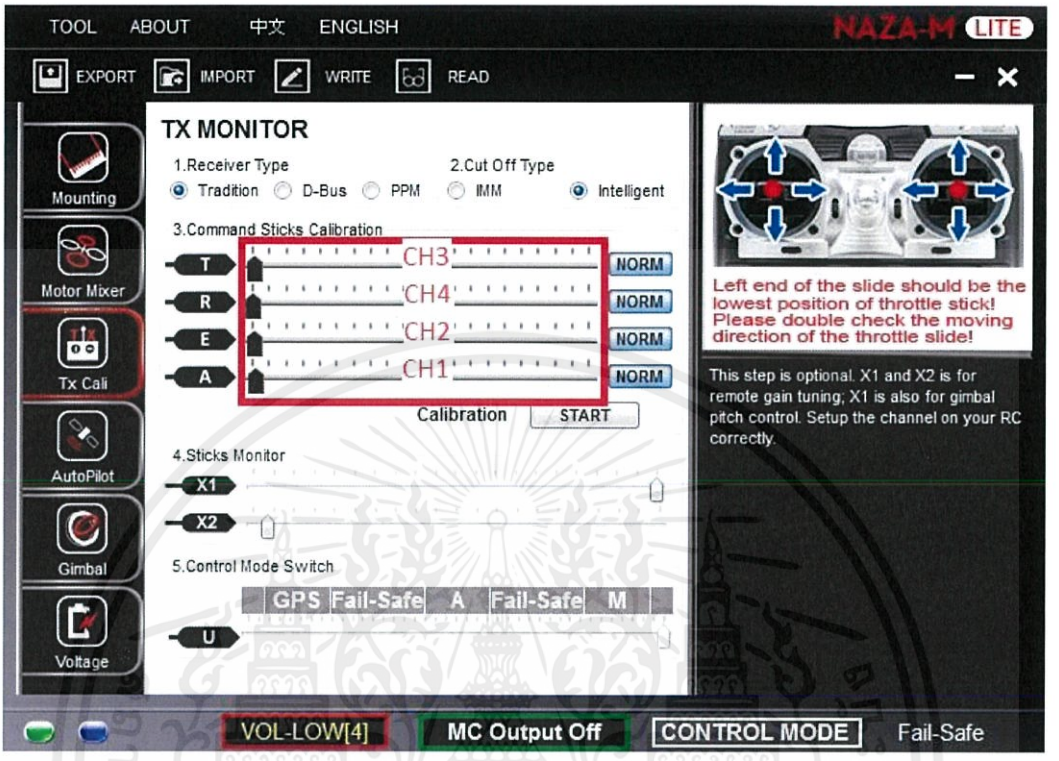
เมื่อป้อนช่องสัญญาณที่ 1 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 1 มิลลิวินาที ดังรูปที่ 4.9 จึงได้เอาต์พุตที่อ่านได้จากโปรแกรมที่ใช้สำหรับตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ ดังรูปที่ 4.10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.9 สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 (CH1)เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 (CH3) เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 (CH4)
เป็น 1 มิลลิวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

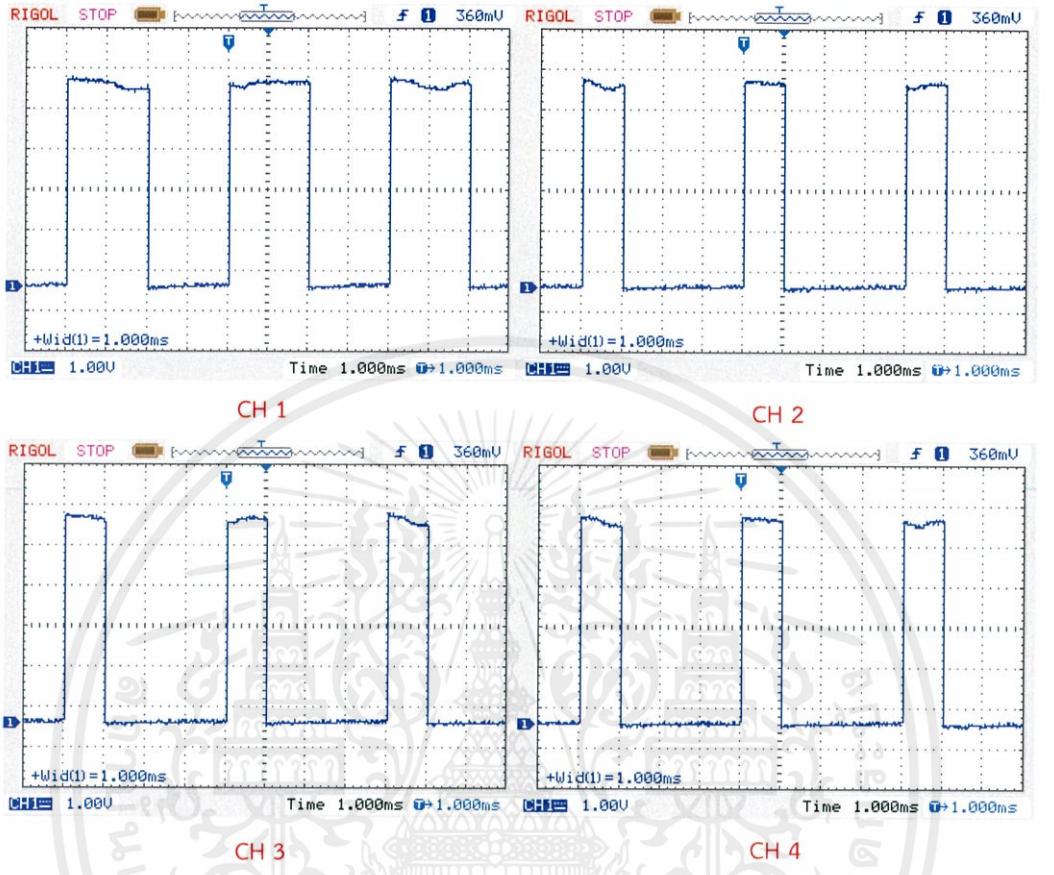


รูปที่ 4.10 หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาณที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที

จะเห็นได้ว่า แถวบาร์ทุกช่องสัญญาณมีค่าเป็นต่ำสุดหมดทุกอัน เนื่องจากว่า สัญญาณที่เราป้อนเข้ามานั้น มีค่าความกว้างพัลส์เป็น 1 มิลลิวินาทีทุกอัน

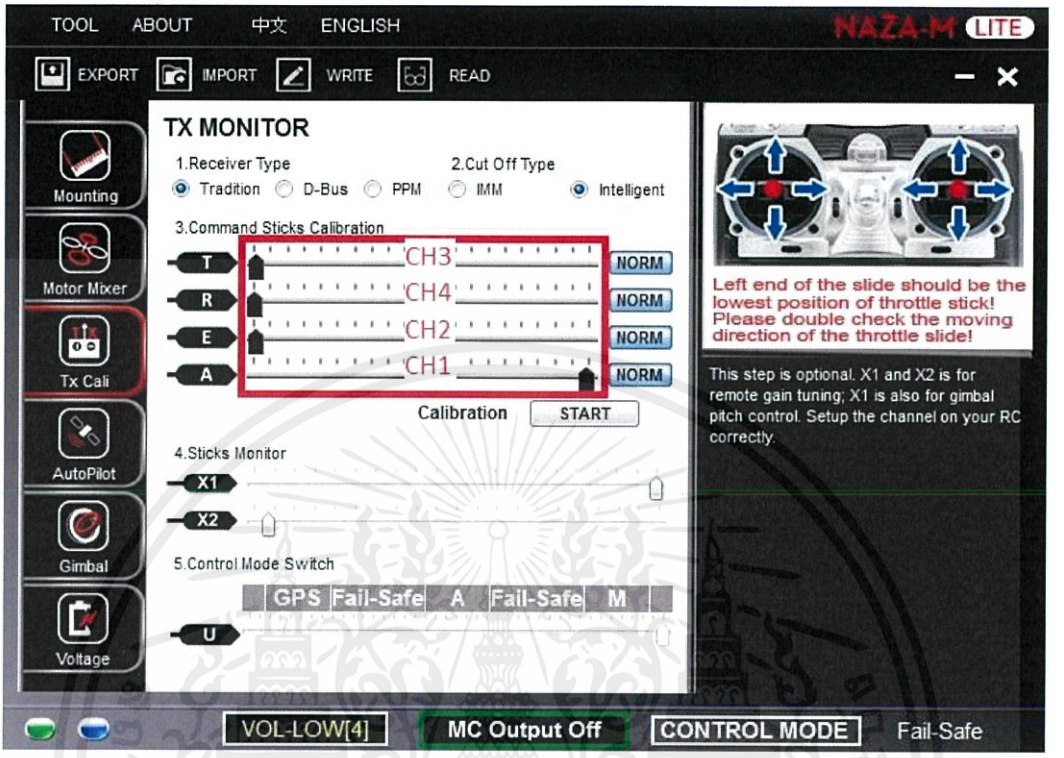
เมื่อป้อนช่องสัญญาณที่ 1 (CH1) เป็น 2 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 (CH3) เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 (CH4) เป็น 1 มิลลิวินาที ดังรูปที่ 4.11 จึงได้เอาต์พุตที่อ่านได้จากโปรแกรมที่ใช้สำหรับตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ ดังรูปที่ 4.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.11 สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 (CH1) เป็น 2 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 (CH3) เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 (CH4) เป็น 1 มิลลิวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

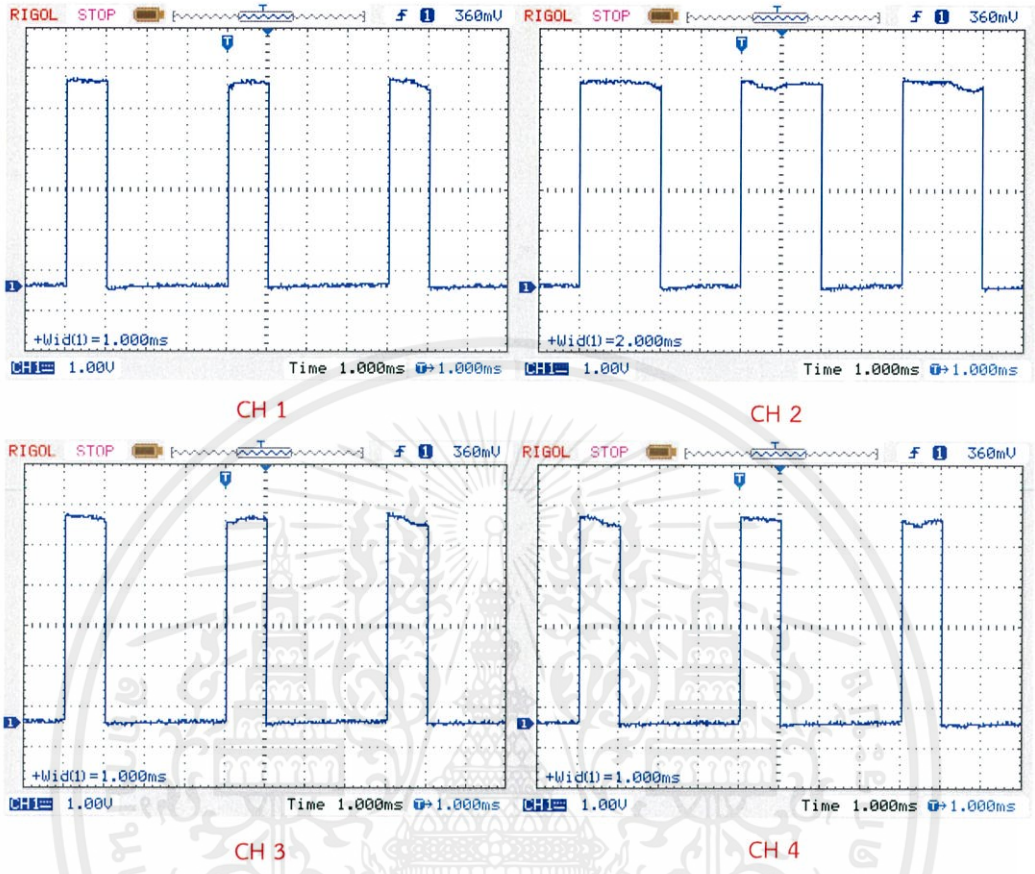


รูปที่ 4.12 หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาณที่เข้ามาช่องสัญญาณ เอ (A) เป็น 2 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที

จะเห็นได้ว่า แถวบาร์ของช่องสัญญาณที่ 1 จะมีค่าสูงสุด เนื่องจากว่า มีเพียงช่องสัญญาณที่ 1 เท่านั้น ที่ถูกจ่ายด้วยสัญญาณที่มีความกว้างพัลส์ 2 มิลลิวินาที นอกจากนั้นถูกจ่ายด้วยสัญญาณที่ความกว้างพัลส์ 1 มิลลิวินาที ทั้งหมด

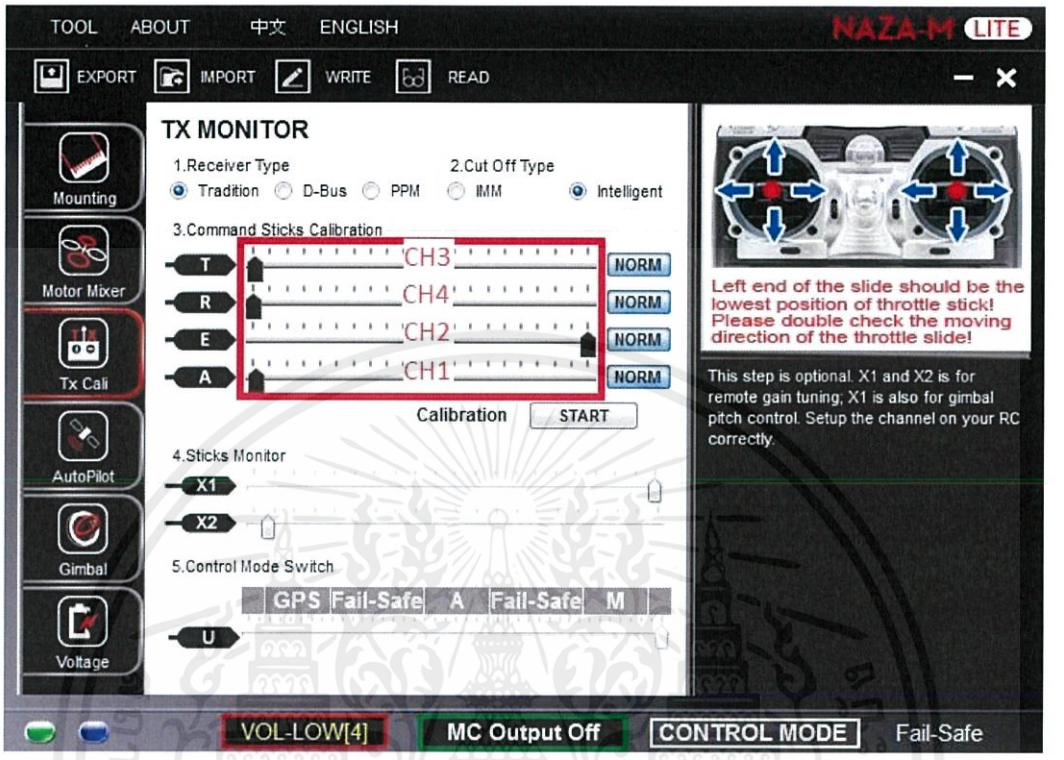
เมื่อป้อนช่องสัญญาณที่ 1 (CH1) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 2 มิลลิวินาที ช่องสัญญาณที่ 3 (CH3) เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 (CH4) เป็น 1 มิลลิวินาที ดังรูปที่ 4.13 จึงได้เอาต์พุตที่อ่านได้จากโปรแกรมที่ใช้สำหรับตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ ดังรูปที่ 4.14

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.13 สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 (CH1) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 2 มิลลิวินาที ช่องสัญญาณที่ 3 (CH3) เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 (CH4) เป็น 1 มิลลิวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

