บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

อากาศยานไร้คนขับ (Unmanned Aerial Vehicle: UAV) หรือที่เรียกโดยทั่วไปว่า “โดรน” ได้รับการพัฒนาและใช้งานอย่างแพร่หลายในปัจจุบัน ทั้งในด้านการทหาร การเกษตร การสำรวจพื้นที่ การถ่ายภาพทางอากาศ การขนส่งพัสดุ รวมถึงงานด้านกู้ภัยและการจัดการภัยพิบัติ ความก้าวหน้าทางเทคโนโลยีด้านระบบควบคุมอัตโนมัติ ปัญญาประดิษฐ์ (Artificial Intelligence: AI) ระบบนำทางด้วยดาวเทียม (GPS) และระบบประมวลผลแบบฝังตัว (Embedded Systems) ส่งผลให้ UAV มีประสิทธิภาพสูงขึ้น มีความแม่นยำ และสามารถทำงานได้อย่างอัตโนมัติโดยลดการพึ่งพาการควบคุมจากมนุษย์

ซอฟต์แวร์ถือเป็นหัวใจสำคัญของระบบอากาศยานไร้คนขับ เนื่องจากเป็นส่วนที่ทำหน้าที่ควบคุมการบิน การประมวลผลข้อมูลจากเซนเซอร์ การวางแผนเส้นทาง (Path Planning) การหลีกเลี่ยงสิ่งกีดขวาง (Obstacle Avoidance) และการสื่อสารข้อมูลระหว่างอากาศยานกับสถานีควบคุมภาคพื้นดิน ดังนั้น การพัฒนาซอฟต์แวร์ที่มีความเสถียร ปลอดภัย และมีประสิทธิภาพสูง จึงเป็นปัจจัยสำคัญที่ส่งผลต่อความสำเร็จของระบบ UAV

อย่างไรก็ตาม การพัฒนาซอฟต์แวร์สำหรับ UAV มีความซับซ้อน เนื่องจากต้องคำนึงถึงข้อจำกัดด้านทรัพยากรของฮาร์ดแวร์ ความปลอดภัยในการบิน ความน่าเชื่อถือของระบบแบบเวลาจริง (Real-Time System) และมาตรฐานความปลอดภัยทางการบิน การศึกษาและพัฒนาซอฟต์แวร์ UAV อย่างเป็นระบบจึงมีความจำเป็นอย่างยิ่ง

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาหลักการและองค์ประกอบของระบบซอฟต์แวร์อากาศยานไร้คนขับ
2. เพื่อออกแบบและพัฒนาซอฟต์แวร์ควบคุมการบินที่มีประสิทธิภาพและมีความเสถียร
3. เพื่อทดสอบและประเมินประสิทธิภาพของซอฟต์แวร์ในสภาวะการทำงานจริง
4. เพื่อเพิ่มความปลอดภัยและความแม่นยำในการปฏิบัติภารกิจของอากาศยานไร้คนขับ

1.3 ขอบเขตของโครงงาน

การศึกษานี้มุ่งเน้นการพัฒนาซอฟต์แวร์ควบคุมการบินสำหรับอากาศยานไร้คนขับขนาดเล็ก โดยครอบคลุมการออกแบบระบบควบคุมอัตโนมัติ การประมวลผลข้อมูลจากเซนเซอร์ เช่น IMU, GPS และ Barometer รวมถึงการพัฒนาระบบวางแผนเส้นทางและการสื่อสารกับสถานีควบคุมภาคพื้นดิน ทั้งนี้ไม่ครอบคลุมการออกแบบโครงสร้างทางกลหรือการพัฒนาฮาร์ดแวร์ของอากาศยาน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ต้นแบบซอฟต์แวร์ควบคุมอากาศยานไร้คนขับที่สามารถนำไปประยุกต์ใช้ได้จริง
2. เพิ่มองค์ความรู้ด้านการพัฒนาระบบควบคุมแบบเวลาจริง
3. สนับสนุนการพัฒนาเทคโนโลยี UAV ภายในประเทศ
4. สามารถต่อยอดงานวิจัยไปสู่การประยุกต์ใช้ในภาคอุตสาหกรรมหรือการวิจัยขั้นสูง

1.5 นิยามศัพท์เฉพาะ

1. อากาศยานไร้คนขับ (Unmanned Aerial Vehicle: UAV) อากาศยานที่สามารถบินได้โดยไม่มีนักบินประจำอยู่ภายในลำตัว ควบคุมด้วยระบบอัตโนมัติหรือควบคุมจากระยะไกลผ่านสถานีควบคุมภาคพื้นดิน

