**ĐẠI HỌC QUỐC GIA HÀ NỘI**
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**


**Phạm Thành Nam**

# Phát triển chương trình điều khiển cân bằng Quadcopter trên hệ điều hành thời gian thực

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Công nghệ hàng không vũ trụ**

**HÀ NỘI – 2024**

**ĐẠI HỌC QUỐC GIA HÀ NỘI**
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**Phạm Thành Nam**

# Phát triển chương trình điều khiển cân bằng Quadcopter trên hệ điều hành thời gian thực

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Công nghệ hàng không vũ trụ**

**Cán bộ hướng dẫn: TS. Nguyễn Hoàng Quân**

**HÀ NỘI – 2024**

# Phát triển chương trình điều khiển cân bằng Quadcopter trên hệ điều hành thời gian thực (RTOS)

**Phạm Thành Nam**

Khóa QH-2020-I/CQ-A-E

**Tóm tắt:** Quadcopter là một dạng thiết bị bay không người lại (UAV) đang được ứng dụng nhiều hơn trong nhiều lĩnh vực bao gồm viễn thám, giám sát, giải trí, nghệ thuật, vận chuyển, nông nghiệp, cứu hộ và quân sự. Việc thiết kế UAV hiện nay yêu cầu tích hợp nhiều loại cảm biến, khả năng tự cân bằng khi bay và khả năng gửi dữ liệu về người dùng. Trong đồ án tốt nghiệp thực hiện các bước phát triển chương trình cân bằng Quadcopter trên hệ điều hành thời gian thực. Nội dung của đồ án tốt nghiệp tập trung về hệ thống cảm biến IMU, thuật toán lọc bù, thuật toán điều khiển PID, thuật toán điều khiển động cơ trên Quadcopter, hệ điều hành thời gian thực, các chuẩn giao tiếp phần cứng, giao thức MAVLink và quá trình thực hiện và thử nghiệm hệ thống.

***Từ khóa*:** UAV, Quadcopter, MAVLink, RTOS, FreeRTOS, PID

# LỜI CẢM ƠN

Lời đầu tiên, tôi xin gửi lời cảm ơn chân thành nhất tới hai thầy giáo hướng dẫn là TS Nguyễn Hoàng Quân và KS Trần Đăng Huy lời cảm ơn sâu sắc nhất, các thầy đã trực tiếp hướng dẫn tận tình, cho tôi những ý kiến đóng góp quý báu và chia sẻ những kinh nghiệm trong suốt thời gian thực hiện đề tài để em hoàn thành đồ án của mình.

Tôi cũng xin gửi lời cảm ơn tới tất cả các giảng viên trong Viện Công nghệ Hàng không Vũ trụ đã truyền đạt những kiến thức nền tàng quý báu cho em trong suốt hơn 4 năm học vừa qua.

Bên cạnh đó, tôi cũng đã nhận được sự ủng hộ nhiệt tình cũng như các ý kiến đóng góp từ bạn bè cùng lớp cũng như những người anh chị nơi tôi làm việc. Tôi cũng xin cảm ơn bạn Vũ Trung Hiếu cũng như những em sinh viên khóa dưới là những người bạn đã hỗ trợ tôi rất nhiều trong quá trình hoàn thành đồ án này. Tôi xin gửi lời cảm ơn đến gia đình và bạn bè đã hỗ trợ, chia sẻ và tạo điều kiện tốt nhất cho tôi học tập, nghiên cứu trong suốt thời gian những năm học vừa qua, cũng như thời gian hoàn thành đồ án tốt nghiệp này.

Cuối cùng, mặc dù đồ án đã hoàn thành tuy nhiên sẽ không thể tránh khỏi những sai sót do sự hạn chế về kiến thức cũng như kinh nghiệm nghiên cứu của bản thân. Tôi hi vọng sẽ nhận được những góp ý quý báu từ thầy cô cũng như bạn bè để có thể hoàn thiện hơn đồ án này và những dự án sau hơn nữa.

## LỜI CAM ĐOAN

Tôi là Phạm Thành Nam, sinh viên lớp K65 của Viện Công nghệ Hàng không Vũ trụ, trường Đại học Công nghệ - Đại Học Quốc Gia Hà Nội.

Tôi xin cam đoan toàn bộ nội dung, số liệu và dẫn chứng trong bài báo cáo này đều là kết quả nghiên cứu và tìm hiểu của tôi dưới sự giúp đỡ của giáo viên hướng dẫn TS Nguyễn Hoàng Quân và KS. Trần Đăng Huy trong quá trình thực hiện. Mọi tài liệu được sử dụng trong bài báo cáo này đều được nêu rõ nguồn gốc, không chỉnh sửa sai lệch nội dung gây ảnh hưởng tới tác giả.

Nếu có bất kì phát hiện gian lận hay sao chép nào tôi xin được chịu hoàn toàn trách nghiệm.

Hà Nội, ngày ..... tháng ….. năm 2024

Sinh viên

**Phạm Thành Nam**

# Danh sách hình ảnh

# Danh sách bảng

# Danh mục từ viết tắt và thuật ngữ

| Từ viết tắt | Tên đầy đủ | Ý nghĩa |
|---|---|---|
| ACK | Acknowledge | Xác nhận |
| ADC | Analog Digital Converter | Bộ chuyển đổi tin hiệu tương tự sang tín hiệu số |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| BLDC | BrushLess Direct Current | Không chổi than dòng một chiều |
| CAN | Controller Area Network | Mạng lưới điều khiển |
| CCR | Capture/Compare Register | Thang ghi so sánh/bắt bộ đếm |
| CPU | Central Processing Unit | Bộ điều khiển trung tâm |
| DAC | Digital Analog Converter | Bộ chuyển đổi tín hiệu số sang tín hiệu tương tự |
| DMP | Digital Motion Processor | Vi xử lý chuyển động |
| EDF | Earliest-Deadline-First | Ưu tiên đến hạn trước |
| EDF | Earliest Deadline First | Thuật toán lập lịch ưu tiên tác vụ đến hạn |
| ESC | Electronic Speed Controller | Thiết bị điều khiển vận tốc |
| FIFO | First-In-First-Out | Vào trước ra trước |
| FIFO | First In First Out | Thuật toán lập lịch vào trước ra trước |
| GPOS | General Purpose Operating System | Hệ điều hành nhiệm vụ chung |
| GPS | Global Positioning System | Hệ thống định vị toàn cầu |
| HPF | High Pass Filter | Bộ lọc thông cao |
| I2C | Inter-Integrated Circuit | Chuẩn truyền thông nối tiếp |
| ID | Identify | Mã nhận dạng |
| IMU | Inertial Measurement Unit | Đơn vị đo chuyển động của vật thể |
| LED | Light Emitting Diode | Diode phát quang |
| LPF | Low Pass Filter | Bộ lọc thông thấp |
| LSB | Least Significant Bit | Bit ít quan trọng nhất |
| MAVLink | Micro Air Vehicle Link | Phương thức giao tiếp phương triện trên không |
| MCU | MicroController Unit | Bộ điều khiển |
| MEMS | Micro-Electro-Mechanical Systems | Hệ thống điện tử kích cỡ micro |
| NACK | Non Acknowledge | Không xác nhận |
| OS | Opearating System | Hệ điều hành |
| PCB | Printed Circuit Board | Bảng mạch in |

| | | |
|---|---|---|
| PID | Proportional-Integral-Derivative | Bộ điều khiển độ lợi, tích phân, đạo hàm |
| PPM | Pulse Position Modulation | Điều chế xung bằng vị trí |
| PWM | Pulse Width Modulation | Điều chế xung bằng độ rộng |
| RAM | Random Access Memory | Bộ nhớ truy cập ngẫu nhiên |
| RC | Radio Control | Điều khiển vô tuyến |
| rpm | round per minute | Vòng trên phút |
| RPV | Remotely Piloted vehicle | Phương tiện điều khiển từ xa |
| RTOS | Real Time Operating System | Hệ điều hành thời gian thực |
| RX | Receiver | Đầu nhận tín hiệu |
| SCL | Serial Clock | Đường dẫn xung tín hiệu |
| SDA | Serial Data | Đường dẫn dữ liệu |
| SDIO | Secure Digital Input Output | Đường truyền tín hiệu số vào ra bảo mật |
| SPI | Serial Peripheral Interface | Kết nối ngoại vi nối tiếp |
| TX | Transmitter | Đầu truyền tín hiệu |
| UART | Universal Asynchronous Receiver/Transmitter | Chuẩn truyền thông truyền nhận không đồng bộ |
| UAV | Unmanned Aerial Vehicle | Phương tiện bay không người lái |
| USAR | Universal Asynchronous/Synchronous Receiver/Transmitteris | Chuẩn truyền thông truyền nhận vừa đồng bộ/không đồng bộ |
| USB | Universal Serial Bus | Chuẩn kết nối tuần tự |
| USB OTG | Universal Serial Bus On-The-Go | Chuẩn kỹ thuật kết nối thiết bị ngoại vi |

# Danh mục ký hiệu

| | |
|---|---|
| $P$ | Áp suất |
| $v$ | Vận tốc |
| $h$ | Độ cao |
| $g$ | Gia tốc trọng trường |
| $L$ | Lực nâng |
| $\rho$ | Mật độ chất lỏng |
| $D$ | Đường kính |
| $C_L$ | Hệ số lực nâng |
| $n$ | Vận tốc |
| $A_X, A_Y, Az$ | Gia tốc theo 3 trục X, Y, Z |
| $G_X, G_Y, Gz$ | Vận tốc góc theo 3 trục X, Y, Z |
| $M_X, M_Y, Mz$ | Từ trường theo 3 trục X, Y, Z |
| $\alpha$ | Hệ số lọc bù |
| $\tau$ | Khoảng thời gian cố định |
| $dt$ | Tần số lấy mẫu |
| $Kp, Ki, Kd$ | Hệ số độ lợi, hệ số tích phân, hệ số đạo hàm |
| μ | Đầu ra bộ PID |
| $e$ | Sai lệch đầu vào bộ PID |
| $\phi, \theta, \psi$ | Góc Roll, Pitch, Yaw |

# MỤC LỤC

# MỞ ĐẦU

## Tính cấp thiết của đề tài

Ngày nay, việc sử dụng các phương tiện bay không người lái (UAV) ngày càng trở nên phổ biến. Đây là tên gọi chung cho các loại thiết bị bay mà không có người điều khiển trong buồng lái, hoạt động tự động và có thể điều khiển và giám sát từ xa. Chúng được sử dụng thay thế con người trong nhiều lĩnh vực từ đời sống bao gồm vận chuyển, giải trí, nghệ thuật đến nghiên cứu như trong nông nghiệp, viễn thám và cả trong quân sự, cứu nạn do có khả năng điều hành tự động cũng như có thể hoạt động ở những nơi mà con người khó tiếp cận được trong thời gian ngắn. Việc ứng dụng UAV ở Việt Nam hiện đang ngày càng phát triển hơn tuy nhiên chưa có nhiều nghiên cứu đi sâu vào việc tích hợp giao điều khiển vào hệ điều hành thời gian thực nhằm kiểm soát hiệu soát hiệu suất cũng như giám sát hoạt động của UAV.

Xuất phát từ thực tế trên, đồ án đi sâu vào việc phát triển chương trình điều khiển cân bằng Quadcopter trên hệ điều hành thời gian thực (RTOS), qua đó nắm bắt lý thuyết và công nghệ nhằm áp dụng vào các nghiên cứu sau này.

## Ý nghĩa khoa học và thực tiễn

Về ý nghĩa khoa học, đề tài "Phát triển chương trình điều khiển cân bằng Quadcopter trên hệ điều hành thời gian thực (RTOS)" có thể ứng dụng phát triển cho các bài nghiên cứu sau này về UAV. Về ý nghĩa thực tiễn, đề tài này giúp làm chủ và ứng dụng công nghệ trong việc chế tạo UAV nhằm phục vụ hoạt động cho các lĩnh vực từ dân sự đến quân sự.

## Nội dung nghiên cứu đề tài

Nội dung nghiên cứu của đề tài bao gồm kiến thức cơ bản khi điều khiển Quadcopter, về các loại cảm biến, các giao tiếp chuẩn, giao thức truyền thông, bộ cân bằng PID, bộ lọc bù và hệ điều hành thời gian thực.

## Phương pháp nghiên cứu

Tìm hiểu các tài liệu có trong và ngoài nước.

Tìm hiểu và nắm vững cơ sở lý thuyết.

Thiết kế bo điều khiển và chương trình điều khiển bay, hoàn thiện sản phẩm và đánh giá khả năng hoạt động thông qua thử nghiệm.

## Cấu trúc của đồ án tốt nghiệp

Cấu trúc của đồ án được chia làm 4 chương trong đó **chương 1** trình bày các kiến thức tổng quan về UAV, Quadcopter và RTOS, **chương 2** trình bày các cơ sở lý thuyết về

Quadcopter, **chương 3** trình bày các bước thiết kế mạch và chương trình điều khiển bay, **chương 4** thiết lập thử nghiệm kiểm tra khả năng hoạt động.

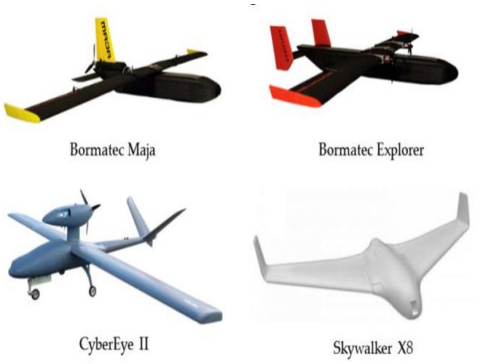
# CHƯƠNG 1: TỔNG QUAN

## 1.1. Giới thiệu về UAV

### 1.1.1. Khái niệm

Cụm từ thiết bị bay không người lái (UAV) bắt đầu được sử dụng vào đầu những năm 1990 để miêu tả các thiết bị bay tự động thay thế cho định nghĩa phương tiện điều khiển từ xa (RPV), được sử dụng trong chiến tranh Việt Nam và các cuộc chiến sau này. Trong từ điển, UAV được miêu tả là "Một thiết bị hoạt động trên không, không cần điều khiển trực tiếp từ phi công, sử dụng lực khí động để tạo lực nâng cho thiết bị, có thể bay tự động hoặc điều khiển từ xa, có thể thay thế và sửa chữa, và có thể mang hoặc không mang theo vũ khí. Các tên lửa đạn đạo và pháo binh không được coi là thiết bị bay không người lái". Sau này UAV được ứng dụng ngày càng nhiều trong cả lĩnh vực quân sự lẫn dân sự [1].

### 1.1.2. Phân loại UAV




***(a)** UAV cánh bằng [2]*


***(b)** UAV cánh quay [2]*

***(c)** UAV dạng hỗn hợp [3]*

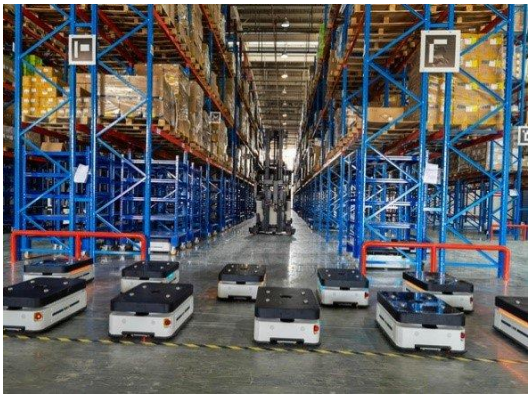
***Hình 1:** Phân loại một số UAV*

UAV thường được chia ra làm 3 biến thể chính là UAV cánh cố định hoặc cánh bằng, UAV cánh quạt hay đa rô-to và lai giữa cánh cố định với cánh quạt. UAV cánh

bằng có cánh dài theo chiều ngang (**Hình 1-a**), khi có lực đẩy về phía trước chênh lệch áp suất xuất hiện bên trên và dưới của cánh tạo ra lực nâng cho thiết bị, yêu cầu của loại UAV này là cần có đường băng để có thể cất cánh tuy nhiên tiêu tốn ít năng lượng để bay. UAV cánh quạt sử dụng lực nâng của cánh quạt để bay lên và di chuyển, UAV cánh quạt có thể có từ một rô-to như máy bay trực thăng đến nhiều rô-to như, Quadcopter, hexacopter để bay lên (**Hình 1-b**), UAV này sử dụng ít diện tích hơn để cất cánh và sử dụng nhiều năng lượng để cất cánh và di chuyển hơn UAV cánh bằng [2]. Loại UAV thứ ba, lai của 2 biến thể UAV trước, UAV này vừa có thể bay theo chiều ngang như UAV cánh bằng hoặc bay lên như UAV cánh quạt (**Hình 1-c**), UAV này được sử dụng trong điều kiện yêu cầu sự chuyển đổi linh hoạt giữa hai trạng thái cho các mục đích khác nhau [3].

### 1.1.3. Ứng dụng của UAV

***(a)*** *Trong nông nghiệp [6]*



***(b)*** *Trong vận chuyển [5]*

***(c)*** *Trong công tác cứu hộ [5]*

***(d)*** *Trong quân sự [5]*

***Hình 2:*** *Một số ứng dụng của UAV*

UAV hiện nay được ứng dụng trong rất nhiều lĩnh vực, một vài trong số các lĩnh vực có thể kể đến như trong nông nghiệp, vận chuyển, cứu hộ và quân sự. Trong nông

nghiệp UAV được dùng trong việc trồng và chăm sóc cây như gieo hạt, phun thuốc và tưới nước (**Hình 2-a**). UAV còn được dùng để quét khu vực để thu ảnh nhằm thu thập dữ liệu về cây trồng đẻ người nông dân có thể tìm phương án nâng cao năng suất hoặc đưa ra các giải quyết cho các sự cố về cây. Trong lĩnh vực vận chuyển, thay vì giao hàng theo phương thức truyền thống UAV đã được đưa vào thử nghiệm để nâng cao hiệu suất giao hàng (**Hình 2-b**) [4]. UAV được xác định trong việc phản ứng với thiện tại, bằng cách sử dụng GPS để xác định vị trí và cứu người bị nạn (**Hình 2-c**). UAV cũng được sử dụng trong quân sự để do thám địa hình và cung cấp dữ liệu trực quan (**Hình 2**-d) [5].

### 1.1.4. Các hướng nghiên cứu về UAV hiện nay

Một trong số những nghiên cứu mang tính đột phá hiện nay về UAV là về việc tích hợp trí tuệ nhân tạo (AI) vào việc giáo sát tự động. Các thiết bị UAV hiện đại ngày nay đã được tích hợp thêm trí tuệ nhân tạo vào hệ thống giúp cho phép thực hiện các nhiệm vụ mang tính ít phụ thuộc vào sự điều khiển và can thiệp của con người. Hệ thống này có thể đưa ra kết luận sát với thời gian thực, có thể xác định vật thể và có thể tự sửa chữa. Việc tích hợp trí tuệ nhân tạo vào UAV không chỉ cải thiện hiệu suất của UAV mà còn mở rộng các ứng dụng của UAV trong nhiều lĩnh vực đặc biệt là cách lĩnh vực yêu cầu sự chính xác cao [7].

Một số cải tiến về UAV hiện nay còn bao gồm cả việc nâng cao thời gian và hiệu suất hoạt động. Vào thời gian đầu vấn đề về thời gian hoạt động và tuổi thọ pin là một trở lại lớn đối với lĩnh vực này. Nhưng hiện nay đã có những bước đột phá trong cụ thể là việc đưa ra thế hệ UAV tiếp theo sử dụng pin lithium-sulfur mới giúp cho thời gian hoạt động của UAV được tăng thêm 50%. Sự cải tiến này là quan trọng trong các ứng dụng đòi hỏi thời gian hoạt động dài như giám sát, viễn thám, kiểm soát nông nghiệp. Thêm vào đó, các cải tiến về hiệu suất hoạt động giúp cho giảm đi lượng năng lượng tiêu thụ của UAV và giúp cho UAV trở nên thân thiện với môi trường hơn [7].

Nhờ vào các tiến bộ khoa học về nghiên cứu vật liệu và khí động lực học mà UAV hiện nay đã có thể vận chuyển được các vật phẩm có khối lượng cao hơn ngày trước. Bằng cách sử dụng các vật liệu composite nhẹ và bền mà UAV hiện nay đã có thể mang theo bên mình nhiều hệ thống cảm biến, camera và hàng hóa hơn. Bước pháp triển này đem lại rất nhiều lĩnh vực công nghiệp như phân tich, vận chuyển và trong lĩnh vực cứu hộ cứu nạn [7].

Ngày nay UAV được sử dụng ngày càng nhiều và ngày càng phát triển, kéo theo đó những hệ thống thiết kế cho UAV này cũng ngày càng phức tạp hơn trong đó yêu cầu UAV phải hoạt động nhanh và chính xác. Một trong những phương pháp để giải quyết vấn đề này là sử dụng hệ điều hành thời gian thực (RTOS) để vận hành UAV [8]. Do đó việc ứng dụng RTOS vào UAV cũng là đang là một trong những hướng phát triển hiện nay.

## 1.2. Giới thiệu về Quadcopter và hệ điều hành thời gian thực

### 1.2.1. Khái niệm

Quadcopter là một loại thiết bị bay UAV đa động cơ bao gồm 4 động cơ, kiểu thiết kế UAV này mang lại lợi ích như hạn chế chi phí và dễ dàng lắp ráp. Quadcopter cung cấp khả năng bay tương đối ổn định và phù hợp cho các nhiệm vụ như viễn thám hoặc giám sát. Quadcopter được chia làm nhiều biến thể với nhiều kích thước từ nhỏ đến lớn khác nhau [9].

Hiện nay, các nghiên cứu và ứng dụng về Quadcopter nói riêng và UAV nói chung đang trở thành một đề tài quan trọng, cùng với đó lượng nhiệm vụ được tích hợp vào Quadcopter càng ngày càng phức tạp hơn, dẫn đến thiết kế ứng dụng trở nên phức tạp hơn. Vậy nên hệ điều hành thời gian thực (RTOS) được tích hợp và để pháp triển những ứng dụng phức tạp đó. RTOS là một loại hệ thống điều hành (OS) được thiết kế để hỗ trợ các ứng dụng thời gian thực như khả năng lập lịch, đồng bộ, quản lý tài nguyên, chính xác về mặt thời gian, giao tiếp và giao tiếp vào ra. Vì những đặc tính này mà RTOS được tích hợp vào Quadcopter để đảm bảo khả năng hoạt động an toàn cho Quadcopter .Sự khác biệt giữa RTOS và hệ điều hành nhiệm vụ chung (GPOS) là RTOS cung cấp khả năng lập cố định tức khả năng cung cấp phản hồi trực tiếp từ thế giới thực [10]. **Bảng 1** là mã nguồn của một số bộ điều khiển bay sử dụng hệ điều hành thời gian thực RTOS cùng với nhiệm vụ của chúng.

***Bảng 1:*** *Một số bộ điều khiển bay sử dụng RTOS*

| Mã nguồn điều khiển | Hệ điều hành thời gian thực | Chức năng |
|---|---|---|
| PX4 | Nuttx[11] | Giám sát CPU<br>Điều khiển các chuẩn giao tiếp |
| ArduPilot | ChibiOS[12] | Điều khiển các chuẩn giao tiếp<br>Nhận tín hiệu điều khiển |
| DJI Flight Controller | Android OS[13] | Điều khiển động cơ<br>Kết hợp cảm biến<br>Xử lý video<br>Truyền thông |
| LibrePilot | FreeRTOS[14] | Đọc dữ liệu cảm biến<br>Điều khiển động cơ<br>Truyền thông |

### 1.2.2. Hệ điều hành thời gian thực được sử dụng trong đồ án

Trong đồ án này, hệ điều hành được sử dụng là hệ điều hành thời gian thực FreeRTOS. Đây là một hệ điều hành mã nguồn mở, đi đầu thị trường và thường được sử

dụng trong UAV. FreeRTOS được phát triển bởi Richard Barry vào năm 2003, hiện tại hệ điều hành này có khả năng hỗ trợ cho ít nhất 35 hệ kiến trúc và hàng trăm loại vi xử lý. Vì là phần mềm mã nguồn mở nên chương trình từ các phiên bản trước luôn có thể truy cập được. FreeRTOS được sử dụng chủ yếu bằng ngôn ngữ lập trình C với một vài hàm được sử dụng bằng ngôn ngữ assembler [10]. FreeRTOS được sử dụng trong đồ án này do những lợi thế như hoàn toàn miễn phí, chiếm kích thước bộ nhớ nhỏ, cấu trúc đơn giản, ngôn ngữ lập trình C là chủ yếu, không yêu cầu thêm điều khoản gì khi sử dụng [15].