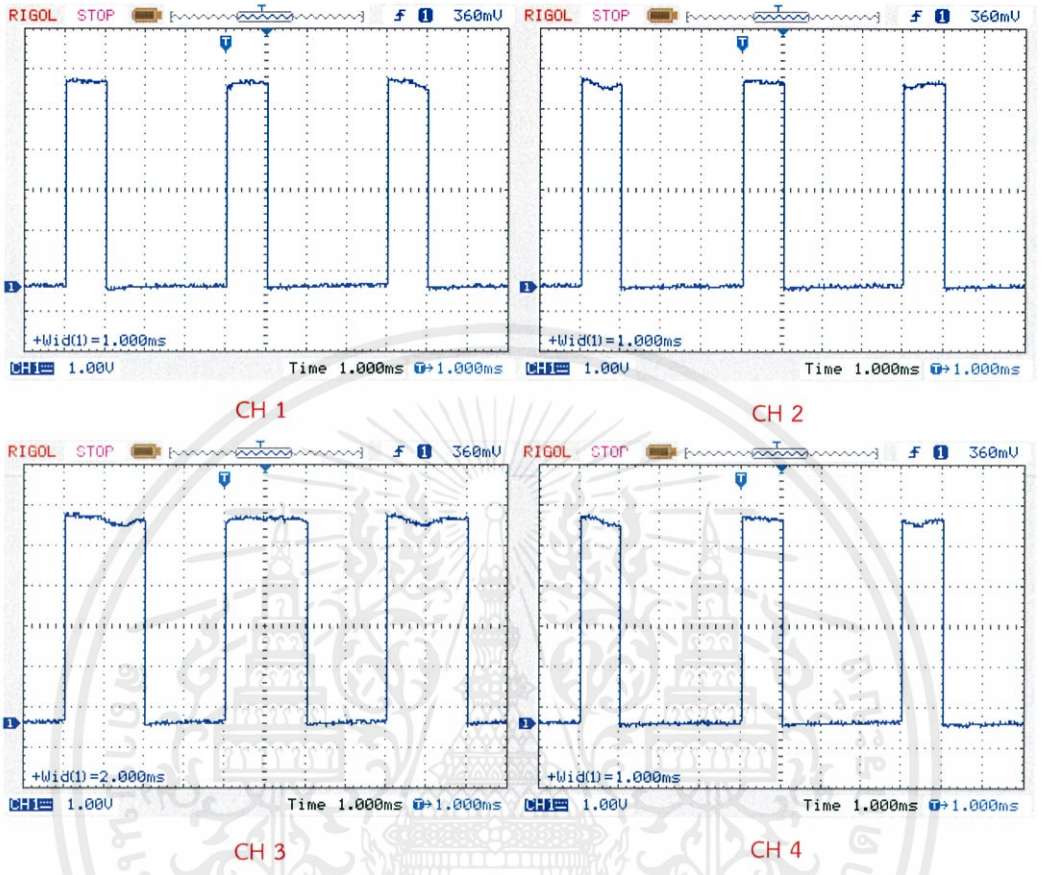


รูปที่ 4.14 หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาณที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 2 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที

จะเห็นได้ว่า แถวบาร์ของช่องสัญญาณที่ 2 จะมีค่าสูงสุด เนื่องจากว่า มีเพียงช่องสัญญาณที่ 1 เท่านั้น ที่ถูกจ่ายด้วยสัญญาณที่มีความกว้างพัลส์ 2 มิลลิวินาที นอกจากนั้นถูกจ่ายด้วยสัญญาณที่ความกว้างพัลส์ 1 มิลลิวินาที ทั้งหมด

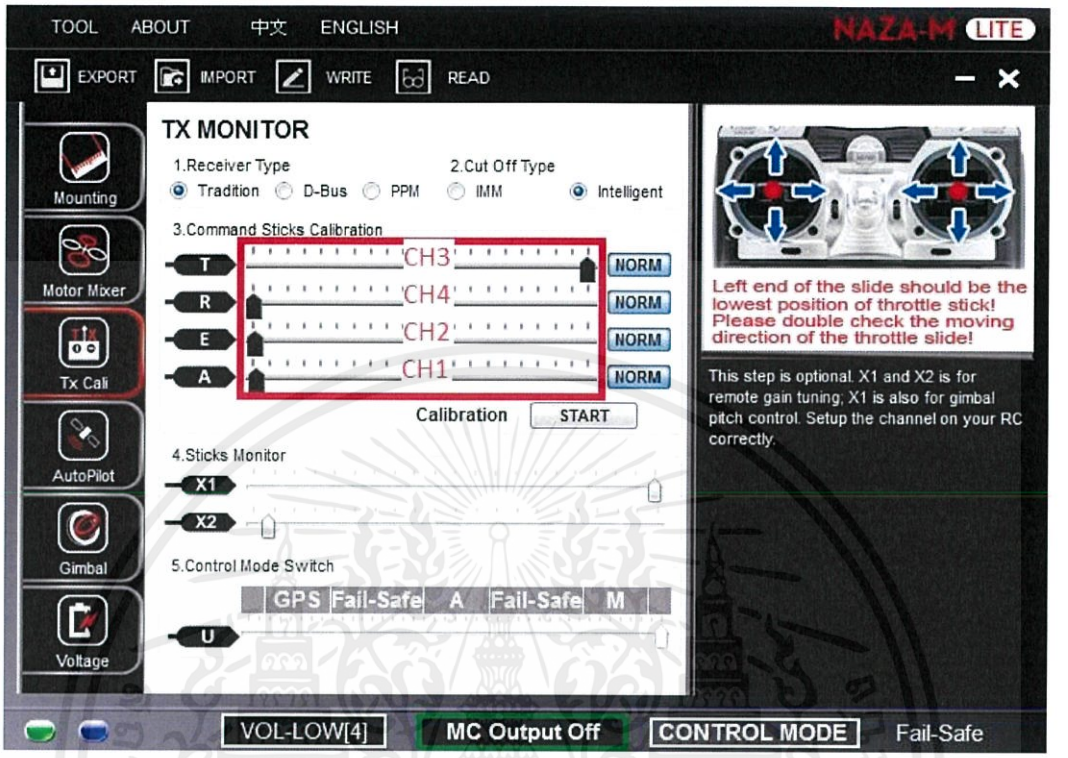
เมื่อป้อนช่องสัญญาณที่ 1 (CH1) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 (CH3) เป็น 2 มิลลิวินาที และช่องสัญญาณที่ 4 (CH4) เป็น 1 มิลลิวินาที ดังรูปที่ 4.15 จึงได้เอาต์พุตที่อ่านได้จากโปรแกรมที่ใช้สำหรับตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ ดังรูปที่ 4.16

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.15 สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 (CH1) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 (CH3) เป็น 2 มิลลิวินาที และช่องสัญญาณที่ 4 (CH4) เป็น 1 มิลลิวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

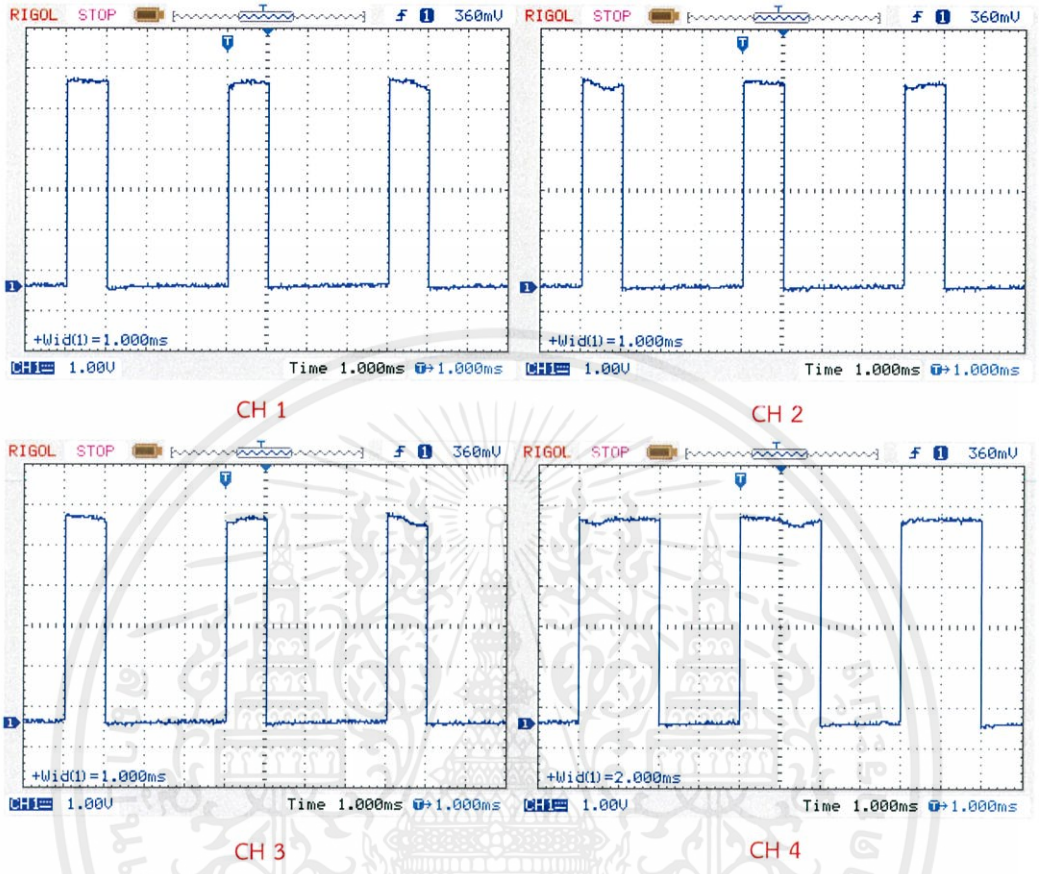


รูปที่ 4.16 หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ เมื่อสัญญาณที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 2 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 1 มิลลิวินาที

จะเห็นได้ว่า แถวบาร์ของช่องสัญญาณที่ 3 จะมีค่าสูงสุด เนื่องจากว่า มีเพียงช่องสัญญาณที่ 1 เท่านั้น ที่ถูกจ่ายด้วยสัญญาณที่มีความกว้างพัลส์ 2 มิลลิวินาที นอกจากนั้นถูกจ่ายด้วยสัญญาณที่ความกว้างพัลส์ 1 มิลลิวินาที ทั้งหมด

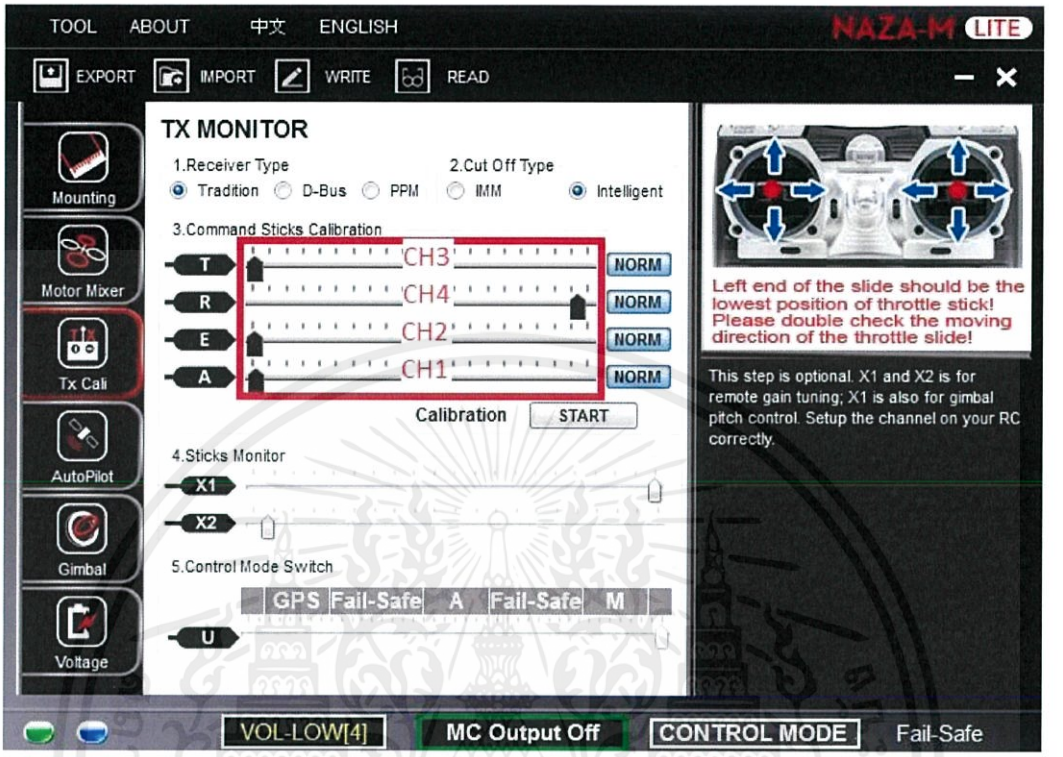
เมื่อป้อนช่องสัญญาณที่ 1 (CH1) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 (CH2) เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 2 มิลลิวินาที ดังรูปที่ 4.17 จึงได้เอาต์พุตที่อ่านได้จากโปรแกรมที่ใช้สำหรับตั้งค่ากล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ ดังรูปที่ 4.18

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.17 สัญญาณที่ป้อนเข้าไปให้กล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ โดยที่ให้ช่องสัญญาณที่ 1 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 2 เป็น 1 มิลลิวินาที ช่องสัญญาณที่ 3 เป็น 1 มิลลิวินาที และช่องสัญญาณที่ 4 เป็น 2 มิลลิวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



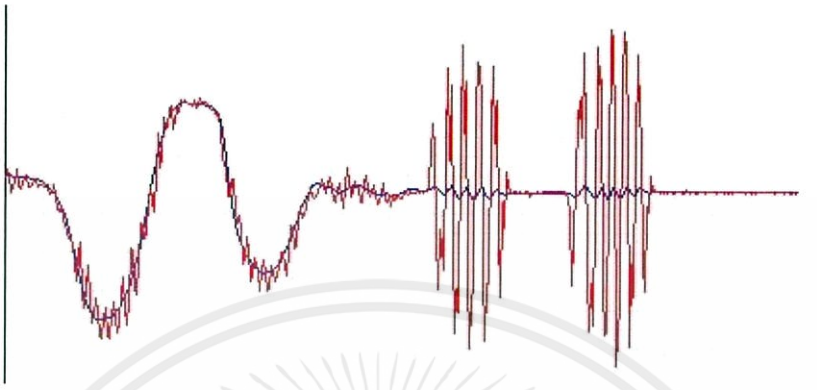
รูปที่ 4.18 หน้าต่างของโปรแกรมตั้งค่ากล่องควบคุมสมดุล นาซา เอ็ม ไลท์ เมื่อสัญญาณที่เข้ามาช่องสัญญาณ เอ (A) เป็น 1 มิลลิวินาที ช่องสัญญาณ อี (E) เป็น 1 มิลลิวินาที ช่องสัญญาณ ที (T) เป็น 1 มิลลิวินาที และช่องสัญญาณ อาร์ (R) เป็น 2 มิลลิวินาที

จะเห็นได้ว่า แถวบาร์ของช่องสัญญาณที่ 4 จะมีค่าสูงสุด เนื่องจากว่า มีเพียงช่องสัญญาณที่ 1 เท่านั้น ที่ถูกจ่ายด้วยสัญญาณที่มีความกว้างพัลส์ 2 มิลลิวินาที นอกจากนั้นถูกจ่ายด้วยสัญญาณที่ความกว้างพัลส์ 1 มิลลิวินาที ทั้งหมด

4.4 การทดลองการทำงานของระบบการกรองสัญญาณแบบคอมพลีเมนทารี่

ในการวัดมุมความเอียงกับพื้นโลกที่จะได้ค่าที่มีความผิดพลาดน้อยนั้น จะต้องอาศัยค่าวงจรกรองสัญญาณแบบคอมพลีเมนทารี่ โดยการเขียนโปรแกรมตามบล็อกไดอะแกรมดังรูปที่ 3.7 โดยนำค่าที่จากเซ็นเซอร์วัดความเร่งและเซ็นเซอร์วัดความเอียงมาคำนวณผ่านวงจรกรองแบบคอมพลีเมนทารี่ ซึ่งได้ผลการเปรียบเทียบระหว่างค่ามุมความเอียงที่ไม่ผ่านวงจรกรองแบบคอมพลีเมนทารี่ และค่ามุมความเอียงที่ผ่านวงจรกรองแบบคอมพลีเมนทารี่ ดังรูปที่ 4.19

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



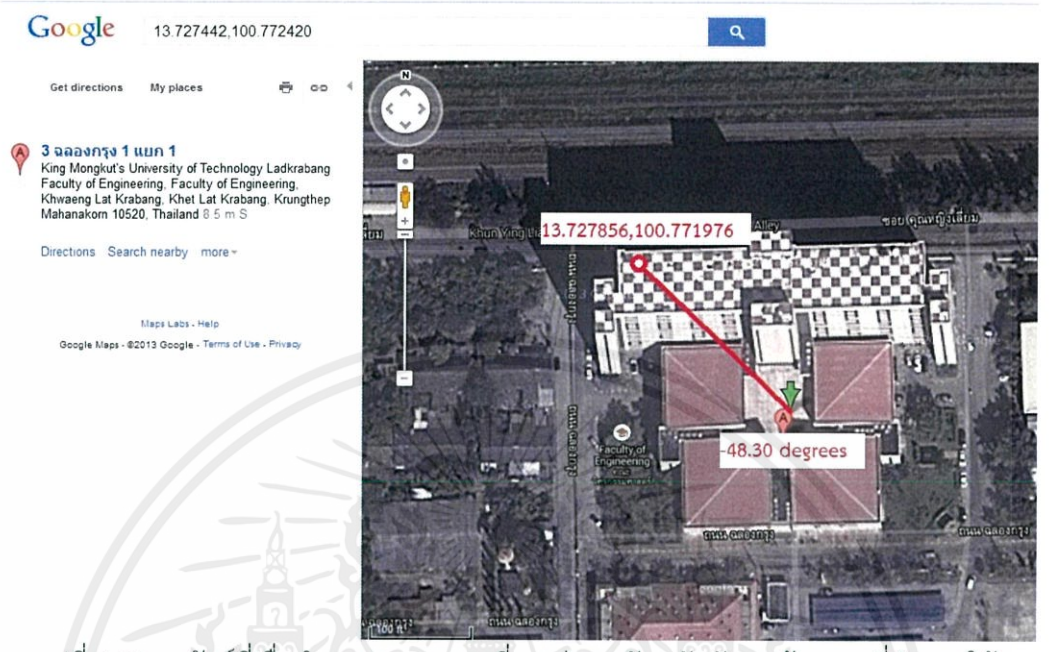
รูปที่ 4.19 กราฟค่ามุมความเอียงที่ได้จากการคำนวณค่าในเซ็นเซอร์วัดความเร่งและเซ็นเซอร์-วัดความเร็วเชิงมุม โดยกราฟสีแดงจะเป็นกราฟของค่ามุมความเอียงที่ไม่ได้ผ่านวงจรกรองแบบคอมพลีเมนทารี่ และกราฟสีน้ำเงินเป็นกราฟของค่ามุมความเอียงที่ผ่านวงจรกรองแบบคอมพลีเมนทารี่