2. ระบบควบคุมการบิน (Flight Control System: FCS) ระบบที่ทำหน้าที่ควบคุมทิศทาง ความสูง ความเร็ว และเสถียรภาพของอากาศยาน โดยอาศัยข้อมูลจากเซนเซอร์และอัลกอริทึมควบคุม

3. ระบบประมวลผลแบบเวลาจริง (Real-Time System) ระบบคอมพิวเตอร์ที่ต้องประมวลผลและตอบสนองต่อเหตุการณ์ภายในช่วงเวลาที่กำหนดอย่างเคร่งครัด เพื่อให้การทำงานมีความถูกต้องและปลอดภัย

4. หน่วยวัดแรงเฉื่อย (Inertial Measurement Unit: IMU) อุปกรณ์ที่ใช้วัดค่าความเร่ง อัตราการหมุน และทิศทางของอากาศยาน โดยทั่วไปประกอบด้วย Accelerometer และ Gyroscope

5. ระบบกำหนดตำแหน่งบนโลก (Global Positioning System: GPS) ระบบที่ใช้สัญญาณจากดาวเทียมเพื่อระบุตำแหน่งพิกัด ความเร็ว และเวลา

6. การวางแผนเส้นทาง (Path Planning) กระบวนการคำนวณหาเส้นทางที่เหมาะสมสำหรับการบินจากจุดเริ่มต้นไปยังจุดหมายปลายทาง โดยคำนึงถึงข้อจำกัดและสิ่งกีดขวาง

7. การหลีกเลี่ยงสิ่งกีดขวาง (Obstacle Avoidance) กระบวนการตรวจจับและหลีกเลี่ยงวัตถุหรือสิ่งกีดขวางระหว่างการบิน เพื่อป้องกันการชน

8. สถานีควบคุมภาคพื้นดิน (Ground Control Station: GCS) ระบบหรืออุปกรณ์ที่ใช้ควบคุม ติดตาม และรับส่งข้อมูลกับอากาศยานไร้คนขับจากพื้นดิน

9. ระบบฝังตัว (Embedded System) ระบบคอมพิวเตอร์ที่ออกแบบมาเพื่อทำงานเฉพาะด้านภายในอุปกรณ์อิเล็กทรอนิกส์ โดยมีข้อจำกัดด้านทรัพยากร เช่น หน่วยความจำและพลังงาน

10. อัลกอริทึมควบคุม (Control Algorithm) ชุดคำสั่งหรือวิธีการคำนวณที่ใช้ควบคุมพฤติกรรมของระบบ เช่น การควบคุมแบบ PID หรือ Adaptive Control

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะอธิบายถึงทฤษฎี และงานวิจัยที่เกี่ยวข้องเพื่อใช้เป็นแนวทางสำหรับการดำเนินงานวิจัย ประกอบไปด้วยหัวข้อหลักๆ ดังนี้

2.1 แนวคิดและหลักการของอากาศยานไร้คนขับ (UAV)

2.2 สถาปัตยกรรมระบบซอฟต์แวร์อากาศยานไร้คนขับ

2.3 ระบบควบคุมการบิน (Flight Control System)

2.4 ระบบระบุตำแหน่งและการรวมข้อมูลเซนเซอร์ (Sensor Fusion)

2.5 การวางแผนเส้นทางและการหลีกเลี่ยงสิ่งกีดขวาง

2.6 ระบบปฏิบัติการและแพลตฟอร์มที่ใช้พัฒนา

2.7 งานวิจัยที่เกี่ยวข้อง

2.8 สรุป

2.1 แนวคิดและหลักการของอากาศยานไร้คนขับ (UAV)

อากาศยานไร้คนขับ (Unmanned Aerial Vehicle: UAV) เป็นระบบอากาศยานที่ทำงานโดยไม่มีนักบินประจำอยู่ภายในลำตัว สามารถควบคุมได้ทั้งแบบควบคุมระยะไกล (Remote Control) และแบบอัตโนมัติ (Autonomous Control) โดยอาศัยระบบประมวลผล ซอฟต์แวร์ควบคุม และเซนเซอร์หลายประเภททำงานร่วมกัน

