|  |  |
| --- | --- |
|  |  |
| NGUYỄN HUY TUẤN | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC CNKT ĐIỆN TỬ - VIỄN THÔNG |
|  |
| **THIẾT KẾ MẠCH ĐIỀU KHIỂN MÁY BAY KHÔNG NGƯỜI LÁI** |
|  |
|  |
| **CBHD: Ths. Đinh Thị Kim Phượng** |
| CNKT ĐIỆN TỬ - VIỄN THÔNG | **Sinh viên: Nguyễn Huy Tuấn** |
| **Mã số sinh viên: 1141050451** |
|  |
|  |
|  |
| **Hà Nội – 2020** |

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

**Giáo viên hướng dẫn**

**Th.s Đinh Thị Kim Phượng**

**LỜI CẢM ƠN**

Trong thời gian thực hiện đồ án tốt nghiệp em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy cô, gia đình và các bạn.

Để hoàn thành đồ án tốt nghiệp ngày hôm nay em cũng xin gửi lời cảm ơn chân thành đến các thầy cô Trường Đại Học Công Nghiệp Hà Nội nói chung, các thầy cô khoa Điện tử nói riêng đã dạy cho em kiến thức chuyên ngành, giúp em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ em suốt quá trình học tập.

Em xin gửi lời cảm ơn chân thành đến cô Đinh Thị Kim Phượng, đã tận tình hướng dẫn, chỉ dạy, giúp đỡ và cung cấp những tài liệu cũng như kinh nghiệm quý báu giúp em hoàn thành các nhiệm vụ được giao trong quá trình thực hiện.

Hà Nội, Ngày ... tháng ... năm ...

Sinh viên thực hiện

**Nguyễn Huy Tuấn**

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc41344514)

[Chương 1. Tìm hiểu về máy bay không người lái 2](#_Toc41344515)

[1.1. Tổng quan về máy bay không người lái 2](#_Toc41344516)

[1.2. Nguyên lý hoạt động của quadcopter 3](#_Toc41344517)

[1.3. Tìm hiều về mô hình tuyến tính máy bay 5](#_Toc41344518)

[1.3.1. Hệ quy chiếu 5](#_Toc41344519)

[1.3.2. Các góc Euler 6](#_Toc41344520)

[1.3.3. Ma trận chuyển đổi hệ trục tọa độ 6](#_Toc41344521)

[1.3.4. Hệ phương trình chuyển động 7](#_Toc41344522)

[1.4. Mô hình động cơ và cánh quạt 18](#_Toc41344523)

[1.5. Mô phỏng tìm hệ số PID 19](#_Toc41344524)

[Chương 2. Tìm hiểu bộ điều khiển PID 20](#_Toc41344525)

[2.1. Lý thuyết điều khiển PID 20](#_Toc41344526)

[2.1.1. Khâu tỉ lệ 20](#_Toc41344527)

[2.1.2. Khâu tích phân 20](#_Toc41344528)

[2.1.3. Khâu vi phân 21](#_Toc41344529)

[2.1.4. Tổng hợp 3 khâu 22](#_Toc41344530)

[2.2. Điều chỉnh vòng lặp 22](#_Toc41344531)

[2.2.1. Độ ổn định 23](#_Toc41344532)

[2.2.2. Tối ưu hóa hành vi 23](#_Toc41344533)

[2.3. Các phương pháp điều chỉnh vòng lặp 24](#_Toc41344534)

[2.3.1. Phương pháp thủ công 24](#_Toc41344535)

[2.3.2. Phương pháp Ziegler-Nichols 25](#_Toc41344536)

[2.3.3. Sử dụng phần mềm điều chỉnh PID 25](#_Toc41344537)

[2.4. Các cải tiến đối với thuật toán PID 26](#_Toc41344538)

[2.4.1. Tích phân khởi động 26](#_Toc41344539)

[2.4.2. Đóng băng chức năng tích phân trong trường hợp bị nhiễu 26](#_Toc41344540)

[2.4.3. Thay chức năng tích phân bằng một phần dựa trên mô hình 27](#_Toc41344541)

[Chương 3. Các bước thiết kế quadcopter 28](#_Toc41344542)

[3.1. Thiết kế cơ khí và mạch điện tử 28](#_Toc41344543)

[3.1.1. Thiết kế cơ khí 28](#_Toc41344544)

[3.1.2. Thiết kế điện tử 29](#_Toc41344545)

[3.2. Sơ đồ các module và mạch nguyên lý 37](#_Toc41344546)