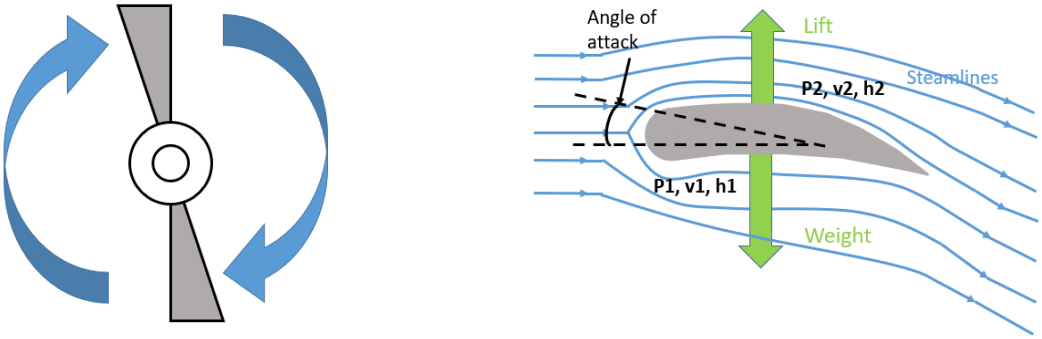

***Hình 3:*** *Hệ điều hành FreeRTOS [15]*

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1. Nguyên lý hoạt động cơ bản

### 2.1.1. Nguyên lý sinh ra lực nâng của cánh quạt

Cánh quạt là một bộ phận quan trọng trong kết của của Quadcopter, nó tạo ra lực nâng để giúp cho Quadcopter bay lên. **Hình 4-a** bên dưới là hình ảnh một cánh quạt của Quadcopter khi quay, cắt một mặt theo chiều dọc của cảnh quạt, hình dạng của mặt cắt có dạng như **Hình 4-b**. Phần mặt cắt quảnh quạt này có hình dạng để không khí ở bề mặt phía trên mặt cắt nhanh hơn ở bề mặt phía dưới.




***(a)** hình ảnh cánh quạt quay* *(b) hình ảnh mặt cắt của cánh quạt [16]*

***Hình 4:** Nguyên lý tạo lực nâng cánh quạt*

Tuân theo định luật Bernoulli [17] tổng lượng năng lượng của chất lỏng tại mọi điểm trong dòng chảy là không thay đổi, công thức của phương trình Bernoulli có dạng:

$$\frac{P_1}{\rho g} + \frac{{v_1}^2}{2g} + h_1 = \frac{P_2}{\rho g} + \frac{{v_2}^2}{2g} + h_2 \quad (1)$$

Trong đó:

- $P$ là áp suất của chất lỏng
- $v$ là vận tốc của chất lỏng
- $h$ là độ cao của chất lỏng
- $\rho$ là mật độ chất lỏng
- $g$ là gia tốc trọng trường

Trong công thức (1) các giá trị $P_1$, $v_1$ và $h_1$ là các đại lượng của dòng chảy tại mặt dưới của mặt cắt cánh quạt, các giá trị $P_2$, $v_2$ và $h_2$ là các đại lượng của dòng chảy ở mặt phía trên cánh quạt. Các đại lượng $g$ và $\rho$ là các hằng số, giả định rằng $h_1$ và $h_2$ xấp xỉ bằng nhau do đó biểu thức phụ thuộc vào 2 đại lượng là $P$ và $v$. Do hình dạng cùaa mặt cắt cánh quạt làm cho vận tốc dòng chảy mặt cắt bên trên nhanh hơn mặt cắt bên dưới tức

$v_2 > v_1$ do đó áp suất mặt cắt dưới $P_1 > P_2$ là áp suất mặt cắt trên. Sự chênh lệch áp suất tại mặt dưới và mặt trên của cánh quạt hình thành lên lực nâng, khi tổng lực của các cánh quạt của Quadcopter lớn hơn khối lượng của cánh quạt sẽ làm cho Quadcopter bay lên. Theo [18] lực nâng của cánh quạt được biểu diễn dưới dạng công thức:

$$L = \rho * D^2 * C_L * n^4 \quad (2)$$

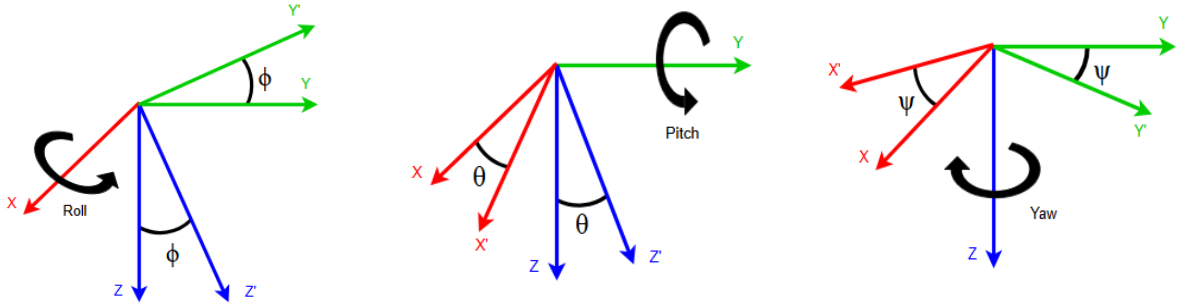
Trong đó

- $\rho$ là mật độ chất lỏng
- $D$ là đường kính cánh quạt
- $C_L$ là hệ số lực nâng
- $n$ là vận tốc góc

Đồ án này xét Quadcopter bay trong môi trường không khí đồng chất cho nên $\rho$ là hằng số và giả định rằng $D$ và $C_L$ là các giá trị cố định do sử dụng một loại cánh quạt có sẵn, do đó lực nâng của cánh quạt sẽ phụ thuộc vào vận tốc góc n của động cơ, vận tốc của cánh quạt càng lớn thì lực nâng càng lớn.

### 2.1.2. Góc Euler

Góc Euler được dùng để mô tả chuyển động quay của một vật theo bằng tổng trình tự quay của 3 chuyển động quanh trục tọa độ của vật thể. Cách biểu thức xoay này mang tính diễn giải cao nhất và không cần phải giảm dữ liệu vì chỉ cần 3 số thực. Trình tự quay trục khác nhau tạo ra kết quả xoay khác nhau, vì vậy góc Euler được định nghĩa bằng các trình tự có sẵn [19]. Trong hệ tọa độ 3 trục X-Y-Z thì ba góc quay Euler được đặt tên và kí hiệu là roll (ϕ - phi), pitch (θ - theta) và yaw (ψ - psi) tương ứng với 3 trục (**Hình 5**).



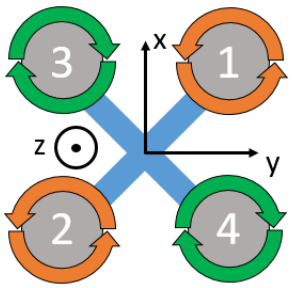

***(a)** góc roll* ***(b)** góc pitch* ***(c)** góc yaw*

***Hình 5:** Góc Roll, Pitch, Yaw [14]*

### 2.1.3. Nguyên lý động lực học của Quadcopter

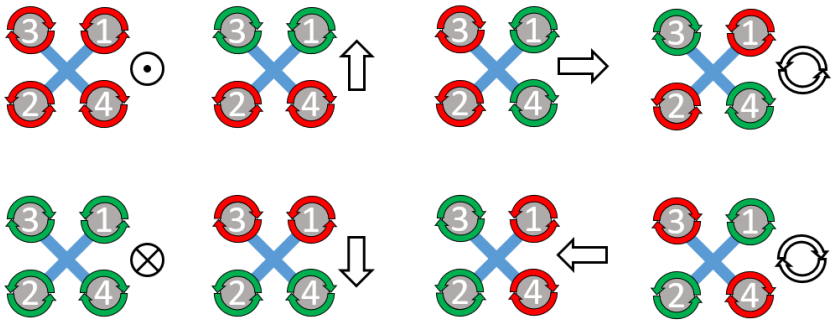
Quadcopter là một loại khí cụ bay sử dụng lực đẩy của bốn động cơ cánh quạt và bao gồm hai dạng là dạng dấu + và dạng chữ X [20]. Trong đồ án này dạng Quadcopter

được sử dụng là dạng chữ X như trong **Hình 6** trong đó các động cơ được đánh số từ 1 đến 4 với hai cặp động cơ 1,2 quay ngược chiều kim đồng hồ và 3,4 quay cùng chiều kim đồng hộ nhằm triệt tiêu mô men quay khi bay . Chiều dương trục x của Quadcopter hướng về phía trước, chiều dương trục y hướng về tay phải và chiều dương dương z là hướng lên trên. Tương đương góc roll, pitch, yaw của Quadcopter sẽ tương ứng với ba trục x, y và z.




***Hình 6:*** *Mô hình Quadcopter X [15]*

Chuyển động của Quadcopter được phân ra làm bốn loại chuyển động bao gồm cất và hạ cánh, di chuyển theo góc roll, pitch và yaw. **Hình 7** mô tả các chuyển động bay của Quadcopter với các động cơ có chiều quay màu đỏ sẽ là các động cơ quay ở tốc độ cao hơn, các động cơ có chiều màu xanh là các động cơ quay chậm hơn. Khi cất cánh, cả 4 động cơ quay ở vận tốc lớn còn khi hạ cánh thì cả 4 động cơ sẽ quay chậm lại. Để di chuyển theo góc pitch về phía trước động cơ 1 và 3 quay chậm hơn động cơ 2 và 4 và ngược lại khi di chuyển về phía sau. Khi di chuyển theo góc roll sang phía phải, động cơ 1,4 quay chậm hơn động cơ 2,3 và ngược lại khi di chuyển sang trái. Để quay Quadcopter theo góc yaw sang phải động cơ 3,4 sẽ quay nhanh hơn động cơ 1,2 và khi quay sang trái động cơ 1,2 sẽ quay nhanh hơn động cơ 3,4 [20].



***Hình 7:*** *Mô tả chuyển động Quadcopter X*

### 2.1.4. Mô hình kết hợp mixer

Ở phần 2.1.3 Quadcopter được chia ra làm 4 loại chuyển động là cất/hạ cánh , di chuyển theo góc roll, pitch, yaw. Do tại một thời điểm Quadcopter có thể thực hiện nhiều hơn một loại chuyển động nên cần có một phương pháp để điều khiển vận tốc các cánh

quạt phụ thuộc vào giá trị điều khiển trạng thái bay đầu vào. Một phương pháp để điều khiển vận tốc động cơ là mô hình điều khiển Mixer là phương pháp đưa ra kết quả điều khiển động cơ dựa theo tổng hợp chuyển động của vật thể. Theo mô hình điều khiển Mixer ở [21] xung PWM để điều khiển mỗi động cơ phụ thuộc vào giá:trị điều khiển bay, vị trí động cơ và trục tọa độ của Quadcopter, nếu sử dụng thiết lập mô hình Quadcopter như **Hình 6** và mô tả chuyển động như hình **Hình 7** thì xung PWM cho mỗi động cơ được tính theo công thức sau:

$$PWM1 = thrust - roll + pitch + yaw \quad (3)$$
$$PWM2 = thrust + roll - pitch + yaw \quad (4)$$
$$PWM3 = thrust + roll + pitch - yaw \quad (5)$$
$$PWM4 = thrust - roll - pitch - yaw \quad (6)$$

Trong đó:

- $PWM1, \mathrm{PWM2}, \mathrm{PWM3}, \mathrm{PWM4}$ là giá trị xung PWM xuất ra 4 động cơ
- $thrust, roll, pitch, yaw$ là giá trị các kiểu điều khiển trạng thái bay

Tối giản công thức (3), (4), (5) và (6) về dạng biểu thức ma trận thì công thức cuối cùng có dạng:

$$\begin{bmatrix} PWM1 \\ PWM2 \\ PWM3 \\ PWM4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} * \begin{bmatrix} thrust \\ roll \\ pitch \\ yaw \end{bmatrix} \quad (7)$$

## 2.2. Lý thuyết về các linh kiện điện tử

### 2.2.1. MEMS và hệ thống IMU

**MEMS:** là thiết bị có tích hợp các thành phần điện tử và máy móc có kích thước nano và micrô. MEMS được cho là một trong những chủ đề ấn tượng do tính ứng dụng trong công nghiệp và tiêu dùng. MEMS được tạo thành bởi những thành phần có kích cỡ nằm trong khoảng từ 1-100 micrô-mét, và các cảm biến MEMS thường có kích cỡ từ 20 micrô-mét đến 1 mili-mét. [22]

Để xác định các yếu tố ảnh hưởng bên trong môi trường, các cảm biến MEMS được sử dụng để đo các thông số như: chuyển động, từ trường, hóa chất, âm thanh, nhiệt độ hay điện từ trường. MEMS được chia làm nhiều loại bao gồm thiết bị cảm ứng âm thanh, gia tốc kế, con quay hồi chuyển, áp kế, công tắc, thước đo nhiệt độ tùy vào mục đích sử dụng. Lợi ích của MEMS là kích thước nhỏ, tỉ lệ tín hiệu/nhiễu cao, không bị ảnh hưởng bởi thiết bị khác, có thể hoạt động trong môi trường khắc nghiệt. Các cảm biến MEMS được sử dụng rộng rãi trong ngành tự động, hàng không, điện tử, bảo vệ, công nghiệp, y tế, khoa học đời sống và liên lạc từ xa. [22]

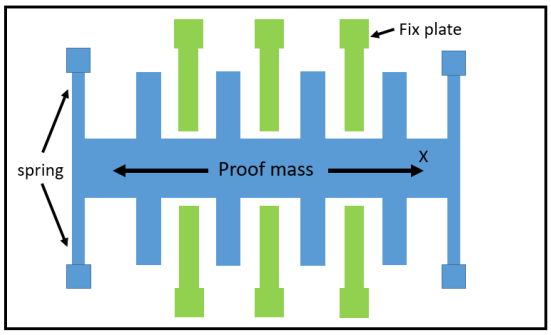
**Hệ thống IMU:** là một thiết bị được trang bị các cảm biến MEMS bên trong. Một IMU bao gồm một gia tốc kế con quay hồi chuyển và từ trường kế, mỗi cảm biến đo các thành phần theo 3 trục, do đó IMU là một thiết bị đo 9 trục.

IMU được phân loại theo thứ tự là cấp người tiêu dùng, doanh nghiệp, chiến lược và viễn thám. Các phân loại càng về sau sẽ càng có giá cao hơn và có độ chính xác cao hơn. IMU cấp người tiêu dùng thường được tích hợp vào trong điện thoại thông minh. IMU cấp độ doanh nghiệp được sử dụng bởi các UAV, IMU cấp chiến lược và viễn thám thường được sử dụng trong quân sự hoặc thiết bị đòi hỏi độ chính xác cao [23].

Một nhược điểm của các IMU dưới mức chiến lược có giá trị đo bị sai lệch sau một khoảng thời gian do chịu ảnh hưởng từ nhiễu, nhiệt độ. Để khắc phục nhược điểm này, một số phương pháp bao gồm tích hợp thêm những cảm biến khác hoặc sử dụng bộ lọc để chỉnh lại các sai lệch trong quá trình hoạt động của IMU. Bộ lọc bù được sử dụng trong bài này là một trong những phương pháp lọc để khắc phục vấn đề sai lệch giá trị đo được.

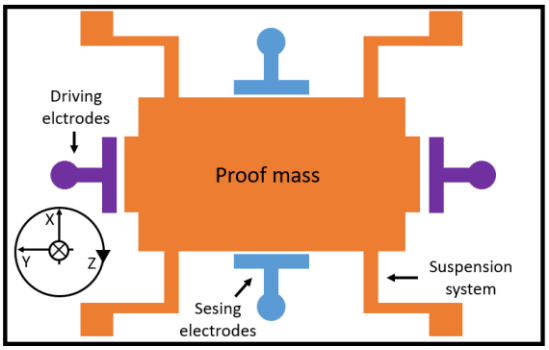
### 2.2.2. Một số loại cảm biến MEMS

**Cảm biến gia tốc kế:** Gia tốc kế là một thiết bị MEMS được dùng để đo gia tốc của vật thể theo 3 trục X,Y và Z. Kết cấu bên trong của một gia tốc kế bao gồm một khối lượng được cố định bằng lò xo, các lớp cố định xếp xen kẽ khối lượng là các điện cực. Khi khối lượng dao động theo trục X, lượng điện tích giữa các điện cực có sự thay đổi. Gia tốc theo trục X sẽ được tính toán dựa trên mức độ thay đổi điện tích này (**Hình 8**).



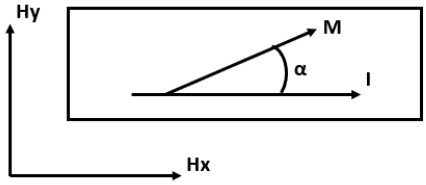

***Hình 8:*** *Gia tốc kế một trục [19]*

**Cảm biến con quay hồi chuyển:** Con quay hồi chuyển là một thiết bị MEMS dùng để đo vận tốc góc của vật thể quay quanh 3 trục X,Y và Z trong không gian. Kết cấu bên trong của cảm biến con quay hồi chuyển bao gồm hệ giao động hai trục có một khối lượng đặt ở giữa (**Hình 9**). Khối lượng của con quay hồi chuyển sẽ di chuyển đồng đều quanh trục Y bởi các cực điều khiển. Khi xuất hiện vận tốc góc theo trục Z, trục X xuất hiện giao động được tạo bởi gia tốc Coriolis. Biên độ của giao động theo trục X đo được tỷ lệ với vận tốc góc. Biên độ này được đo bằng cách đo lượng thay đổi điện dung giữa khối lượng và cực cảm biến. [24]



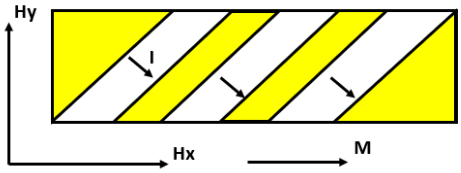

***Hình 9:*** *Con quay hồi chuyển một trục*

**Cảm biến từ trường kế:** Từ trường kế là một thiết bị MEMS được dùng để đo hướng và độ lớn từ trường, một trong những ứng dụng phổ biến của thiết bị này là la bàn. Một trong những kiểu từ trường kế bao gồm vật liệu gọi là trở từ trường(**Hình 10**) hoạt động theo hiệu ứng từ điện trở từ trường, khi có từ trường đi qua thì khả năng cản trở dòng điện của vật liệu thay đổi [25].



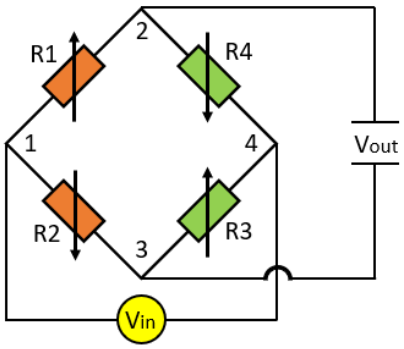

***Hình 10:*** *Cấu tạo trở từ trường [21]*

Để cho điện từ trường có thể đủ nhạy để cảm nhận từ trường yếu trở từ trường sẽ có cấu tạo theo cấu trúc barber poles bao gồm một cuộn dây dẫn điện cuộn theo trục X để ép dòng điện đi theo góc 45 độ theo trục X như **Hình 11** [25].




***Hình 11:*** *Cấu tạo trở từ trường barber poles [21]*

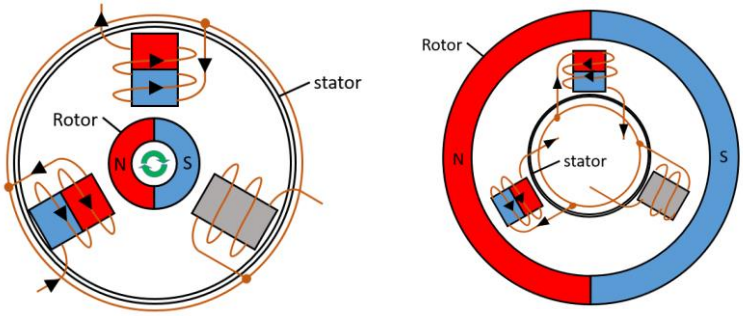
Để đo được độ thay đổi của từ trường theo một trong 3 trục X,Y hoặc Z, một cấu trúc có tên là cấu trúc cầu Wheatstone bao gồm bốn trở từ trường được đặt như **Hình 12** với hai điện trở đối diện nhau được đặt theo cùng cấu trúc Barber poles 45 độ hoặc 135 độ. Điện áp ra Vout có mối quan hệ với điện áp vào Vin. Một từ trường kế bao gồm cảm biến từ trường theo cả 3 trục X, Y và Z [25].




***Hình 12:*** *Từ trường kế một trục [21]*

### 2.2.3. Động cơ không chổi than

Động cơ BLDC hay động cơ không chổi than là động cơ motor sử dụng dòng một chiều không sử dụng chổi than như các loại động cơ thông thường. Động cơ không chổi than bao gồm 2 phần là rô-to và stato với rô-to là một nam châm vĩnh cửu và được quay bởi từ trường tạo ra bởi 3 cuộn dây quấn quanh stato. Bằng cách đổi pha  dòng điện đi qua 3 cuộn dây này rô-to khiến cho rô-to quay (**Hình 13**) do đó động cơ này còn được gọi là động cơ 3 pha.Động cơ không chổi than được chia làm 2 loại là động cơ có rô-to ở trong (**Hình 13-a**) và động cơ có rô-to ở ngoài (**Hình 13-b**).




***(a)*** *rô-to ở trong [20]* ***(b)*** *rô-to ở ngoài*

***Hình 13:*** *Động cơ không chổi than*

Trong một động cơ BLDC thiết bị điều khiển vận tốc động cơ điều khiển động cơ bằng cách chuyển đổi dòng điện giữa các lõi dây một cách nhanh chóng tạo ra một từ trường quay quay lõi động cơ và làm quay rô-to. Lõi rô-to nam châm được kết nối với một tay quay và làm nó quay theo. Bên trên động cơ, cánh quạt sẽ được lắp vào tay quay và lực nâng được tạo ra bằng cách quay cánh quạt trong không khí [20].

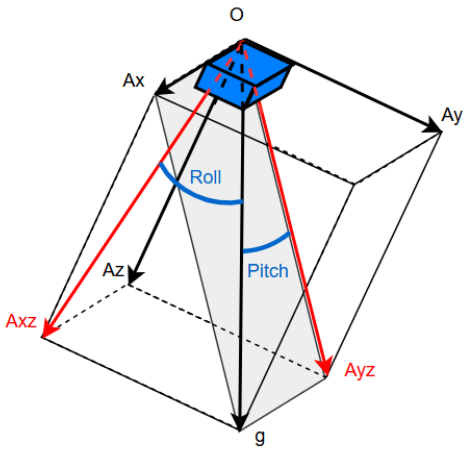
Một động cơ với hệ số Kv thấp có thể tạo ra lực quay lớn hơn và có thể quay cánh quạt lớn hơn so với động cơ có hệ số Kv cao. Các Quadcopter dùng để đua thường dùng cánh quạt nhỏ với động cơ có hệ số Kv lớn từ đó có thể quay nhanh hơn nhưng lại có thể

nâng ít hơn Quadcopter dùng động cơ có hệ số Kv thấp. Kv đại diện cho giá trí vòng trên phút mỗi vôn khi không có tải lên động cơ và có đơn vị là rpm/V, điện áp được ấp vào càng nhiều thì động cơ quay càng nhanh. Trong thực tế số vòng quay trên phút chậm do ảnh hưởng của lực cản và thông số của cánh [20].

## 2.3. Lý thuyết về điều khiển

### 2.3.1. Tính góc toán góc Euler từ cảm biến

**Tính toán bằng cảm biến gia tốc góc:** Cảm biến gia tốc góc đo gia tốc tính góc roll và pitch dựa trên góc nghiêng của mặt phẳng Oxz và Oyz so với hướng gia tốc trọng trường $g$ của trái đất trong đó $g$ là hợp của 3 gia tốc theo 3 phương X,Y và Z , theo như **Hình 14** công thức dùng để tính góc roll và pitch bao gồm những công thức sau.