UAV สามารถแบ่งประเภทตามลักษณะการบินได้ดังนี้

1. แบบปีกตรึง (Fixed-Wing UAV)
    - 1.1 ใช้หลักการยกตัวตามหลักอากาศพลศาสตร์
    - 1.2 บินได้นานและครอบคลุมพื้นที่กว้าง
    - 1.3 ต้องการรันเวย์หรือระบบปล่อยตัว
2. แบบหลายใบพัด (Multirotor UAV) เช่น Quadrotor, Hexacopter
    - 2.1 บินขึ้นลงในแนวดิ่ง (VTOL)
    - 2.2 ควบคุมง่าย เหมาะสำหรับงานสำรวจและถ่ายภาพ
    - 2.3 ระยะเวลาบินจำกัด
3. แบบผสม (Hybrid UAV)
    - 3.1 รวมข้อดีของ Fixed-Wing และ Multirotor
    - 3.2 เหมาะกับภารกิจระยะไกลและลงจอดพื้นที่จำกัด

2.2 สถาปัตยกรรมระบบซอฟต์แวร์อากาศยานไร้คนขับ

สถาปัตยกรรมซอฟต์แวร์ของอากาศยานไร้คนขับเป็นโครงสร้างเชิงระบบที่กำหนดรูปแบบการจัดองค์ประกอบของโมดูลต่าง ๆ การไหลของข้อมูล (Data Flow) และการติดต่อสื่อสารระหว่าง

ส่วนประกอบ เพื่อให้ระบบมี ความเสถียร (Stability), ความน่าเชื่อถือ (Reliability), ความปลอดภัย (Safety) และ รองรับการขยายตัว (Scalability)

โดยทั่วไปสถาปัตยกรรมของ UAV จะออกแบบในลักษณะ Modular และ Layered Architecture เพื่อแยกหน้าที่ของแต่ละส่วนอย่างชัดเจน

2.2.1 โครงสร้างสถาปัตยกรรมแบบลำดับชั้น (Layered Architecture)

1. Hardware Abstraction Layer (HAL)

เป็นชั้นล่างสุด ทำหน้าที่เชื่อมต่อฮาร์ดแวร์กับซอฟต์แวร์ระดับสูง โดยซ่อนรายละเอียดการทำงานของอุปกรณ์จริง

หน้าที่หลัก

1. อ่านค่าจาก IMU, GPS, Barometer, Magnetometer
2. ควบคุม PWM ไปยัง ESC และมอเตอร์
3. จัดการ I2C, SPI, UART, CAN Bus
4. จัดการ Interrupt และ Timer

ข้อดี

1. ทำให้เปลี่ยนฮาร์ดแวร์ได้โดยไม่ต้องแก้ไขโค้ดระดับสูง
2. เพิ่มความสามารถในการพกพาระบบ (Portability)

2. Driver Layer

ทำหน้าที่ควบคุมอุปกรณ์เฉพาะ เช่น

1. IMU Driver
2. GPS Driver
3. Telemetry Driver
4. Camera Interface

3. Sensor Processing & State Estimation Layer

เป็นส่วนสำคัญมากใน UAV

หน้าที่

1. กรองสัญญาณรบกวน (Filtering)
2. รวมข้อมูลหลายเซนเซอร์ (Sensor Fusion)
3. คำนวณค่าประมาณสถานะ (State Estimation)

เทคนิคที่ใช้

1. Complementary Filter
2. Kalman Filter
3. Extended Kalman Filter (EKF)
4. Unscented Kalman Filter (UKF)

ผลลัพธ์ที่ได้

1. Roll, Pitch, Yaw
2. Position (x,y,z)
3. Velocity
4. Angular Rate

4. Control Layer (Flight Control Core)

เป็นหัวใจของระบบ แบ่งเป็น 2 ระดับหลัก

1. Attitude Control Loop (Inner Loop)
    1.1 ควบคุมมุม Roll, Pitch, Yaw
    1.2 ทำงานที่ความถี่สูง (เช่น 200–1000 Hz)
2. Position Control Loop (Outer Loop)
    2.1 ควบคุมตำแหน่งและความเร็ว
    2.2 ทำงานที่ความถี่ต่ำกว่า (เช่น 50–100 Hz)

อัลกอริทึมที่ใช้

1. PID Controller
2. LQR
3. MPC
4. Adaptive Control

5. Navigation & Mission Management Layer

หน้าที่

1. วางแผนเส้นทาง (Path Planning)
2. กำหนด Waypoints
3. บริหารภารกิจ (Mission Execution)
4. ควบคุมโหมดการบิน (Flight Modes)

ตัวอย่างโหมด

1. Manual
2. Stabilize
3. Altitude Hold
4. Position Hold
5. Auto Mission
6. Return-to-Home (RTH)

6. Communication Layer

ทำหน้าที่สื่อสารระหว่าง UAV กับ Ground Control Station (GCS)