[3.2.1. Các khối module 37](#_Toc41344547)

[3.2.2. Mạch nguyên lý khối nhận và xử lý tín hiệu 38](#_Toc41344548)

[3.2.3. Mạch nguyên lý khối truyền tín hiệu 39](#_Toc41344549)

[3.3. Thiết kế bộ điều khiển 39](#_Toc41344550)

[3.3.1. Tìm hiểu về và các phương pháp thiết kế bộ điều khiển PID 39](#_Toc41344551)

[3.3.2. Phương pháp tuyến tính kết hợp mô phỏng 40](#_Toc41344552)

[3.3.3. Phương pháp dò thực nghiệm kết hợp Zieler-Nichols 40](#_Toc41344553)

[3.3.4. Lưu đồ dải thuật cân bằng 41](#_Toc41344554)

[3.4. Quá trình thực nghiệm 42](#_Toc41344555)

[3.4.1. Quá trình dò các thông số PID 42](#_Toc41344556)

[3.4.2. Bay thử nghiểm sản phẩm 44](#_Toc41344557)

[Chương 4. Kết luận 45](#_Toc41344558)

**KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| UAV | * + - * 1. Máy bay không người lái (Unmanned Aerial Vehicle) |
| PID | * + - * 1. Tỷ lệ - tích phân – vi phân (Proportional – Integral – Derivative) |
| ESC | * + - * 1. Bộ điều tốc (Electronic Speed Controller) |
| RX | * + - * 1. Bộ thu sóng radio (Receiver) |
| TX | * + - * 1. Bộ phát sóng radio (Transmitter) |
| BLDC | * + - * 1. Động cơ một chiều không chổi than (Brushless DC Motor) |
| PWM | * + - * 1. Điều chỉnh độ rộng xung (Pulse Width Modulatio) |

**DANH MỤC HÌNH ẢNH**

[Hình 1.1 Mô tả các chuyển động của Quadcopter 4](#_Toc41331099)

[Hình 1.2 Hệ tọa độ vật (hình trái) và hệ tọa độ quán tính (hình phải) 6](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331100)

[Hình 1.3 Quy ước các phương chiều 7](#_Toc41331101)

[Hình 1.4 Sơ đồ mô phỏng để tìm hệ số PID theo sơ đồ sau: 20](#_Toc41331102)

[Hình 2.1 Thiết kế khung cơ khí 21](#_Toc41331103)

[Hình 2.2 Arduino nano 22](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331104)

[Hình 2.3 Esc cho động cơ không chổi than 26](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331105)

[Hình 2.4 Pin Li-Po 27](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331106)

[Hình 2.5 Động cơ 3 pha DC không chổi than 27](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331107)

[Hình 2.6 Module NRF24L01 28](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331108)

[Hình 2.7 Mô hình truyền nhận của Module NRF24L01 28](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331109)

[Hình 2.8 Joystick 29](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331110)

[Hình 2.9 Hardware Diagram 30](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331111)

[Hình 2.10 Mạch nguyên lý khối nhận và xử lý tín hiệu 31](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331112)

[Hình 2.11 Mạch nguyên lý khối truyền tín hiệu 32](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331113)

[Hình 2.12 Lưu đồ dải thuật cân bằng 35](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331114)

[Hình 2.13 Phần mềm hỗ trợ điều chỉnh PID 36](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331115)

[Hình 2.14 Bay thử nghiểm sản phẩm 37](file:///C:\Users\admin\Desktop\doAnLeNgocHung.docx#_Toc41331116)

**DANH MỤC BẢNG BIỂU**

[Bảng 2.1 Thông số của kỹ thuật của Arduino nano 23](#_Toc41312461)

[Bảng 2.2 Thông số kỹ thuật của Cảm biến IMU MPU-6050 25](#_Toc41312462)

[Bảng 2.3 Thông số kỹ thuật của Pin Li-Po WildScorpion 26](#_Toc41312463)

[Bảng 2.4 Thông số kỹ thuật PID dò được 37](#_Toc41312464)

[Bảng 3.1 Thông số kỹ thuật Module thu phát Wifi NodeMcu 39](#_Toc41312465)

MỞ ĐẦU

Máy bay không người lái (UAV) hiện nay được sử dụng khá phổ biến trong các lĩnh vực, nó cung cấp góc quan sát từ trên cao như trong các lĩnh vực giải trí, thể thao.