***Hình 14:*** *Góc tính từ giá trị gia tốc*

$$g = \sqrt{{A_X}^2 + {A_Y}^2 + {A_Z}^2} \quad (8)$$

$$Pitch_{accel} = \frac{180}{\pi} * ArcTan\left(\frac{-Ax}{\sqrt{Ay^2 + Az^2}}\right) \quad (9)$$

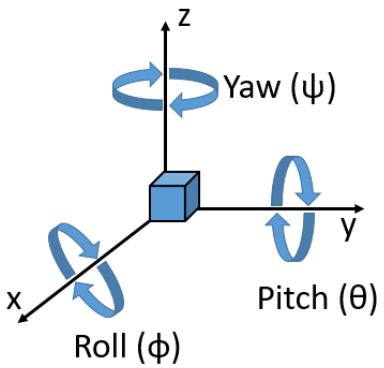
$$Roll_{accel} = \frac{180}{\pi} * ArcTan\left(\frac{Ay}{\sqrt{Ax^2 + Az^2}}\right) \quad (10)$$

Trong đó:

- $g$ là gia tốc trọng trường
- $A_X, A_Y, A_Z$ là gia tốc theo 3 trục X,Y và Z

**Tính toán bằng cảm biến con quay hồi chuyển:** Các giá trị Pitch và Roll của con quay hồi chuyển được tính dựa trên đại lượng vận tốc góc tương ứng với hai trục x và y

(**Hình 15**), do sử dụng đại lượng vận tốc để tính góc nên góc của cảm biến là tổng các thay đổi của góc theo thời gian. Biểu thức để tính góc roll và góc pitch theo con quay hồi chuyển có dạng:




***Hình 15:*** *Góc tính từ giá trị vận tốc góc*

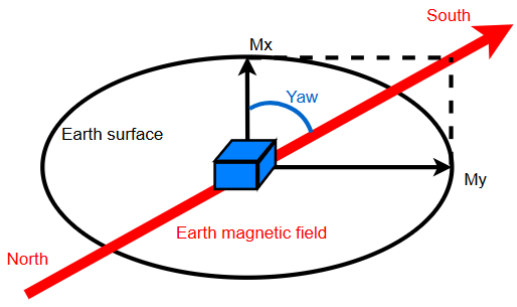
$$Pitch_{gyro} = Pitch_{gyroPrev} + Gy * \triangle t \quad (11)$$

$$Roll_{gyro} = Roll_{gyroPrev} + Gx * \triangle t \quad (12)$$

Trong đó:

- $Gx, Gy$ là vận tốc góc quanh hai trục X và Y
- $Pitch_{gyroPrev}$ và $Roll_{gyroPrev}$ là các giá trị từ lần đọc trước
- $\triangle t$ là khoảng thời gian giữa hai lần đọc dữ liệu

**Tính toán bằng từ trường kế:** Giá trị góc yaw được lấy từ trường kế trong đó từ trường kế sẽ tính toán góc lệch của cảm biến so với từ trường của trái đất thông qua đo sự thay đổi về từ trường của cảm biến theo hai góc X và Y (**Hình 16**). Công thức để tính góc yaw dựa trên cảm biến từ trường kế có dạng:




***Hình 16:*** *Góc tính từ giá trị từ trường*

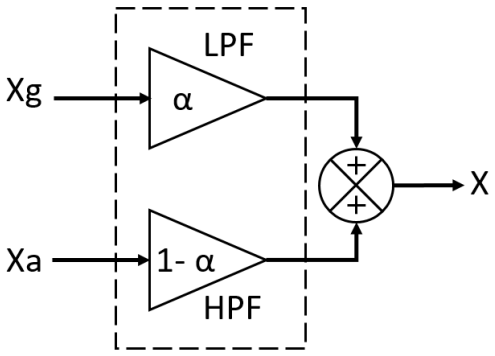
$$\text{Yaw} = \frac{180}{\pi} * ArcTan\left(\frac{\text{My}}{\text{Mx}}\right) \quad (13)$$

Trong đó:

- Mx, My là giá trị từ trường theo hai trục X và Y

### 2.3.2. Bộ lọc bù

Do việc đọc dữ liệu giá trị góc roll và góc pitch bởi gia tốc kế và con quay hồi chuyển sẽ bị sai lệch theo thời gian do rung động và trôi giá trị, một trong những phương pháp để tính toán chính xác hơn giá trị của 2 góc này là bộ lọc bù. Bộ lọc bù là một phương pháp kết hợp cảm biến chi phí thấp bao gồm một bộ lọc thông cao (HPF) và một bộ lọc thông thấp (LPF) (**Hình 17**). Một số khái niệm liên quan đến bộ lọc bù bao gồm:




***Hình 17:*** *Bộ lọc bù [22]*

**Bộ lọc thông thấp:** Mục đích của bộ lọc thông thấp là chỉ để tín hiệu thay đổi dài hạn đi qua và lọc bỏ những nhiễu ngắn hạn. Một trong những phương pháp để thực hiện là ép buộc giá trị thay đổi tăng lên từng chút một trong những lần lặp tiếp theo thông qua vòng lặp chương trình [23].

**Bộ lọc thông cao:** Mục đích của bộ lọc thông cao cho phép tín hiệu có chu kì ngắn đi qua trong khi lọc bỏ những tín hiệu không đổi theo thời gian và có thể sử dụng để loại bỏ hiện tượng trượt giá trị [23].

**Tần số lấy mẫu:** Khoảng thời gian lặp lại giữa những lần thu tín hiệu, ký hiệu của tần số lấy mẫu là dt [23].

**Thời gian cố định:** khoảng thời gian cố định của một bộ lọc là khoảng thời gian để lọc bỏ tín hiệu. Đối với bộ lọc thông thấp, tín hiệu dài hơn khoảng thời gian cố định sẽ được đi qua trong khi tín hiệu ngắn hơn sẽ bị lọc bỏ. Điều ngược lại sẽ xảy ra với bộ lọc thông cao. Ký hiệu của khoảng thời gian cố định là $\tau$ [23].

Bộ lọc bao gồm 2 đầu vào Xg là giá trị vận tốc góc của con quay hồi chuyển sẽ đi qua bộ lọc thông thấp và Xa là giá trị góc quay của gia tốc kế sẽ đi qua bộ lọc thông cao với kết quả đầu ra là tổng kết quả của 2 bộ lọc (**Hình 17**). Theo [26] công thức của bộ lọc này có dạng:

$$Pitch = (\text{Pitch} + Pitch_{gyro}) * \alpha + Pitch_{accel} * (1 - \alpha) \quad (14)$$

$$Roll = (\text{Roll} + Roll_{Gyro}) * \alpha + Roll_{accel} * (1 - \alpha) \quad (15)$$

Trong đó:

- $\alpha$ là hệ số của bộ lọc bù
- $Pitch_{gyro}$ và $Roll_{Gyro}$ là giá trị góc quay của con quay hồi chuyển
- $Pitch_{accel}$ và $Roll_{accel}$ là giá trị góc quay của gia tốc kế

Theo [27] công thức để tính hệ số lọc bù α có dạng như sau:

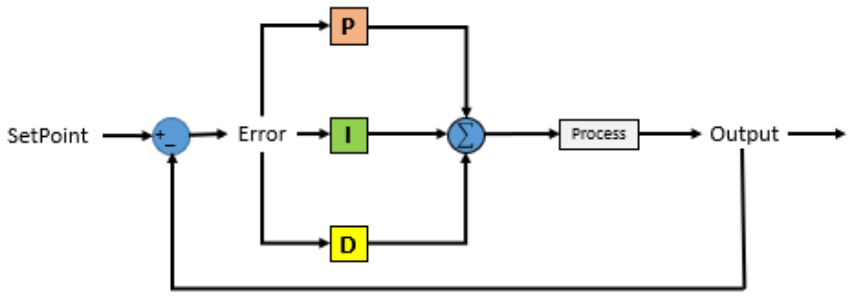
$$\alpha = \frac{\tau}{\tau + dt} \quad (16)$$

Trong đó:

- $\alpha$ là hệ số của bộ lọc bù
- $\tau$ là khoảng thời gian cố định
- $dt$ là tần số lấy mẫu

### 2.3.3. Thuật toán điều khiển PID

PID là một thuật toán điều khiển để xác định độ chính xác của một hệ thống phản hồi. **Hình 18** mô tả mô hình khối của bộ điều khiển PID bao gồm ba thành phần chính là khối tỷ lệ (P), khối tích phân (I) và khối đạo hàm (D) với đầu vào là sai sai lệch giữa giá trị mong muốn và giá trị thực tế đầu ra. Bộ điều khiển này dựa vào sai lệch giá trị trong hiện tại, sự tích lũy sai lệch theo thời gian và dự đoán giá trị tương lai dựa trên tốc độ thay đổi. Ba khối điều khiển có thể dùng cùng nhau hoặc riêng biệt như bộ điều khiển P, PI hoặc PID tùy theo loại phản hồi mong muốn của người dùng. Ưu điểm của bộ PID là chỉ phụ thuộc vào bước đo đạc giá trị mà không quan tâm đến quá trình giúp nó được sử dụng rộng rãi. Phản hồi của bộ điều khiển có thể được giải thích thông qua các phản hồi sai lệch, lượng vượt mức so với điểm đặt và sự giao động của hệ thống. Thuật toán PID không yêu cầu sự tối ưu hay ổn định của hệ thống [28].




***Hình 18:*** *Mô hình bộ điều khiển PID [24]*

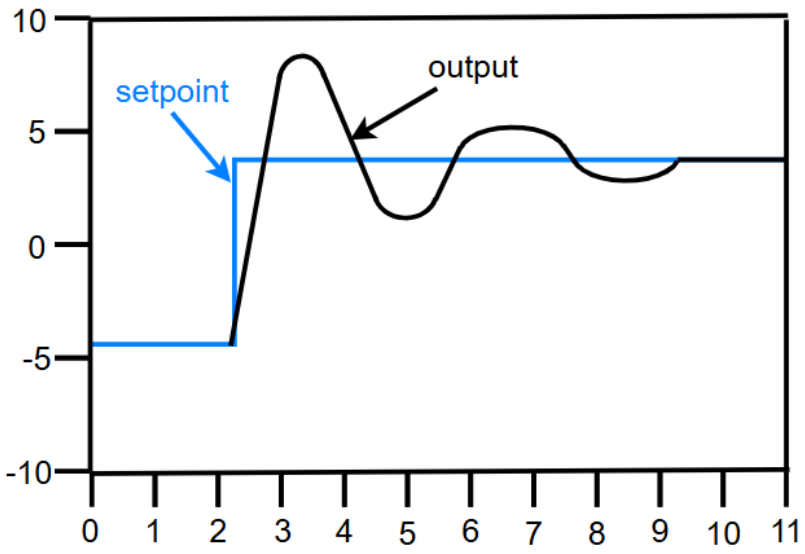
Biểu thức của một bộ điều khiển PID có dạng

$$\mu(\mathrm{m}) = Kp * e(m) + Ki * \sum_{k=0}^{m} e(m) + Kd * \big(e(m) - e(m-1)\big) \quad (17)$$

Trong đó:

- $Kp$ là hằng số tỷ lệ
- $Ki$ là hằng số tích phân
- $Kd$ là hằng số đạo hàm
- $e(m)$ là sai lệch giữa giá trị đo được và giá trị mong muốn
- $e(m-1)$ là sai lệch của lần đo trước
- $\mu(\mathrm{m})$ là kết quả của bộ điều khiển.

Có nhiều cách để xác định giá trị của $Kp$, $Ki$ và $Kd$. Một trong số đó là hiệu chỉnh từng hệ số một. Bắt đầu là hệ số tỷ lệ $Kp$, hệ số $Kp$ cần tìm phải đảm bảo hệ thống phản hồi kịp thời. Sau khi phản hồi ở mức tương đối, bước tiếp theo là tìm giá trị hằng số đạo hàm $Kd$ với mục đích để giảm biên độ để triệt tiêu giao động của hệ thống. Bước cuối cùng là tìm hằng số tích phân $Ki$ vì nó giúp tình trạng hệ thống có một khoảng sai an toàn giữa giá trị đo được và giá trị mong muốn [28]. Một bộ PID lý tưởng sẽ có kết quả đầu ra giống như **Hình 19** trong đó giá trị đầu ra hội tụ về giá trị đặt mong muốn.




***Hình 19:*** *Đầu ra một bộ PID lý tưởng [29]*

## 2.4. Lý thuyết về giao thức truyền thông chuẩn

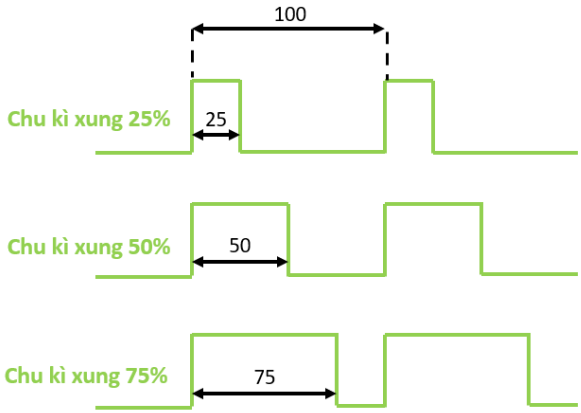
### 2.4.1 Nguyên lý điều chế xung PWM

Phương pháp điều chế xung bằng độ rộng (PWM) là một phương pháp điều chỉnh độ rộng của xung trong hệ thống điện để ổn định công suất đầu ra đến một tải. Xung

PWM thường được sử dụng phổ biến trong việc ổn định bộ khuếch đại âm thanh, điều khiển vận tốc động cơ và độ sáng của đèn. Phương pháp này được sử dụng rộng rãi trong các vi điều khiển và mạch tích hợp IC [30]. Các đặc trưng chính của phương pháp điều chế xung PWM bao gồm:

**Chu kỳ nhiệm vụ xung PWM:** Chu kỳ nhiệm vụ xung PWM là tỉ lệ giữa thời gian xuất xung lên so với chu kì hoạt động của một xung (**Hình 20**) và được biểu diễn dưới dạng công thức:

$$Chu\ k\grave{y}\ \text{nhiệm vụ}\ xung = \frac{Th\grave{o}i\ gian\ xu\acute{a}t\ xung\ l\hat{e}n}{Chu\ k\grave{y}\ c\acute{u}a\ m\hat{o}t\ xung} \quad (18)$$



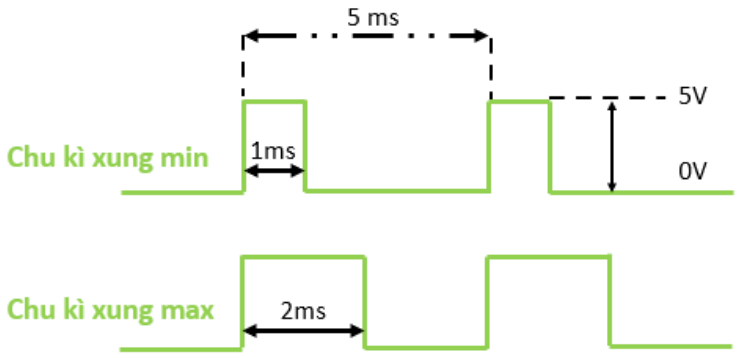

***Hình 20:*** *Chu kì nhiệm vụ xung PWM [25]*

**Tần số xung PWM:** là số chu kì xung PWM được xuất ra trong một giây có dạng công thức:

$$\text{Tần số xung} = \frac{1}{Chu\ k\grave{i}\ \text{một xung}} \quad (19)$$

**Điện áp ra xung PWM:** là điện áp ra của xung PWM khi ở xung lên.

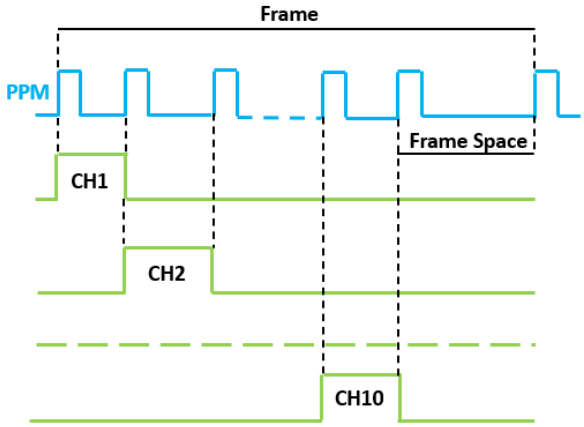
Trong đồ án này, xung PWM được dùng để điều khiển vận tốc quay của động cơ Quadcopter có tần số 200Hz hay chu kì 5ms và có chu kì nhiệm vụ xung từ 1ms đến 2ms với điện áp ra 5V (**Hình 21**).




***Hình 21:*** *Xung PWM điều khiển động cơ Quadcopter*

### 2.4.2. Điều khiển tín hiệu vô tuyến bằng xung PPM

Phương pháp điều chế bằng khoảng cách xung (PPM) là một phương pháp điều chế trong đó các xung có độ rộng như nhau như khoảng cách giữa các xung với nhau là không bằng nhau. PPM được ứng dụng trong hệ thống giao tiếp quang học, điều khiển vô tuyến và trong quân sự. Trong đồ án này, tín hiệu PPM được nhận từ bộ nhận tín hiệu vô tuyến bao gồm một chuỗi các xung, xung đầu tiên xác định bắt đầu của một khung tín hiệu, trong đó khoảng thời gian giữa các lần xung lên sẽ được dùng để xác định tín hiệu của từng kênh. Sau khi gửi hết chuỗi các xung thì tay cầy sẽ không gửi gì và để trống một khoảng thời gian, khoảng trống này để xác nhận sự kết thúc của 1 khung tín hiệu và xung tiếp theo sẽ báo hiệu sự bắt đầu của một khung tín hiệu mới [31]. Tay cầm điều khiển của đồ án này sẽ gửi xung PPM bao gồm 10 kênh và khoảng cách giữa các kênh được tính theo mili giây (**Hình 22**).

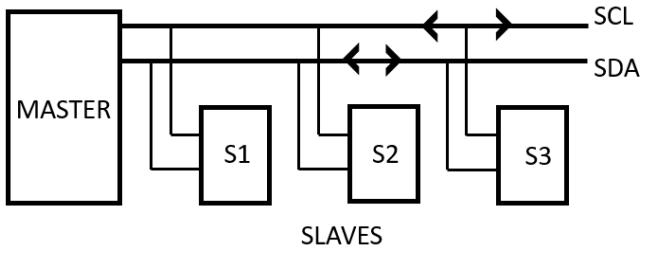



***Hình 22:*** *Xung tín hiệu PPM [26]*

### 2.4.3. Giao thức truyền thông I2C

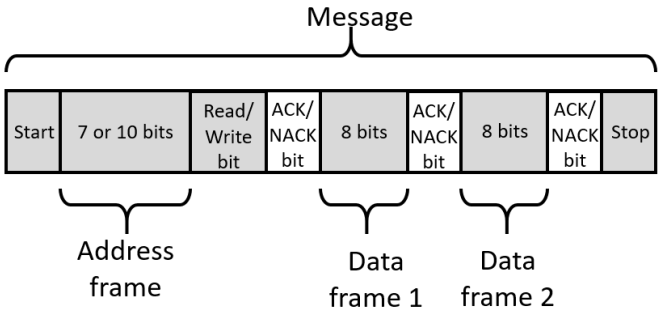
Inter-Integrated Circuit (I2C) là một chuẩn giao thức truyền thông tin đơn giản không dễ bị mất tín hiệu, giao thức cung cấp tốc độ cao hơn so với các giao thức khác. Giao thức này được sử dụng để có thể giao tiếp ở tốc độ cao và có thể điều khiển các

thanh ghi bên trong thiết bị cũng như các dữ liệu được lưu bên trong thanh ghi. I2C là một trong những giao thức sử dụng địa chỉ phần mềm do đó có thể giao tiếp và kết nối với nhiều thiết bị trong khi chỉ cần 2 dây kết nối là dây truyền xung (SCL) truyền xung và dây dữ liệu (SDA) truyền dữ liệu (**Hình 23**). Mỗi thiết bị được định danh bởi một địa chỉ riêng biệt và có thể hoạt động như cả thiết bị truyền và nhận, tùy thuộc vào chức năng của mỗi thiết bị [32]. Việc thêm hoặc loại bỏ thiết bị khỏi giao thức I2C đem lại lợi ích trong việc dễ dàng bảo trì và điều khiển trong lập trình nhúng . Tiêu chuẩn tần số xung đồng hồ của giao thức I2C là và có thể tăng tần số này lên đến 400kHz.




***Hình 23:*** *Mô hình kết nối giao thức I2C [27]*

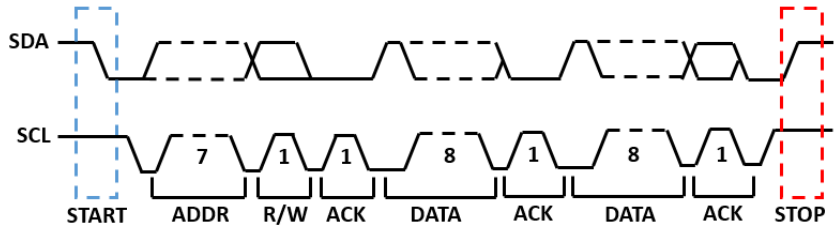
Trong giao thức I2C, dữ liệu được truyền dưới dạng tin nhắn, tin nhắn này được chia làm nhiều phần như được mô tả trong **Hình 24.** Bit Start đóng vai trò là tín hiệu đến tất cả các thiết bị nhận trong mạng lưới giao thức rằng dữ liệu chuẩn bị được truyền đi. Sau Bit Start là Address Frame có độ dài 7 hoặc 10 bit chứa địa chỉ của thiết bị mà thiết bị truyền muốn giao tiếp. Bit Read/Write có độ dài 1 bit để thông báo rằng thiết bị truyền muốn truyền dữ liệu hay yêu cầu thiết bị gửi dữ liệu về. Data Frame có độ dài 8 bit được dùng để truyền dữ liệu, dữ liệu quan trọng nhất sẽ được gửi đi trước, cuối mỗi Data Frame thiết bị nhận sẽ gửi ngược một bit ACK về thiết bị truyền để thông báo tin nhắn có được gửi thành công. Bit ACK/NACK có độ dài 1 bit nằm sau bit địa chỉ và bit dữ liệu và được gửi ngược từ thiết bị đến thiết bị chủ để thông báo rằng dữ liệu được nhận thành công hay không. Sau khi truyền hết dữ liệu tệp tin kết thúc bằng một bit Stop để thông báo thiết bị nhận gói tin đã kết thúc, sau khi nhận bit stop thiết bị sẽ trở lại trạng thái sẵn sàng để chờ tin nhắn tiếp theo được gửi đi [33].



***Hình 24:*** *Cấu trúc tin nhắn giao thức I2C [28]*

Nguyên lý hoạt động của giao thức I2C dựa trên cách hoạt động của việc bật hoặc tắt tín hiệu điện điện áp ở dây SDA đồng thời với xung được gửi đi từ dây SCL. Việc bắt đầu gửi tiến hiệu được thực hiện bằng việc giảm điện áp ở SDA về 0V khi điện áp ở SCL vẫn đang ở mức cao. Thay đổi điện áp ở SDA khi điện áp ở SCL là tín hiệu để bắt đầu của một gói tin. Nếu điện áp của SDA không đổi khi SCL ở điện áp cao, các thiết bị I2C sẽ mặc định là chưa có gói tin nào được gửi đi [34].

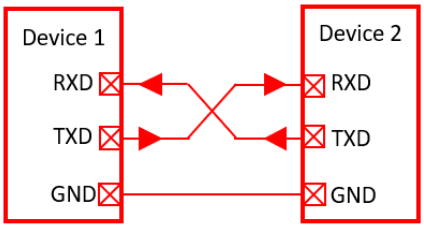
Sau khi bắt đầu gói tin các thông tin về địa chỉ, chức năng, nội dung của gói tin sẽ được gửi ở dưới dạng bit nhị phân với 1 là điện áp cao và 0 là điện áp thấp mỗi khi có một xung SCL. 7 hoặc 10 bit đầu tiên sẽ là địa chỉ gói tin được gửi đi, 1 bit tiếp theo là bit Read/Write nhằm xác định chức năng của gói tin. Sau đó bit ACK sẽ được thiết bị nhận đặt ở mức thấp để thông báo thiết bị đã sẵn sàng để nhận gói tin. Sau mỗi 8 bit dữ liệu được gửi đi, thiết bị nhận sẽ gửi một xung ACK ở mức thấp để thông báo gói tin nhận thành côn, và ngược lại thiết bị nhận sẽ thông báo không nhận được gói tin bằng việc để điện áp SDA ở mức cao. Cuối cùng để đánh dấu kết thúc gói tin, điện áp của đầu SDA sẽ được giữ ở mức cao nhằm thông báo gói tin đã kết thúc [34]. **Hình 25** là mô tả nguyên lý gửi tín hiệu của giao thức I2C.