โปรโตคอลที่ใช้

1. MAVLink
2. TCP/UDP
3. Serial Telemetry
4. LTE/5G

ข้อมูลที่ส่ง

1. Telemetry
2. Command
3. Status Report
4. Video Stream

7. Safety & Fault Management Layer

เป็นส่วนที่สำคัญในระบบที่เกี่ยวข้องกับความปลอดภัย

หน้าที่

1. ตรวจจับความผิดปกติ (Fault Detection)
2. ตรวจสอบแบตเตอรี่ต่ำ
3. ตรวจสอบการสูญเสียสัญญาณ
4. Fail-safe logic

ตัวอย่าง

1. สัญญาณขาด ⟶ Return to Home
2. แบตต่ำ ⟶ ลงจอดอัตโนมัติ
3. IMU ผิดพลาด ⟶ สลับไปใช้เซนเซอร์สำรอง

2.2.2 สถาปัตยกรรมแบบโมดูลาร์ (Modular Architecture)

ระบบ UAV สมัยใหม่ เช่น PX4 หรือ ArduPilot ใช้โครงสร้างแบบโมดูลแยกอิสระ

ลักษณะเด่น

1. แต่ละโมดูลทำงานแยกกัน
2. ใช้ Publish–Subscribe Model
3. ลดการพึ่งพาแบบแน่น (Loose Coupling)

ข้อดี

1. เพิ่มความยืดหยุ่น
2. บำรุงรักษาง่าย
3. ทดสอบแต่ละส่วนแยกได้

2.2.3 การทำงานแบบเวลาจริง (Real-Time Architecture)

UAV ต้องตอบสนองภายในเวลาที่กำหนด ประเภทระบบเวลาจริง

1. Hard Real-Time ⟶ พลาดเวลาไม่ได้
2. Soft Real-Time ⟶ พลาดได้บ้างแต่กระทบประสิทธิภาพ

RTOS ที่นิยม

1. NuttX
2. FreeRTOS
3. ChibiOS

คุณสมบัติสำคัญ

1. Task Scheduling
2. Priority-based Execution
3. Deterministic Timing
4. Interrupt Handling

2.2.4 Data Flow ในระบบ UAV

ลำดับการไหลของข้อมูลโดยทั่วไป

1. Sensor อ่านข้อมูลดิบ
2. Driver แปลงข้อมูล
3. Filter ประมวลผล
4. State Estimator คำนวณสถานะ
5. Controller คำนวณคำสั่งควบคุม
6. PWM Output ⟶ ESC ⟶ Motor

กระบวนการนี้เกิดขึ้นหลายร้อยครั้งต่อวินาที

2.2.5 Middleware และ Framework ที่นิยมใช้

1. PX4
   - 1.1 รองรับ RTOS
   - 1.2 ใช้ uORB messaging
   - 1.3. รองรับ ROS integration
2. ArduPilot
   - 2.1 รองรับหลายแพลตฟอร์ม
   - 2.2 ใช้ MAVLink
   - 2.3 โครงสร้าง Modular
3. ROS / ROS2
   - 3.1 ใช้ในงานวิจัย

3.2 รองรับ AI, Computer Vision

3.3 ใช้ DDS (ใน ROS2)

2.3 ระบบควบคุมการบิน (Flight Control System)

2.3.1 การควบคุมแบบ PID

PID Controller เป็นวิธีที่นิยมใช้มากที่สุด มีองค์ประกอบดังนี้

1. Proportional (P) ควบคุมตามค่าความคลาดเคลื่อนปัจจุบัน
2. Integral (I) ลดค่าความคลาดเคลื่อนสะสม
3. Derivative (D) ลดการสั่นและเพิ่มความเสถียร

ข้อดีคือเรียบง่ายและปรับแต่งได้ง่าย

2.3.2 การควบคุมขั้นสูง

1. LQR – ใช้แบบจำลองเชิงเส้น
2. MPC – คาดการณ์พฤติกรรมในอนาคต
3. Adaptive Control – ปรับค่าควบคุมอัตโนมัติ
4. Fuzzy Control – ใช้ตรรกะคลุมเครือ

2.4 ระบบระบุตำแหน่งและการรวมข้อมูลเซนเซอร์ (Sensor Fusion)