Ngoài ra UAV còn được sử dụng nhiều trong lĩnh vực quân sự như do thám, cứu hộ và trong nhiệm vụ vận chuyển bưu kiện có khối lượng nhỏ do có ưu điểm linh hoạt, khó bị phát hiện do kích thước và tiếng ồn nhỏ, chi phí vận hành thấp hơn nhiều so với các loại máy bay khác.

UAV hoạt động dự trên nguyên lý cân bằng góc nghiêng của từng cặp động cơ đối diện nhau. Vấn đề đặt ra là cần có một thuật toán điều khiển bốn động cơ, giúp cho máy bay có thể cân bằng từng trục, kết hợp cân bằng các trục với nhau, triệt tiêu quán tính và điều khiển di chuyển ổn định.

Yếu tố quan trọng nhất để có thể điều khiển cân bằng và di chuyển đố là các góc nghiêng phản hồi đọc từ cảm biến phải chính xác, thuật toán điều khiển PID phải chọn được thông số phù hợp để cân bằng và di chuyển mô hình bay.

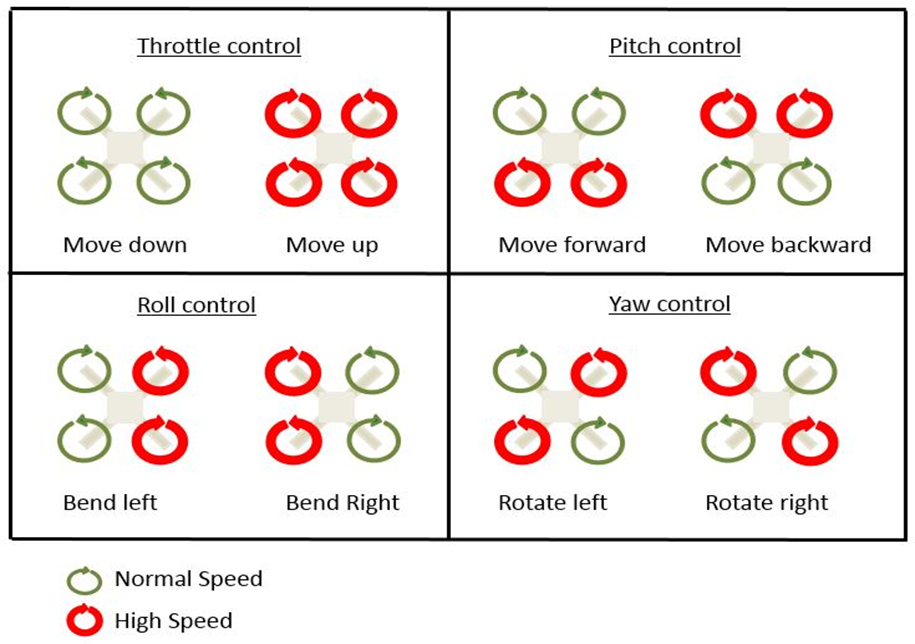
1. Tìm hiểu về máy bay không người lái
   1. Tổng quan về máy bay không người lái

Máy bay không người lái (UAV) là loại máy bay có thể được điều khiển từ xa thông qua hệ thống điều khiển tự động hoặc bán tự động. Trong số các loại máy bay không người lái, quadcopter là một trong những loại máy bay có khả năng cất hạ cánh thẳng đứng mà không cần đường băng. Bộ phận tạo ra lực đẩy và đồng thời cũng làm nhiệm vụ cân bằng cho máy bay là bốn động cơ. Toàn bộ các chuyển động của máy bay được điều khiển thông qua tăng giảm tốc độ quay của động cơ (không thông qua thay đổi góc tấn các cánh), từ đó dẫn đến thay đổi lực đẩy trên các cánh quạt. Do đó, so với các loại máy bay khác, quadcopter có kết cấu cơ khí đơn giản hơn. Điều này đồng nghĩa với việc chế tạo và bảo trì sẽ đơn giản và ít tốn kém so với các loại máy bay khác. Mặt khác, quadcopter tạo lực đẩy từ bốn cánh quạt nên cho phép tạo ra lực đẩy tương đương với các máy bay khác có kích thước cồng kềnh hơn do sử dụng cánh quạt có kích thước lớn hơn. Điều này là ưu thế cho phép quadcopter hoạt động trong các môi trường chật hẹp và nhiều vật cản như trong thành phố, ở các nơi không có quá nhiều không gian cho việc cất hạ cánh. Bên cạnh các ưu điểm, một trong số các hạn chế của quadcopter là hiệu suất lực đẩy của máy bay thấp do toàn bộ lực nâng của máy bay sinh ra từ lực đẩy của động cơ. Mặt khác, sử dụng cánh quạt có đường kính nhỏ cho tổn thất khí động lớn hơn so với cánh quạt có đường kính lớn. Tuy vây, trong tương lai, việc chế tạo máy bay bằng các vật liệu mới có khối lượng nhẹ, sẽ đóng góp đáng kể vào việc nâng cao hiệu suất tiêu thụ năng lượng của máy bay. Quadcopter đã và đang cho thấy sự hữu ích trong muôn vàn các ứng dụng trong cuộc sống. Với kích thước nhỏ gọn với khả năng mang theo tải trọng lớn, quadcopter có thể làm nhiệm vụ cứu hộ một cách hiệu quả (cứu thương, tìm kiếm cứu hộ sau thảm họa, cứu hỏa…), đặc biệt trong thành phố, nơi mà nguy cơ tắc nghẽn giao thông đường bộ luôn có khả năng làm chậm trễ việc tiếp cận bằng các phương tiện cứu hộ đường bộ. Bên cạnh đó là rất nhiều ứng dụng trong việc vận chuyển hàng hóa, hay thậm chí cả trong quân sự (trinh sát, do thám…).