***Hình 25:*** *Nguyên lý truyền tin giao thức I2C [29]*

### 2.4.4. Giao thức truyền thông UART

UART là một chuẩn giao thức truyền thông nối tiếp không đồng bộ, thường là cho khoảng cách ngắn, tốc độ thấp, chi phí thấp giữa máy chủ và ngoại vi. UART bao gồm 3 mô-đun chính là bộ khởi tạo tốc độ truyền dữ liệu, bộ nhận và bộ truyền . Về cơ bản, giao thức UART chỉ cần kết nối 2 dây để nối với 2 cổng TXD là đầu truyền dữ liệu, đầu ra của UART và cổng RXD lầ đầu nhận dữ liệu, đầu vào của UART như được mô tả ở **Hình 26**. Đầu TXD ở thiết bị này sẽ nối với đầu RXD ở thiết bị còn lại và ngược lại, chiều di chuyển của dữ liệu được mô tả như trên hình. Dữ liệu truyền và nhận chỉ có hai trạng thái là 1 và 0. Khi đường truyền không hoạt động thì dữ liệu được truyền đi sẽ luôn ở trạng thái cao [35].

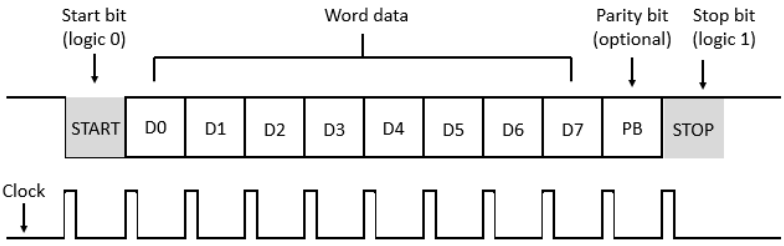



***Hình 26:*** *Mô hình kết nối giao thức UART [31]*

**Hình 27** mô tả cấu trúc dữ liệu UART được truyền đi, bit đầu tiên khi được truyền đi sẽ là Start Bit, lúc này bit dữ liệu chuyển từ trạng thái từ 1 về 0. Start Bit được sử dụng để thông báo cho bên nhận rằng sắp có dữ liệu được truyền đi và cưỡng ép bộ tạo xung ở bên nhận đường bộ với bên phát tín hiệu. Hai xung đồng hồ này phải đủ chính xác để có tần số lệch nhau không quá 10% trong quá trình truyền tải dữ liệu của tệp tin [35].

Sau Start Bit, các bit chứa dữ liệu được truyền đi với các bit có ít tầm quan trọng nhất (LSB) được gửi đi trước. Mỗi bit được truyền đi với cùng khoảng thời gian như nhau, và thiết bị nhận sử dụng một nửa chu kỳ đó để xác định đó là bit 1 hay 0.

Sau khi toàn bộ dữ liệu được gửi đi hết, thiết bị truyền tín hiệu sẽ gửi đi một bit có tên là Parity Bit để thiết bị nhận thực hiện kiểm tra lỗi đơn giản. Sau Parity Bit là bit dừng dùng để kết thúc tệp tin được gửi đi. Nếu thiết bị nhận không phát hiện ra bit dừng thì giao thức UART sẽ mặc định rằng tập tin đó vô nghĩa và sẽ thông báo lỗi về bo điều khiển chính khi đọc dữ liệu. Nguyên nhân chính của lỗi này là do bộ truyền và bộ nhận không chạy cùng thời điểm với nhau hoặc đường truyền tín hiệu bị can thiệp [35].

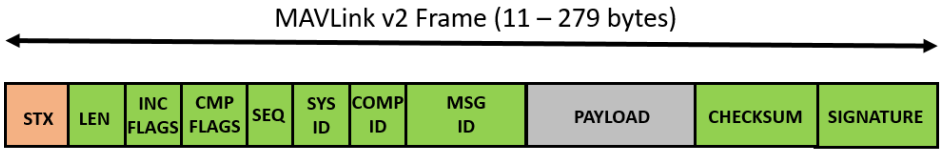



***Hình 27:*** *Nguyên lý truyền tin giao thức UART [30]*

## 2.5. Lý thuyết giao thức truyền thông từ xa MAVLink

Giao thức MAVLink đặc trưng ở khả năng đảm bảo việc giao tiếp giữa thiết bị và trạm mặt đất. MAVLink là giao thức truyền tin tương đối nhẹ với kiểu dữ liệu mà nó truyền đi có dạng chuỗi nhị phân. MAVLink có thể truyền đi thông qua wifi, Bluetooth, mạng Ethernet hoặc thiết bị truyền tin telemetry. Giao thức MAVLink được sử dụng trong bài nghiên này là giao thức MAVLink 2.0. **Hình 28** và **Bảng 2** bên dưới cho biết cấu trúc, kích thước, công dụng từng khối của một gói tin MAVLink. Hình ảnh cho biết

kích thước tối thiểu của gói tin là 11 bytes khi mà không có tin truyền đi và tối đa là 279 bytes [36].

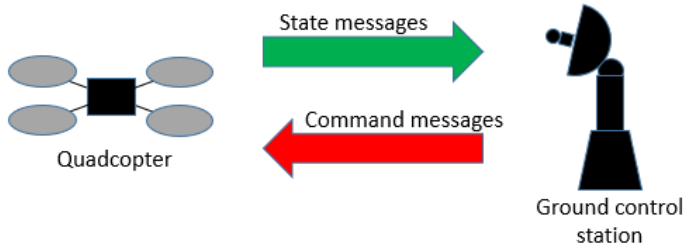



***Hình 28:*** *Cấu trúc tệp tin MAVLink [37]*

***Bảng 2:*** *Cấu trúc thành phần của MAVlink [37]*

| Khối | Tên | Kích thước (bytes) | Chức năng |
|---|---|---|---|
| STX | Start | 1 | Đánh dấu khởi đầu gói tin, luôn có giá trị 0xFE |
| LEN | Length | 1 | Độ dài của gói tin |
| INC FLAGS | Incompatibility flags | 1 | Không ảnh hưởng đến tệp tin nếu không hiểu được |
| CMP FLAGS | Compatibility flags | 1 | Quyết định kết cấu của tập tin |
| SEQ | Sequence number | 1 | Trình tự gói tin, tự động về không khi đạt 255 |
| SYS ID | System ID | 1 | ID của thiết bị (tối đa 254 thiết bị)<br>Trạm mặt đất ID mặc định là 255 |
| COMP ID | Component ID | 1 | ID các thành phần phần cứng trên thiết bị, cho biết thành phần nào đang gửi tin nhắn (27 thành phần có ID sẵn) |
| MSG ID | Message ID | 3 | ID của tin nhắn, cho biết loại tin nhắn trong PAYLOAD |
| PAYLOAD | | 0 - 255 | Chứa tin nhắn được gửi đi |
| CHECKSUM | | 2 | Đảm bảo gói tin không bị chỉnh sửa khi gửi đi |
| SIGNATURE | | 13 | Đảm bảo kết nối không bị nhiễu |

Gói tin MAVLink được phân loại làm 2 loại: tin nhắn trạng thái hay state message và tin nhắn ra lệnh hay command messages. Tin nhắn trạng thái là loại tin được gửi từ UAV về trạm mặt đất và chưa thông tin về thiết bị đó bao gồm ID, vị trí, vận tốc, độ cao. Ngược lại, tin nhắn ra lệnh được gửi từ trạm mặt đất đến thiết bị để yêu cầu thiết bị thực hiện chế độ hoặc nhiệm vụ [36]. **Hình 29** mô tả việc giao tiếp giữa trạm mặt đất và UAV thông qua MAVLink. Trong đồ án này giao thức MAVLink sẽ chỉ được dừng để gửi tin nhắn trạng thái từ Quadcopter về trạm mặt đất.




***Hình 29:*** *Kiểu truyền tin MAVLink*

## 2.6. Lý thuyết về hệ điều hành thời gian thực

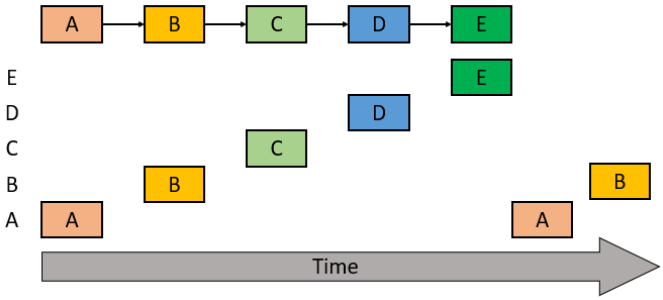
### 2.6.1. Khái niệm

RTOS hay hệ điều hành thời gian thực là một hệ điều hành được đặc trưng bởi khả năng yêu cầu các tác vụ phải hoàn thành trong một thời gian cho phép, nếu không thì dữ liệu từ tác vụ đó sẽ bị coi là sai sót, bất kể giá trị là gì [38]. Một chức năng khác của hệ điều hành này là khả năng phân cấp mức độ ưu tiên của các tác vụ trong đó khi hệ điều hành đang thực hiện một tác vụ khi gặp một yêu cầu thực hiện tác vụ có mức ưu tiên cao hơn sẽ tạm ngưng tác vụ hiện tại để thực hiện tác vụ cao hơn đó [39].

### 2.6.2. Các phương pháp lập lịch cho các tác vụ

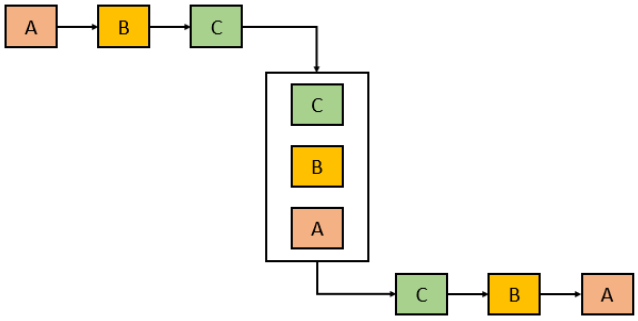
Các phương pháp lập lịch cho các tác vụ dưới đây được sử dụng để quyết định tác vụ nào được đưa vào chạy trước khi các tác vụ này có mức độ ưu tiên như nhau, một số phương pháp lập lịch bao gồm:

**Round robin:** Phương thức lập lịch Round robin sử dụng phân chia thời gian vi xử lý và cho mỗi tác vụ thời gian xử lý và mức độ ưu tiên như nhau với thứ tự hoạt động cố định, mỗi tác vụ này sẽ được phân cho một lượng vi xử lý nhất định. Khi một tác vụ được xử lý hết thì tác vụ đó sẽ bị đưa ra khỏi vi xử lý để có tài nguyên cho các tác vụ khác sử dụng (**Hình 30**).[40]

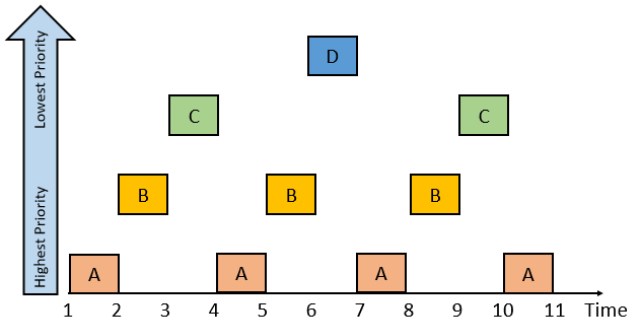



***Hình 30:*** *Phương thức lập lịch Round robin [35]*

**First-In-First-Out:** Phương thức lập lịch First-In-First-Out hay FIFO sẽ cố gắng thực hiện các tác vụ có mức độ ưu tiên cao nhất, khi có các tác vụ có cùng mức độ ưu tiên thì việc ưu tiên sẽ dựa vào việc tác vụ nào được gọi đến trước. Các tác vụ sau sẽ chỉ được thực hiện khi tác vụ trước đó đã hoàn thành hoặc tác vụ được chạy có mức độ ưu tiên cao hơn mức độ ưu tiên cao hơn tác vụ trước (**Hình 31**) [40].




***Hình 31:*** *Phương thức lập lịch First-In-First-Out [35]*

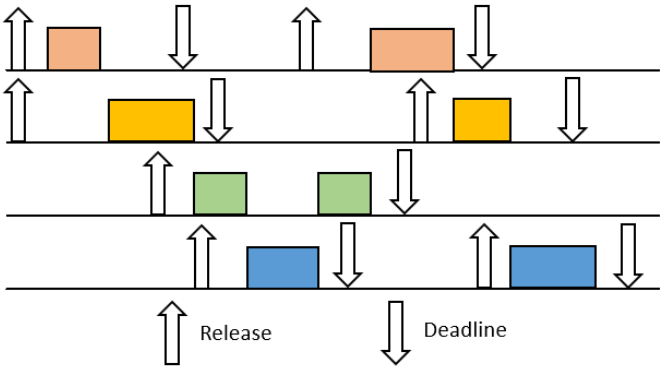



***Hình 32:*** *Phương thức lập lịch Rate-monotonic [35]*

**Rate-monotonic:** Phương pháp này là một thuật toán mức độ ưu tiên tĩnh đặt mức độ ưu tiên phụ thuộc vào thứ tự của các chu kỳ thông tin được thu thập. Các tác vụ có chu kì ngắn sẽ được thực hiện thường xuyên hơn, ngược lại các tác vụ có chu kì dài sẽ ít được thực hiện hơn. Các tác vụ chu kỳ ngắn sẽ được đặt mức độ ưu tiên cao hơn các tác vụ có

chu kì hoạt động dài. Phương pháp này phổ biến khi các tác vụ sử dụng cùng lượng thời gian của vi xử lý chính (**Hình 32**) [40].

**Earliest-Deadline-First :** Kiểu lập lịch Earliest-Deadline-First hay EDF sẽ tính toán mức độ ưu tiên cửu các tác vụ dựa trên thời gian hoạt động, thời gian yêu cầu giới hạn và các dữ liệu cần đầu vào cần thiết từ đó xếp các tác vụ có thời gian giới hạn sớm nhất trước. EDF có khả năng làm cho tất cả các thời hạn được đáp ứng khi tải hệ thống cao khi so sánh với phương pháp rate-monotonic (**Hình 33**) [40].




***Hình 33:*** *Phương thức lập lịch Earliest-Deadline-First [35]*

### 2.6.3. Nhân xử lý

RTOS hỗ trợ việc đặt giới hạn về thời gian vô hiệu hóa các ngắt, điều này cho phép một tác vụ có mức độ ưu tiên cao chạy đến khi hoàn thành mà không bị gián đoạn và giúp cho các tác vụ có thể hoàn thành trước thời gian yêu cầu giới hạn. Nhân xử lý của RTOS có gắng sử dụng lượng tài nguyên ít nhất có thể. Một trong những lợi ích của việc sử dụng nhân xử lý RTOS là nó sử dụng một lượng rất ít hệ thống để có thể hoạt động bao gồm tín hiệu, bộ đếm thời gian và bộ lập lịch tác vụ [40].

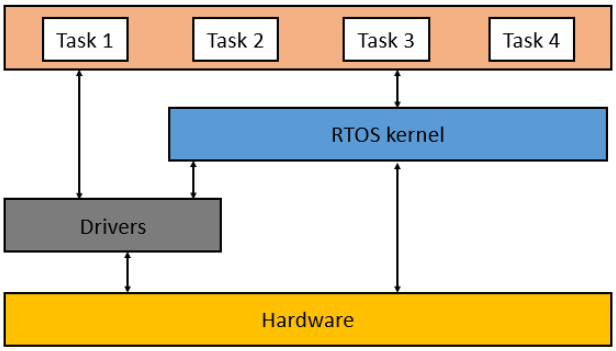
### 2.6.4. Đảo mức độ ưu tiên

Đảo mức độ ưu tiên xảy ra khi có hai tác vụ có mức độ ưu tiên khác nhau cùng sử dụng chung nguồn tài nguyên và tác vụ có mức ưu tiên cao hơn không thể truy cập vào lượng tài nguyên cần thiết mà tác vụ có mức ưu tiên thấp hơn đang sử dụng. Đây là một vấn đề nghiêm trọng vì nó ngăn cản các tác vụ hoàn thành trước thời gian cho phép. Một giao thức được sử dụng để giải quyết vấn đề này là giao thức thừa kế mức độ ưu tiên [40].

Giao thức kế thừa mức độ ưu tiên được thực hiện khi tác vụ có mức độ cao hơn phải đợi tác vụ có mức độ thấp hơn do đang sử dụng nguồn tài nguyên cần thiết thì thuật toán lập lịch sẽ tạm thời đặt cho tác vụ thấp hơn đó mức độ ưu tiên cao nhất để tác vụ thấp đó có thể hoàn thành và giải phóng tài nguyên cho tác vụ có mức độ ưu tiên cao hơn kia. Sau khi tác vụ ưu tiên thấp kia hoàn thành thì mức độ ưu tiên của tác vụ này sẽ đặt về mức ban đầu [40].

### 2.6.5. Hệ điều hành thời gian thực FreeRTOS

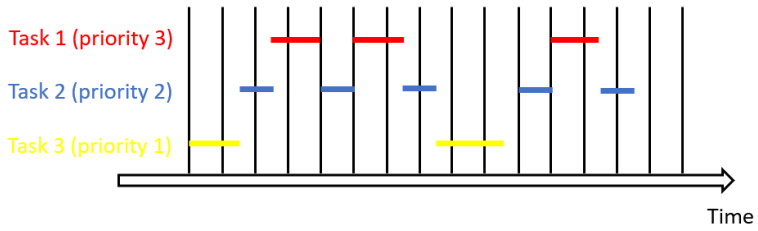
Hệ điều hành thời gian thực FreeRTOS tận dụng một nhân vi xử lý để thực thi tác tác vụ thời gian thực. Nhân của hệ điều hành này cung cấp bộ lập lịch dựa trên mức độ ưu tiên , có khả năng chận và tránh việc trì nghẽn và đình chỉ lập lịch [40]. (**Hình 34**)




***Hình 34:*** *Hệ điều hành thời gian thực FreeRTOS [35]*

Một số đặc trưng có trong hệ điều hành FreeRTOS khi lập lịch cho các tác vụ bao gồm lập lịch ưu tiên tác vụ cao hơn và cố định mức ưu tiên. đối với các tác vụ có mức độ ưu tiên giống nhau FreeRTOS sử dụng kiểu lập lịch round-robin kết hợp chia nhỏ thành các khoảng thời gian [41] trong đó:

**Ưu tiên tác vụ có mức ưu tiên cao:** nghĩa là bộ lập lịch sẽ chạy tác vụ có mức độ ưu tiên cao nhất đầu tiên  ngay cả khi có tác vụ khác thấp hơn đang hoạt động (**Hình 35**).

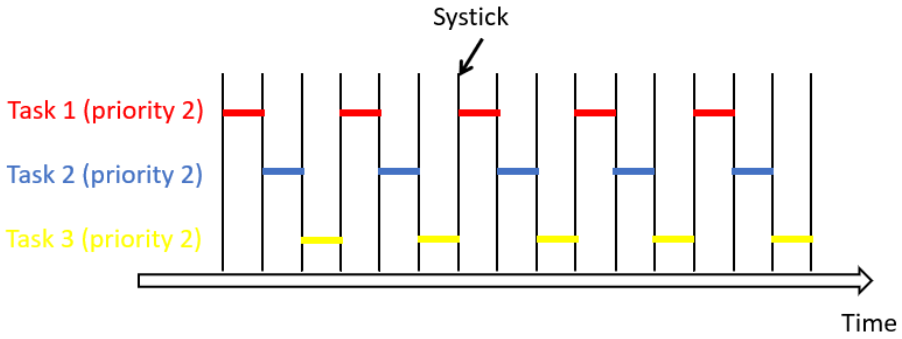



***Hình 35:*** *Phương thức lập lịch dựa trên mức độ ưu tiên*

**Cố định mức độ ưu tiên:** nghĩa là bộ lập lịch sẽ không thay đổi mức ưu tiên của tác vụ khác một cách vĩnh viễn khi trong trương hợp xảy ra kế thừa tác vụ ưu tiên.

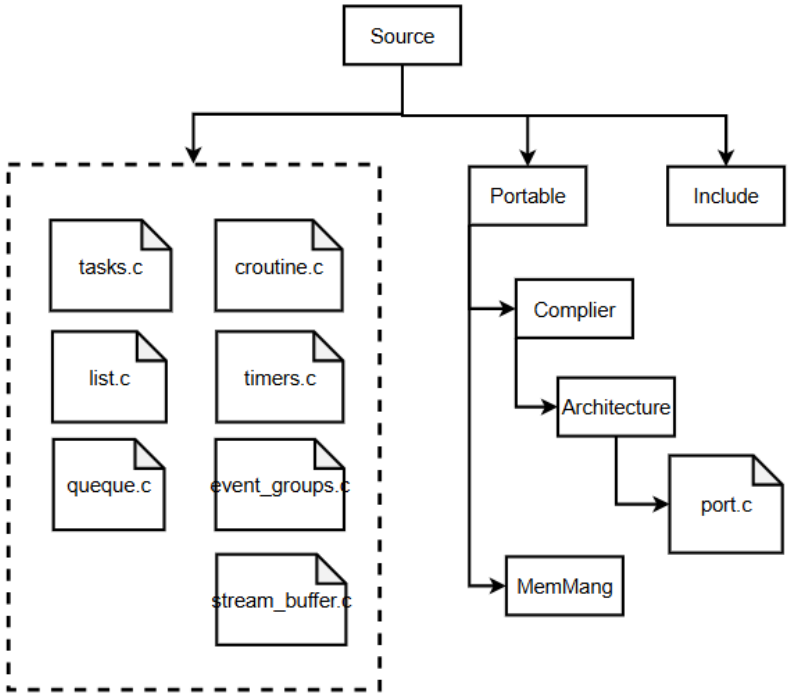
**Round-robin:** nghĩa là các tác vụ có mức độ ưu tiên như nhau sẽ thay phiên nhau tiến vào trạng thái hoạt động (**Hình 36**).

**Chia nhỏ thời gian:** nghĩa là các bộ lập lịch sẽ chuyển đổi chế độ hoạt động của các tác vụ có mức ưu tiên bằng nhau mỗi khi có ngắt thời gian của hệ điều hành (mỗi 1ms, có thể thay đổi bởi người dùng). **Hình 36** là ví dụ khi các tác vụ 1,2, 3 có mức ưu tiên như nhau thay phiên nhau hoạt động mỗi khi có ngắt thời gian của hệ điều hành.




***Hình 36:*** *Phương thức lập lịch các tác vụ có cùng mức ưu tiên [34]*

Đối với tất cả phiên bản của FreeRTOS, cấu trúc hệ điều hành bao gồm 3 files: tasks.c, list.c và queue.c (**Hình 37**) trong đó chức năng lập lịch được đưa vào trong file task.c, các hàm và biến sử dụng bởi bộ lập lịch được đặt trong file list.c, file queue.c bao gồm hàm đợi được sử dụng trong giao tiếp của các tác vụ và đồng bộ, bên cạnh các file này còn bao gồm các file bổ sung như routine.c, timers.c, event_groups.c và stream_buffer.c được tích hợp các hàm đồng bộ, bộ hẹn giờ phần mềm, nhóm sự kiện và bộ đệm luồng. thư viện include bao gồm các file header của hệ thống. Thư viện portable bao gồm chương trình phụ thuộc vào kiến trúc, trong thư viện này, mỗi chương trình dịch bao gồm một chương trình kiến trúc đặc trưng gọi là port.c để hỗ trợ kiến trúc xử lý. FreeRTOS lưu giao thức phân bố bộ nhớ trong tầng portable và thực hiện phân bố bộ nhớ heap trong thư viện MemMang [10].