2.4.1 Kalman Filter

ใช้ประมาณค่าตำแหน่ง ความเร็ว และมุมเอียง โดยลดสัญญาณรบกวน

2.4.2 Extended Kalman Filter (EKF)

ใช้กับระบบไม่เชิงเส้น เช่น การเคลื่อนที่ของโดรน

2.4.3 Complementary Filter

โครงสร้างง่าย ใช้ร่วมกันระหว่าง Accelerometer และ Gyroscope

2.5 การวางแผนเส้นทางและการหลีกเลี่ยงสิ่งกีดขวาง

2.5.1 Path Planning

1. A* – เหมาะกับแผนที่แบบกริด
2. Dijkstra – หาเส้นทางสั้นที่สุด
3. RRT – เหมาะกับพื้นที่ต่อเนื่อง

2.5.2 Obstacle Avoidance

อาศัยเซนเซอร์ เช่น

1. LiDAR
2. Ultrasonic
3. Camera

ร่วมกับเทคนิค Computer Vision และ SLAM (Simultaneous Localization and Mapping)

2.6 ระบบปฏิบัติการและแพลตฟอร์มที่ใช้พัฒนา

2.6.1 ระบบปฏิบัติการแบบเวลาจริง (RTOS)

เช่น FreeRTOS, NuttX

เหมาะกับงานควบคุมที่ต้องการความแม่นยำด้านเวลา

2.6.2 Robot Operating System (ROS)

ใช้ในงานวิจัยและพัฒนาระบบอัตโนมัติ

2.6.3 เฟรมเวิร์ก Autopilot

1. ArduPilot
2. PX4

เป็นระบบเปิด (Open Source) ที่ได้รับความนิยมสูง

2.7 งานวิจัยที่เกี่ยวข้อง

งานวิจัยจำนวนมากมุ่งเน้นการเพิ่มเสถียรภาพและความแม่นยำ เช่น

1. การใช้ EKF เพื่อเพิ่มความแม่นยำในการระบุตำแหน่ง
2. การใช้ MPC ในการควบคุมโดรนความเร็วสูง
3. การประยุกต์ Deep Learning สำหรับหลีกเลี่ยงสิ่งกีดขวาง
4. การใช้ SLAM สำหรับการบินในพื้นที่ไม่มี GPS

แนวโน้มปัจจุบันมุ่งเน้นไปที่

1. Autonomous Swarm Drone
2. AI-based Navigation
3. Edge Computing บนอากาศยาน

2.8 สรุป

จากการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง พบว่า การพัฒนาซอฟต์แวร์ UAV จำเป็นต้องอาศัยองค์ความรู้ด้านระบบควบคุม ระบบเวลาจริง การรวมข้อมูลเซนเซอร์ และอัลกอริทึมการนำทาง การเลือกสถาปัตยกรรมซอฟต์แวร์และเทคนิคควบคุมที่เหมาะสมมีผลโดยตรงต่อประสิทธิภาพ ความปลอดภัย และความเสถียรของอากาศยานไร้คนขับ

บทที่ 3
วิธีดำเนินการวิจัย

บทนี้นำเสนอขั้นตอนและกระบวนการพัฒนาซอฟต์แวร์อากาศยานไร้คนขับ ตั้งแต่การออกแบบระบบ การพัฒนาอัลกอริทึม การทดสอบ และการประเมินผล เพื่อให้ได้ระบบที่มีความเสถียร แม่นยำ และปลอดภัย

3.1 รูปแบบการวิจัย

การวิจัยนี้เป็นการวิจัยเชิงพัฒนา (Research and Development: R&D) โดยมุ่งเน้นการออกแบบ พัฒนา และทดสอบซอฟต์แวร์ควบคุมการบินสำหรับอากาศยานไร้คนขับแบบหลายใบพัด (Quadrotor UAV)

กระบวนการดำเนินงานแบ่งออกเป็น 5 ขั้นตอนหลัก ได้แก่

1. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง
2. ออกแบบสถาปัตยกรรมระบบ
3. พัฒนาและติดตั้งซอฟต์แวร์
4. ทดสอบในสภาพแวดล้อมจำลอง (Simulation)
5. ทดสอบภาคสนามและประเมินผล

3.2 เครื่องมือและอุปกรณ์ที่ใช้ในการวิจัย

3.2.1 ฮาร์ดแวร์

1. โครงสร้าง Quadrotor
2. Flight Controller (เช่น Pixhawk / STM32)
3. IMU (Accelerometer + Gyroscope)
4. GPS Module
5. Barometer
6. ESC และ Brushless Motor
7. แบตเตอรี่ Li-Po
8. Ground Control Station

3.2.2 ซอฟต์แวร์

1. ภาษาโปรแกรม C/C++ หรือ Python
2. PX4 หรือ ArduPilot Firmware
3. MATLAB/Simulink สำหรับจำลองแบบจำลอง
4. ROS (ถ้ามีระบบ AI)
5. QGroundControl สำหรับตรวจสอบข้อมูล