* 1. Nguyên lý hoạt động của quadcopter

Quadcopter được thiết kế trong đồ án này có dạng dấu nhân “X”. Máy bay có 2 cặp cánh quạt thuận nghịch, với các cánh đặt đối diện nhau là cùng phía. Bên cạnh việc đóng góp vào lực đẩy để lấy độ cao cho máy bay, các cặp cánh còn tạo ra các chuyển động của máy bay xung quanh trọng tâm. Bằng cách tăng và giảm đồng thời tốc độ các động cơ đặt cùng bên, sẽ xuất hiện mô-men lực làm máy bay nghiêng một góc. Từ đó ta có thể điều khiển máy bay di chuyển tịnh tiến theo hướng ngang hoặc dọc so với hướng của máy bay.

Với các cánh quạt quay giống nhau quay ở cùng tốc độ, các mô-men cản do không khí tác dụng lên các cánh quạt sẽ triêt tiêu lẫn nhau. Áp dụng nguyên tắc này, ta có thể ổn định góc hướng của máy bay theo hướng mong muốn, bằng cách thay đổi đồng thời tốc độ (cùng tăng hoặc cùng giảm) của cặp cánh đối diện nhau.



Hình 1.1 Mô tả các chuyển động của Quadcopter

Với trạng thái đứng yên (lơ lửng) của may bay các cánh quạt của máy bay quay giống nhau quay ở cùng tốc độ, các mô-men cản do không khí tác dụng lên các cánh quạt sẽ triêt tiêu lẫn nhau. Lực nâng của 4 động cơ cân bằng với trọng lượng máy bay giúp máy bay giữ nguyên trạng thái. Khi muốn máy bay di chuyển lên hoặc xuống ta chỉ cần thay đổi tốc độ của 4 động cơ.

Di chuyển tiến về phía trước (Move forward) bằng cách giảm tốc độ cặp động cơ phía trước máy bay, tăng tốc độ 2 động cơ còn lại.

Di chuyên lùi về phía sau (Move backward) bằng cách giảm tốc độ cặp động cơ phía sau và tăng tốc độ 2 động cơ phía trước.

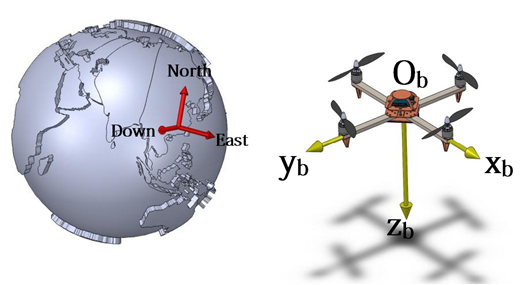
Di chuyển sang bên trái (Bend left) bằng cách giảm tốc độ 2 động cơ bên trái và tốc độ 2 động cơ bên phải.

Di chuyển sang bên phải (Bend right) bằng cách giảm tốc độ 2 động cơ bên phải và tang tốc độ 2 động cơ bên trái.

Xoay tròn tại chỗ (Yaw control) bằng cách thay đổi tốc độ hai động cơ đối xứng nhau, dẫn đến mô men xoắn thay đổi làm máy bay xoay tại chỗ.