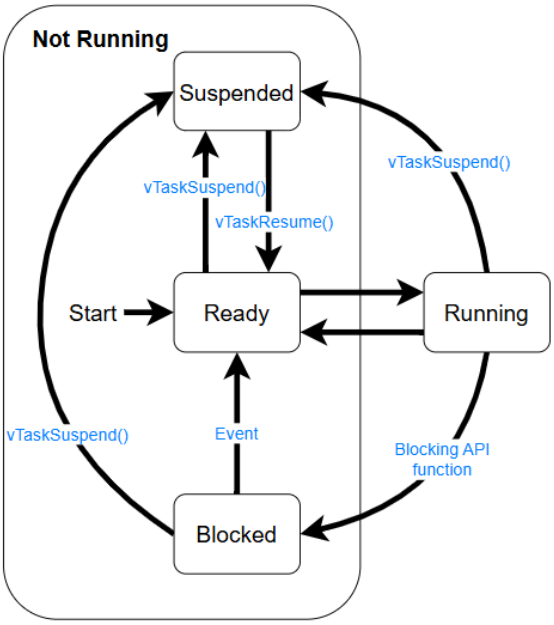



***Hình 37:*** *Cấu trúc hệ điều hành FreeRTOS [9]*

### 2.6.6. Hoạt động của tác vụ trong hệ điều hành FreeRTOS

Trong hệ thống thời gian thực FreeRTOS, hoạt động của các thành phần chương trình được thực hiện thông qua các tác vụ, một khi tác vụ được gọi tới thì tác vụ đó sẽ chạy trong vòng lặp vĩnh viễn và không thoát ra ngoài, một khi tác vụ không được gọi đến nữa thì tác vụ sẽ không trả lại mà sẽ bị xóa thông qua hàm deleted(). Mỗi hàm tác vụ được định nghĩa có thể được sử dụng để tạo các hàm tác vụ khác với mỗi tác vụ được tạo là một phiên bản thực thi riêng biệt, với ngăn xếp riêng và bản sao riêng của bất kì biến được tạo tự động bên trong tác vụ [42].

Việc khởi tạo các tác vụ được thực hiện thông qua hàm xTaskCreate(), hàm này sẽ được dùng để đặt tên, định nghĩa, đặt mức độ ưu tiên và cấu hình dung lượng bộ nhớ cho tác vụ. Sau khi tất cả tác vụ được khởi tạo cần gọi thêm một hàm vTaskStartScheduler() để bắt đầu lập lịch cho các tác vụ và đưa vào chạy. Khi bắt đầu chương trình, một tác vụ khi không bị chặn hoặc treo và đang đợi tác vụ có mức ưu tiên cao hơn chạy xong sẽ ở trong trạng thái sẵn sàng. Khi được gọi tới để chạy tác vụ sẽ được đặt vào trạng thái đang chạy và khi chạy xong tác vụ sẽ được đặt lại vào trạng thái sẵn sàng cho lần gọi tiếp theo. Tác vụ sẽ rơi vào trạng thái chặn khi gặp một trong hai trường hợp: chặn tạm thời do sử dụng hàm delay để khởi tạo tần số hoạt động cho tác vụ hoặc có một tác vụ khác hoặc ngắt được gọi, khi này tác vụ sẽ trở lại trạng thái sẵn sàng khi có sự kiện xảy ra. Trạng thái treo có thể được gọi tại bất kì thời điểm nào của ba trạng thái còn lại, trạng thái này cũng mang ý nghĩa là ngăn không cho tác vụ chạy tuy nhiên khác với trạng thái chặn tác vụ vẫn nằm trong bộ lập lịch thì khi trong trạng thái treo treo thì tác vụ sẽ bị đưa ra khỏi bộ lập lịch và chỉ quay lại trạng thái sẵn sàng khi được gọi lại [42]. (**Hình 38**)




***Hình 38:*** *Trạng thái hoạt động của một tác vụ [37]*

# CHƯƠNG 3: THIẾT KẾ QUADCOPTER

## 3.1. Khung và cánh Quadcopter được sử dụng cho đồ án

### 3.1.1. Khung Quadcopter

Khung Quadcopter là khung xương dùng để đặt các thiết bị điện tử như bảng mạch điều khiển, pin, ESC, motor cũng như bảo vệ các thiết bị bên trong nó. Trong đồ án này khung Quadcopter được sử dụng là khung 405F nylon Fiber Frame Airframe kit của QWinOut [43] có dạng như **Hình 39-a** với bốn dầm đỡ động cơ tạo thành hình chữ X, khoảng cách giữa bốn động cơ là 45cm với tấm đỡ phía trên là nơi để đặt bộ điều khiển và tấm đỡ bên dưới chính giữa mang chức năng là một bộ phân phối năng lượng (**Hình 39-b**) dùng để đặt pin, nhận điện áp từ pin và cấp nguồn cho 4 ESC.


***(a)** Khung đỡ động cơ và bộ điều khiển [44]* ***(b)** Bộ phân phối năng lượng [44]*

***Hình 39:** Khung Quadcopter*

### 3.1.2. Cánh Quadcopter

***Hình 40:** Cánh Quadcopter [40]*

Cánh Quadcopter là bộ phận chính để tạo lực nâng cho Quadcopter, trong đồ án này cánh Quadcopter được sử dụng là cánh 1045 Carbon Fiber Propeller [43], cánh có đường kính 25.4cm, năng 7g và có thể được sử dụng để đặt được lên nhiều loại trục động cơ với kích thước khác nhau, chiều của cánh sẽ được biểu thị bằng kí hiệu L hoặc R trên cánh

## 3.2. Thiết kế mạch điện Quadcopter

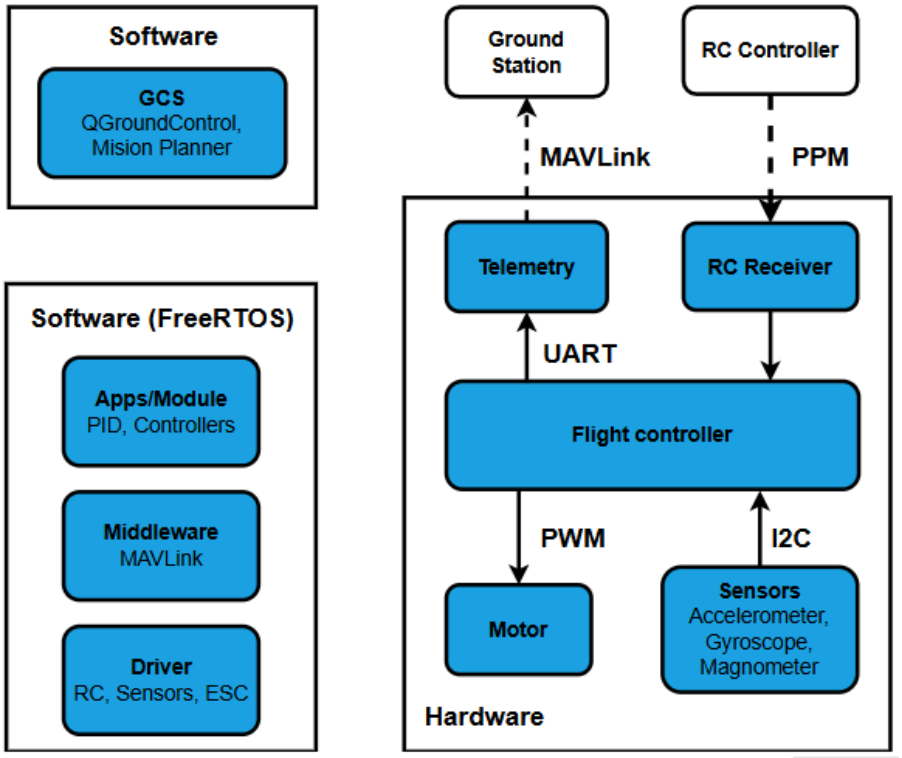
### 3.2.1. Sơ đồ bộ điều khiển bay Quadcopter

**Hình 41** là sơ mô tả hệ thống điều khiển bay của Quadcopter được sử dụng trong đồ án này, hệ thống bao gồm phần cứng và phần mềm trong đó bao gồm phần mềm dùng cho trạm mặt đất và phần mềm dùng cho bộ điều khiển bay. Phần cứng của Quadcopter bao gồm bộ điều khiển bay, các cảm biến, motor, mô-đun nhận sóng vô tuyến và mô-đun truyền tín hiệu trong đó các giao thức truyền thông chuẩn được sử dụng là I2C, UART, PWM, PPM và giao thức truyền thông MAVLink. Phần mềm của trạm mặt đất được sử dụng là 2 phần mềm QGroundControl và Mission Planner. Phần mềm của Quadcopter được tích hợp hệ điều hành thời gian thực FreeRTOS bao gồm 3 lớp là lớp Driver, Middleware và Application. Trong đó:

**Driver**: là chương trình dùng để điều khiển một phần cứng có sẵn, hệ điều hành có thể giao tiếp với các thiết bị phần cứng thông qua chương trình [45]. Trong đồ án này lớp Driver được sử dụng để giao tiếp với các mô-đun cảm biến, thu tín hiệu RC từ tay điều khiển, điều khiển vận tốc động cơ.

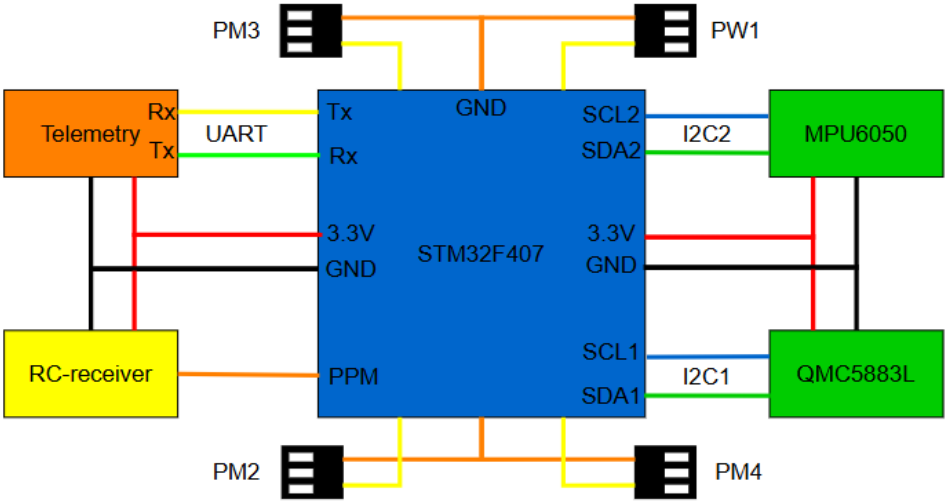
**Middleware**: là chương trình được dùng để kết nối 2 phần mềm với nhau, trong đó chương trình được kết nối có thể tồn tại trong cùng một máy hay trong hay máy khác nhau [45]. Trong đồ án này lớp Middleware được sử dụng để đọc các dữ liệu của cảm biến từ lớp Driver và giao tiếp với chương trình trạm mặt đất bằng giao thức truyền thông MAVLink.

**Applications:** là phần ứng dụng nằm ở trên cùng của hệ điều hành, các ứng dụng này không thể giao tiếp trực tiếp với phần cứng mà giao tiếp và quản lý hoạt động của các ứng dụng thông qua các dữ liệu nhận từ lớp Middleware. Trong đồ án này lớp Applications sẽ được tích hợp chương trình điều khiển bay ,bộ cân bằng PID và các chương trình lập trình tính toán dữ liệu cảm biến và giải mã tín hiệu điều khiển vô tuyến.




***Hình 41:*** *Sơ đồ mô tả hệ thống [46]*

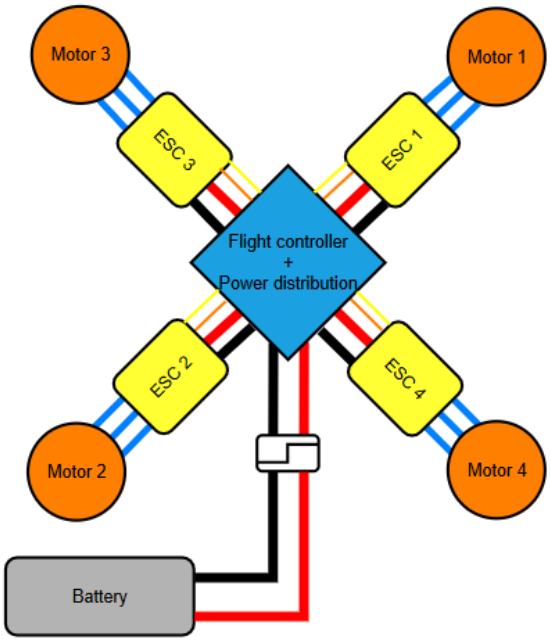
Từ thiết kế phần cứng của Quadcopter có thể đưa ra sơ đồ mạch điện như **Hình 42** là sơ đồ đấu nối của bộ điều khiển bay bao gồm bo điều khiển ST32F4 ở trung tâm, hai mô-đun cảm biến là MPU6050 và QMC5883L kết nối bằng giao thức I2C, mô-đun telemetry kết nối bằng giao thức UART, mô-đun RC-receiver gửi xung PPM về bo điều khiển và 4 cổng PWM theo thứ tự từ 1 đến 4, .




***Hình 42:*** *Sơ đồ đấu nối bộ điều khiển bay*

### 3.2.2. Sơ đồ mạch điện Quadcopter

**Hình 43** là sơ đồ đấu nối mạch điện của toàn bộ Quadcopter, sơ đồ gồm bộ điều khiển bay và bộ phân phối năng lượng ở trung tâm, bộ phân phối năng lượng được kết nối với pin để cấp nguồn cho ESC và bộ điều khiển bay, ESC được sử dụng để thay đổi vận tốc motor và được điều khiển bởi xung PWM phát từ bộ điều khiển bay.




***Hình 43:*** *Sơ đồ đấu nối mạch điện Quadcopter*

### 3.2.3. Các linh kiện được sử dụng cho đồ án

***Hình 44:*** *Bo điều khiển STM32F4-DISC*

**Bo điều khiển STM32F4-DISCOVERY:** Bo điều khiển STM32F4-DISCOVERY sử dụng chip xử lý STM32F407VG là một lõi xử lý hiệu suất cao ARM Cortex M4 32 bit. Bo điều khiển được trang bị nhiều ngoại vi bên trên mạch bao gồm một công cụ debug

nhúng ST-LINK/V2-A, một mic-crô số, một cổng âm thanh ADC, LED, nút nhấn và một cổng kết nối USB OTG Micro-AB. Bo điều khiển được cung cấp thư viện và ứng dụng trong gói STM32CubeF4 MCU trong phần mềm lập trình STM32. Điện áp cấp cho bo mạch nằm trong khoảng từ 3-5V và có thể cấp nguồn bằng nhiều phương thức: ST-LINK, USB $V_{BUS}$ hoặc nguồn ngoài, bo có bộ nhớ Flash 1Mbyte và bộ nhớ RAM 192 Kbyte .

**MPU-6050:** MPU-6050 là mô-đun cảm biến IMU bao gồm gia tốc kế 3 trục và con quay hồi chuyển 3 trục trong nó. MPU-6050 giúp đo gia tốc, vận tốc, góc quay, vị trí. Mô-đun giao tiếp với vi điều khiển chính bằng giao thức I2C thông qua 2 cổng SCL và SDA. Mô-đun được tích hợp một vi xử lý chuyển động (DMP) bên trong giúp giảm công việc tính toán phức tạp cho vi điều khiển chính. Mô-đun còn được tích hợp thêm 2 cổng phụ XCL và XDA để thực hiện kết nối bằng giao thức I2C với mô-đun từ trường kế mặc dù rất hạn chế. Tài liệu về sơ đồ thanh ghi để lập trình và lấy dữ liệu của mô-đun này là nguồn mở nên không khó để lập trình. Một điểm đặc biệt khác của mô-đun này là nó có thể điều chỉnh giải đo của gia tốc kế và con quay hồi chuyển thông qua việc truy cập trực tiếp vào thanh ghi.


***Hình 45:*** *MPU-6050*

**QMC5883L:** QMC5883L là một cảm biến từ trường đa vi xử lý ba trục được tích hợp cảm biến từ trường bên trong được sử dụng trong nhiều ứng dụng như drone, robot và thiết bị di động. QMC5883L sử dụng giao thức I2C để đọc dữ liệu và sử dụng bộ phân giải 16-bit và bao gồm nhiều lợi ích như nhiễu tháp, độ chính xác cao, ít tiêu thụ năng lượng, xóa bù và bù nhiệt có độ sai lệch thấp.

***Hình 46:*** *QMC5993L*

**RC Receiver R6DS:** Mô-đun R6DS có nhiệm vụ thu tín hiệu từ tay cầm ở tần số 2.4GHz và xuất xung ra tương ứng với các kênh tín hiệu trên các chân mô-đun. Mô-đun có thể chuyển đổi giữa 2 chế độ hoạt động 6 kênh và 10 kênh thông qua việc giữ nút bấm trên mô-đun trong khoảng 1 giấy, ở mỗi chế độ mô-đun sẽ phát ra đèn khác nhau. Khi đèn LED có màu đỏ mô-đun đang ở trong trạng thái xuất 6 kênh PWM tại 6 chân đầu ra của mô-đun. Khi đèn LED có màu xanh pha tím mô-đun ở trạng thái xuất 10 kênh, chân 1 của mô-đun sẽ xuất xung PPM mang thông tin của 10 kênh tín hiệu, chân 2 của mô-đun sẽ xuất xung SBUS, chân 3 sẽ xuất xung PWM, chân 4 đến chân 6 sẽ xuất ra xung PWM độc lập liên tục. Trong đồ án này, Quadcopter sẽ chỉ sử dụng chân 1 của mô-đun R6DS để nhận tín hiệu từ tay cầm dưới dạng xung PPM.

Đẻ có thể kết nối tay cầm với mô-đun người dùng cần: đặt tay cầm và mô-đun trong khoảng 1 mét, bật tay cầm rồi cấp nguồn cho R6DS, kết nối R6DS với bọ điều khiển, ESC hoặc servo, giữ nút trên motor cho đến khi đèn trên mô-đun nhấp nháy, sau khi nhấp nháy khoảng 8 lần đèn LED sẽ sáng liên tục để thông báo kết nối thành công.

Một số điểm lưu ý khi dùng mô-đun này bao gồm: giữ cho ăng ten mô-đun thẳng nhất có thể để tối đa khoảng nhận tín hiệu, đặt cách xa các vật làm bằng kim loại hoặc cacbon, đặt ăng ten cách xa motor, ESC hoặc các nguồn nhiễu nhiều nhất có thể và mô-đun nên bọc giảm xóc khi lắp đặt vào Quadcopter.


***Hình 47:*** *Mô-đun thu tín hiệu RC R6DS*

**Mô-đun Telemetry:** Mô-đun telemetry là một thiết bị thu thập và truyền tải dữ liệu từ xa đến trạm mặt đất, telemetry gửi đi những dữ liệu từ thiết bị bay bao gồm độ cao, vận tốc, tình trạng pin. Trong đồ án này, mô-đun telemetry cung cấp dữ liệu thời gian thực thông qua giao thức UART của bo điều khiển về trạm mặt đất, điều này giúp đảm bảo người điều khiển có thể xác định tình trạng của của Quadcopter. Mô-đun hoạt động ở hai tần số là 433MHz và 915MHz.


***Hình 48:*** *Mô-đun telemetry [40]*

**ESC:** ESC là một loại thiết bị điện tử được dùng để thay đổi vận tốc của động cơ 3 pha. ESC nhận tín hiệu xung PWM và đo chu kì nhiệm vụ của xung PWM để điều khiển vận tốc động cơ. Khi lần đầu sử dụng ta cần phải hiệu chỉnh cho ESC, hiệu chỉnh ở đây là đặt khoảng lớn nhất và nhỏ nhất để điều khiển vận tốc bởi mỗi thiết bị xuất xung trong khoảng chu kì nhiệm vụ khác nhau.

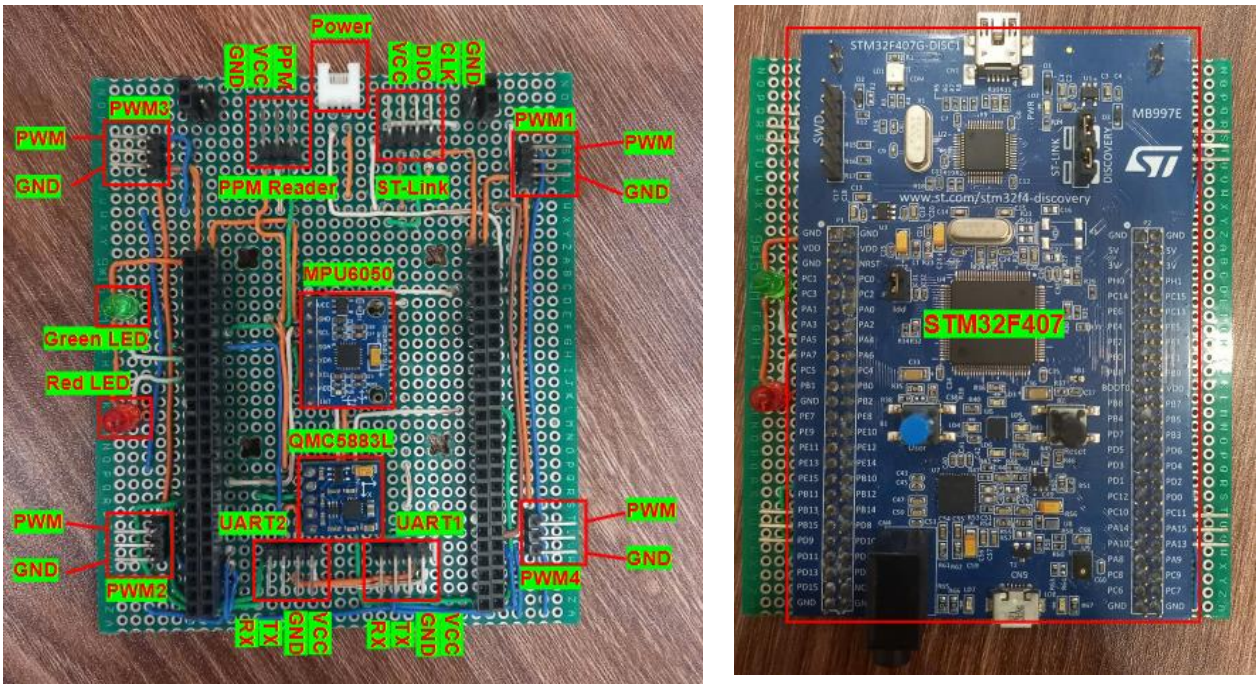

***Hình 49:*** *ESC [41]*

**BLDC Motor A2212/13T 1000KV:** Động cơ được sử dụng ở đây là động cơ không chổi than A2212/13T 1000KV sử dụng rô-to ở ngoài. Động cơ được cấp nguồn bởi pin Lithium và công suất tối đa là 150W.


***Hình 50:*** *Động cơ BLDC A2212/13T 1000KV [42]*

### 3.2.4. Sản phẩm hoàn thiện

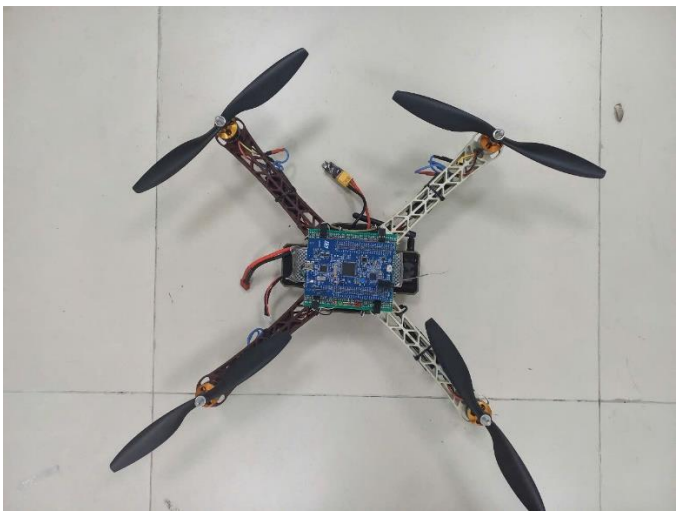
**Hình 51-a** là hình ảnh lớp phía bên dưới bo mạch điều khiển bay hoàn thiện, mặt dưới bộ điều khiển gồm mô-đun MPU6050 và QMC5883L được cố định trên bo mạch, hai cổng kết nối phía trên là cổng đọc tín hiệu PPM và cổng ST-Link dùng để nạp chương trình điều khiển, cổng kết nối trên cùng là cổng cấp nguồn được kết nối từ bộ phân phối năng lượng, cổng kết nối bên dưới là cổng giao thức UART, cổng đầu ra ở 4 góc của bo mạch là đầu ra xung PWM, thêm vào đó là 2 đèn led đỏ và xanh để thể hiện tình trạng của Quadcopter. Sau khi kết nối với bo điều khiển STM32F407 bo điều khiển bay có thiết kế hoàn thiện như **Hình 51-b**.