จะสังเกตได้ว่า ที่สภาวะที่ไม่เกิดการสั่นสะเทือนมาก ค่ามุมความเอียงที่ผ่านและไม่ผ่านวงจรกรองแบบคอมพลีเมนทารี่ จะมีค่าค่อนข้างใกล้เคียงกัน แต่เมื่อเกิดการสั่นสะเทือน ค่ามุมความเอียงที่ไม่ผ่านวงจรกรองแบบคอมพลีเมนทารี่ จะเกิดการสะบัดอย่างรุนแรงเกิดขึ้น ในขณะเดียวกัน ค่ามุมความเอียงที่ผ่านวงจรกรองแบบคอมพลีเมนทารี่ จะเกิดการสะบัดเพียงเล็กน้อยเท่านั้น

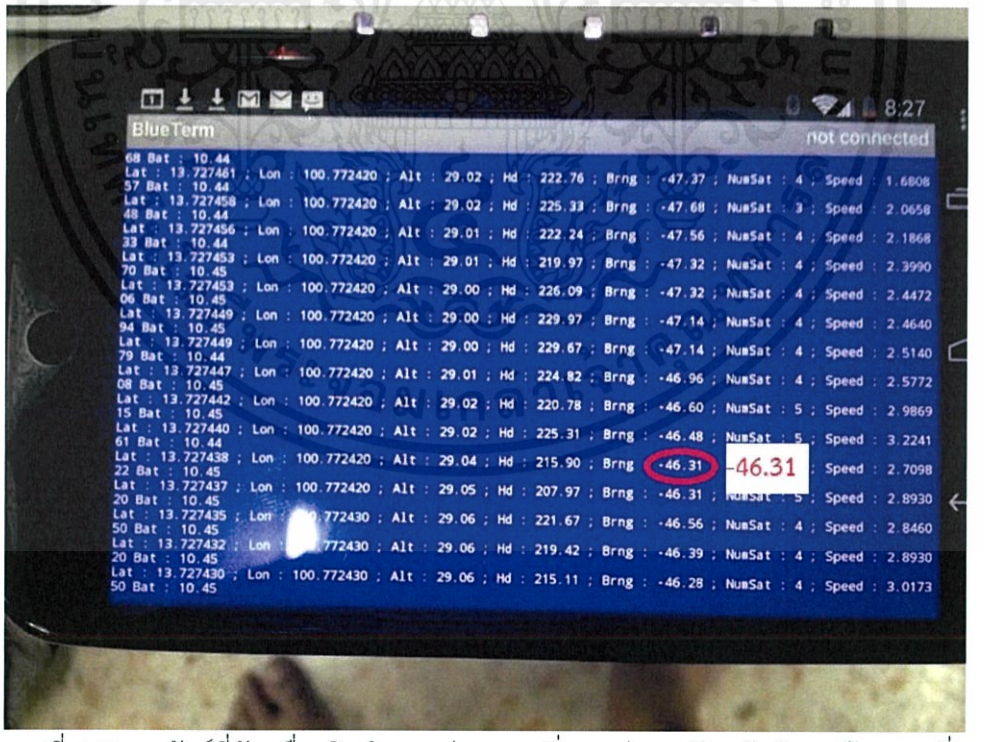
4.5 การทดลองการทำงานของระบบหาค่ามุมแบริ่ง

จากสมการการคำนวณหามุมแบริ่งดังสมการที่ 2.1 ก็ได้ทำการทดลองโดยทดสอบกำหนดจุดพิกัดเป้าหมายให้กับโปรแกรมคำนวณหามุมแบริ่ง และหลังจากนั้นให้โปรแกรมทำการคำนวณหามุมแบริ่งจากจุดที่อยู่ปัจจุบันไปยังจุดเป้าหมาย แสดงตำแหน่งของจุดปัจจุบันและจุดเป้าหมายได้ดังรูปที่ 4.20 และแสดงค่าที่ได้จากการให้โปรแกรมคำนวณดังรูปที่ 4.21

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.20 ผลลัพธ์ที่เมื่อนำมาแสดงบนแผนที่ระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้

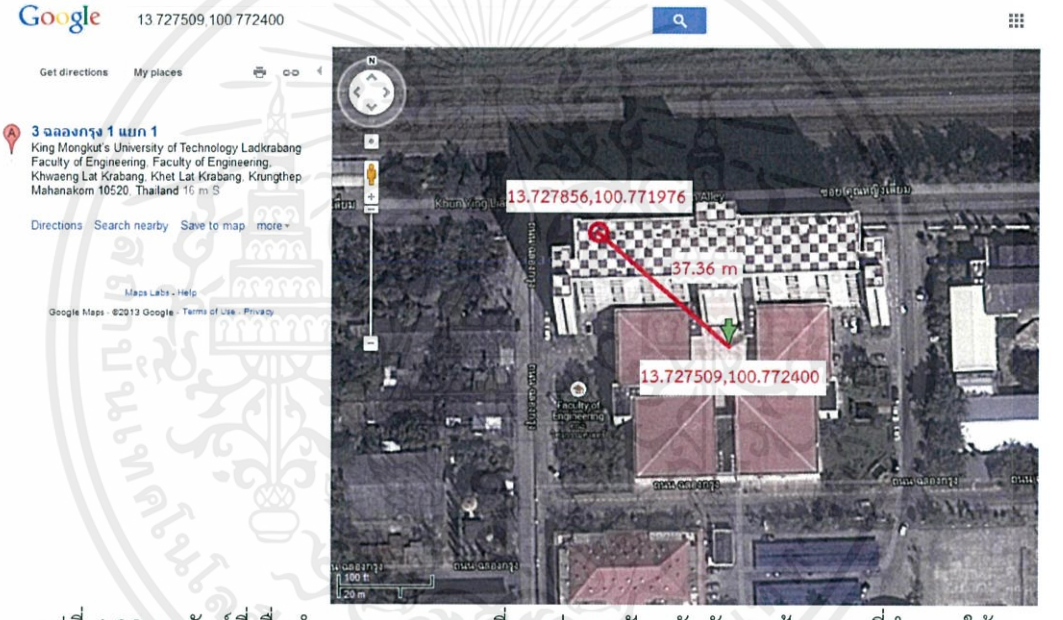


รูปที่ 4.21 ผลลัพธ์ที่ตัวเครื่องบินคำนวณค่ามุมแบรริ่งระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

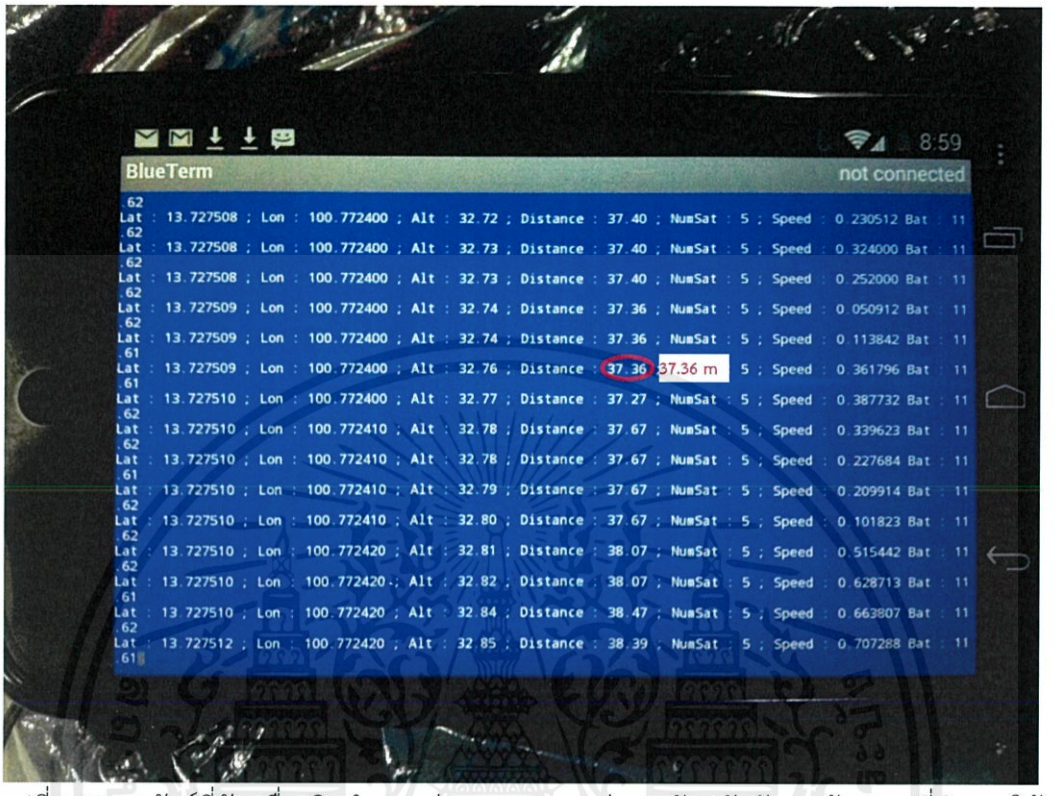
4.6 การทดลองการทำงานของระบบหาค่าระยะทางระหว่างจุดสองจุดบนโลก

จากสมการการคำนวณหาระยะทางระหว่างจุดสองจุดบนโลกดังสมการที่ 2.2 สมการที่ 2.3 และสมการที่ 2.4 ก็ได้ทำการทดลองโดยทดสอบกำหนดจุดพิกัดเป้าหมายให้กับโปรแกรมคำนวณหาระยะทางระหว่างจุดสองจุดนั้น โดยการทดสอบกำหนดจุดเป้าหมายให้กับโปรแกรมคำนวณ และให้โปรแกรมคำนวณหาค่าระยะทางระหว่างจุดที่อยู่ปัจจุบันและจุดเป้าหมาย แสดงตำแหน่งที่อยู่ปัจจุบันและจุดเป้าหมายได้ดังรูปที่ 4.22 และแสดงค่าที่ได้จากการคำนวณของโปรแกรมดังรูปที่ 4.23



รูปที่ 4.22 ผลลัพธ์ที่เมื่อนำมาแสดงบนแผนที่ระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

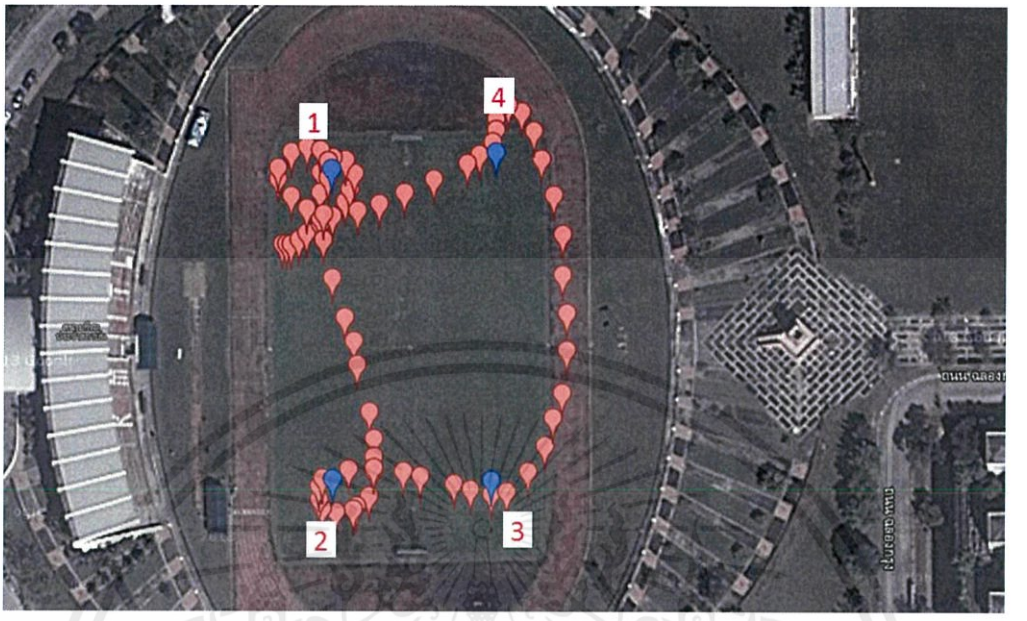


รูปที่ 4.23 ผลลัพธ์ที่ตัวเครื่องบินคำนวณค่าระยะทางระหว่างจุดปัจจุบันกับจุดเป้าหมายที่กำหนดให้

4.7 การทดลองการทำงานของระบบการบินแบบอัตโนมัติ

การทดลองการทำงานของระบบบินแบบอัตโนมัตินั้น เป็นการทดสอบโดยการระบุจุดการบินทั้งหมด 4 จุด และบันทึกค่าโดยการเขียนโปรแกรมให้ทำการเก็บค่าพิกัดตำแหน่งปัจจุบันของเครื่องบินทุกๆ 1 วินาที แล้วนำค่าตำแหน่งที่ได้ทำการเก็บไว้ มาพล็อตลองในแผนที่ เริ่มการทดสอบโดยการปล่อยให้เครื่องบินทำการบินจากจุดที่ 1 ไปจุดที่ 2 แล้วบินจากจุดที่ 2 ไปจุดที่ 3 แล้วบินจากจุดที่ 3 ไปจุดที่ 4 แล้วบินจากจุดที่ 4 กลับมาจุดที่ 1 แสดงผลการทดลองได้ดังรูปที่ 2.24 รูปที่ 2.25 รูปที่ 2.26 รูปที่ 2.27 และรูปที่ 2.28

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



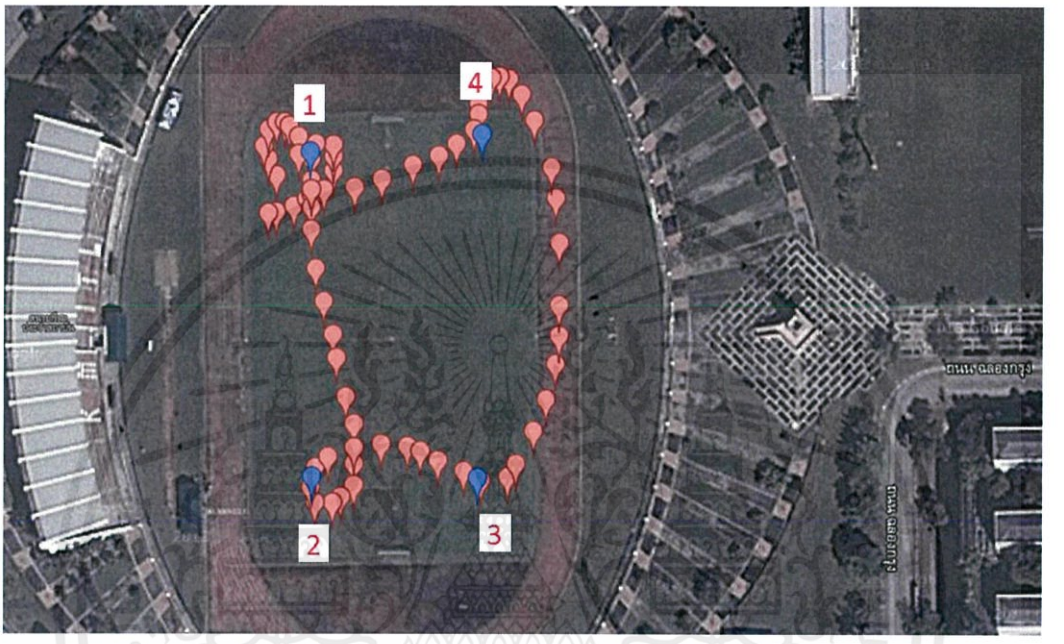
รูปที่ 4.24 เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ แบบยังไม่มีการปรับแต่งค่า