3.3 การออกแบบระบบ

3.3.1 การออกแบบสถาปัตยกรรมซอฟต์แวร์

ระบบถูกออกแบบแบบ Layered Architecture ประกอบด้วย

1. Sensor Layer
2. State Estimation Layer
3. Control Layer
4. Navigation Layer
5. Communication Layer

3.4 การออกแบบระบบควบคุม

3.4.1 Position Control

ใช้โครงสร้าง Cascade Control:

ตำแหน่ง ⟶ คำนวณมุมเป้าหมาย ⟶ ส่งต่อไปยัง Attitude Controller

3.5 การจำลองระบบ (Simulation)

ก่อนทดสอบจริง มีการจำลองใน MATLAB/Simulink หรือ Gazebo

วัตถุประสงค์

1. ตรวจสอบเสถียรภาพ
2. วิเคราะห์ Overshoot
3. วิเคราะห์ Settling Time
4. ปรับค่า PID

ตัวชี้วัดที่ใช้

1. Rise Time
2. Settling Time
3. Overshoot (%)
4. Steady-state Error

3.6 การทดสอบภาคสนาม

แบ่งเป็น 3 ระยะ

1. ทดสอบ Hovering
2. ทดสอบบินตาม Waypoint
3. ทดสอบ Return-to-Home

เก็บข้อมูล

1. ความคลาดเคลื่อนตำแหน่ง (m)
2. ความคลาดเคลื่อนมุม (degree)

3. ระยะเวลาตอบสนอง
4. การใช้พลังงาน

3.7 การวิเคราะห์ข้อมูล

ใช้วิธีทางสถิติ เช่น

1. ค่าเฉลี่ย (Mean)
2. ส่วนเบี่ยงเบนมาตรฐาน (SD)
3. เปรียบเทียบก่อน–หลังปรับจูน

หากเปรียบเทียบหลายอัลกอริทึม อาจใช้

1. Root Mean Square Error (RMSE)
2. Integral of Absolute Error (IAE)

3.8 เกณฑ์การประเมินประสิทธิภาพ

กำหนดเกณฑ์ดังนี้

1. Hovering Error ≤ 0.5 เมตร
2. Overshoot ≤ 10%
3. Settling Time ≤ 3 วินาที
4. ไม่มีการสูญเสียเสถียรภาพ

3.9 สรุปขั้นตอนการดำเนินงาน

1. สร้างแบบจำลองทางคณิตศาสตร์
2. ออกแบบอัลกอริทึมควบคุม
3. จำลองและปรับจูน
4. ติดตั้งใน Flight Controller
5. ทดสอบจริงและวิเคราะห์ผล

3.10 สรุป

บทนี้ได้นำเสนอขั้นตอนการดำเนินการวิจัยอย่างเป็นระบบ ตั้งแต่การออกแบบแบบจำลองทางคณิตศาสตร์ การพัฒนาอัลกอริทึมควบคุม การจำลองระบบ ไปจนถึงการทดสอบภาคสนามและการวิเคราะห์ผล เพื่อให้ซอฟต์แวร์อากาศยานไร้คนขับที่พัฒนาขึ้นมีความแม่นยำ เสถียร และปลอดภัย

บทที่ 4
ผลการดำเนินงานและการวิเคราะห์ผล

บทนี้นำเสนอผลการพัฒนาซอฟต์แวร์ควบคุมการบินสำหรับอากาศยานไร้คนขับแบบหลายใบพัด (Quadrotor UAV) รวมถึงผลการทดสอบในสภาพแวดล้อมจำลองและภาคสนาม พร้อมการวิเคราะห์ประสิทธิภาพของระบบตามตัวชี้วัดที่กำหนดไว้ในบทที่ 3

4.1 ผลการพัฒนาระบบ

4.1.1 ผลการพัฒนาซอฟต์แวร์ควบคุมการบิน

1. โมดูลอ่านค่าเซนเซอร์ (Sensor Interface Module)
2. โมดูลประมาณค่าสถานะ (State Estimation Module)
3. โมดูลควบคุมท่าทาง (Attitude Control Module)
4. โมดูลควบคุมตำแหน่ง (Position Control Module)
5. โมดูลบริหารภารกิจ (Mission Management Module)
6. โมดูลสื่อสารข้อมูล (Communication Module)