***(a)*** *Lớp phía dưới bộ điều khiển* ***(b)*** *Lớp phía trên bộ điều khiển*

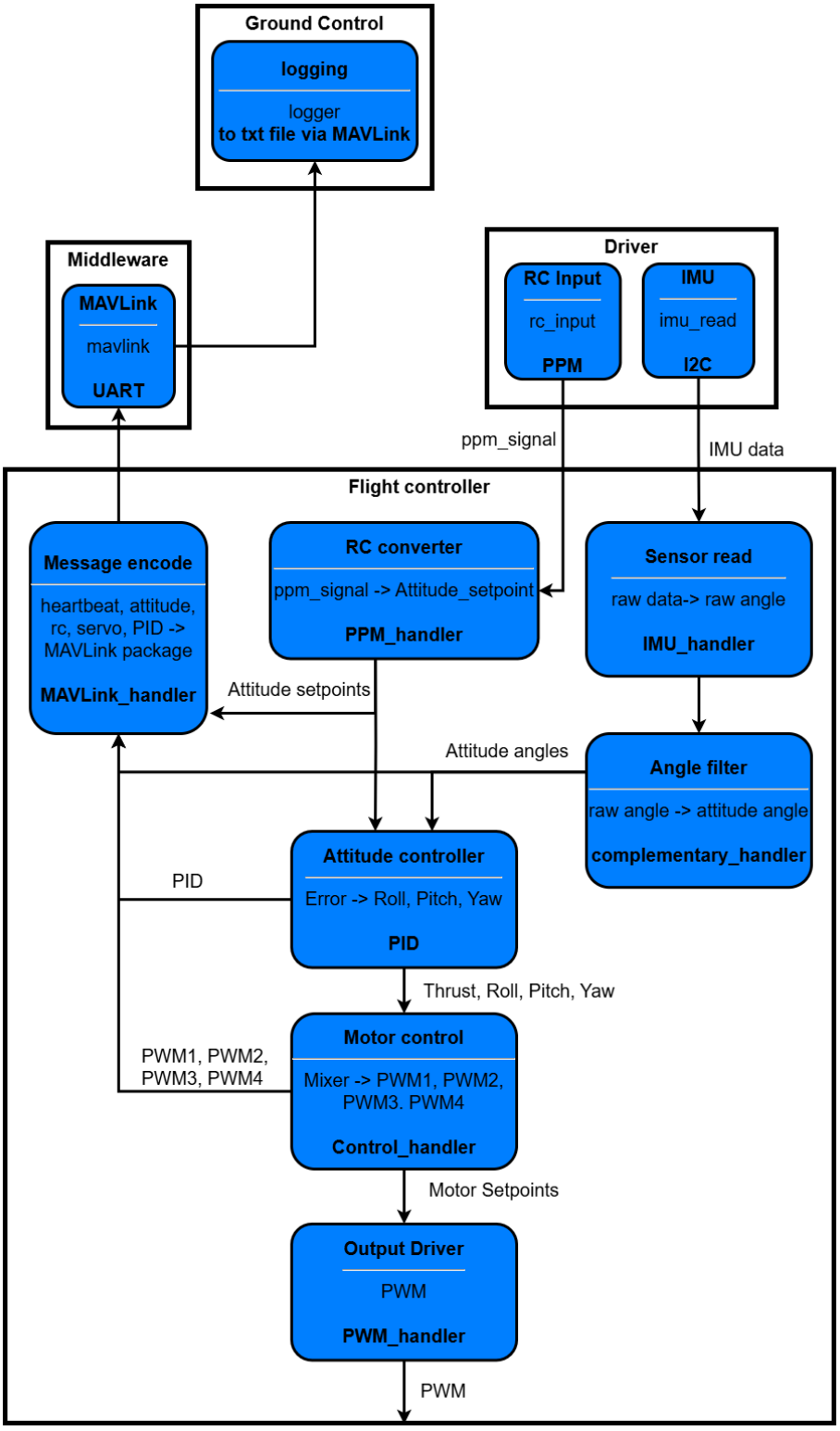
***Hình 51:*** *Bộ điều khiển bay*

**Hình 52** phía bên dưới là hình ảnh Quadcopter hoàn thiện sau khi được đặt thêm pin, ESC, động cơ, cánh quạt bộ điều khiển bay cũng như các mô-đun lên khung Quadcopter.



***Hình 52:*** *Quadcopter hoàn chỉnh*

## 3.3. Thiết kế thuật toán cho Quadcopter

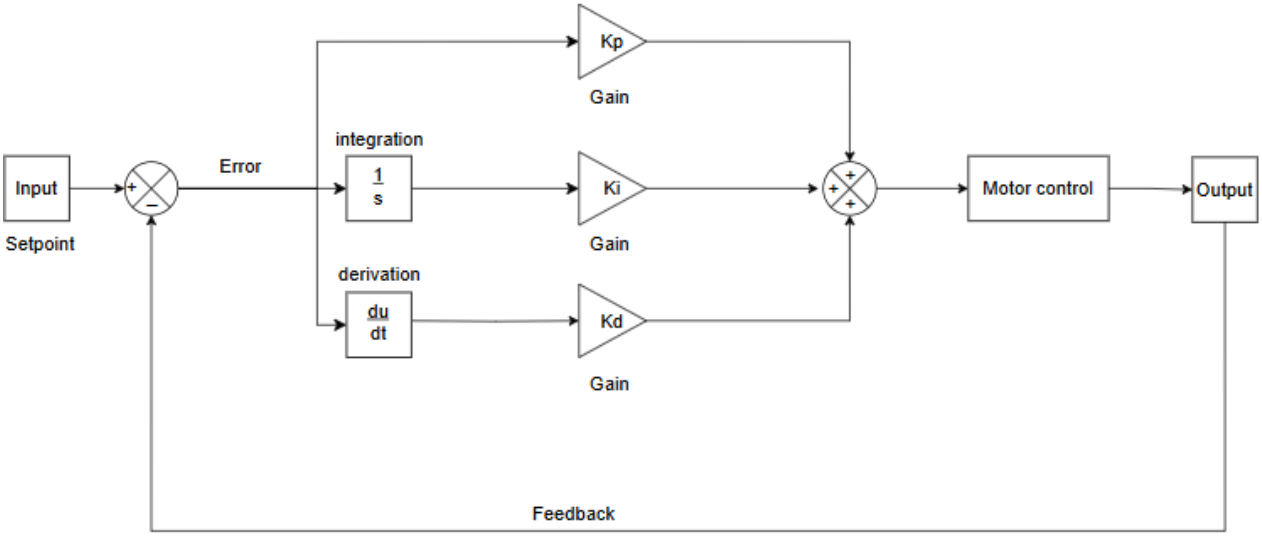



***Hình 53:*** *Sơ đồ kiến trúc phần mềm chương trình Quadcopter*

Từ sơ đồ mô tả hệ thống ở **Hình 53** có thể suy ra được sơ đồ kiến trúc phần mềm của chương trình điều khiển Quadcopter, ở lớp Driver chương trình sẽ đọc dữ liệu giá trị góc từ cảm biến và tín hiệu RC từ tay điều khiển trong đó dữ liệu cảm biến sẽ được tính toán thành góc quay của Quadcopter và tín hiệu RC sẽ được giải mã sang giá trị điều khiển bay của Quadcopter. Hai dữ liệu giá trị góc quay và giá trị điều khiển bay sẽ được đưa vào bộ PID và bộ Mixer để tính toán ra vận tốc của từng động cơ. Cuối cùng tín hiệu điều khiển động cơ sẽ được xuất ra ESC bằng xung ESC, các dữ liệu về góc quay, điều khiển, tình trạng, PID và giá trị điều khiển động cơ sẽ được gói vào một tệp tin MAVLink và gửi về trạm mặt đất để lưu dữ liệu phân tích. Các thuật toán tính toán PID, bộ lọc bù và điều khiển bay sẽ được biểu diễn dưới dạng mã giả ở các phần sau.

### 3.3.1. Thiết kế thuật toán chương trình tính PID

Trong đồ án này thuật toán thuật toán chương trình PID sẽ có dạng sơ đồ khối như **Hình 54.** Input của thuật toán là tín hiệu góc Setpoint nhận từ tay cầm điều khiển, Output của thuật toán bao gồm góc đo Feedback lấy từ cảm biến sau khi đưa kết quả bộ PID vào thuật toán điều khiển động cơ, Error của thuật toán là sai khác giữa góc Input của tay cầm điều khiển và giá trị Feedback khi đo từ cảm biến.



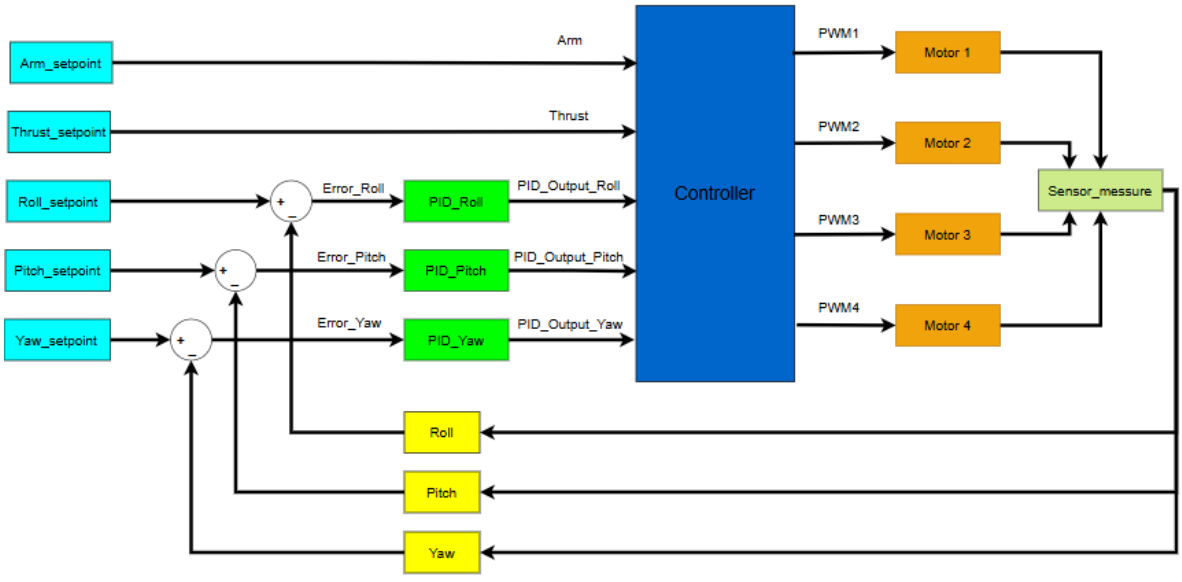

***Hình 54:*** *Sơ đồ khối thuật toán PID*

Sơ đồ khối của thuật toán có thể chuyển sang dạng mã giả của thuật toán chương trình tính toán PID như sau:

```
Input: setpoint, feedback
Output: PID  output
Function PID_Calculate(setpoint, feedback)
    error ← setpoint - feedback
    P ← Kp * error
    I  ← I + error * Ki * dt
    D ← (error – previousError) * Kd / dt
    PID  output ← P + I + D
    previousError ← error
    return PID  output
end function
```

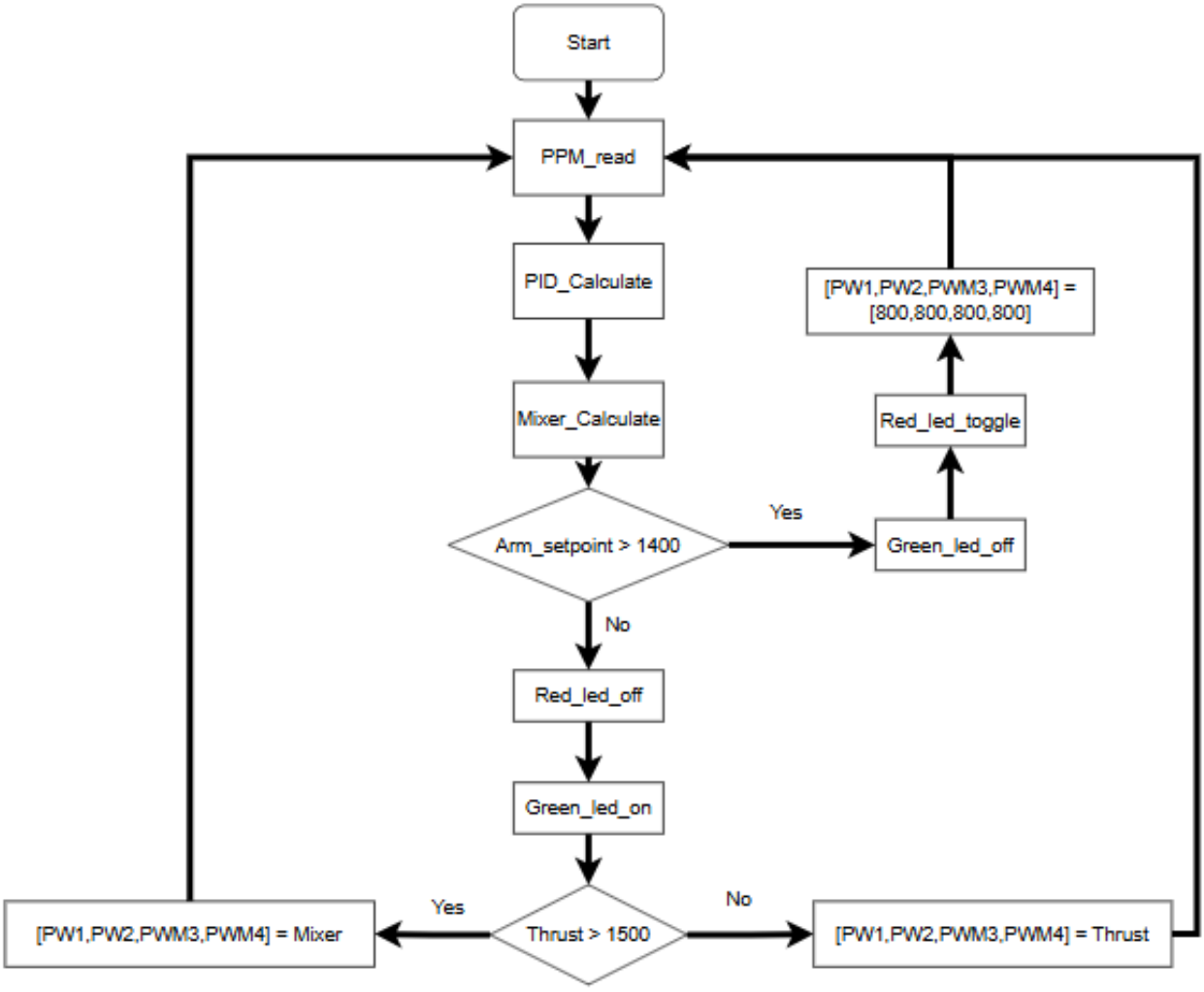
### 3.3.2. Thiết kế thuật toán điều khiển bay cho Quadcopter

Chương trình điều khiển dùng để điều khiển chuyển động Quadcopter đồng thời tự cân bằng Quadcopter khi bay. Sơ đồ trong **Hình 55** mô tả sơ đồ khối thuật toán điều khiển chuyển động Quadcopter có trung tâm thuật toán nằm trong khối Controller và đầu vào của sơ đồ là dữ liệu từ 5 kênh tín hiệu từ tay cầm điều khiển là Arm_setpoint, Thrust_setpoin, Roll_setpoint, Pitch_setpoint và Yaw_setpoint trong đó Arm_setpoint và Thrust_setpiont sẽ được đưa trực tiếp vào thuật toán điều khiển còn Roll_setpoint, Pitch_setpoint và Yaw_setpoint sẽ được đưa vào bộ tính toán PID cùng với dữ liệu giá trị góc được lấy từ cảm biến ở đầu ra thuật toán. Sau khi đi qua bộ tính toán PID kết quả đầu ra PID của 3 góc Euler sẽ được đưa vào thuật  toán để tính toán vận tốc đầu ra của cộng cơ sau đó thuật sẽ trả lại kết quả vận tốc của 4 động cơ dưới dạng kết quả xung PWM. Sau khi xuất xung điều khiển động cơ Motor chương trình sẽ lấy dữ liệu giá trị góc Euler từ cảm biến để đưa lại vào bộ PID cho lần tính toán tiếp theo.




***Hình 55:*** *Sơ đồ khối thuật toán điều khiển bay Quadcopter*

**Hình 56** bên dưới là lưu đồ của thuật toán điều khiển bay Quadcopter. Sau khi bắt đầu, thuật toán sẽ đọc xung PPM của 5 kênh từ tay điều khiển, tính toán PID cho 3 góc quay và tính toán đầu ra thuật toán Mixer, sau đó thuật toán sẽ kiểm tra xem giá trị Arm_setpoint có lớn hơn giá trị đặt trước 1400 không, nếu lớn hơn tức Quadcopter chưa được phép hoạt động thì động cơ sẽ không quay và xung PWM sẽ chỉ được xuất ra ở chu kì nhiệm vụ 800ms, đồng thời đèn xanh của bộ điều khiển tắt và đèn đỏ nhấp nháy. Nếu Arm_setpoint nhỏ hơn 1400 thì Quadcopter sẽ vào trạng thái hoạt động bằng cách tắt đèn đỏ và bật đèn xanh. Lúc này thuật toán sẽ đi vào câu lệnh điều kiện kiểm tra xem giá trị Thrust có lớn hơn 1500 không, nếu không lớn hơn thì động sẽ vào trạng thái cất cánh và vận tốc 4 động cơ sẽ bằng nhau và bằng giá trị Thrust còn nếu lớn hơn thì vận tốc 4 động cơ sẽ là đầu ra của thuật toán Mixer để Quadcopter tự cân bằng khi bay. Sau bước này, thuật toán sẽ kết thúc vòng lặp và trở về hàm đọc xung PPM.




***Hình 56:*** *Lưu đồ thuật toán điều khiển bay Quadcopter*

Thông qua sơ đồ và lưu đồ của thuật toán  thì chương trình mã giả của thuật toán điều khiển Quadcopter có dạng như sau:

```
Input: Arm_setpoint, Thrust
Output: PW1M, PWM2, PWM3, PWM4
function ppm_handler()
    if Arm_setpoint > 1400 then
        PWM1, PWM2, PWM3, PWM4 ← 800, 800, 800, 800
        Red_led_on()
        Green_led_off()
    else
        Red_led_off()
        Green_led_on()
        if Thrust >= 1550 then
            PWM1, PWM2, PWM3, PWM4 ← Mixer1, Mixer2, Mixer3, Mixer4
        else
            PWM1, PWM2, PWM3, PWM4 ← Thrust
        end if
    end if
end function
```

### 3.3.3. Thiết kế thuật toán bộ lọc bù

Từ kiến thức về bộ lọc bù ở phần 2.3.2 mã giả của thuật toán lọc bù như dưới đây trong đó biến angle là giá trị góc từ lần tính trước highPass là giá trị góc thu được từ con quay hồi chuyển, lowPass là giá trị vận tốc góc tính từ gia tốc kế, samplingTime là thời gian lấy mẫu của mô-đun cảm biến MPU6050 và timeConstant là khoảng thời gian dùng để xử lý tín hiệu. Giá trị góc sau khi tính ra được sẽ được trả về thông qua hàm return

```
Input: highPass, lowpass, samplingTime, timeConstant
Output: filterAngle
function complementary_filter(highPass, lowpass, samplingTime, timeConstant)
    alpha ← timeConstant/(samplingTime + timeConstant)
    filterAngle ← alpha * highpass + (1 – alpha)*lowpass
    return filterAngle
end function
```

### 3.3.4. Thiết kế thuật toán gửi gói tin MAVLink

Chương trình gửi gói tin MAVLink được khởi tạo trong đồ án này được đặt tần số hoạt động là 100Hz và được sử dụng để gửi 5 gói tin khác nhau, mỗi khi vào trạng thái chạy thuật toán sẽ gửi một trong 5 gói tin trên do đó tần số hoạt động của thuật toán được chia đều cho 5 gói tin và mỗi gói tin gửi đi có tần số bằng nhau là 20Hz. **Bảng 3** là thông tin của 5 gói tin được gửi đi bao gồm ID, tên gói tin, tần số gửi khởi tạo cho mỗi gói tin và nội dung gói tin gửi đi. Nội dung của mỗi gói tin được thay mới mỗi khi cảm biến đọc được dữ liệu mới và khi giá trị điều khiển bay thay đổi. Sau khi gói tin được cập nhập gói

tin gửi đi sẽ được mã hóa, gửi đi và giải mã lại sau khi gói tín được tiếp nhận thành công tại trạm mặt đất.

***Bảng 3:*** *Các tệp tin MAVLink được gửi đi*

| ID | Tên | Tần số | Nội dung |
|---|---|---|---|
| 0 | HEARTBEAT | 20Hz | Thông báo đang có thiết bi kết nối, cung cấp thông tin về thiết bị bay đang gửi gói tin. |
| 30 | ATTITUDE | 20Hz | Gửi thông tin về góc và vận tốc góc Euler |
| 36 | SERVO_OUTPUT_RAW | 20Hz | Gửi thông tin về chu kì nhiệm vụ xung PWM của các động cơ |
| 81 | MANUAL_SETPOINt | 20Hz | Gửi thông tin về giá trị điều khiển bay (roll_setpoint, pitch_setpoint, yaw_setpoint, thrust) |
| 194 | PID_TURNING | 20Hz | Gửi thông tin về giá trị của bộ cân bằng PID |

Từ các yêu cầu khởi tạo và từ **Bảng 3** mã giả sử dụng để gửi gói tin MAVLink có dạng như sau

```
Input: message_index
Output: mavlink_message
function MAVLink_sending(message_index)
    if message_index == 0 then
        mavlink_message ← Hearbeat_message_encode()
        message_index ← 1
    else if message_index == 1 then
        mavlink_message ← Attitude_message_encode()
        message_index ← 2
    else if message_index == 2 then
        mavlink_message ← Servo_message_encode()
        message_index ← 3
    else if message  index == 3 then
        mavlink_message ← Servo_message_encode()
        message_index ← 4
    else if message_index == 4 then
        mavlink_message ← Servo_message_encode()
        message_index ← 0
    end if
    UART_send_messsage(mavlink_message)
end function
```

### 3.3.5. Khởi tạo các tác vụ cho hệ điều hành FreeRTOS

**Hình 57** là lưu đồ chương trình điều khiển hoạt động của Quadcoptet trong đó 6 tác vụ được khởi tạo bao gồm mpu6050_hanlder, pwm_handler, complementary_handler, qmc5883l_hanlder, control_handler và mavlink_hanlder. Mỗi tác vụ sẽ được đặt trong một hàm while chạy vĩnh viễn và được xác định tần số hoạt động bằng hàm TaskDelay().


***Hình 57:*** *Lưu đồ hoạt động của chương trình điều khiển Quadcopter*

**Bảng 4** bên dưới là bảng lập mức độ ưu tiên các tác vụ của các chức năng trong Quadcopter thực hiện trong đồ án trong đó tác vụ có giá trị ưu tiên càng cao có nghĩa mức ưu tiên của nó càng cao, các ô trống không có tác vụ gì. Các tác vụ cũng được đặt tần số hoạt động khác nhau tùy theo mục đích của mỗi tác vụ.