จากรูปที่ 4.24 จุดสีน้ำเงินแสดงจุดเป้าหมายที่ต้องการให้เครื่องบินบินไป จุดสีแดงแสดงจุดที่เครื่องบินเคลื่อนที่ผ่าน

รูปที่ 4.25 เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

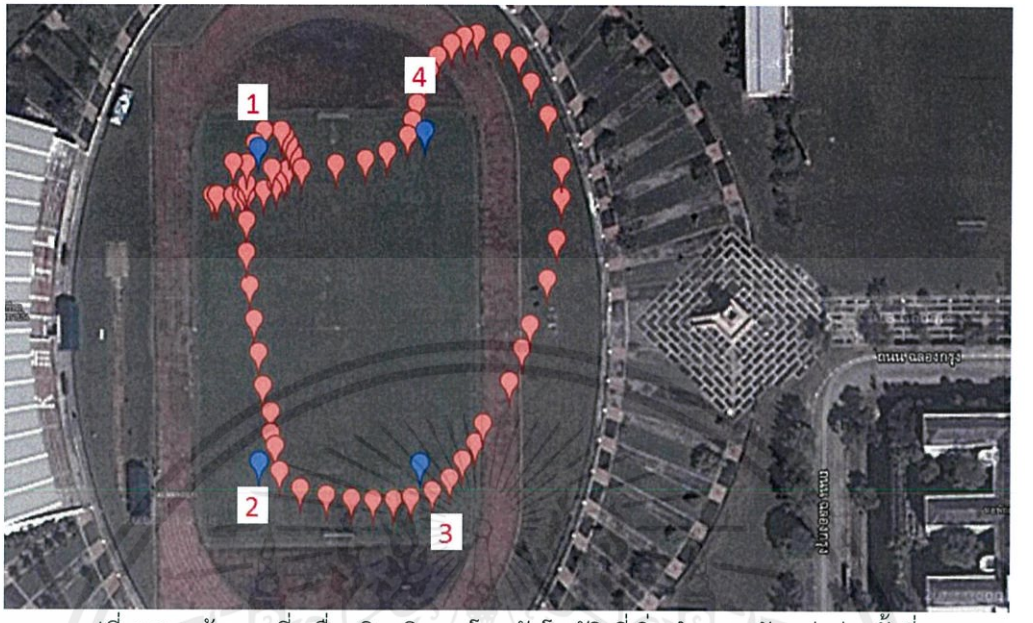
จากรูปที่ 4.25 จุดสีน้ำเงินแสดงจุดเป้าหมายที่ต้องการให้เครื่องบินบินไป จุดสีแดงแสดงจุดที่เครื่องบินเคลื่อนที่ผ่าน โดยการปรับแต่งค่าครั้งนี้ ให้ค่าชดเชยของมุมที่อ่านได้จากเข็มทิศเป็น 15 องศา



รูปที่ 4.26 เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 2

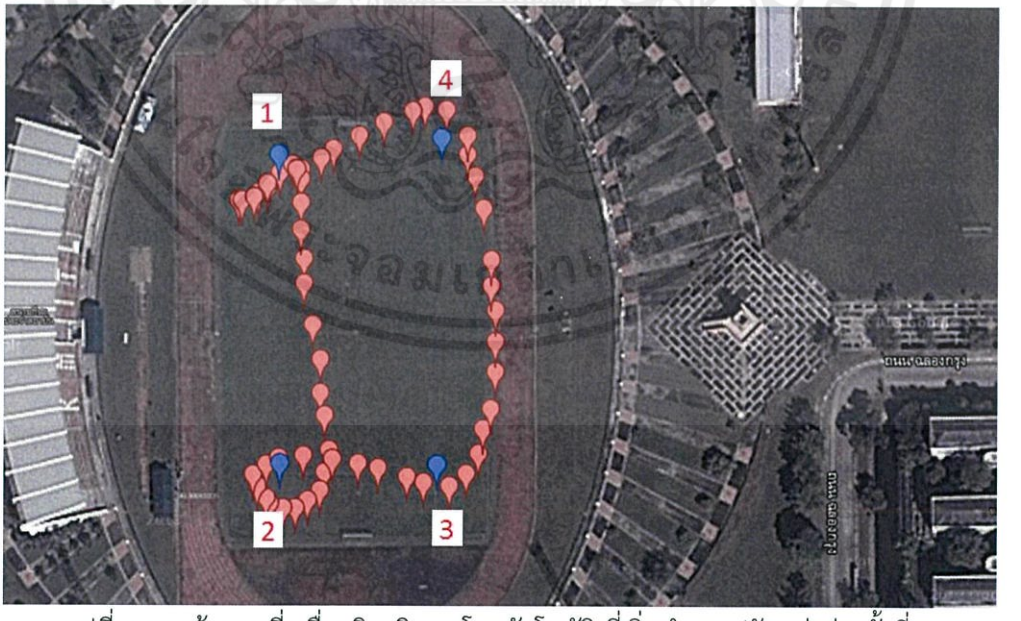
จากรูปที่ 4.26 จุดสีน้ำเงินแสดงจุดเป้าหมายที่ต้องการให้เครื่องบินบินไป จุดสีแดงแสดงจุดที่เครื่องบินเคลื่อนที่ผ่าน โดยการปรับแต่งค่าครั้งนี้ ให้ค่าชดเชยของมุมที่อ่านได้จากเข็มทิศเป็น 0 องศา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.27 เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 3

จากรูปที่ 4.27 จุดสีน้ำเงินแสดงจุดเป้าหมายที่ต้องการให้เครื่องบินบินไป จุดสีแดงแสดงจุดที่เครื่องบินเคลื่อนที่ผ่าน โดยการปรับแต่งค่าครั้งนี้ ให้ค่าชดเชยของมุมที่อ่านได้จากเข็มทิศเป็น -10 องศา



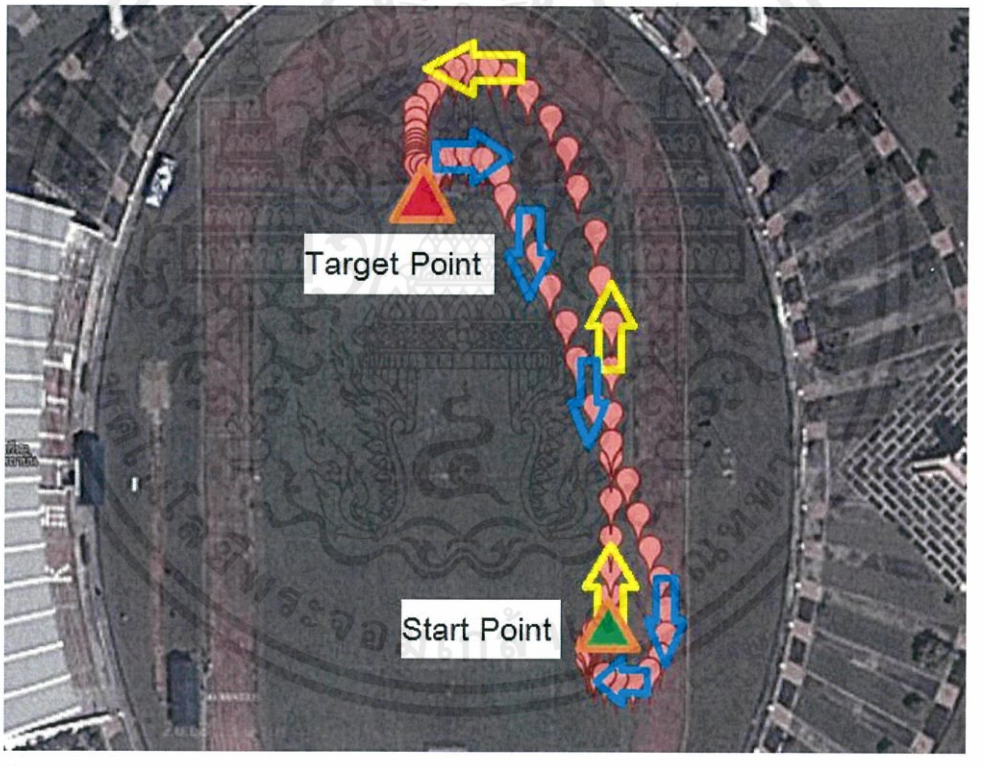
รูปที่ 4.28 เส้นทางที่เครื่องบิน บินเองโดยอัตโนมัติ ที่เริ่มทำการปรับแต่งค่าครั้งที่ 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 4.28 จุดสีน้ำเงินแสดงจุดเป้าหมายที่ต้องการให้เครื่องบินบินไป จุดสีแดงแสดงจุดที่เครื่องบินเคลื่อนที่ผ่าน โดยการปรับแต่งค่าครั้งนี้ ให้ค่าชดเชยของมุมที่อ่านได้จากเข็มทิศเป็น 5 องศา และเป็นครั้งที่ปรับแต่งได้ดีที่สุด จึงใช้ค่า ชดเชยนี้เป็นค่าชดเชยที่ใช้ในการบินจริง

4.8 การทดลองการทำงานของระบบการบินแบบเต็มรูปแบบ

การทดลองการทำงานของระบบการบินแบบเต็มรูปแบบ ทำการทดลองโดยระบุจุดเป้าหมายของการถ่ายภาพให้ แล้วให้เครื่องทำการบินเองทั้งหมด ตั้งแต่การขึ้นบิน การบินไปยังเป้าหมาย การถ่ายรูป การบินกลับสู่จุดออกบิน และการลงจอด โดยแสดงเส้นทางการบินทั้งหมดได้ดังรูปที่ 4.29 และได้ภาพตัวอย่างที่ได้จากการถ่ายจากเครื่องบินดังรูปที่ 4.30



รูปที่ 4.29 เส้นทางการบินแบบเต็มรูปแบบของระบบการบินถ่ายภาพอัตโนมัติ ที่บินจากจุดออกบินไปยังจุดเป้าหมายและถ่ายภาพ แล้วทำการบินกลับมายังจุดออกบินและลงจอดได้เองโดยอัตโนมัติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.30 ภาพถ่ายทางอากาศที่ได้จากเครื่องบินถ่ายภาพไร้คนขับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้ เป็นการออกแบบและจัดสร้างระบบเครื่องบินถ่ายภาพไร้คนขับที่สามารถขึ้นบินและลงจอดได้เองโดยอัตโนมัติ โดยใช้กล่องควบคุม นาซ่า เอ็ม ไลท์ (Naza m lite) เป็นระบบควบคุมสมดุลต่างๆของเครื่องบิน ใช้กล่องควบคุม อาดูไพล็อต (Ardupilot) ทำหน้าที่ในการนำร่องและเป็นส่วนที่ใช้ในการติดต่อกลางของระบบทั้งหมด ใช้บอร์ด ราสเบอรี่พาย (Raspberry PI) เป็นส่วนประมวลผลทางภาพ เพื่อตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด โดยทางผู้จัดทำได้ทำการจัดสร้างโปรแกรมลงในกล่องควบคุม อาดูไพล็อต เพื่อไปสั่งการกล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ ให้ออกบิน บินไปยังเป้าหมาย บินกลับมาฐานออกบิน และลงจอด และจัดสร้างโปรแกรมที่ใช้ในการตรวจจับสัญลักษณ์ที่ใช้ในการลงจอด ซึ่งกล่องควบคุมอาดูไพล็อตติดต่อกับบอร์ดราสเบอรี่พายเพื่อรับตำแหน่งของสัญลักษณ์ที่ใช้ในการลงจอดและสั่งถ่ายภาพ ผ่านทาง พอร์ตอนุกรม

จากผลการทดลอง ทำให้เราได้รู้ถึงข้อผิดพลาดต่างๆ ซึ่งเป็นผลทำให้การทดลองของเราเกิดการคลาดเคลื่อนและไม่สามารถทำงานตามที่คาดไว้ ซึ่งจะแยกอธิบายถึงเหตุผลของข้อผิดพลาดเป็นส่วนๆ

5.1.1 ความผิดเพี้ยนจากเซ็นเซอร์เข็มทิศ

ความผิดเพี้ยนของเข็มทิศ เกิดจากการรบกวนของสนามแม่เหล็กที่อยู่ใกล้ๆตัวเซ็นเซอร์เข็มทิศ ซึ่งทำให้ค่าสนามแม่เหล็กที่วัดได้มานั้นผิดเพี้ยนไป สนามแม่เหล็กที่มารบกวนเป็นส่วนมาก มาจากสนามแม่เหล็กที่ถูกปล่อยออกมาจากมอเตอร์ขับใบพัด ของระบบการบิน จึงทำให้ค่ามุมองศา ที่วัดได้จากเซ็นเซอร์เข็มทิศ ผิดเพี้ยนไปจากที่ควรจะเป็น

5.1.2 ความผิดเพี้ยนจากเซ็นเซอร์ระบุพิกัดจีพีเอส

ความผิดเพี้ยนของเซ็นเซอร์ระบุพิกัดจีพีเอส เกิดจากหลายปัจจัย ไม่ว่าจะมาจากสภาพอากาศขณะนั้น การรบกวนจากคลื่นความถี่วิทยุ ที่เป็นความถี่ในช่วงที่ใช้ในระบบจีพีเอส หรือแม้แต่ความสามารถของโมดูลจีพีเอส โดยแต่ละตัวก็จะมีความสามารถในการระบุพิกัดต่างกัน ซึ่งความผิดเพี้ยนที่เกิดจากการระบุพิกัดนี้ ส่งผลกับระบบในขั้นตอนการลงจอด ซึ่งถ้าค่าความผิดเพี้ยนเกินกว่าที่กำหนด จะส่งผลให้ระบบการลงจอดทำงานผิดเพี้ยน เพราะว่าในขั้นตอนการลงจอดนั้นต้องใช้ระบบระบุพิกัดจีพีเอสช่วยนำร่องเครื่องบินให้เข้ามาใกล้กับที่ลงจอดในระดับที่กล้องที่ใช้ในระบบจับตำแหน่งการลงจอด สามารถจับภาพสัญลักษณ์ในการลงจอดได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.3 ความผิดเพี้ยนจากเซ็นเซอร์ระบุความสูง

เซ็นเซอร์ระบุความสูง จะใช้หลักการการวัดค่าความดันบรรยากาศรอบๆ แล้วนำค่าความดันมาคำนวณกลับเป็นความสูง เนื่องจากว่าการวัดความสูงเป็นการวัดความดัน จึงทำให้ความผิดเพี้ยนเกิดได้ง่ายมาก ไม่ว่าจะเป็นจากอุณหภูมิ จากลมที่เกิดจากใบพัดของเครื่องบิน

5.2 ข้อเสนอแนะ

5.2.1 ข้อเสนอแนะการใช้งาน

1) ก่อนเปิดการใช้งานกล่องควบคุมสมดุล นาซ่า เอ็ม ไลท์ ต้องจ่ายสัญญาณควบคุมมาก่อนเสมอ
2) สัญญาณควบคุมที่ใช้ในระบบบังคับวิทยุ ที่จะนำมาป้อนนั้น ไม่ควรมีค่าคาบเวลาน้อยเกินไป ถ้าหากน้อยเกินไป บางครั้งระบบตรวจจับในปริญญานิพนธ์นี้ ไม่สามารถตรวจจับได้

5.2.2 ข้อเสนอแนะวิธีการแก้ไข

1) **ข้อเสนอแนะวิธีการแก้ไขความผิดเพี้ยนจากเซ็นเซอร์เข็มทิศ**
   - เปลี่ยนที่ติดตั้งเซ็นเซอร์เข็มทิศ ให้อยู่ในที่ที่ห่างกับมอเตอร์ส่งกำลังให้มากที่สุด และยกระดับของที่ติดตั้งเซ็นเซอร์เข็มทิศ ไม่ให้อยู่ในระดับเดียวกับระดับของมอเตอร์ส่งกำลัง
   - นำการกรองแบบคอมพรีเมนทารี่ มาช่วยในการลดค่าความผิดเพี้ยน โดยใช้เซ็นเซอร์ไจโรสโคปในแนวแกน Z เข้ามาช่วย
2) **ข้อเสนอแนะวิธีการแก้ไขความผิดเพี้ยนจากเซ็นเซอร์ระบุพิกัดจีพีเอส**
   - ตำแหน่งการติดตั้งของโมดูลจีพีเอส ต้องไม่ถูกบดบังโดยสิ่งใดๆ
   - เปลี่ยนโมดูลจีพีเอส เป็นรุ่นที่มีความสามารถในการระบุที่มากขึ้น แต่ราคาก็จะสูงขึ้น
3) **ข้อเสนอแนะวิธีการแก้ไขความผิดเพี้ยนจากเซ็นเซอร์ระบุความสูง**
   - ตำแหน่งที่ติดตั้งเซ็นเซอร์ จะต้องเป็นตำแหน่งที่มีลมน้อยที่สุด เพื่อลดความผิดเพี้ยนจากลม
   - ติดตั้งเซ็นเซอร์วัดอุณหภูมิตรงตำแหน่งของเซ็นเซอร์วัดความสูงด้วย เพื่อนำค่าอุณหภูมิมาร่วมคำนวณเพื่อเพิ่มความแม่นยำของค่าความสูงที่ได้จากการคำนวณ
   - เปลี่ยนไปใช้เซ็นเซอร์ที่มีค่าความแม่นยำมากขึ้น แต่ราคาก็จะสูงขึ้นตาม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] ÅSTRÖM, KARL J. PID Controllers: Theory, Design, and Tuning. 2nd ed. p. cm.