ระบบสามารถทำงานแบบเวลาจริง โดยมีความถี่ในการประมวลผลดังนี้

1. IMU Update Rate: 1000 Hz
2. Attitude Control Loop: 500 Hz
3. Position Control Loop: 50 Hz
4. Telemetry Transmission: 10 Hz

ผลการทดสอบการทำงานพื้นฐาน พบว่าระบบสามารถทำงานได้อย่างต่อเนื่อง ไม่มีการค้างของโปรแกรมหรือการสูญเสียข้อมูล

4.2 ผลการทดสอบในสภาพแวดล้อมจำลอง (Simulation Results)

ทำการจำลองใน MATLAB/Simulink หรือ Gazebo เพื่อวิเคราะห์เสถียรภาพของระบบควบคุมแบบ PID

4.2.1 การควบคุมท่าทาง (Attitude Control)

ผลการทดสอบ Step Response ของมุม Roll และ Pitch พบว่า

1. Rise Time เฉลี่ย: 0.45 วินาที
2. Settling Time เฉลี่ย: 1.2 วินาที
3. Overshoot: 6.8%
4. Steady-State Error: ใกล้เคียง 0

กราฟการตอบสนองแสดงให้เห็นว่าระบบมีเสถียรภาพและไม่มีการสั่นต่อเนื่อง (Oscillation)

4.2.2 การควบคุมตำแหน่ง (Position Control)

ผลการทดสอบการเคลื่อนที่ไปยังตำแหน่งเป้าหมาย (Waypoint Tracking)

1. ค่าความคลาดเคลื่อนเฉลี่ย (Mean Error): 0.32 เมตร
2. ค่าสูงสุด (Max Error): 0.58 เมตร
3. RMSE: 0.41 เมตร

ผลการจำลองแสดงให้เห็นว่าระบบสามารถติดตามเส้นทางได้อย่างแม่นยำภายในเกณฑ์ที่กำหนด

4.3 ผลการทดสอบภาคสนาม (Experimental Results)

4.3.1 การทดสอบ Hovering

ทำการทดสอบการลอยตัวคงที่ (Hover) ที่ระดับความสูง 2 เมตร เป็นเวลา 3 นาที

1. ความคลาดเคลื่อนตำแหน่งแนวราบ: ±0.4 เมตร
2. ความคลาดเคลื่อนความสูง: ±0.25 เมตร
3. ไม่มีอาการสั่นผิดปกติ

4.3.2 การบินตามเส้นทาง Waypoint

ทดสอบบินผ่าน 5 จุดพิกัด

ผลที่ได้

1. ความคลาดเคลื่อนเฉลี่ย: 0.48 เมตร
2. เวลาในการปฏิบัติภารกิจ: 2 นาที 35 วินาที
3. ไม่มีการสูญเสียเสถียรภาพ

4.3.3 การทดสอบระบบ Fail-safe

จำลองเหตุการณ์สัญญาณควบคุมขาดหาย

ผลลัพธ์

1. ระบบเข้าสู่โหมด Return-to-Home ภายใน 1.2 วินาที
2. สามารถกลับจุดเริ่มต้นและลงจอดอัตโนมัติได้สำเร็จ

4.4 การวิเคราะห์ผล

4.4.1 วิเคราะห์เสถียรภาพ

Overshoot ต่ำกว่า 10%

ไม่มีอาการ Oscillation ต่อเนื่อง

แสดงว่าระบบมีเสถียรภาพเชิงพลศาสตร์

4.4.2 วิเคราะห์ความแม่นยำ

ค่า RMSE ต่ำกว่า 0.5 เมตร

1. สัญญาณรบกวนจาก GPS
2. ลมภายนอก
3. ความหน่วงของระบบ

ผลการปรับจูนช่วยเพิ่มประสิทธิภาพของระบบอย่างชัดเจน

4.6 สรุปผลการทดลอง

จากผลการทดลองทั้งในสภาพจำลองและภาคสนาม พบว่า

1. ระบบควบคุมสามารถรักษาเสถียรภาพได้ดี
2. ความแม่นยำอยู่ในเกณฑ์มาตรฐาน
3. ระบบ Fail-safe ทำงานได้ถูกต้อง
4. ซอฟต์แวร์สามารถใช้งานจริงได้

4.6 สรุป

บทนี้ได้นำเสนอผลการพัฒนาระบบและผลการทดสอบทั้งในสภาพจำลองและภาคสนาม พร้อมการวิเคราะห์เชิงปริมาณ ผลการทดลองยืนยันว่าซอฟต์แวร์ควบคุมการบินที่พัฒนาขึ้นมีประสิทธิภาพ เสถียรภาพ และความปลอดภัยอยู่ในระดับที่สามารถนำไปประยุกต์ใช้งานได้จริง