***Bảng 4:*** *Bảng phân mức độ ưu tiên cho các tác vụ*

| Tác vụ | Mức ưu tiên | Tần số hoạt động |
|---|---|---|
| idle | 0 | |
| | 1 | |
| mavlink_handler | 2 | 100 Hz (10ms) |
| control_handler<br>qmc5883l_handler | 3 | 200 Hz (5ms)<br>200 Hz (5ms) |
| pwm_handler<br>complementary_handler<br>mpu6050_handler | 4 | 200 Hz (5ms)<br>200 Hz (5ms)<br>200 Hz (5ms) |

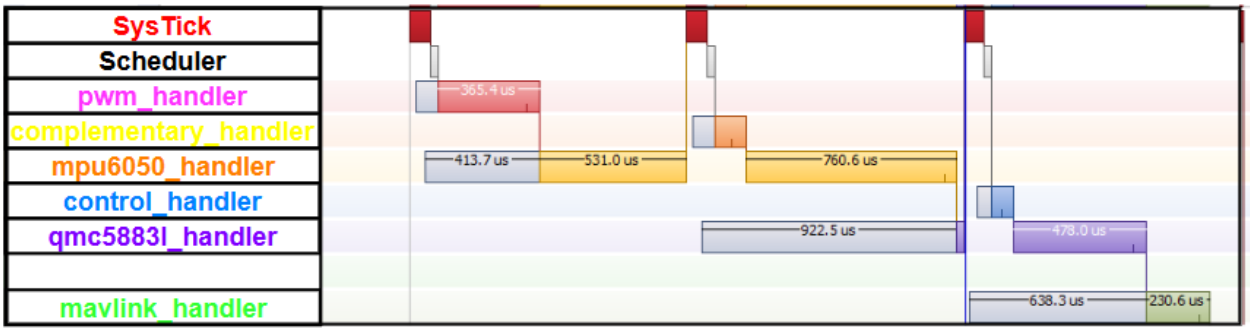
Tác vụ mpu6050_handler là tác vụ có mức ưu tiên 4 và tần số 200Hz dùng để đọc giá trị góc roll và pitch từ gia tốc kế và con quay hồi chuyển. Tác vụ complementary_hanlder có mức ưu tiên 4 và tần số hoạt động 200Hz dùng để dùng bộ lọc bù để tính giá trị góc roll và pitch. Tác vụ pwm_hanlder có mức ưu tiên 4 và tần số hoạt động 200Hz dùng để điều khiển vận tốc động cơ. Tác vụ qmc5883l_hanlder dùng để đọc giá trị góc yaw từ từ trường kế có mức ưu tiên là 3 và tần số hoạt động 200Hz. Tác vụ control_handler có mức ưu tiên 3 và tần số hoạt động 200Hz dùng để giải mã tín hiệu từ tay điều khiển để điều khiển Quadcopter. Tác vụ mavlink_hanlder có mức ưu tiên 2 và tần số hoạt động 100Hz dùng để gửi gói tin Mavlink về trạm mặt đất.

# CHƯƠNG 4: THIẾT LẬP THỬ NGHIỆM

## 4.1. Kiểm nghiệm chương trình điều khiển Quadcopter

### 4.1.1. Kiểm nghiệm khả năng hoạt động của hệ điều hành thời gian thực

Trong phần kiểm nghiệm này hoạt động của hệ điều hành thời gian thực được kiểm tra thông qua phần mềm SEGGER's SystemView, đây là một phần mềm giúp cho người sử dụng quan sát cách một hệ điều hành nhúng hoạt động. Đây là một công cụ mạnh mẽ có thể cung cấp cho người dùng nhiều khả năng trừ ghi lại, phân tích, hiển thị dữ liệu theo thời gian thực, tác vụ, ngắt và các sự kiện [47]. **Hình 58** mô tả kết quả kiểm nghiệm hoạt động các tác vụ của hệ điều hành FreeRTOS được thu được bằng phần mềm SEGGER's SystemView trong đó chức năng chia nhỏ thời gian của hệ điều hành được thực hiện bằng tác vụ Systick mỗi 1 mili giây sẽ được gọi tới một lần. Sau mỗi 1 Systick nếu có tác vụ được gọi lên thực hiện thì bộ lập lịch Scheduler sẽ được gọi để xác xếp thứ tự hoạt động các tác vụ dựa trên mức độ ưu tiên, thứ tự được gọi đến. Bộ lập lịch round robin cũng được thực thi khi xử lý tác vụ complementary_handler và mpu6050_handler khi này có cùng mức ưu tiên sau khi tác vụ Systick xảy ra tác vụ đang được thực thi là mpu6050_handler bị thay thế bằng tác vụ complementary_handler và sau khi tác vụ complementary_handler hoàn thành thì hệ điều hành trả lại quyền thực thi cho tác vụ mpu6050_handler.




***Hình 58:*** *Hoạt động các tác vụ thu được bằng phần mềm*

Sau khi sử dụng phần mềm SEGGER's SystemView đồ án thu được các thông số trung bình của các tác vụ trong chương trình điều khiển sau 5 lần kiểm thử trong **Bảng 5. Bảng 5** cho biết thông tin về tần số hoạt động của các tác vụ, thời gian thực hiện, mức ưu tiên và dung lượng CPU sử dụng cho các tác vụ đó. Có thể thấy rằng tác vụ mpu6050_handler có tần số hoạt động 200Hz, có mức ưu tiên cao nhất và có thời gian hoàn thành tối đa lớn hơn các tác vụ còn lại do đó tác vụ này chiếm 25.21% dung lượng CPU. Hai tác vụ complementary_handler và control_handler có thời gian hoạt động tương đối ngắn trong khoảng từ 0.088ms đến 0.120ms nên hai tác vụ này chỉ chiếm khoảng 2% dung lượng CPU. Hai tác vụ pwm_handler và qmc5883l_handler có tần số hoạt động 200Hz và có thời gian hoạt động tối đa tương đối lớn từ 0.38ms đến 0.48ms nên dung

lượng CPU chiếm 7.36% và 9.44%. Tác vụ mavlink_handler có thời gian hoạt động tối đa 0.23ms tuy nhiên tần số hoạt động của tác vụ này chỉ là 100Hz nên chỉ chiếm 2.04% CPU. Tác vụ Systick mặc định hoạt động mỗi 1 mili giây cho nên không cần có mức ưu tien và chiếm 4.91% CPU. Tác vụ Scheduler chỉ được gọi khi có tác vụ thông báo hoạt động nên không thể xác định được tần số hoạt động của tác vụ này. Từ **Bảng 5** có thể thấy tổng dung lượng của các tác vụ đã sử dụng chiếm 52.47% CPU.

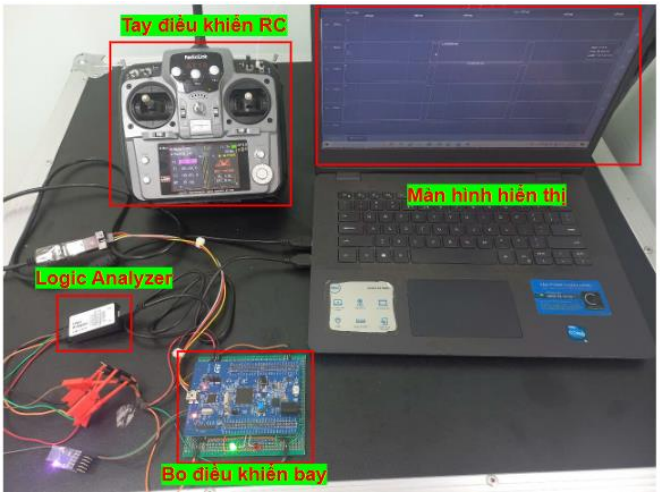
***Bảng 5 :*** *Thông số các tác vụ trong chương trình điều khiển*

| Tác vụ | Mức ưu tiên | Tần số hoạt động | Dung lượng CPU | Thời gian hoàn thành tối đa |
|---|---|---|---|---|
| mpu6050_handler | 4 | 200Hz | 25.21% | 0.774ms |
| complementary_handler | 4 | 200Hz | 2.20% | 0.120ms |
| pwm_handler | 4 | 200Hz | 7.36% | 0.383ms |
| qmc5883l_handler | 3 | 200Hz | 9.44% | 0.486ms |
| control_handler | 3 | 200Hz | 1.61% | 0.088ms |
| mavlink_hanlder | 2 | 100Hz | 2.04% | 0.230ms |
| Systick | | 1000Hz | 4.91% | 0.092ms |
| Scheduler | | | 1.62% | 0.044ms |

### 4.1.2. Kiểm nghiệm khả năng điều khiển động cơ

Trong phần kiểm nghiệm này độ chính xác của xung PWM xuất ra từ động cơ được kiểm tra bằng phần mềm và thiết bị hỗ trợ có tên là Logic Analyzer (Error! R eference source not found.**-a**), đây là một thiết bị dùng để bắt tín hiệu và hiển thị dữ liệu bắt được lên màn hình. Thiết kế thử nghiệm kiểm tra xung PWM của bộ điều khiển bay bao gồm các thiết bị như bo điều khiển bay, Logic Analyzer, tay điều khiển RC để gửi tín hiệu về bộ điều khiển bay và máy tính để hiển thị kết quả lên màn hình (**Hình 59-b**).

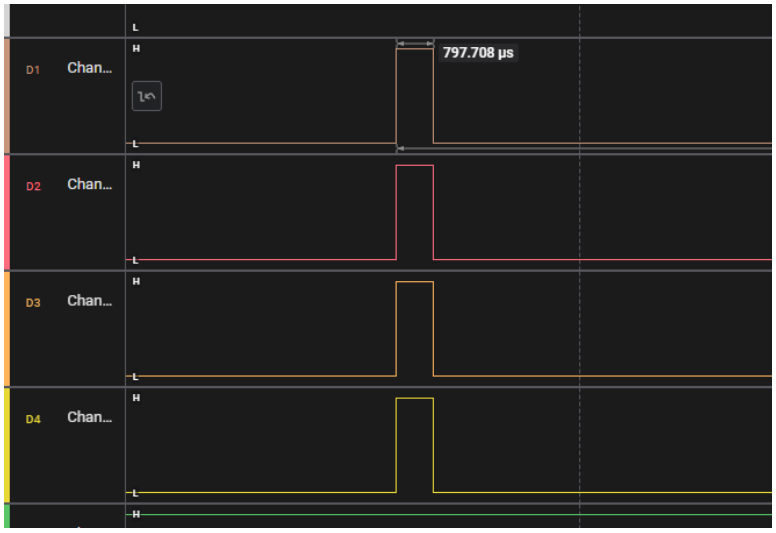

*(a) Logic Analyzer*



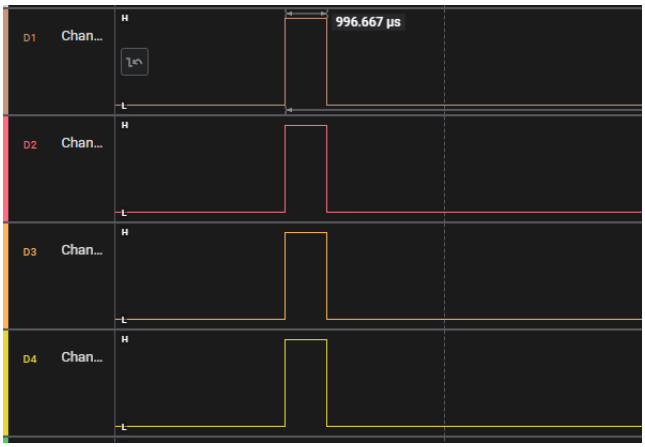

*(b) Thành phần trong thử nghiệm*

***Hình 59:*** *Thiết lập thử nghiệm kiểm tra xung PWM*

**Hình 60** là kết quả khi đo xung đầu ra của 4 chân PWM của bộ điều khiển bay tương với thứ tự từng kết quả tương ứng với từng ESC mà bộ điều khiển cấp xung, cả 4 chân PWM đều hoạt động ở tần số 200Hz. **Hình 60-a** là xung PWM đo được khi Quadcopter đang trong trạng thái Unarmed lúc này chu kì nhiệm vụ ở cả 4 chân PWM là 800 micrô giây. **Hình 60-b** là xung PWM đo được trong trạng thái armed khi cần gạt throttle của tay điều khiển đang ở trạng thái thấp nhất chu kỳ nhiệm vụ xung PWM của cả 4 chân lúc này là 1000 micrô giây còn **Hình 60-c** là kết quả xung đo được khi cần gạt throttle của tay điều khiển đang ở trạng thái cao nhất, lúc này chu kì nhiệm vụ của cả 4 chân là 2000 micrô giây.




*(a) Xung PWM chế độ Unarmed*




*(b) Xung PWM khi throttle min*

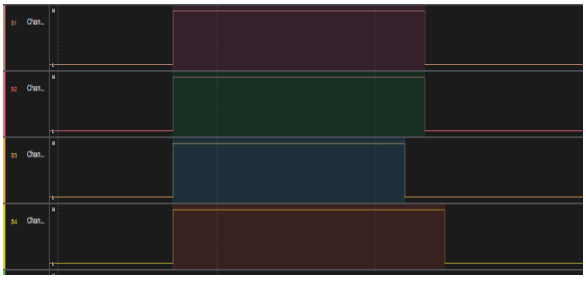

*(c) Xung PWM khi throttle max*

***Hình 60:*** *Xung PWM đo được bằng phần mềm*

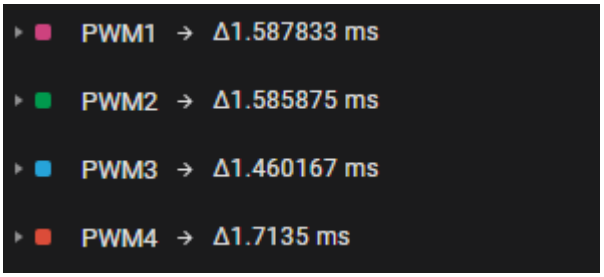
**Hình 61** là xung PWM đo được khi gạt tay điều khiển tại một vị trí bất kì, **Hình 61-a** là chu kỳ nhiệm vụ xung PWM của 4 chân được gửi về máy tính bởi bộ điều khiển bay với các giá trị CCR1,CCR2,CCR3,CCR4 là các giá trị được ghi trực tiếp vào thanh ghi bo điều khiển đại diện chu kỳ nhiệm vụ đầu ra 4 chân PWM1,PWM2,PWM3,PWM4. **Hình 61-b** và **Hình 61-c** là chu kì nhiệm vụ xung PWM được đo bởi phần mềm logic analyzer. Qua những kết quả này ta có thể thấy rằng xung PWM xuất ra có chu kì nhiệm vụ gần đúng với giá trị được gửi từ tay điều khiển qua đó ta có thể xác định xung PWM được cấp ra đúng với chương trình đã thiết kế.

| | |
|---|---|
| (x)= htim4.Instance->CCR1 | 1600 |
| (x)= htim4.Instance->CCR2 | 1589 |
| (x)= htim4.Instance->CCR3 | 1460 |
| (x)= htim4.Instance->CCR4 | 1720 |

*(a) Xung PWM gửi về từ vi điều khiển*



*(b) Xung PWM đo được bằng phần mềm*

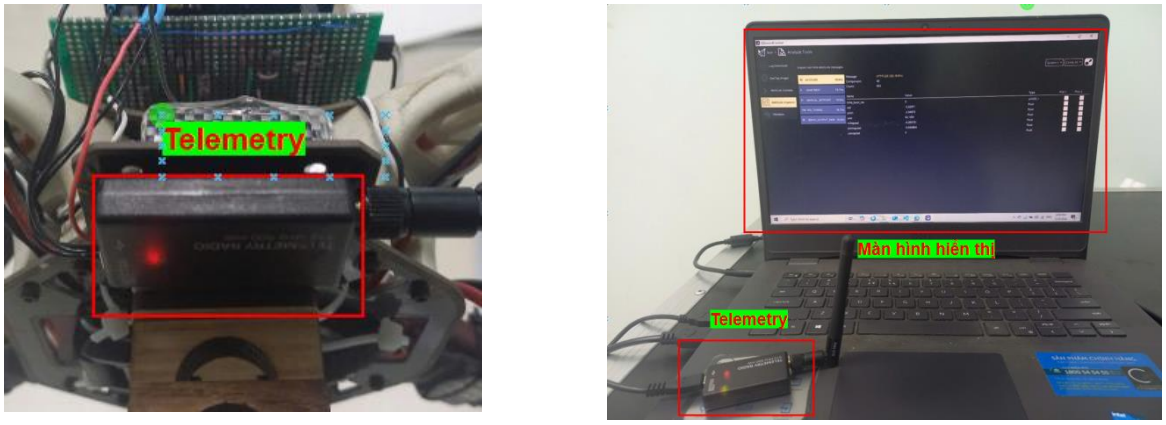



*(c) chu kì nhiệm vụ PWM đo được*

***Hình 61:*** *Xung PWM tại một mức điều khiển bất kì*

### 4.1.3. Thử nghiệm khả năng truyền giao thức Mavlink

Trong phần thử nghiệm này ta sẽ sử dụng 2 phần mềm trạm mặt đất hay GCS đó là QGroundControl và MissionPlanner để đọc tập tin MAVLink được gửi về và kiểm nghiệm độ chính xác của gói tin bằng cách giải mã các gói tin và hiển thị thông số góc Euler trên màn hình. GCS là phần mềm cho phép người dùng có thể giao tiếp và điều khiển Quadcopter hoặc các loại UAV khác, có thể bằng cách cài đặt các thông số để hoạt động tự động hoặc điều khiển trực tiếp UAV [48]. Để có thể kết nối với trạm mặt đất bo điều khiển phải gửi tin nhắn HEARTBEAT về trạm mặt đất để thông báo trạm mặt đất đang có thiết bị gửi tín hiệu đến. **Hình 62** là thiết kế thử nghiệm khả năng truyền tin của gói tin MAVLink, **Hình 62-a** là cài đặt mô-đun telemetry trên Quadcopter đóng vai trò là thiết bị truyền, **Hình 62-b** là thiết lập trạm mặt đất bao gồm mô-đun telemetry đóng vai trò là thiết bị nhận ,màn hình máy tính dùng để khởi tạo phần mềm GCS và hiển thị tình trạng nhận gói tin MAVLink.




***(a)** Thiết lập trên Quadcopter* ***(b)** Thiết lập trạm mặt đất*

***Hình 62:** Thiết lập thử nghiệm khả năng truyền gói tin MAVLink*

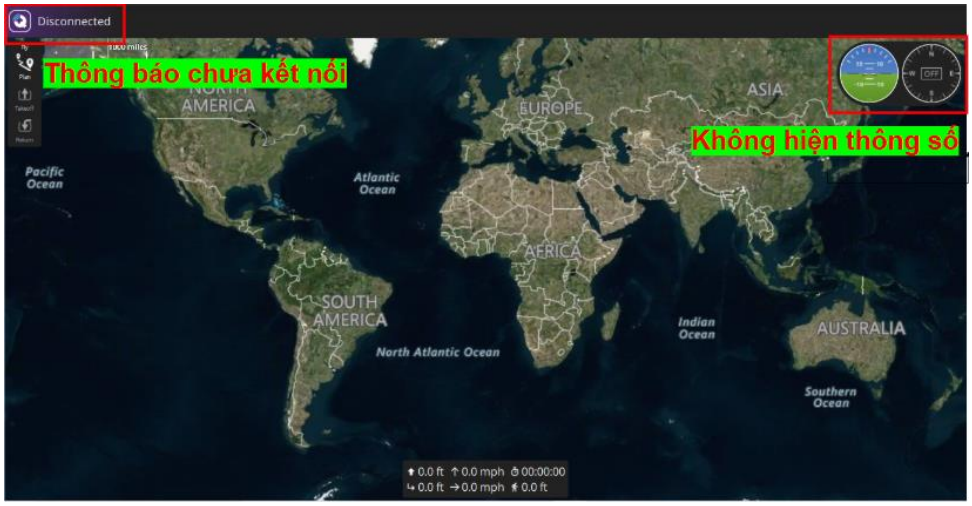
**Hình 63-a** là hình ảnh màn hình phần mềm Mission Planner khi chưa kết nối với Quadcopter bằng giao thức Mavlink, khi này ở phía bên phải màn hình sẽ thông báo biểu tượng chưa được kết nối và bảng thông số phía bên trái màn hình sẽ không hiển thị bất kì thông số nào. **Hình 63-b** là hình ảnh màn hình phần mềm trạm mặt đất Mission Planner sau khi đã kết nối thành công với Quadcopter, ở góc phải phía trên màn hình là thông báo đã kết nối thành công, cổng đang kết nối, tần số truyền tín hiệu và thông báo thiết bị đang gửi tin về trạm mặt đất là Quadcopter, ở phái bên trái màn hình là thông tin về 3 góc Euler là roll, pitch, yaw đang được gửi về trạm mặt đất.


***(a)*** *Màn hình hiển thị khi chưa kết nối*


***(b)*** *Màn hình hiển thị khi đã kết nối*

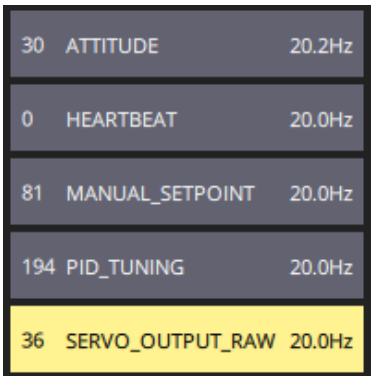
***Hình 63:*** *Phần mềm trạm mặt đất Mission Planner*

**Hình 64-a** là màn hình phần mềm QGroundControl khi chưa kết nối mavlink với Quadcopter, khi này ở phía bên trái màn hình sẽ hiện thông báo chưa kết nối và 2 thước đo ở phía bên phải màn hình không hiển thị giá trị gì. **Hình 64-b** là màn hình phần mềm QGroundControl khi đã kết nối Mavlink với Quadcopter, khi này 2 thước đo ở góc phải màn hình sẽ hiển thị dữ liệu giá trị góc Euler. **Hình 64-c** là danh sách hiển thị các tin nhắn Mavlink được gửi gửi bộ điều khiển bay với giá trị bên phải là giá trị ID của tin nhắn và giá trị bên trái gói tin là tần số gói tin được gửi. Từ **Hình 64-c** có thể thấy tần số gửi của cả 5 gói tin là 20Hz trùng với thiết lập đã thiết lập từ trước.




***(a)*** *Màn hình hiển thị khi chưa kết nối*


***(b)*** *Màn hình hiển thị khi đã kết nối*




***(c)*** *Danh sách gói tin Mavlink*

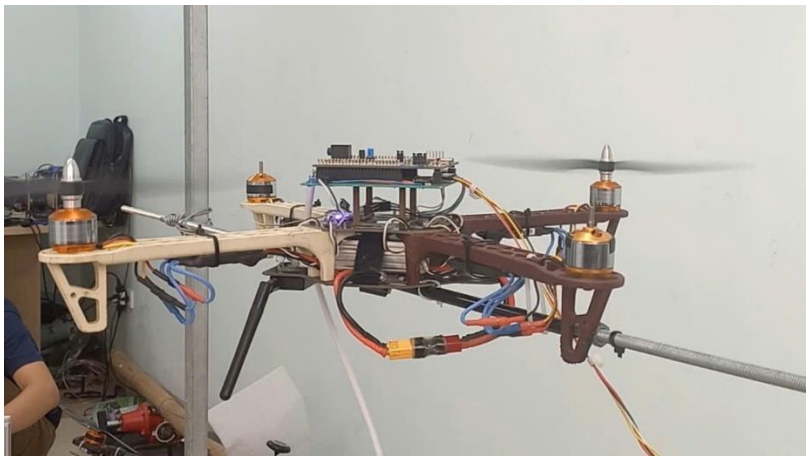
***Hình 64:*** *Phần mềm QGroundControl*

## 4.2. Thử nghiệm khả năng cân bằng của Quadcopter

### 4.2.1. Thử nghiệm với mô hình cân bằng 2 trục

Như đã tìm hiểu ở phần 2.1.4 việc điều khiển trạng thái bay của Quadcopter phụ thuộc vào việc vận tốc quay của từng động cơ mà thông qua mô hình Mixer vận tốc của động cơ phụ thuộc vào 4 giá trị điều khiển bay là lực nâng, giá trị điều khiển theo 3 góc roll, pitch, yaw trong đó kết quả điều khiển của mỗi trục roll, pitch, yaw là kết quả của một bộ điều khiển PID khác nhau do đó cần có 3 bộ PID để điều khiển Quadcopter cân bằng trên không. Để có thể tìm được giá trị của cả 3 bộ PID một cách dễ dàng hơn thì trước tiên cần tìm giá trị của 2 bộ PID roll và pitch, lý do cần tìm kết quả PID của hai góc roll và pitch cùng nhau là vì hệ số Kp, Ki và Kd của hai bộ PID này có giá trị tương tự nhau. Trong lần thử nghiệm này, trục yaw của Quadcopter được giữ nguyên bằng cách cố định động cơ 3, 4 và thân của Quadcopter vào giá treo như ***Hình 65*** đồng thời trong thử

nghiệm này chỉ động cơ 1 và 2 được sử dụng. Bằng cách tác động vào Quadcopter trong quá trình thử nghiệm và quan sát kết quả thu được từ đó tìm ra hệ số phù hợp cho bộ điều khiển PID.