[2] OpenCV Reference. http://docs.opencv.org/modules/refman.html

[3] ArduPilot Reference. http://code.google.com/p/ardupilot-mega/

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

โค้ดโปรแกรมที่ใช้ในการตรวจจับสัญลักษณ์ที่ใช้ในการลงจอดของเครื่องบินถ่ายภาพไร้คนขับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <sys/signal.h>
#include <errno.h>
#include <termios.h>
#include <iostream>
#include <unistd.h>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#define SEND_THROUGH_SERIAL    0

using namespace std;
using namespace cv;

#define THRESHOLD_BINARY      0
#define THRESHOLD_TO_ZERO     3

void Uart_INT(int status);

bool IsHalf(Mat img_in_gray , Point center , int radius , float threshold_low ,
float threshold_high)
{
   Mat img_thresholded;
   Size size;
   uint sqr_length;
   uint64 sum_all_pixel;
   uint i,j;
   uint start_x;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
uint start_y;
uint end_x;
uint end_y;

float percent;

size = img_in_gray.size();

if((center.x + radius) > size.width) return 0;
if((center.x - radius) < 0) return 0;

if((center.y + radius) > size.height) return 0;
if((center.y - radius) < 0) return 0;

sqr_length = (uint)sqrt((double)pow(radius,2) / 2) * 2;

start_x = center.x - (sqr_length / 2);
end_x = center.x + (sqr_length / 2);

start_y = center.y - (sqr_length / 2);
end_y = center.y + (sqr_length / 2);

threshold(img_in_gray, img_thresholded, 127, 255, THRESHOLD_BINARY);

sum_all_pixel = 0;
for(j = start_y ; j <= end_y ; j++)
{
   for(i = start_x ; i <= end_x ; i++)
   {
      sum_all_pixel += img_thresholded.at<uchar>(Point(i,j));
   }
}

percent = (float)sum_all_pixel / (255 * sqr_length * sqr_length);
if((percent > threshold_low) && (percent < threshold_high))
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
        return true;
    }
    else
    {
        return false;
    }

}

int Uart_Init(void)
{
    int fd;
    int baudRate;
    struct termios termAttr;
    struct sigaction saio;
    fd = open("/dev/ttyUSB0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (fd == -1)
    {
      perror("open_port: Unable to open /dev/ttyUSB0\n");
      return -1;
    }

    saio.sa_handler = Uart_INT;
    saio.sa_flags = 0;
    saio.sa_restorer = NULL;
    sigaction(SIGIO,&saio,NULL);

    fcntl(fd, F_SETFL, FNDELAY);
    fcntl(fd, F_SETOWN, getpid());

    tcgetattr(fd,&termAttr);
    baudRate = B115200;
    cfsetispeed(&termAttr,B115200);
    cfsetospeed(&termAttr,B115200);
    termAttr.c_cflag &= ~PARENB;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    termAttr.c_cflag &= ~CSTOPB;
    termAttr.c_cflag &= ~CSIZE;
    termAttr.c_cflag |= CS8;
    termAttr.c_cflag |= (CLOCAL | CREAD);
    termAttr.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
    termAttr.c_iflag &= ~(IXON | IXOFF | IXANY);
    termAttr.c_oflag &= ~OPOST;
    tcsetattr(fd,TCSANOW,&termAttr);

    return fd;

}

int Uart_Write_Buf(int fd , unsigned char * buf , int count)
{
  if(write(fd,buf,count) == -1)
    return -1;
  return 0;
}

int main()
{

  Mat src, src_gray;
  CvCapture * capture = cvCreateCameraCapture(0);
  float resolution;

  int img_width = 320;
  int img_height;

#if SEND_THROUGH_SERIAL == 1
  int port_fd;
  port_fd = Uart_Init();
  string str_out;
  char str_out_c[20];
#endif
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#if SEND_THROUGH_SERIAL == 1
   if(port_fd == -1)
   {
      cout << "Uart_Init Error";
      return 0;
   }
#endif

   cvSetCaptureProperty(capture,CV_CAP_PROP_FRAME_WIDTH,320);
   cvSetCaptureProperty(capture,CV_CAP_PROP_FRAME_HEIGHT,240);

   while(waitKey(1) != 13)
   {
      //for(int i=0 ; i < 5 ; i++)
         cvGrabFrame(capture);

      src = cvRetrieveFrame(capture);

      // Convert it to gray
      cvtColor( src, src_gray, CV_BGR2GRAY );


      // Reduce the noise so we avoid false circle detection
      GaussianBlur( src_gray, src_gray, Size(3, 3), 2, 2 );

      vector<Vec3f> circles;

      // Apply the Hough Transform to find the circles
      HoughCircles( src_gray, circles, CV_HOUGH_GRADIENT, 1, src_gray.rows/8,
200, 50, 0, 0 );

      // Draw the circles detected
      for( size_t i = 0; i < circles.size(); i++ )
      {
         Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            int radius = cvRound(circles[i][2]);

            // circle center
            circle( src, center, 3, Scalar(0,255,0), -1, 8, 0 );
            // circle outline
            circle( src, center, radius, Scalar(0,255,0), 3, 8, 0 );

            if(IsHalf(src_gray,center,radius,0.35,0.65))
            {

              // circle center
              circle( src, center, 3, Scalar(0,255,0), -1, 8, 0 );
              // circle outline
              circle( src, center, radius, Scalar(0,0,255), 3, 8, 0 );

              cout << "X: " << center.x << " Y: " << center.y << "\r\n";

#if SEND_THROUGH_SERIAL == 1
              sprintf(str_out_c,"X: %d Y: %d\r\n",center.x,center.y);

              write(port_fd,str_out_c,strlen(str_out_c));
#endif

            }


        }

        imshow("Output",src);

    }
}
void Uart_INT(int status)
{
  printf("Received data from Uart\r\n");
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

โค๊ดโปรแกรมที่ใช้ในระบบการควบคุมการบินอัตโนมัติทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define POSITION_LOG_LENGTH_ADDR        0x0000
#define POSITION_LOG_START_ADDR         0x0010
#define POSITION_LOG_UPDATE_INTERVAL    1000

uint16_t logger_position_current_addr;
uint8_t logger_position_stat;
uint16_t logger_position_length;

void Logger_Init(void)
{

  logger_position_current_addr = POSITION_LOG_START_ADDR;
  logger_position_length = 0;

}

void Logger_Position_Write(double lat , double lon)
{
 int32_t lat_int,lon_int;
 lat_int = (int32_t)(lat * 10000000);
 lon_int = (int32_t)(lon * 10000000);
 hal.storage->write_dword(logger_position_current_addr,lat_int);
 logger_position_current_addr += 4;
 hal.storage->write_dword(logger_position_current_addr,lon_int);
 logger_position_current_addr += 4;
 logger_position_length++;
}

void Logger_Update(void)
{
 static uint32_t logger_position_last_update;
 uint32_t now;

 now = hal.scheduler->millis();
 if(logger_position_stat)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
      if(now - logger_position_last_update > POSITION_LOG_UPDATE_INTERVAL)
      {
        Logger_Position_Update();
        logger_position_last_update = now;
      }
    }
  }

  void Logger_Position_Print_Out(void)
  {
    uint16_t current_addr = POSITION_LOG_START_ADDR;
    double lat;
    int32_t lat_int;
    double lon;
    int32_t lon_int;
    uint16_t length;
    uint16_t i;

    length = hal.storage->read_word(POSITION_LOG_LENGTH_ADDR);

    hal.console->printf("Latitude,Longitude,Number\r\n");
    for(i = 0 ; i < length ; i++)
    {
      lat_int = hal.storage->read_dword(current_addr);
      lat = (double)lat_int / 10000000;
      current_addr += 4;
      lon_int = hal.storage->read_dword(current_addr);
      lon = (double)lon_int / 10000000;
      current_addr += 4;
      hal.console->printf("%.7lf,%.7lf,%d\r\n",lat,lon,i);
    }
  }

  void Logger_Position_Start(void)
  {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        logger_position_stat = 1;
      }

      void Logger_Position_End(void)
      {
        logger_position_stat = 0;
        hal.storage-
>write_word(POSITION_LOG_LENGTH_ADDR,logger_position_length);
      }

      void Logger_Position_Update(void)
      {
        Logger_Position_Write(DJI_GPS_Get_Lat(),DJI_GPS_Get_Lon());
}

      #include <AP_Common.h>
      #include <AP_Progmem.h>
      #include <AP_HAL.h>
      #include <AP_Param.h>
      #include <AP_Math.h>

      #include <AP_HAL_AVR.h>
      #include <AP_HAL_AVR_SITL.h>
      #include <AP_HAL_Empty.h>

      #define CH_1 0
      #define CH_2 1
      #define CH_3 2
      #define CH_4 3
      #define CH_5 4
      #define CH_6 5
      #define CH_7 6
      #define CH_8 7

      #define SW_A_THRESHOLD     500
      #define SW_B_THRESHOLD     500
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define SW_D_THRESHOLD      500

uint8_t sw_thr_stat;
uint8_t sw_rud_stat;
uint8_t sw_ele_stat;
uint8_t sw_ail_stat;

uint8_t override_control = 0;

uint8_t ch1_request_control , ch2_request_control , ch3_request_control ,
ch4_request_control;
uint8_t ch1_failsafe_control , ch2_failsafe_control , ch3_failsafe_control ,
ch4_failsafe_control;
uint16_t ch1_control_value = 0;
uint16_t ch2_control_value = 1500;
uint16_t ch3_control_value = 1500;
uint16_t ch4_control_value = 1500;

uint16_t rcin_pure_1;
uint16_t rcin_pure_2;
uint16_t rcin_pure_3;
uint16_t rcin_pure_4;
uint16_t rcin_pure_5;
uint16_t rcin_pure_6;
uint16_t rcin_pure_7;
uint16_t rcin_pure_8;

uint8_t output_en = 0;

void Remote_Init(void)
{
 hal.rcout->write(CH_1,1500);
 hal.rcout->write(CH_2,1500);
 hal.rcout->write(CH_3,1000);
 hal.rcout->write(CH_4,1500);
 hal.rcout->write(CH_5,1000);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  hal.rcout->write(CH_6,1500);
}

void Remote_Update(void)
{

 uint16_t rc_1,rc_2,rc_3,rc_4,rc_5,rc_6,rc_7,rc_8;
 uint16_t rc_out1,rc_out2,rc_out3,rc_out4,rc_out5,rc_out6,rc_out7;
 static uint16_t last_t_rc;
 uint32_t now = hal.scheduler->millis();

 uint16_t sw_ch6;
 uint16_t sw_ch7;

 if((now - last_t_rc) > 10)
 {
  last_t_rc = now;

  rc_1 = hal.rcin->read(CH_1);
  rc_2 = hal.rcin->read(CH_2);
  rc_3 = hal.rcin->read(CH_3);
  rc_4 = hal.rcin->read(CH_4);
  rc_5 = hal.rcin->read(CH_5);
  rc_6 = hal.rcin->read(CH_6);
  rc_7 = hal.rcin->read(CH_7);
  rc_8 = hal.rcin->read(CH_8);

  rcin_pure_3 = rc_3;

  sw_ch6 = rc_6 - 1000;
  sw_ch7 = rc_7 - 1000;

  //=========================Detect all
switch==============================
  if(sw_ch6 >= 500)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 {
   sw_thr_stat = 1;
 }
 else
 {
   sw_thr_stat = 0;
 }

 if((sw_ch6 % 500) >= 250)
 {
   sw_rud_stat = 1;
 }
 else
 {
   sw_rud_stat = 0;
 }

 if(sw_ch7 >= 500)
 {
   sw_ele_stat = 1;
 }
 else
 {
   sw_ele_stat = 0;
 }

 if((sw_ch7 % 500) >= 250)
 {
   sw_ail_stat = 1;
 }
 else
 {
   sw_ail_stat = 0;
 }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//=============================================================
=======


//============================SW_D_Detecting=====================
=====
        //Dim sw_thr as override_control
        if(sw_thr_stat != 0)
        {
          override_control = 1;
        }
        else
        {
          override_control = 0;
        }

//=============================================================
=======


//========================RC_IN_RANGE_PROTECTION=================
====
        if(rc_1 > 2000) rc_1 = 2000;
        if(rc_1 < 1000) rc_1 = 1000;
        if(rc_2 > 2000) rc_2 = 2000;
        if(rc_2 < 1000) rc_2 = 1000;
        if(rc_3 > 2000) rc_3 = 2000;
        if(rc_3 < 1000) rc_3 = 1000;
        if(rc_4 > 2000) rc_4 = 2000;
        if(rc_4 < 1000) rc_4 = 1000;
        if(rc_5 > 2000) rc_5 = 2000;
        if(rc_5 < 1000) rc_5 = 1000;
        if(rc_6 > 2000) rc_6 = 2000;
        if(rc_6 < 1000) rc_6 = 1000;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        if(rc_7 > 2000) rc_7 = 2000;
        if(rc_7 < 1000) rc_7 = 1000;
        if(rc_8 > 2000) rc_8 = 2000;
        if(rc_8 < 1000) rc_8 = 1000;

//=================================================================
=======


        //=========================A-E-R-
D_Control_Decision=====================
        if(override_control)
        {
         rc_out1 = rc_1;
         rc_out2 = rc_2;
         rc_out3 = rc_3;
         rc_out4 = rc_4;
        }
        else
        {
         if(ch1_request_control) rc_out1 = ch1_control_value;
         else rc_out1 = rc_1;

         if(ch2_request_control) rc_out2 = ch2_control_value;
         else rc_out2 = rc_2;

         if(ch3_request_control) rc_out3 = ch3_control_value;
         else rc_out3 = rc_3;

         if(ch4_request_control) rc_out4 = ch4_control_value;
         else rc_out4 = rc_4;
        }

        rc_out5 = rc_5;
        rc_out6 = rc_7;
        rc_out7 = rc_8;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//=================================================================
=======


//=========================RC_OUT_RANGE_PROTECTION================
======
        if(rc_out1 > 2000) rc_out1 = 2000;
        if(rc_out1 < 1000) rc_out1 = 1000;
        if(rc_out2 > 2000) rc_out2 = 2000;
        if(rc_out2 < 1000) rc_out2 = 1000;
        if(rc_out3 > 2000) rc_out3 = 2000;
        if(rc_out3 < 1000) rc_out3 = 1000;
        if(rc_out4 > 2000) rc_out4 = 2000;
        if(rc_out4 < 1000) rc_out4 = 1000;
        if(rc_out5 > 2000) rc_out5 = 2000;
        if(rc_out5 < 1000) rc_out5 = 1000;
        if(rc_out6 > 2000) rc_out6 = 2000;
        if(rc_out6 < 1000) rc_out6 = 1000;
        if(rc_out7 > 2000) rc_out7 = 2000;
        if(rc_out7 < 1000) rc_out7 = 1000;

//=================================================================
=======

        if(output_en)
        {
         hal.rcout->write(CH_1,rc_out1);
         hal.rcout->write(CH_2,rc_out2);
         hal.rcout->write(CH_3,rc_out3);
         hal.rcout->write(CH_4,rc_out4);
         hal.rcout->write(CH_5,rc_out5);
         hal.rcout->write(CH_6,rc_out6);
         hal.rcout->write(CH_7,rc_out7);
        }
        else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {

    }

  }
 }

 void Remote_Output_Enable(void)
 {
  output_en = 1;
 }

 uint8_t Remote_Get_Override_Stat(void)
 {
  return override_control;
 }

 unsigned int Remote_Get_Pure_Ch3(void)
 {
  return rcin_pure_3;
 }

 uint8_t Remote_Sw_Thr_Stat(void)
 {
  return sw_thr_stat;
 }

 uint8_t Remote_Sw_Rud_Stat(void)
 {
  return sw_rud_stat;
 }

 uint8_t Remote_Sw_Ele_Stat(void)
 {
  return sw_ele_stat;
 }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
uint8_t Remote_Sw_Ail_Stat(void)
{
  return sw_ail_stat;
}

//====================Ch1 Request
Control=========================
void Remote_Ch1_Request_Control(uint8_t stat)
{
  if(stat)
  {
    ch1_request_control = 1;
  }
  else
  {
    ch1_request_control = 0;
  }
}

void Remote_Ch1_Control_Value(unsigned int value)
{
  ch1_control_value = value;
}
//============================================================
======


//====================Ch2 Request
Control=========================
void Remote_Ch2_Request_Control(uint8_t stat)
{
  if(stat)
  {
    ch2_request_control = 1;
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    else
    {
      ch2_request_control = 0;
    }
  }

  void Remote_Ch2_Control_Value(unsigned int value)
  {
    ch2_control_value = value;
  }
  //============================================================
======