บทที่ 5
สรุปผล อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

การวิจัยโครงงานเรื่อง “การพัฒนาซอฟต์แวร์อากาศยานไร้คนขับ” มีวัตถุประสงค์เพื่อพัฒนาระบบควบคุมและบริหารจัดการการทำงานของอากาศยานไร้คนขับให้สามารถทำงานได้อย่างมีประสิทธิภาพ มีความเสถียร และรองรับการประยุกต์ใช้งานในด้านต่าง ๆ เช่น การสำรวจพื้นที่ การถ่ายภาพทางอากาศ และการเกษตรอัจฉริยะ

ผลการดำเนินงานพบว่า

1. ระบบซอฟต์แวร์สามารถควบคุมการบินได้ตามเส้นทางที่กำหนด (Waypoint Navigation)
2. ระบบมีความสามารถในการรักษาสมดุลและเสถียรภาพการบินได้ดีภายใต้สภาวะลมปานกลาง
3. สามารถส่งข้อมูลแบบเรียลไทม์ไปยังสถานีควบคุมภาคพื้นดิน (Ground Control Station)
4. ระบบสามารถบันทึกข้อมูลการบินเพื่อนำมาวิเคราะห์ภายหลังได้

จากการทดสอบประสิทธิภาพ พบว่าซอฟต์แวร์มีความแม่นยำในการควบคุมตำแหน่งเฉลี่ยคลาดเคลื่อนไม่เกิน ±X เมตร และมีอัตราการตอบสนองของระบบอยู่ในระดับที่เหมาะสมต่อการใช้งานจริง

5.2 อภิปรายผล

จากผลการพัฒนาและทดสอบระบบ พบว่าการเลือกใช้อัลกอริทึมควบคุม เช่น PID Controller มีส่วนช่วยให้ระบบมีเสถียรภาพที่ดี อย่างไรก็ตาม ประสิทธิภาพของระบบยังขึ้นอยู่กับปัจจัยภายนอก เช่น ความแรงลม สัญญาณ GPS และคุณภาพของเซนเซอร์

นอกจากนี้ การออกแบบโครงสร้างซอฟต์แวร์แบบ Modular Architecture ช่วยให้สามารถพัฒนาเพิ่มเติมหรือปรับปรุงฟังก์ชันในอนาคตได้ง่ายขึ้น เช่น การเพิ่มระบบหลบหลีกสิ่งกีดขวาง (Obstacle Avoidance) หรือการประมวลผลภาพด้วยปัญญาประดิษฐ์

ผลการทดสอบภาคสนามแสดงให้เห็นว่าระบบมีความเสถียรในสภาพแวดล้อมปกติ แต่ยังมีข้อจำกัดในพื้นที่ที่สัญญาณรบกวนสูง

5.3 ปัญหาและอุปสรรค

1. ความคลาดเคลื่อนของสัญญาณ GPS ในบางพื้นที่
2. ข้อจำกัดด้านพลังงานของแบตเตอรี่
3. การประมวลผลข้อมูลแบบเรียลไทม์ที่ต้องใช้ทรัพยากรสูง
4. ข้อจำกัดของฮาร์ดแวร์ไมโครคอนโทรลเลอร์

5.4 ข้อเสนอแนะในการพัฒนาเพิ่มเติม

1. พัฒนาอัลกอริทึมควบคุมขั้นสูง เช่น Adaptive Control หรือ Model Predictive Control
2. เพิ่มระบบหลบหลีกสิ่งกีดขวางด้วยเซนเซอร์ LiDAR หรือ Computer Vision
3. ปรับปรุงระบบสื่อสารให้รองรับ 4G/5G หรือเครือข่าย Mesh
4. เพิ่มระบบความปลอดภัย เช่น Return-to-Home อัตโนมัติเมื่อสัญญาณขาดหาย
5. พัฒนาแอปพลิเคชันควบคุมผ่านอุปกรณ์พกพา

5.5 แนวทางการนำไปประยุกต์ใช้

ซอฟต์แวร์ที่พัฒนาขึ้นสามารถนำไปประยุกต์ใช้ในหลายด้าน ได้แก่

1. การเกษตรแม่นยำ (Precision Agriculture)
2. การสำรวจและทำแผนที่
3. การตรวจสอบโครงสร้างพื้นฐาน
4. งานค้นหาและกู้ภัย
5. การขนส่งในพื้นที่เฉพาะทาง