**Hình 65:** *Thiết lập thử nghiệm cân bằng Quadcopter 2 trục*

Từ những thiết lập trong phần này và từ kiến thức được trình bày về mô hình Mixer ở phần 2.1.4 giá trị vận tốc của hai động cơ 1 và 2 được tính bằng biểu thức ma trận có dạng:

$$\begin{bmatrix} PWM1 \\ PWM2 \\ PWM3 \\ PWM4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} thrust \\ roll \\ pitch \\ yaw \end{bmatrix} \quad (20)$$

Công thức (20) còn có thể đưa về dưới dạng biểu thức sau:

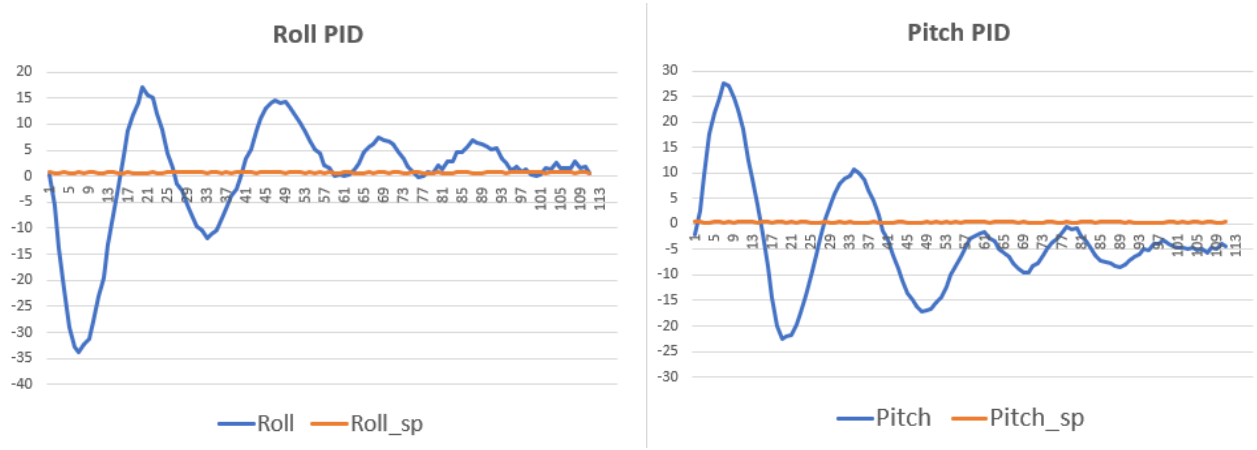
$$PWM1 = thrust - roll + pitch \quad (21)$$

$$PWM1 = thrust - roll + pitch \quad (22)$$

Như có thể thấy ở công thức (21) và (22) bằng cách thiết lập thử nghiệm như trên thì việc điều khiển cân bằng Quadcopter chỉ còn phụ thuộc vào kết quả bộ PID của 2 trục roll và pitch.

Bằng phương pháp thực nghiệm thì hệ số của hai bộ PID trục roll và pitch tìm được có giá trị bằng nhau lần lượt là $K_p = 10.0, K_i = 0.5, K_d = 0.9$. **Hình 66** dưới đây là dữ liệu về góc thu được khi thử nghiệm trong đó giá trị Roll và Pitch là giá trị góc thu được từ cảm biến, giá trị Roll_sp và Pitch_sp là giá trị điều khiển mong muốn của 2 trục roll và pitch và bằng 0. Như có thể thấy ở **Hình 66-a** và **Hình 66-b** giá trị góc đo được của 2 trục roll và pitch hội tụ về giá trị điều khiển đúng như kết quả đầu ra của một bộ

PID lý tưởng qua đó khẳng định hệ số cho 2 bộ điều khiển roll và pitch là phù hợp để sử dụng cho phần thử nghiệm sau.




***(a)** Kết quả bộ PID trục Roll* ***(b)** Kết quả bộ PID trục Pitch*

***Hình 66:** Kết quả thử nghiệm cân bằng 2 trục*

### 4.2.2. Thử nghiệm với mô hình cân bằng 3 trục

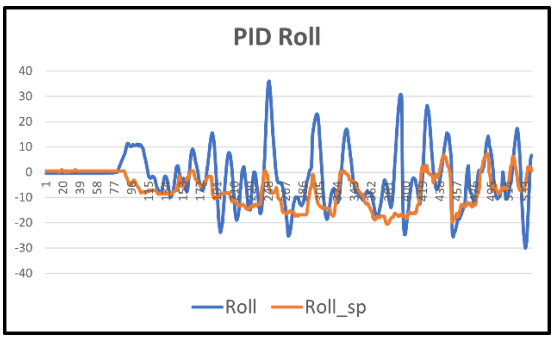
Trong phần thử nghiệm này, cả 3 bộ PID của Quadcopter đều sẽ được thử nghiệm trong đó hệ số bộ PID của hai trục roll và pitch đã tìm được thông qua thử nghiệm trước do đó nhiệm vụ trong thử nghiệm này chỉ cần phải tìm hệ số bộ PID của trục yaw. **Error! Reference source not found.** là thiết lập của thử nghiệm này trong đó Quadcopter sẽ được thử nghiệm bay trong môi trường thực tế để tìm ra hệ số PID.

***Hình 67:*** *Thiết lập thử nghiệm cân bằng Quadcopter 3 trục*

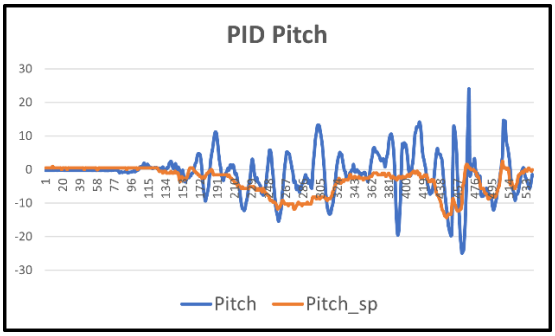
Bằng phương pháp thực nghiệm hệ số PID của 3 trục Roll, Pitch, Yaw được tìm ra và được thể hiện trong **Bảng 6**.

***Bảng 6:*** *Kết quả hệ số PID cân bằng 3 trục*

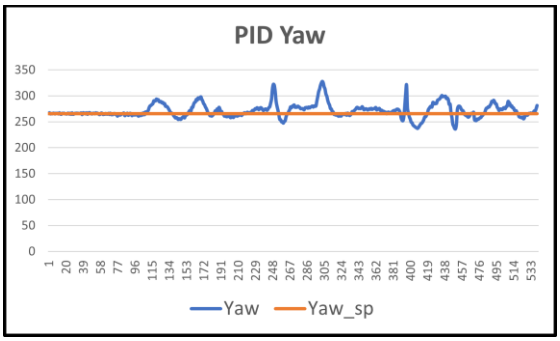
| | **Kp** | **Ki** | **Kd** |
|---|---|---|---|
| **Roll** | 10.9 | 0.5 | 0.9 |
| **Pitch** | 10 | 0.5 | 0.9 |
| **Yaw** | 15.7 | 0.2 | 0.8 |




*() Kết quả PID Roll 3 trục*




*() Kết quả PID Pitch 3 trục*




*() Kết quả PID Yaw 3 trục*

***Hình 68:** Kết quả thử nghiệm cân bằng 3 trục*

**Hình 68** dưới đây là kết quả góc đo được của 3 trục Roll, Pitch, Yaw được gửi về phần mềm trạm mặt đất trong thử nghiệm này. Mặc dù đã tìm ra hệ số PID để Quadcopter cân bằng khi bay tuy nhiên hệ số này chưa phải là hệ số tối ưu nhất do Quadcopter vẫn xuất hiện hiện tượng trôi và rung do đó cần phải nghiên cứu thêm trong tương lai.

## 4.3. Kết luận

Qua các thử nghiệm của chương 4 có thể đưa ra một số kết luận sau bao gồm:

Thông qua thử nghiệm với hệ điều hành FreeRTOS được tích hợp trên bộ điều khiển bay Quadcopter có thể thấy khả năng lập lịch các tác vụ hoàn toàn chính xác như những lý thuyết đã được đưa ra và tần số hoạt động của các tác vụ cũng đáp ứng được yêu cầu trong bước khởi tạo.

Thông qua thử nghiệm kiểm tra xung đầu ra PWM của bộ điều khiển bay có thể thấy bộ điều khiển bay gửi xung đúng như những cài đặt về tần số và chu kì nhiệm vụ đã khởi tạo từ trước đồng thời việc điều khiển xung PWM bằng tay cầm điều khiển RC cũng cho thấy kết quả điều khiển tương đối chính xác.

Thử nghiệm gói tin MAVLink khẳng định khả năng giao tiếp với đa dạng phần mềm trạm mặt đất bằng giao thức truyền thông MAVLink đồng thời tần số gửi của các gói tin cũng bằng tần số gửi gói tin đã được khởi tạo trong bước thiết lập thuật toán.

Thông qua thử nghiệm PID có thể đưa ra hệ số PID của hai trục roll và pitch một các tương đối tuy nhiên việc tìm hệ số tối ưu nhất cho trục yaw cần có thêm thời gian nghiên cứu.

# KẾT LUẬN

## Kết quả đạt được

Sau khi hoàn thành đồ án có thể rút ra một số kết luận của từng chương như sau:

- **Chương 1** của đồ án đã đưa ra các thông tin tổng quan về UAV, Quadcopter và hệ điều hành thời gian thực
- **Chương 2** của đồ án đã đưa ra được các lý thuyết cơ bản về điều khiển Quadcopter bao gồm nguyên lý hoạt động của các cảm biến, nguyên lý tạo lực nâng Quadcopter, nguyên lý điều khiển bay, lý thuyết về bộ cân bằng PID, các chuẩn giao tiếp và giao thức MAVLink.
- **Chương 3** của đồ án đã đưa ra được các bước để thiết kế một bộ điều khiển bay Quadcopter hoàn chỉnh và các thuật toán điều khiển bay được sử dụng để điều khiển Quadcopter.
- **Chương 4** của đồ án đã đưa ra một số kết quả thực nghiệm khả năng hoạt động của Quadcopter bao gồm khả năng hoạt động của hệ điều hành thời gian thực, khả năng điều khiển động cơ, khả năng truyền gói tin MAVLink và thiết lập thử nghiệm để tìm ra hệ số PID phù hợp để cân bằng Quadcopter.

## Kết quả chưa đạt được

Tuy nhiên kết quả đạt được vẫn gặp phải thiếu sót bao gồm:

- Mặc dù đã tìm ra hệ số PID giúp cho Quadcopter cân bằng tuy nhiên chưa phải là hệ số tối ưu nhất.
- Hệ điều hành FreeRTOS mặc dù đã hoạt động như đã khởi tạo tuy nhiên hệ điều hành này mặc định chỉ hoạt động chính xác ở mức mili giây

## Định hướng phát triển

- Tìm hệ số PID phù hợp.
- Thiết kế bộ điều khiển bay bằng bảng mạch PCB
- Tích hợp thêm nhiều cảm biến (GPS, mô-đun đo khoảng cách, Camera) lên bộ điều khiển bay.
- Tích hợp thêm nhiều chế độ bay cho Quadcopter (giao nhiệm vụ bay thông qua giao thức MAVLink, khả năng tự cân bằng độ cao, khả năng tự động tránh vật cản, khả năng tự trở về khi mất tín hiệu).
- Ứng dụng thêm nhiều hệ điều hành thời gian thực nhằm tìm ra hệ điều hành hoạt động chính xác ở mức cao hơn.

# TÀI LIỆU THAM KHẢO


[1] L. R. Newcome, *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles*. AIAA, 2004.

[2] N. Barrera, *Unmanned Aerial Vehicles*. in Robotics Research and Technology Ser. New York: Nova Science Publishers, Incorporated, 2021.

[3] "VTOL Drone for Intelligence, Surveillance, and reconnaissance (ISR) - Helvetis." Accessed: Nov. 26, 2024. [Online]. Available: https://helvetis.com/vtol-uav-isr/

[4] G. N. Muchiri and S. Kimathi, "A Review of Applications and Potential Applications of UAV," *Proceedings of the Sustainable Research and Innovation Conference*, pp. 280–283, Apr. 2022.

[5] M. Mendez, "The Critical Role of INS in Disaster Management," Inertial Labs. Accessed: Nov. 26, 2024. [Online]. Available: https://inertiallabs.com/the-critical-role-of-ins-in-disaster-management/

[6] "Consumer demand, new technology and trade tariffs: What are the trends impacting Agriculture businesses in 2019?" Accessed: Nov. 26, 2024. [Online]. Available: https://www.hlb.global/consumer-demand-new-technology-and-trade-tariffs-what-are-the-trends-impacting-agriculture-businesses-in-2019/

[7] "UAV 2024: The Future of Unmanned Aerial Vehicles," Aeronautics. Accessed: Nov. 26, 2024. [Online]. Available: https://aeronautics-sys.com/the-future-of-unmanned-aerial-vehicles/

[8] A. Abbas, "Real Time Operating Systems(RTOS) For Drones | Asad Abbas," Oct. 2018. doi: 10.13140/RG.2.2.10243.35369.

[9] Oscar, "What is a Quadcopter and How does it Fly?," Oscar Liang. Accessed: Dec. 05, 2024. [Online]. Available: https://oscarliang.com/what-is-quadcopter/

[10] Z. Zheng and G. Xiao, "Evolution analysis of a UAV real-time operating system from a network perspective," *Chinese Journal of Aeronautics*, vol. 32, no. 1, pp. 176–185, Jan. 2019, doi: 10.1016/j.cja.2018.04.011.

[11] "NuttX Board Porting Guide | PX4 Guide (main)." Accessed: Dec. 09, 2024. [Online]. Available: https://docs.px4.io/main/en/hardware/porting_guide_nuttx.html

[12] "ardupilot/libraries/AP_HAL_ChibiOS/hwdef/scripts at master · ArduPilot/ardupilot," GitHub. Accessed: Dec. 10, 2024. [Online]. Available: https://github.com/ArduPilot/ardupilot/tree/master/libraries/AP_HAL_ChibiOS/hwdef/scripts

[13] N. Schiller *et al.*, "Drone Security and the Mysterious Case of DJI's DroneID," in *Proceedings 2023 Network and Distributed System Security Symposium*, San Diego, CA, USA: Internet Society, 2023. doi: 10.14722/ndss.2023.24217.

[14] “LibrePilot System Architecture - LibrePilot Documentation - LibrePilot.” Accessed: Dec. 10, 2024. [Online]. Available: https://librepilot.atlassian.net/wiki/spaces/LPDOC/pages/100523730/LibrePilot+System+Architecture

[15] “Why use FreeRTOS? - FreeRTOS™.” Accessed: Nov. 27, 2024. [Online]. Available: https://freertos.org/Why-FreeRTOS/Why-FreeRTOS

[16] mtimmons, “Propeller Blade, ‘Lift,’” MTwallets. Accessed: Nov. 27, 2024. [Online]. Available: https://www.mtwallets.com/propeller-blade-lift/

[17] “14.8: Bernoulli’s Equation,” Physics LibreTexts. Accessed: Nov. 27, 2024. [Online]. Available: https://phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/Book%3A_University_Physics_I_-_Mechanics_Sound_Oscillations_and_Waves_(OpenStax)/14%3A_Fluid_Mechanics/14.08%3A_Bernoullis_Equation

[18] O. Gur, “Maximum Propeller Efficiency Estimation,” *Journal of Aircraft*, vol. 51, no. 6, pp. 2035–2038, Nov. 2014, doi: 10.2514/1.C032557.

[19] A. Janota, V. Šimák, D. Nemec, and J. Hrbček, “Improving the Precision and Speed of Euler Angles Computation from Low-Cost Rotation Sensor Data,” *Sensors*, vol. 15, no. 3, Art. no. 3, Mar. 2015, doi: 10.3390/s150307016.

[20] Aleksi Olkkonen, “A Quadcopter Flight Controller”.

[21] “Mixer Theory.” Accessed: Nov. 27, 2024. [Online]. Available: https://cookierobotics.com/066/

[22] J. Zhu *et al.*, “Development Trends and Perspectives of Future Sensors and MEMS/NEMS,” *Micromachines*, vol. 11, no. 1, p. 7, Dec. 2019, doi: 10.3390/mi11010007.

[23] “What is an IMU? Understanding Inertial Measurement Unit,” GuideNav: The Global Standard in Inertial Navigation Excellence. Accessed: Nov. 27, 2024. [Online]. Available: https://guidenav.com/what-is-an-imu-understanding-inertial-measurement-units/

[24] B. Choi, S.-Y. Lee, T. Kim, and S. S. Baek, “Dynamic Characteristics of Vertically Coupled Structures and the Design of a Decoupled Micro Gyroscope,” *Sensors*, vol. 8, no. 6, pp. 3706–3718, Jun. 2008, doi: 10.3390/s8063706.

[25] “BASICS OF MAGNETORESISTIVE (MR) SENSORS”.

[26] P. Narkhede, S. Poddar, R. Walambe, G. Ghinea, and K. Kotecha, “Cascaded Complementary Filter Architecture for Sensor Fusion in Attitude Estimation,” *Sensors*, vol. 21, no. 6, p. 1937, Mar. 2021, doi: 10.3390/s21061937.

[27] M. Muneeb, “The Balance Filter A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform”, Accessed: Nov. 27, 2024. [Online]. Available: https://www.academia.edu/27146601/The_Balance_Filter_A_Simple_Solution_for_Integrating_Accelerometer_and_Gyroscope_Measurements_for_a_Balancing_Platform

[28] M. M. Bachtiar, F. Ardilla, M. F. Hasbi, and I. K. Wibowo, “PID Control System on Brushless DC Motor for Quadcopter Balance,” *Inf. J. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 6, no. 2, pp. 110–114, Jul. 2021, doi: 10.25139/inform.v6i2.3999.

[29] Theautomization, “Pid Control Basics In Detail : Part 2,” The Automization. Accessed: Dec. 10, 2024. [Online]. Available: https://theautomization.com/pid-control-basics-in-detail-part-2/

[30] “Pulse Width Modulation - Types, Important Parameters and Applications,” GeeksforGeeks. Accessed: Nov. 27, 2024. [Online]. Available: https://www.geeksforgeeks.org/pulse-width-modulation-pwm/

[31] Can_i_trade?, “RCArduino: How To Read RC Receiver PPM Stream,” RCArduino. Accessed: Nov. 27, 2024. [Online]. Available: http://rcarduino.blogspot.com/2012/11/how-to-read-rc-receiver-ppm-stream.html

[32] A. Sahu, D. Ravi, R. Mishra, and P. Gour, “An Implementation of I2C using VHDL for DATA surveillance,” *International Journal on Computer Science and Engineering*, vol. 3, May 2011.

[33] S. Campbell, “Basics of the I2C Communication Protocol,” Circuit Basics. Accessed: Nov. 27, 2024. [Online]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[34] “I2C: How it works and how to use it? – ensatellite.” Accessed: Nov. 27, 2024. [Online]. Available: https://ensatellite.com/i2c/

[35] K. Amrita and A. Kumari, “DESIGN AND VERIFICATION OF UART USING VERILOG HDL”.

[36] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, “Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey,” *IEEE Access*, vol. 7, pp. 87658–87680, 2019, doi: 10.1109/ACCESS.2019.2924410.

[37] “MAVLINK Common Message Set (common.xml) | MAVLink Guide.” Accessed: Dec. 10, 2024. [Online]. Available: https://mavlink.io/en/messages/common.html

[38] R. Le Moigne, O. Pasquier, and J.-P. Calvez, “A generic RTOS model for real-time systems simulation with systemC,” in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Paris, France: IEEE Comput. Soc, 2004, pp. 82–87. doi: 10.1109/DATE.2004.1269211.

[39] M.-Y. Zhu, “Understanding FreeRTOS: A Requirement Analysis,” 2011, *Unpublished*. doi: 10.13140/RG.2.2.12419.09767.

[40] A. Serino and L. Cheng, “A Survey of Real-Time Operating Systems”.

[41] “FreeRTOS scheduling (single-core, AMP and SMP) - FreeRTOS™.” Accessed: Nov. 30, 2024. [Online]. Available: https://freertos.org/Documentation/02-Kernel/02-Kernel-features/01-Tasks-and-co-routines/04-Task-scheduling

[42] R. Barry, “USING THE FREERTOS REAL TIME KERNEL,” 2009.

[43] “QWinOut DIY FPV Drone Quadcopter 4-axle Aircraft Kit :F450 450 Frame + PXI PX4 Flight Control + 920KV Motor +GPS + AT9 Transmitter + Gimbal Camera Mount,” QWinOut. Accessed: Nov. 28, 2024. [Online]. Available: https://qwinout.com/products/qwinout-diy-fpv-drone-quadcopter-4-axle-aircraft-kit-f450-450-frame-pxi-px4-flight-control-920kv-motor-gps-at9-transmitter-gimbal-camera-mount

[44] “QWinOut F450 Drone Frame Kit 4-Axis Airframe 450mm Quadcopter Frame K.” Accessed: Nov. 28, 2024. [Online]. Available: https://qwinout.com/products/qwinout-f450-drone-frame-kit-4-axis-airframe-450mm-quadcopter-frame-kit-with-landing-skid-gear

[45] B. Gunasekaran, “Hardware, Software, Firmware, Middleware, Drivers, OS & Applications, The Difference?,” Embedded Inventor. Accessed: Dec. 05, 2024. [Online]. Available: https://embeddedinventor.com/hardware-software-firmware-middleware-drivers-os-applications-the-difference/

[46] “PX4 System Architecture | PX4 Guide (main).” Accessed: Dec. 06, 2024. [Online]. Available: https://docs.px4.io/main/en/concept/px4_systems_architecture.html

[47] “SEGGER Verify.” Accessed: Nov. 29, 2024. [Online]. Available: https://www.segger.com/verify-embedded-systems/

[48] “Ground Control Stations (GCS) for UAV, Drones and Robotics,” Unmanned Systems Technology. Accessed: Dec. 08, 2024. [Online]. Available: https://www.unmannedsystemstechnology.com/expo/ground-control-stations-gcs/

# Phụ lục: mã giả chương trình điều khiển Quadcopter

```
1:  function mpu6050_handler()
2:      while TRUE
3:          read roll_accel, roll_gyro, pitch_accel, pitch_gyro
4:          taskdelay(5)
5:      end while
6:  end function
7:  function complementary_handler()
8:      while TRUE
9:          roll = complementary_filter()
10:         pitch = complementary_filter()
11:         taskdelay(5)
12:     end while
13: end function
14: function qmc5883l_handler()
15:     while TRUE
16:         read yaw
17:         taskdelay(5)
18:     end while
19: end function
20: function pwm_handler()
21:     while TRUE
22:         If Arm_setpoint > 1400 then
23:             PWM1, PWM2, PWM3, PWM4 ← 800, 800, 800, 800
24:             Red_led_on()
25:             Green_led_off()
26:         Else
27:             Red_led_off()
28:             Green_led_on()
29:             If Thrust >= 1550 then
30:                 PWM1, PWM2, PWM3, PWM4 ← Mixer1, Mixer2, Mixer3, Mixer4
31:             Else
32:                 PWM1, PWM2, PWM3, PWM4 ← Thrust
33:             end if
34:         end if
35:         taskdelay(5)
36:     end while
37: end function
38: function mavlink_handler()
39:     while TRUE
40:         MAVLink_sending(message_index)
```

```
41:         taskdelay(10)
42:     end while
43: end function
44: function control_handler()
45:     while TRUE
46:         throttle_setpoint ← rc_channel3
47:         roll_setpoint ← rc_channel1
48:         pitch_setpoint ← rc_channel2
49:         yaw_setpoint ← rc_channel4
50:         Throttle_setpoint ← rc_channel5
51:         taskdelay(10)
52:     end while
53: end function
```