  //====================Ch3 Request
Control=========================
  void Remote_Ch3_Request_Control(uint8_t stat)
  {
   if(stat)
   {
    ch3_request_control = 1;
   }
   else
   {
    ch3_request_control = 0;
   }
  }

  void Remote_Ch3_Control_Value(unsigned int value)
  {
    ch3_control_value = value;
  }

  //============================================================
======
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        //====================Ch4 Request
Control=========================
        void Remote_Ch4_Request_Control(uint8_t stat)
        {
          if(stat)
          {
            ch4_request_control = 1;
          }
          else
          {
            ch4_request_control = 0;
          }
        }

        void Remote_Ch4_Control_Value(unsigned int value)
        {
         ch4_control_value = value;
}

        #define CH_1 0
        #define CH_2 1
        #define CH_3 2
        #define CH_4 3
        #define CH_5 4
        #define CH_6 5
        #define CH_7 6
        #define CH_8 7

        #include <AP_HAL.h>

        void RC_Out_Init(void)
        {
         hal.rcout->set_freq(0xF, 490);
         hal.rcout->enable_mask(0xFF);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}

#define CH_1 0
#define CH_2 1
#define CH_3 2
#define CH_4 3
#define CH_5 4
#define CH_6 5
#define CH_7 6
#define CH_8 7

#include <AP_HAL.h>
#include <RC_Channel.h>

RC_Channel rc_1(CH_1);
RC_Channel rc_2(CH_2);
RC_Channel rc_3(CH_3);
RC_Channel rc_4(CH_4);
RC_Channel rc_5(CH_5);
RC_Channel rc_6(CH_6);


void RC_In_Init(void)
{

    RC_In_Setup_Radio();

    // set type of output, symmetrical angles or a number range;
    // XXX BROKEN
    rc_1.set_angle(100);
    rc_1.set_dead_zone(2);
    rc_1.set_type(RC_CHANNEL_TYPE_ANGLE_RAW);
    // XXX BROKEN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    rc_2.set_angle(100);
    rc_2.set_dead_zone(2);
    rc_2.set_type(RC_CHANNEL_TYPE_ANGLE_RAW);

    rc_3.set_range(0,100);
    rc_3.set_dead_zone(2);

    rc_4.set_angle(100);
    rc_4.set_dead_zone(2);
    rc_4.set_type(RC_CHANNEL_TYPE_ANGLE_RAW);

    rc_5.set_range(0,100);
    rc_6.set_range(0,100);

}

void RC_In_Read_Radio(void)
{
    rc_1.set_pwm(hal.rcin->read(CH_1));
    rc_2.set_pwm(hal.rcin->read(CH_2));
    rc_3.set_pwm(hal.rcin->read(CH_3));
    rc_4.set_pwm(hal.rcin->read(CH_4));
    rc_5.set_pwm(hal.rcin->read(CH_5));
    rc_6.set_pwm(hal.rcin->read(CH_6));
}

void RC_In_Setup_Radio(void)
{
        hal.console->println("\n\nRadio Setup:");
        uint8_t i;

        for(i = 0; i < 100;i++){
                hal.scheduler->delay(20);
                RC_In_Read_Radio();
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
rc_1.radio_min = rc_1.radio_in;
rc_2.radio_min = rc_2.radio_in;
rc_3.radio_min = rc_3.radio_in;
rc_4.radio_min = rc_4.radio_in;
rc_5.radio_min = rc_5.radio_in;
rc_6.radio_min = rc_6.radio_in;

rc_1.radio_max = rc_1.radio_in;
rc_2.radio_max = rc_2.radio_in;
rc_3.radio_max = rc_3.radio_in;
rc_4.radio_max = rc_4.radio_in;
rc_5.radio_max = rc_5.radio_in;
rc_6.radio_max = rc_6.radio_in;

rc_1.radio_trim = rc_1.radio_in;
rc_2.radio_trim = rc_2.radio_in;
rc_4.radio_trim = rc_4.radio_in;
// 3 is not trimed
rc_5.radio_trim = 1500;
rc_6.radio_trim = 1500;

hal.console->println("\nMove all controls to each extreme. Hit Enter to
save:");
while(1){

    hal.scheduler->delay(20);
    // Filters radio input - adjust filters in the radio.pde file
    // ----------------------------------------
    RC_In_Read_Radio();

    rc_1.update_min_max();
    rc_2.update_min_max();
    rc_3.update_min_max();
    rc_4.update_min_max();
    rc_5.update_min_max();
    rc_6.update_min_max();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            if(hal.console->available() > 0) {
                hal.console->println("Radio calibrated, Showing control values:");
                break;
            }
        }
        return;
    }

    int RC_In_CH1(void)
    {
      return rc_1.control_in;
    }

    int RC_In_CH2(void)
    {
      return rc_2.control_in;
    }

    int RC_In_CH3(void)
    {
      return rc_3.control_in;
    }

    int RC_In_CH4(void)
    {
      return rc_4.control_in;
    }

    int RC_In_CH5(void)
    {
      return rc_5.control_in;
    }

    int RC_In_CH6(void)
    {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    return rc_6.control_in;
}

#include <AP_Common.h>
#include <AP_Progmem.h>
#include <AP_HAL.h>
#include <AP_Param.h>
#include <AP_Math.h>

#include <AP_HAL_AVR.h>
#include <AP_HAL_AVR_SITL.h>
#include <AP_HAL_Empty.h>

#define CH_1 0
#define CH_2 1
#define CH_3 2
#define CH_4 3
#define CH_5 4
#define CH_6 5
#define CH_7 6
#define CH_8 7

#define FLYING_HEIGHT           20

#define TAKE_OFF_MAX_RATE       300
#define TAKE_OFF_START_RATE     300
#define TAKE_OFF_ARRIVE_THRESHOLD  0.3
#define TALE_OFF_HOLD_T_TO_ARRIVE  3

#define LANDING_MAX_RATE        100

#define FORWARD_VALUE_HIGH      300
#define FORWARD_VALUE_MID       200
#define FORWARD_VALUE_LOW       150
#define MAX_FORWARD_VALUE       300
#define MAX_FORWARD_RATE        300
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define MAX_ROTATE_RATE          100
#define MAX_LIFT_RATE            100
#define MAX_LANDING_RATE          50

#define ADJ2TARGET_MAX_MOVE_RATE   100
#define ADJ2TARGET_ARRIVE_DISTANCE 1

#define TARGET_ARRIVE_LV1_DISTANCE 20
#define TARGET_ARRIVE_LV2_DISTANCE 10
#define TARGET_ARRIVE_LV3_DISTANCE 5


float kp_heading = 7;
float kp_alt = 100;
float lock_heading_degree;
float lock_alt_meter;
float gps_lock_alt_meter;
double movement_target_lat;
double movement_target_lon;
double movement_home_lat;
double movement_home_lon;
double movement_last_gps_lat;
double movement_last_gps_lon;
int16_t movement_const_forward_value;
uint8_t lock_heading_stat;
uint8_t lock_heading2target;
uint8_t lock_alt_stat;
uint8_t gps_lock_alt_stat;
uint8_t landing_stat;
uint8_t go2target_stat;
uint8_t go2target_arrive_stat;
uint8_t go2target_arrive_lv;
float go2target_current_heading;
uint8_t go2home_stat;
uint8_t go2home_arrive_stat;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น รวมถึงการดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 uint8_t go2home_arrive_lv;
 float go2home_current_heading;

 uint8_t takeoff_stat;
 uint8_t takeoff_is_finish;
 float takeoff_kp = 100;
 uint8_t takeoff_first_arrive_stat;
 uint8_t takeoff_stage;

 uint8_t adj2target_stat;
 uint8_t adj2target_arrive_stat;
 double adj2target_lat;
 double adj2target_lon;

 double adj2target_kp = 5000000;

uint8_t const_forward_stat;


void Movement_Init(void)
{

}


void Movement_Update(void)
{
 static uint32_t last_t_update;
 static uint32_t last_t_go2target_update , last_t_go2home_update;
 uint32_t now;

 now = hal.scheduler->millis();

 if(now - last_t_update > 20)
 {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if(lock_heading_stat)
{
  Lock_Heading_Update();
}

if(lock_heading2target)
{
  Lock_Heading2Target_Update();
}

if(lock_alt_stat)
{
  Baro_Lock_Alt_Update();
}

if(gps_lock_alt_stat)
{
  GPS_Lock_Alt_Update();
}

if(landing_stat)
{

}

if(const_forward_stat)
{
  Movement_ConstForward_Update();
}

if(go2target_stat)
{
  if(now - last_t_go2target_update > 200)
  {
    Movement_Go2Target_Update();
    last_t_go2target_update = now;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
   }

   if(go2home_stat)
   {
    if(now - last_t_go2home_update > 200)
    {
     Movement_Go2Home_Update();
     last_t_go2home_update = now;
    }
   }

   if(takeoff_stat)
   {
    Movement_TakeOff_Update();
   }

   if(adj2target_stat)
   {
    Movement_Adj2Target_Update();
   }

  }

}

void Movement_ConstForward_Set_Value(int value)
{
 if(value < 0) value = 0;
 if(value > MAX_FORWARD_VALUE) value = MAX_FORWARD_VALUE;
 movement_const_forward_value = value;
}

void Movement_ConstForward_Start(int value)
{
 const_forward_stat = 1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  Remote_Ch2_Request_Control(1);
  if(value < 0) value = 0;
  if(value > MAX_FORWARD_VALUE) value = MAX_FORWARD_VALUE;
  movement_const_forward_value = value;
}

void Movement_ConstForward_End()
{
  const_forward_stat = 0;
  Remote_Ch2_Request_Control(0);
}

void Movement_ConstForward_Update()
{
 int16_t rc_2,drive;

 drive = movement_const_forward_value;

 if(drive > MAX_FORWARD_RATE) drive = MAX_FORWARD_RATE;
 if(drive < -MAX_FORWARD_RATE) drive = -MAX_FORWARD_RATE;

 rc_2 = 1500 + drive;

 Remote_Ch2_Control_Value(rc_2);
}

void Movement_LockHeading_Start(float degree)
{
 lock_heading_stat = 1;
 lock_heading_degree = degree;
 Remote_Ch4_Request_Control(1);
}

void Movement_LockHeading_End(void)
{
 lock_heading_stat = 0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Remote_Ch4_Request_Control(0);
  }

  void Movement_LockHeading_Set_Degree(float degree)
  {
    lock_heading_degree = degree;
  }

  void Lock_Heading_Update(void)
  {
    int16_t rc_4 , drive;
    drive = (int16_t)(AHRS_Sub_Angle(AHRS_Get_Yaw(),lock_heading_degree) *
kp_heading);
    if(drive > MAX_ROTATE_RATE) drive = MAX_ROTATE_RATE;
    if(drive < -MAX_ROTATE_RATE) drive = -MAX_ROTATE_RATE;

    rc_4 = 1500 - drive;

    Remote_Ch4_Control_Value(rc_4);
  }

  void Movement_LockHeading2Target_Start(void)
  {
    lock_heading2target = 1;
    Remote_Ch4_Request_Control(1);
  }

  void Movement_LockHeading2Target_End(void)
  {
    lock_heading2target = 0;
    Remote_Ch4_Request_Control(0);
  }

  void Movement_LockHeading2Target_Set_Target(double lat,double lon)
  {
    movement_target_lat = lat;
```

เอกสารนี้เป็นเอกสาร[illegible]ใช้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    movement_target_lon = lon;
  }

  void Lock_Heading2Target_Update(void)
  {
    int16_t rc_4, drive;
    static float bearing;
    float current_heading;
    double current_lat,current_lon;

    current_lat = DJI_GPS_Get_Lat();
    current_lon = DJI_GPS_Get_Lon();
    current_heading =
    (DJI_GPS_Get_Heading());

    bearing =
DJI_GPS_Get_Bearing(current_lat,current_lon,movement_target_lat,movement_target_
lon);

    //hal.console->printf("current_heading : %f ; bearing : %f ; Sub :
%f\r\n",current_heading,bearing,AHRS_Sub_Angle(current_heading,bearing));

    drive = (int16_t)(AHRS_Sub_Angle(current_heading,bearing) * kp_heading);

    if(drive > MAX_ROTATE_RATE) drive = MAX_ROTATE_RATE;
    if(drive < -MAX_ROTATE_RATE) drive = -MAX_ROTATE_RATE;

    rc_4 = 1500 - drive;

    //hal.console->printf("Drive : %d\r\n",drive);

    Remote_Ch4_Control_Value(rc_4);
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void Movement_Baro_LockAlt_Start(float meter)
{
  lock_alt_stat = 1;
  lock_alt_meter = meter;
  Remote_Ch3_Request_Control(1);
}

void Movement_Baro_LockAlt_End(void)
{
  lock_alt_stat = 0;
  Remote_Ch3_Request_Control(0);
}

void Movement_Baro_LockAlt_Set_Meter(float meter)
{
 gps_lock_alt_meter = meter;
}

void Movement_GPS_LockAlt_Start(float meter)
{
 gps_lock_alt_stat = 1;
 gps_lock_alt_meter = meter;
 Remote_Ch3_Request_Control(1);
}

void Movement_GPS_LockAlt_End(void)
{
 gps_lock_alt_stat = 0;
 Remote_Ch3_Request_Control(0);
}

void Movement_GPS_LockAlt_Set_Meter(float meter)
{
 gps_lock_alt_meter = meter;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void Baro_Lock_Alt_Update(void)
{
  int16_t rc_3 , drive;
  drive = (int16_t)((BARO_Get_Alt() - lock_alt_meter) * kp_alt);
  if(drive > MAX_LIFT_RATE) drive = MAX_LIFT_RATE;
  if(drive < -MAX_LIFT_RATE) drive = -MAX_LIFT_RATE;

  rc_3 = 1500 - drive;

  Remote_Ch3_Control_Value(rc_3);
}

void GPS_Lock_Alt_Update(void)
{
 int16_t rc_3 , drive;
 drive = (int16_t)((DJI_GPS_Get_Alt() - gps_lock_alt_meter) * kp_alt);
 if(drive > MAX_LIFT_RATE) drive = MAX_LIFT_RATE;
 if(drive < -MAX_LIFT_RATE) drive = -MAX_LIFT_RATE;

 rc_3 = 1500 - drive;

  Remote_Ch3_Control_Value(rc_3);
}

uint8_t Movement_Go2Target_Arrive_Stat(void)
{
 return go2target_arrive_stat;
}

void Movement_Go2Target_Set_Target(double lat , double lon)
{
 movement_target_lat = lat;
 movement_target_lon = lon;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void Movement_Go2Target_Start(double lat , double lon)
{
  go2target_arrive_stat = 0;
  go2target_arrive_lv = 0;
  movement_target_lat = lat;
  movement_target_lon = lon;
  movement_last_gps_lat = DJI_GPS_Get_Lat();
  movement_last_gps_lon = DJI_GPS_Get_Lon();
  //go2target_current_heading = DJI_GPS_Get_Heading();
  go2target_current_heading = AHRS_Get_Yaw();

  //Hold Altitude
  Remote_Ch3_Request_Control(1);
  Remote_Ch3_Control_Value(1500);


  Movement_ConstForward_Start(FORWARD_VALUE_HIGH);
  Remote_Ch4_Request_Control(1);
  go2target_stat = 1;
}
void Movement_Go2Target_End(void)
{
  Remote_Ch4_Request_Control(0);
  Remote_Ch3_Request_Control(0);
  Movement_ConstForward_End();
  go2target_stat = 0;
}

void Movement_Go2Target_Update(void)
{
  double current_gps_lat,current_gps_lon;
  float current_distance;
  float current_bearing;
  float current_speed;
  uint16_t current_numsat;
  int16_t drive,rc_4;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if(go2target_arrive_stat == 0)
{
  current_gps_lat = DJI_GPS_Get_Lat();
  current_gps_lon = DJI_GPS_Get_Lon();
  current_speed = DJI_GPS_Get_Speed();
  current_numsat = DJI_GPS_Get_NumSat();

  current_distance =
DJI_GPS_Get_Distance(current_gps_lat,current_gps_lon,movement_target_lat,moveme
nt_target_lon);
  current_bearing =
DJI_GPS_Get_Bearing(current_gps_lat,current_gps_lon,movement_target_lat,movemen
t_target_lon);

  //go2target_current_heading = DJI_GPS_Get_Heading();
  go2target_current_heading = AHRS_Get_Yaw();

  drive =
(int16_t)(AHRS_Sub_Angle(go2target_current_heading,current_bearing) * kp_heading);

  if(drive > MAX_ROTATE_RATE) drive = MAX_ROTATE_RATE;
  if(drive < -MAX_ROTATE_RATE) drive = -MAX_ROTATE_RATE;

  rc_4 = 1500 - drive;

  Remote_Ch4_Control_Value(rc_4);


  if(current_distance < TARGET_ARRIVE_LV1_DISTANCE) go2target_arrive_lv =
1;
  if(current_distance < TARGET_ARRIVE_LV2_DISTANCE) go2target_arrive_lv =
2;
  if(current_distance < TARGET_ARRIVE_LV3_DISTANCE) go2target_arrive_lv =
3;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            switch(go2target_arrive_lv)
            {
             case 0: Movement_ConstForward_Set_Value(FORWARD_VALUE_HIGH);
break;
             case 1: Movement_ConstForward_Set_Value(FORWARD_VALUE_MID);
break;
             case 2: Movement_ConstForward_Set_Value(FORWARD_VALUE_LOW);
break;
             case 3: Movement_ConstForward_Set_Value(0);
Remote_Ch4_Control_Value(1500); go2target_arrive_stat = 1; break;
            }

          }
          else
          {
            Movement_ConstForward_Set_Value(0);
          }
        }

        void Movement_Set_Home(double lat , double lon)
        {
          movement_home_lat = lat;
          movement_home_lon = lon;
        }

        void Movement_Go2Home_Start(void)
        {
          go2home_arrive_stat = 0;
          go2home_arrive_lv = 0;
          go2home_current_heading = AHRS_Get_Yaw();
          Remote_Ch3_Request_Control(1);
          Remote_Ch3_Control_Value(1500);
          Movement_ConstForward_Start(FORWARD_VALUE_HIGH);
          Remote_Ch4_Request_Control(1);
          go2home_stat = 1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}

void Movement_Go2Home_End(void)
{
  Remote_Ch4_Request_Control(0);
  Remote_Ch3_Request_Control(0);
  Movement_ConstForward_End();
  go2home_stat = 0;
}

uint8_t Movement_Go2Home_Arrive_Stat(void)
{
  return go2home_arrive_stat;
}

void Movement_Go2Home_Update(void)
{
 double current_gps_lat,current_gps_lon;
 float current_distance;
 float current_bearing;
 float current_speed;
 uint16_t current_numsat;
 int16_t drive,rc_4;

 if(go2home_arrive_stat == 0)
 {
  current_gps_lat = DJI_GPS_Get_Lat();
  current_gps_lon = DJI_GPS_Get_Lon();
  current_speed = DJI_GPS_Get_Speed();
  current_numsat = DJI_GPS_Get_NumSat();

  current_distance =
DJI_GPS_Get_Distance(current_gps_lat,current_gps_lon,movement_home_lat,moveme
nt_home_lon);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            current_bearing =
DJI_GPS_Get_Bearing(current_gps_lat,current_gps_lon,movement_home_lat,movemen
t_home_lon);

            //go2target_current_heading = DJI_GPS_Get_Heading();
            go2home_current_heading = AHRS_Get_Yaw();

            drive =
(int16_t)(AHRS_Sub_Angle(go2home_current_heading,current_bearing) * kp_heading);

            if(drive > MAX_ROTATE_RATE) drive = MAX_ROTATE_RATE;
            if(drive < -MAX_ROTATE_RATE) drive = -MAX_ROTATE_RATE;

            rc_4 = 1500 - drive;

            Remote_Ch4_Control_Value(rc_4);



            if(current_distance < TARGET_ARRIVE_LV1_DISTANCE) go2home_arrive_lv =
1;
            if(current_distance < TARGET_ARRIVE_LV2_DISTANCE) go2home_arrive_lv =
2;
            if(current_distance < TARGET_ARRIVE_LV3_DISTANCE) go2home_arrive_lv =
3;

            switch(go2home_arrive_lv)
            {
              case 0: Movement_ConstForward_Set_Value(FORWARD_VALUE_HIGH);
break;
              case 1: Movement_ConstForward_Set_Value(FORWARD_VALUE_MID);
break;
              case 2: Movement_ConstForward_Set_Value(FORWARD_VALUE_LOW);
break;
              case 3: Movement_ConstForward_Set_Value(0);
Remote_Ch4_Control_Value(1500); go2home_arrive_stat = 1; break;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