* 1. Tìm hiều về mô hình tuyến tính máy bay
     1. Hệ quy chiếu

Trước khi phân tích các yếu tố động lực học đối với máy bay ta cần xác định các hệ quy chiếu. Hệ quy chiếu cho phép người quan sát mô tả các chuyển động của máy bay. Những người quan sát khi đứng trong các hệ quy chiếu khác nhau sẽ có cảm nhận khác nhau về chuyển động của cùng một chiếc máy bay tại cùng một thời điểm. Trong các hệ quy chiếu có gắn các hệ trục tọa độ. Các hệ tọa độ này cho phép tính toán những thay đổi mang tính định lượng về trạng thái động học của máy bay đó. Hệ tọa độ được sử dụng phổ biến là hệ Decartes. Để mô tả các chuyển động của máy bay, cần sử dụng hai hệ quy chiếu sau:

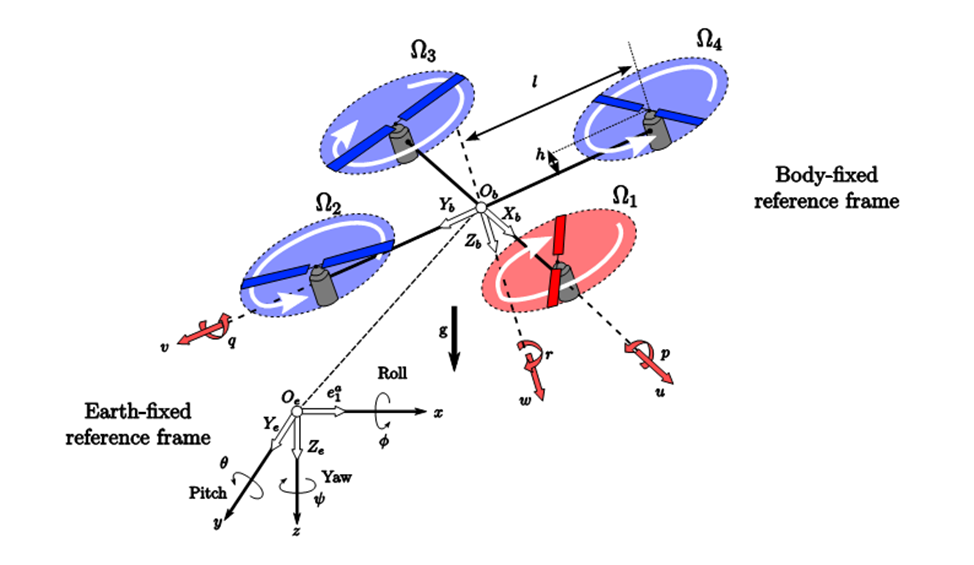
* Hệ quy chiếu quán tính: là hệ quy chiếu không có gia tốc, ở đó các định luật Newton được thỏa mãn. Hệ trục tọa độ gắn với hệ này có gốc tọa độ (kí hiệu: OE) được gắn với một vật mốc cố định so với Trái Đất. Ta kí hiệu hệ là NED do ba trục tọa độ gắn với hệ xE, yE, zE được quy ước lần lượt chỉ theo các hướng bắc (North), đông (East) và hướng về tâm Trái Đất.
* Hệ quy chiếu gắn với vật thể bay (Hệ vật): Hệ quy chiếu được gắn cố định và chuyển động cùng với vật thể bay. Gốc OB của hệ tọa độ gắn với hệ vật được lấy trùng với trọng tâm của máy bay (CG). Các trục tọa độ được kí hiệu lần lượt là xB, yB, zB [hình vẽ]

Hình 1.2 Hệ tọa độ vật (hình trái) và hệ tọa độ quán tính (hình phải)

* + 1. Các góc Euler

Các góc được quay theo quy tắc tay phải xung quanh ba trục tọa độ của một hệ trục thuận được định nghĩa là các góc Euler. Các góc quay không phải là đại lượng vec-tơ, do đó chúng không thỏa mãn tính chất giao hoán. Bởi vậy, thứ tự tiến hành các phép quay là rất quan trọng. Người ta quy ước thực hiện các phép quay theo trình tự sau để đưa hệ trục gắn với máy bay trùng với hệ Trái Đất:

* + OBxByBzB quay một góc roll ϕ (-π/2<ϕ<π/2) xung quanh trục OBxB;
  + Quay một góc pitch θ (-π/2<θ<π/2) xung quanh trục OByB;
  + Quay một góc yaw ψ (-π≤ψ<π) xung quanh trục OBzB và trùng với OExEyEzE.