```
    }

   }
   else
   {
    Movement_ConstForward_Set_Value(0);
   }
  }

 void Movement_TakeOff_Start(void)
 {
  takeoff_is_finish = 0;
  takeoff_first_arrive_stat = 0;
  takeoff_stat = 1;
  Remote_Ch3_Request_Control(1);
 }

 void Movement_TakeOff_End(void)
 {
  takeoff_stat = 0;
  Remote_Ch3_Request_Control(0);
 }

 uint8_t Movement_TakeOff_IsFinish(void)
 {
  return takeoff_is_finish;
 }

 void Movement_TakeOff_Update(void)
 {
  int16_t rc_3 , drive;
  uint32_t now;
  float d_height;
  static uint32_t first_arrive_t;

  //now = hal.scheduler->millis();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    d_height = BARO_Get_Alt() - FLYING_HEIGHT;
    drive = (int16_t)(d_height * kp_alt);
    if(drive > TAKE_OFF_MAX_RATE) drive = TAKE_OFF_MAX_RATE;
    if(drive < -TAKE_OFF_MAX_RATE) drive = -TAKE_OFF_MAX_RATE;

    rc_3 = 1500 - drive;
    Remote_Ch3_Control_Value(rc_3);


    now = hal.scheduler->millis();

    if(!takeoff_first_arrive_stat)
    {
     if(d_height > -TAKE_OFF_ARRIVE_THRESHOLD)
     {
      takeoff_first_arrive_stat = 1;
      first_arrive_t = now;
      //hal.console->printf("Arrived and wait for 3 sec\r\n");
     }
    }
    else
    {
     if((now - first_arrive_t) > (TALE_OFF_HOLD_T_TO_ARRIVE * 1000))
     {
      takeoff_is_finish = 1;
      Movement_TakeOff_End();
      //hal.console->printf("Take Off finished\r\n");
     }
    }

}

void Movement_Landing_Start(void)
{
  Remote_Ch3_Request_Control(1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Remote_Ch3_Control_Value(1500 - LANDING_MAX_RATE);
  }

  void Movement_Landing_End(void)
  {
    Remote_Ch3_Request_Control(0);
  }

  void Movement_Adj2Target_Start(double lat , double lon)
  {
    adj2target_stat = 1;
    adj2target_arrive_stat = 0;
    adj2target_lat = lat;
    adj2target_lon = lon;
    Movement_LockHeading_Start(0);
    Remote_Ch1_Request_Control(1);
    Remote_Ch2_Request_Control(1);
  }

  void Movement_Adj2Target_End(void)
  {
    adj2target_stat = 0;
    Movement_LockHeading_End();
    Remote_Ch1_Request_Control(0);
    Remote_Ch2_Request_Control(0);
  }

  uint8_t Movement_Adj2Target_Arrive_Stat(void)
  {
    return adj2target_arrive_stat;
  }

  void Movement_Adj2Target_Update(void)
  {
    double d_lat,d_lon;
    double x_output,y_output;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
float distance2target;
int16_t x_drive,y_drive;

uint32_t now;
static uint32_t last_t;

now = hal.scheduler->millis();

if(now - last_t > 100)
{
  last_t = now;
  d_lat = adj2target_lat - DJI_GPS_Get_Lat();
  d_lon = adj2target_lon - DJI_GPS_Get_Lon();


  x_output = adj2target_kp * d_lon;
  y_output = adj2target_kp * d_lat;

  if(x_output > 500) x_output = 500;
  if(x_output < -500) x_output = -500;

  if(y_output > 500) y_output = 500;
  if(y_output < -500) y_output = -500;

  x_drive = (int16_t)x_output;
  y_drive = (int16_t)y_output;

  if(x_drive > ADJ2TARGET_MAX_MOVE_RATE) x_drive =
ADJ2TARGET_MAX_MOVE_RATE;
  if(x_drive < -ADJ2TARGET_MAX_MOVE_RATE) x_drive = -
ADJ2TARGET_MAX_MOVE_RATE;

  if(y_drive > ADJ2TARGET_MAX_MOVE_RATE) y_drive =
ADJ2TARGET_MAX_MOVE_RATE;
  if(y_drive < -ADJ2TARGET_MAX_MOVE_RATE) y_drive = -
ADJ2TARGET_MAX_MOVE_RATE;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            //hal.console->printf("Lat:%.7f ; Lon:%.7f ; dLat:%.7f ; dLon:%.7f ;
x_drive:%d ;
y_drive:%d\r\n",DJI_GPS_Get_Lat(),DJI_GPS_Get_Lon(),d_lat,d_lon,x_drive,y_drive);

            Remote_Ch1_Control_Value(1500 + x_drive);
            Remote_Ch2_Control_Value(1500 + y_drive);

            if(!adj2target_arrive_stat)
            {
              distance2target =
DJI_GPS_Get_Distance(DJI_GPS_Get_Lat(),DJI_GPS_Get_Lon(),adj2target_lat,adj2target_l
on);
              if(distance2target < ADJ2TARGET_ARRIVE_DISTANCE)
              {
                //hal.console->printf("Adj2Target Arrived\r\n");
                adj2target_arrive_stat = 1;
              }
            }
          }
        }

        void Movement_Adj2MainLine_Start(void)
        {
          float angle_src2target;
          float angle_current2target;
        }

        void Movement_Adj2MainLine_End(void)
        {

        }

        void Movement_Adj2MainLine_Update(void)
        {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}

#include <FastSerial.h>

#include <AP_Common.h>
#include <AP_Progmem.h>
#include <AP_HAL.h>
#include <AP_Param.h>
#include <AP_Math.h>

#include <AP_HAL_AVR.h>
#include <AP_HAL_AVR_SITL.h>
#include <AP_HAL_Empty.h>

#include <NazaDecoderLib.h>

#define HORIZONTAL_ROUND_LENGTH        40075
#define VERTICAL_ROUND_LENGTH          40008

#define HEADING_ADJ                    -17



uint32_t currTime, attiTime;
uint8_t gps_avail,compass_avail;
float last_bearing;


float degree2form(float ang_in)
{
  if(ang_in > 180) return ang_in - 360;
  else return ang_in;
}

float degree2radian(float ang_in)
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น ทั้งนี้ต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  return ang_in * 0.017453;
 }
 float radian2degree(float ang_in)
 {
  return ang_in * 57.296;
 }

 void DJI_GPS_Init(void)
 {
  hal.uartC->begin(115200,128,16);
 }

 void DJI_GPS_Update(void)
 {
  uint8_t decodedMessage;
  int16_t numc;

  numc = hal.uartC->available();

  for(int16_t i = 0 ; i < numc ; i++)
  {
   decodedMessage = NazaDecoder.decode(hal.uartC->read());
   switch (decodedMessage)
   {
     case NAZA_MESSAGE_GPS:
       gps_avail = 1;
       //hal.uartC->ResetBuffer();
       break;
     case NAZA_MESSAGE_COMPASS:
       compass_avail = 1;
       //hal.uartC->ResetBuffer();
       break;
   }
  }
 }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
uint8_t DJI_GPS_GPS_Avail(void)
{
 if(gps_avail == 1)
 {
  gps_avail = 0;
  return 1;
 }
 else
 {
  return 0;
 }
}

uint8_t DJI_GPS_Compass_Avail(void)
{
 if(compass_avail == 1)
 {
  compass_avail = 0;
  return 1;
 }
 else
 {
  return 0;
 }
}

double DJI_GPS_Get_Lat(void)
{
 return NazaDecoder.getLat();
}

double DJI_GPS_Get_Lon(void)
{
 return NazaDecoder.getLon();
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
double DJI_GPS_Get_Alt(void)
{
  return NazaDecoder.getAlt();
}

uint8_t DJI_GPS_Get_FixType(void)
{
  return NazaDecoder.getFixType();
}

uint8_t DJI_GPS_Get_NumSat(void)
{
  return NazaDecoder.getNumSat();
}

float DJI_GPS_Get_Heading(void)
{
  return degree2form(NazaDecoder.getHeading() + HEADING_ADJ);
}

float DJI_GPS_Get_Speed(void)
{
  return NazaDecoder.getSpeed();
}

float DJI_GPS_Get_Distance(double p1_lat , double p1_lon , double p2_lat ,
double p2_lon)
{
   int radius = 3956.6;
   const float two = 2.0;
   float delta_lat = radians(p2_lat - p1_lat);
   float delta_lon = radians(p2_lon - p1_lon);
   float a = square(sin(delta_lat/two)) + cos(radians (p1_lat)) * cos(radians
(p2_lat)) * square(sin(delta_lon/two));
   float c = two * asin(sqrt(a));
   float d = radius * c;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        return d * 1000;
    }

    float DJI_GPS_Get_Bearing(double p1_lat , double p1_lon , double p2_lat ,
double p2_lon)
    {
        double a = degree2radian(p1_lat);
        double b = degree2radian(p1_lon);
        double c = degree2radian(p2_lat);
        double d = degree2radian(p2_lon);

        if (cos(c) * sin(d - b) == 0)
          if (c > a)
            return 0;
          else
            return 180;
        else
        {
          double angle = atan2(cos(c) * sin(d - b), sin(c) * cos(a) - sin(a) * cos(c) *
cos(d - b));
          return radian2degree(angle);
        }
}

    void Camera_Control_Init(void)
    {
      hal.uartB->begin(115200,128,16);
    }

    void Camera_Control_Take(void)
    {
      hal.uartB->write('t');
}
    #include <AP_Common.h>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <AP_Progmem.h>
#include <AP_Param.h>
#include <AP_Math.h>
#include <AP_HAL.h>
#include <AP_Buffer.h>
#include <Filter.h>
#include <AP_Baro.h>

#include <AP_HAL_AVR.h>

#define BATTERY_AN_CH          0


AP_HAL::AnalogSource* battery_an_source;

void Battery_Init(void)
{
 battery_an_source = hal.analogin->channel(BATTERY_AN_CH);
}

float Battery_Read(void)
{
 return (float)(battery_an_source->read_average() * 0.03);
}


#include <AP_Common.h>
#include <AP_Progmem.h>
#include <AP_Param.h>
#include <AP_Math.h>
#include <AP_HAL.h>
#include <AP_Buffer.h>
#include <Filter.h>
#include <AP_Baro.h>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <AP_HAL_AVR.h>
```

```
AP_Baro_MS5611 baro(&AP_Baro_MS5611::spi);

static uint32_t timer;

static float baro_pressure , baro_temp , baro_alt;

void BARO_Init(void)
{
    /* What's this for? */
    hal.gpio->pinMode(63, GPIO_OUTPUT);
    hal.gpio->write(63, 1);

    baro.init();
    baro.calibrate();

    timer = hal.scheduler->micros();
}

void BARO_Cal(void)
{
  baro.calibrate();
}

void BARO_Update(void)
{
    if((hal.scheduler->micros() - timer) > 100000UL) {
        timer = hal.scheduler->micros();
        baro.read();

        baro_pressure = baro.get_pressure();
        baro_temp = baro.get_temperature();
        baro_alt = baro.get_altitude();
    }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
float BARO_Get_Pressure(void)
{
  return baro_pressure;
}

float BARO_Get_Temp(void)
{
  return baro_temp;
}

float BARO_Get_Alt(void)
{
  return baro_alt;
}

// -*- tab-width: 4; Mode: C++; c-basic-offset: 4; indent-tabs-mode: nil -*-

//
// Simple test for the AP_AHRS interface
//

#include <AP_HAL.h>
#include <AP_Common.h>
#include <AP_Progmem.h>
#include <AP_Math.h>
#include <AP_Param.h>
#include <AP_InertialSensor.h>
#include <AP_ADC.h>
#include <AP_GPS.h>
#include <AP_AHRS.h>
#include <AP_Compass.h>
#include <AP_Declination.h>
#include <AP_Airspeed.h>
#include <AP_Baro.h>
#include <GCS_MAVLink.h>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <Filter.h>
#include <SITL.h>
#include <AP_Buffer.h>

#include <AP_HAL_AVR.h>
#include <AP_HAL_AVR_SITL.h>
#include <AP_HAL_Empty.h>


#if CONFIG_HAL_BOARD == HAL_BOARD_APM2
AP_InertialSensor_MPU6000 ins;
#elif CONFIG_HAL_BOARD == HAL_BOARD_APM1
AP_ADC_ADS7844 adc;
AP_InertialSensor_Oilpan ins( &adc );
#else
AP_InertialSensor_Stub ins;
#endif

AP_Compass_HMC5843 compass;

GPS *g_gps;

AP_GPS_Auto g_gps_driver(&g_gps);

// choose which AHRS system to use
AP_AHRS_DCM  ahrs(&ins, g_gps);
//AP_AHRS_MPU6000  ahrs(&ins, g_gps);          // only works with APM2

AP_Baro_HIL barometer;


#define HIGH 1
#define LOW 0

#if CONFIG_HAL_BOARD == HAL_BOARD_APM2
 # define A_LED_PIN        27
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น

```
 # define C_LED_PIN       25
 # define LED_ON          LOW
 # define LED_OFF         HIGH
#else
 # define A_LED_PIN       37
 # define C_LED_PIN       35
 # define LED_ON          HIGH
 # define LED_OFF         LOW
#endif

#define HEADING_ADJ                 0


static void flash_leds(bool on)
{
    hal.gpio->write(A_LED_PIN, on ? LED_OFF : LED_ON);
    hal.gpio->write(C_LED_PIN, on ? LED_ON : LED_OFF);
}

void AHRS_Leds(bool stat)
{
    hal.gpio->write(A_LED_PIN, stat);
    hal.gpio->write(C_LED_PIN, stat);
}

void AHRS_Init(void)
{

 #ifdef APM2_HARDWARE
    // we need to stop the barometer from holding the SPI bus
    hal.gpio->pinMode(40, GPIO_OUTPUT);
    hal.gpio->write(40, HIGH);
 #endif
    ins.init(AP_InertialSensor::COLD_START,
                        AP_InertialSensor::RATE_100HZ,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                                    flash_leds);
        ins.init_accel(flash_leds);

        ahrs.init();

        if( compass.init() ) {
            hal.console->printf("Enabling compass\n");
            ahrs.set_compass(&compass);
        } else {
            hal.console->printf("No compass detected\n");
        }
        g_gps = &g_gps_driver;
    #if WITH_GPS
        g_gps->init(hal.uartB);
    #endif

}

void AHRS_Update(void)
{
    static uint16_t counter;
    static uint32_t last_t, last_print, last_compass;
    uint32_t now = hal.scheduler->micros();
    float heading = 0;

    if (last_t == 0) {
        last_t = now;
        return;
    }
    last_t = now;

    if (now - last_compass > 100*1000UL &&
        compass.read()) {
        heading = compass.calculate_heading(ahrs.get_dcm_matrix());
        // read compass at 10Hz
        last_compass = now;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#if WITH_GPS
        g_gps->update();
#endif
    }

    ahrs.update();

}

float AHRS_Convert2Format(float ang_in)
{

 float rtn_value;

 if((ang_in > -180) && (ang_in < 180))
 {
  rtn_value = ang_in;
 }

 if(ang_in > 180)
 {
  rtn_value =  ang_in - 360;
 }

 if(ang_in < -180)
 {
  rtn_value = ang_in + 360;
 }

 return rtn_value;

}

float AHRS_Sub_Angle(float first_ang , float second_ang)
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    float rtn_value;

    if(first_ang == 180) first_ang = 179.99;
    if(first_ang == -180) first_ang = -179.99;

    if(second_ang == 180) second_ang = 179.99;
    if(second_ang == -180) second_ang = -179.99;

    first_ang = AHRS_Convert2Format(first_ang);
    second_ang = AHRS_Convert2Format(second_ang);

    if(((first_ang - second_ang) < 180) && ((first_ang - second_ang) > -180))
    {
     rtn_value = first_ang - second_ang;
    }

    if(first_ang - second_ang > 180)
    {
     rtn_value = first_ang - (second_ang + 360);
    }

    if(first_ang - second_ang < -180)
    {
     rtn_value = (first_ang + 360) - second_ang;
    }

    return rtn_value;

}

float AHRS_Get_Roll(void)
{
  return ToDeg(ahrs.roll);
}
float AHRS_Get_Pitch(void)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
  return ToDeg(ahrs.pitch);
}

float AHRS_Get_Yaw(void)
{
  return AHRS_Convert2Format(ToDeg(ahrs.yaw) + HEADING_ADJ - 90);
}


/*
 *      Example of RC_Channel library.
 *      Code by Jason Short. 2010
 *      DIYDrones.com
 *
 */
#include <AP_Common.h>
#include <AP_Progmem.h>
#include <AP_HAL.h>
#include <AP_Param.h>
#include <AP_Math.h>

#include <AP_HAL_AVR.h>
#include <AP_HAL_AVR_SITL.h>
#include <AP_HAL_Empty.h>

#define CH_1 0
#define CH_2 1
#define CH_3 2
#define CH_4 3
#define CH_5 4
#define CH_6 5
#define CH_7 6
#define CH_8 7

#define TARGET_LAT              13.731092
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define TARGET_LON                  100.773679

const AP_HAL::HAL& hal = AP_HAL_BOARD_DRIVER;

uint8_t alt_lock_mode = 0;

uint8_t plan_i;

double home_lat,home_lon;

double plan_target_point[4][2] = {13.730558,100.772027,
                                  13.729838,100.772033,
                                  13.729836,100.772411,
                                  13.730597,100.772419};

double auto_target_point[2] = {13.730578,100.772201};
uint8_t auto_stage;
uint8_t auto_stat;


void setup()
{

 Battery_Init();
 hal.uartA->begin(9600);
 RC_Out_Init();

 //RC_In_Init();
 BARO_Init();
 DJI_GPS_Init();
 AHRS_Init();
 Logger_Init();
 Camera_Control_Init();
 //FailSafe_Init();
 Remote_Update();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        if(Remote_Sw_Rud_Stat())
        {
          Logger_Position_Print_Out();
          while(Remote_Sw_Rud_Stat());
        }

        if(Remote_Sw_Ele_Stat())
        {
          Remote_Update();
          while(1)
          {
            hal.console->printf("THR : %d , RUD : %d , ELE : %d , AIL : %d
\r\n",Remote_Sw_Thr_Stat(),Remote_Sw_Rud_Stat(),Remote_Sw_Ele_Stat(),Remote_S
w_Ail_Stat());
            Remote_Update();
            hal.scheduler->delay(100);
          }
        }

        hal.console->printf("Waiting for locked Satellite >= 7\r\n");
        while((DJI_GPS_Get_NumSat() < 7) && !Remote_Get_Override_Stat())
        {
          DJI_GPS_Update();
          Remote_Update();
          hal.console->printf("Current NumSat : %d   \r",DJI_GPS_Get_NumSat());
        }
        hal.console->printf("\r\n");

        home_lat = DJI_GPS_Get_Lat();
        home_lon = DJI_GPS_Get_Lon();

        Movement_Set_Home(home_lat,home_lon);

        hal.console->printf("Set home to %.6lf,%.6lf\r\n",home_lat,home_lon);
```

เอกสารนี้เป็นเอกสาร[illegible]ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        //=====================Waiting for Start
Condition=========================
        hal.console->printf("Waiting for remote-start-condition...\r\n");
        hal.console->printf("Throttle up to max!!!\r\n");
        while(hal.rcin->read(CH_3) < 1800) hal.scheduler->delay(50);
        hal.console->printf("Throttle down to min!!!\r\n");
        while(hal.rcin->read(CH_3) > 1200) hal.scheduler->delay(50);
        hal.console->printf("Start-Condition Found!!!\r\n");
        Remote_Output_Enable();
        hal.scheduler->delay(1000);

//================================================================
==========

    }

    void Auto_Start(void)
    {
     auto_stat = 1;
     auto_stage = 1;
    }

    void Auto_End(void)
    {
     auto_stat = 0;
     Movement_TakeOff_End();
     Movement_Go2Target_End();
     Movement_Go2Home_End();
     Movement_Landing_End();
     Movement_Adj2Target_End();
    }

    void Auto_Running(void)
    {
     uint32_t now;
     static uint32_t stage_last_t;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
now = hal.scheduler->millis();


switch(auto_stage)
{
 case 1:
  Movement_TakeOff_Start();
  auto_stage++;
  hal.console->printf("Start TakingOff\r\n");
  break;
 case 2:
  if(Movement_TakeOff_IsFinish())
  {
   Movement_TakeOff_End();
   Movement_Go2Target_Start(auto_target_point[0],auto_target_point[1]);
   Logger_Position_Start();
   auto_stage++;
   hal.console->printf("TakeOff Finished\r\n");
   hal.console->printf("Start Going to Target\r\n");
  }
  break;
 case 3:
  if(Movement_Go2Target_Arrive_Stat())
  {
   Movement_Go2Target_End();
   Movement_Baro_LockAlt_Start(BARO_Get_Alt());
   Movement_Adj2Target_Start(auto_target_point[0],auto_target_point[1]);
   auto_stage++;
   hal.console->printf("Arrive Target\r\n");
   hal.console->printf("Finely adjusting to target\r\n");
  }
  break;
 case 4:
  if(Movement_Adj2Target_Arrive_Stat())
  {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        Movement_Adj2Target_End();
        stage_last_t = now;
        auto_stage++;
        hal.console->printf("Adjusting finished\r\n");
        hal.console->printf("Waiting for 3 sec\r\n");
      }
      break;
    case 5:
      if(now - stage_last_t > 3000)
      {
        Camera_Control_Take();
        stage_last_t = now;
        auto_stage++;
        hal.console->printf("Take a photo\r\n");
        hal.console->printf("Waiting for 5 sec\r\n");
      }
      break;
    case 6:
      if(now - stage_last_t > 5000)
      {
        Movement_Baro_LockAlt_End();
        Movement_Go2Home_Start();
        auto_stage++;
        hal.console->printf("Start going to Home\r\n");
      }
      break;
    case 7:
      if(Movement_Go2Home_Arrive_Stat())
      {
        Movement_Go2Home_End();
        Movement_Baro_LockAlt_Start(BARO_Get_Alt());
        Movement_Adj2Target_Start(home_lat,home_lon);
        auto_stage++;
        hal.console->printf("Arrive Home\r\n");
        hal.console->printf("Finely adjusting to home\r\n");
      }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของผลงานทุกครั้งที่มีการนำไปใช้

```
            break;
          case 8:
            if(Movement_Adj2Target_Arrive_Stat())
            {
              Movement_Adj2Target_End();
              Movement_Baro_LockAlt_End();
              Movement_Landing_Start();
              Logger_Position_End();
              auto_stage = 0;
              hal.console->printf("Adjusting finished\r\n");
              hal.console->printf("Start Landing\r\n");
            }
            break;
        }
      }

      void Auto_Update(void)
      {
        if(auto_stat)
        {
          Auto_Running();
        }
      }

      void Plan_Running(void)
      {

        switch(plan_i)
        {
          case 0:

Movement_Go2Target_Start(plan_target_point[0][0],plan_target_point[0][1]);
            plan_i++;
            break;
          case 1:
            if(Movement_Go2Target_Arrive_Stat())
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        {
          Movement_Go2Target_End();
          plan_i++;
        }
        break;
      case 2:

Movement_Go2Target_Start(plan_target_point[1][0],plan_target_point[1][1]);
        plan_i++;
        break;
      case 3:
        if(Movement_Go2Target_Arrive_Stat())
        {
          Movement_Go2Target_End();
          plan_i++;
        }
        break;
      case 4:

Movement_Go2Target_Start(plan_target_point[2][0],plan_target_point[2][1]);
        plan_i++;
        break;
      case 5:
        if(Movement_Go2Target_Arrive_Stat())
        {
          Movement_Go2Target_End();
          plan_i++;
        }
        break;
      case 6:

Movement_Go2Target_Start(plan_target_point[3][0],plan_target_point[3][1]);
        plan_i++;
        break;
      case 7:
        if(Movement_Go2Target_Arrive_Stat())
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
      Movement_Go2Target_End();
      plan_i = 0;
    }
    break;
  }

}

void loop()
{

 static uint32_t last_t_rc,last_t_ahrs, last_print, last_compass,last_kp_heading;
 static double gps_lat,gps_lon,gps_alt,gps_heading;
 static float gps_distance2target,gps_bearing;
 static double compass_heading;
 static uint8_t gps_fixtype,gps_numsat;
 uint32_t now = hal.scheduler->micros();
 uint32_t now_ms = hal.scheduler->millis();

 static uint32_t loop_last_t;
 uint32_t loop_time;

 static uint8_t last_sw_ele_stat;
 static uint8_t last_sw_ail_stat;
 static uint8_t last_sw_rud_stat;

 loop_time = now - loop_last_t;
 loop_last_t = now;

 if(now_ms - last_t_rc > 50)
 {
  last_t_rc = now_ms;
  if(Remote_Sw_Ele_Stat() && !last_sw_ele_stat)
  {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    //Movement_Baro_LockAlt_Start(BARO_Get_Alt());
    //Movement_LockHeading2Target_Set_Target(13.727856,100.771976);
    //Movement_LockHeading2Target_Start();
    //Movement_ConstForward_Start(200);
    //Movement_Go2Target_Start(13.730208,100.772212);
    //Movement_Go2Home_Start();
    Movement_Adj2Target_Start(home_lat,home_lon);
  }

  if(!Remote_Sw_Ele_Stat() && last_sw_ele_stat)
  {
   //Movement_Baro_LockAlt_End();
   //Movement_LockHeading2Target_End();
   //Movement_ConstForward_End();
   //Movement_Go2Target_End();
   //Movement_Go2Home_End();
   Movement_Adj2Target_End();

  }


 if(Remote_Sw_Ail_Stat() && !last_sw_ail_stat)
 {
  //plan_i = 0;
  //Logger_Position_Start();
 }


 if(Remote_Sw_Ail_Stat() && !last_sw_ail_stat)
 {
  //Movement_Baro_LockAlt_Start(BARO_Get_Alt());
  //Movement_LockHeading2Target_Set_Target(13.727856,100.771976);
  //Movement_LockHeading2Target_Start();
  //Movement_ConstForward_Start(200);
  //Movement_Go2Target_Start(TARGET_LAT,TARGET_LON);
  //Plan_Running();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารนี้ทุกครั้งที่มีการนำไปใช้

```
    Camera_Control_Take();

  }

  if(!Remote_Sw_Ail_Stat() && last_sw_ail_stat)
  {
   //Movement_Baro_LockAlt_End();
   //Movement_LockHeading2Target_End();
   //Movement_ConstForward_End();
   //Movement_Go2Target_End();
   //Logger_Position_End();
  }

  if(Remote_Sw_Rud_Stat() && !last_sw_rud_stat)
  {
   //Camera_Control_Take();
   //Movement_Baro_LockAlt_Start(BARO_Get_Alt());
   BARO_Cal();
   //Movement_TakeOff_Start();
   Auto_Start();
  }

  if(!Remote_Sw_Rud_Stat() && last_sw_rud_stat)
  {
   //Movement_Baro_LockAlt_End();
   //Movement_TakeOff_End();
   Auto_End();
  }

  last_sw_ele_stat = Remote_Sw_Ele_Stat();
  last_sw_ail_stat = Remote_Sw_Ail_Stat();
  last_sw_rud_stat = Remote_Sw_Rud_Stat();
}

if(DJI_GPS_GPS_Avail())
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            gps_lat = DJI_GPS_Get_Lat();
            gps_lon = DJI_GPS_Get_Lon();
            gps_alt = DJI_GPS_Get_Alt();
            gps_fixtype = DJI_GPS_Get_FixType();
            gps_numsat = DJI_GPS_Get_NumSat();
            //gps_heading = DJI_GPS_Get_Heading();
            gps_heading = AHRS_Get_Yaw();

            gps_bearing = DJI_GPS_Get_Bearing(gps_lat,gps_lon,13.730201,100.77221);
            gps_distance2target =
DJI_GPS_Get_Distance(gps_lat,gps_lon,13.730201,100.77221);

            //hal.console->printf("Lat : %.6lf ; Lon : %.6lf ; Alt : %.2f ; Hd : %.2f ;
Distance : %.2f ; Brng : %.2f ; NumSat : %d ; Speed : %f Bat :
%.2f\r\n",gps_lat,gps_lon,gps_alt,gps_heading,gps_distance2target,gps_bearing,gps_nu
msat,DJI_GPS_Get_Speed(),Battery_Read());
            //hal.console->printf("Lat : %.7lf ; Lon : %.7lf ; Alt : %.7lf ; FixType : %d ;
NumSat : %d\r\n",gps_lat,gps_lon,gps_alt,gps_fixtype,gps_numsat);
          }

          if(DJI_GPS_Compass_Avail())
          {
            compass_heading = DJI_GPS_Get_Heading();
          }



          if(now_ms - last_print > 100)
          {
            last_print = now_ms;
            //hal.console->printf("Lat : %.6lf ; Lon : %.6lf ; Alt : %.2f ; Hd : %.2f ;
Distance : %.2f ; Brng : %.2f ; NumSat : %d ; Speed : %f Bat :
%.2f\r\n",gps_lat,gps_lon,gps_alt,gps_heading,gps_distance2target,gps_bearing,gps_nu
msat,DJI_GPS_Get_Speed(),Battery_Read());
            //hal.console->printf("%d %d %d %d %d %d %d %d %d\r\n",hal.rcin-
>read(CH_1),hal.rcin->read(CH_2),hal.rcin->read(CH_3),hal.rcin->read(CH_4),hal.rcin-
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
>read(CH_5),hal.rcin->read(CH_6),hal.rcin->read(CH_7),hal.rcin-
>read(CH_8),Remote_Sw_B_Stat());
            //gps_distance2target =
DJI_GPS_Get_Distance(gps_lat,gps_lon,13.727493,100.772486);
            //hal.console->printf("%.2f , %.2f\r\n",BARO_Get_Alt(),BARO_Get_Pressure());
            //hal.console->printf("Baro Height : %.2f\r\n",BARO_Get_Alt());
            //hal.console->printf("%.2f\r\n",BARO_Get_Alt());
            //hal.console->printf("Loop time %ld\r\n",loop_time);
          }



          DJI_GPS_Update();
          BARO_Update();
          AHRS_Update();
          Remote_Update();
          Movement_Update();
          Logger_Update();
          Auto_Update();

        }


AP_HAL_MAIN();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้