Hình 1.3 Quy ước các phương chiều

* + 1. Ma trận chuyển đổi hệ trục tọa độ

Theo lý thuyết điều khiển, các ứng xử động học của một vật được mô tả thông qua các biến trạng thái. Các biến này bao gồm:

* + - * + Các góc Euler giữa hệ vật và hệ quán tính: ϕ, θ, ψ.
        + Các thành phần vận tốc góc của máy bay chiếu trên các trục tọa độ gắn với hệ vật: Obxb, Obyb, Obzb lần lượt là p, q, r.
        + Các thành phần cho biết vị trí của máy bay trong hệ tọa độ OE, xE, yE, zE: x, y, z.
        + Các thành phần vận tốc dài xét trong hệ tọa độ vật được chiếu lên ba trục: Obxb, Obyb, Obzb lần lượt là u, v, w.

Các biến trạng thái đo được trực tiếp bằng các cảm biến điện tử gắn trên máy bay có giá trị đo được trong hệ tọa độ vật. Vì vậy, ta phải sử dụng các ma trận chuyển hệ trục tọa độ, để chuyển đổi các giá trị đo bởi cảm biến về giá trị xác định trong hệ quán tính. Ma trận này thực chất được xây dựng từ trình tự quay ba góc Euler như đã trình bày trong phần trước.

Đối với các đại lượng dài, ta có mối liên hệ giữa vận tốc dài xét trong 2 hệ trục như sau:

Trong đó:

* + 1. Hệ phương trình chuyển động
* Các giả thiết

Để khảo sát chuyển động của quadcopter, ta cần xây dựng hệ phương trình động học mô tả chuyển động của máy bay. Ta đơn giản hóa việc này bằng cách đưa ra một số giả thiết sau đây:

Bỏ qua các yếu tố biến dạng đàn hồi ở kết cấu khung quadcopter, coi máy bay là cứng tuyệt đối, máy bay là đối xứng, bỏ qua hiệu ứng mặt đất.

Các yếu tố này sẽ có tác động tương đối nhỏ hoặc chỉ trong khoảng thời gian nhỏ so với tổng thời gian máy bay hoạt động. Do đó ta có thể bỏ qua các ảnh hưởng này để đơn giản hóa các phương trình chuyển động. Phương pháp này đã được chứng minh về tính hiệu quả trong thực tiễn thông qua các nghiên cứu trước đó. Việc coi kết cấu khung là cứng tuyệt đối đã bỏ qua ảnh hưởng động học của các mối ghép cơ khí, độ đàn hồi của kết cấu khung và cánh quạt, cho phép đưa bài toán về phân tích một vật thể cứng tuyệt đối có 6 bậc tự do. Việc coi máy bay đối xứng qua các mặt phẳng Obxbyb, Obybzb, Obzbxb cho phép bỏ qua các thành phần mô-men tích quán tính.

* Hệ phương trình động học
* Các phương trình cân bằng lực

Kí hiệu tổng các ngoại lực tác động lên máy bay là: với các thành phần Fx, Fy, F­z xác định bởi hệ tọa độ vật.

Theo định luật 2 Newton ta có:

Với = (u,v,w) là vec-tơ vận tốc tuyệt đối của máy bay xét trong hệ tọa độ vật.

Theo định luật Transport [12,12], ta có:

Trong đó: = (p, q , r )

Do máy bay được thiết kế chuyển động với tốc độ chậm, có thể bỏ qua các ảnh hưởng của các lực khí động. Các ngoại lực không thể bỏ qua gồm lực đẩy của bốn động cơ T và trọng lượng của máy bay W.

Do các thành phần Fx, Fy, F¬z xác định bởi hệ tọa độ vật, cần đưa các lực về xét trong hệ tọa độ vật thông qua ma trận chuyển hệ trục tọa độ D.

Sau khi tính toán với ma trận, ta viết lại dưới dạng hệ phương trình như sau:

Trong đó:

(i=1,4): là lực đẩy của lần lượt 4 động cơ.

là tốc độ góc của 4 động cơ.

(Theo lý thuyết phần tử cánh, lực đẩy tạo ra do cánh quạt tỉ lệ với tốc độ góc của cánh theo một hằng số nếu coi biên dạng cánh và điều kiện môi trường xung quanh là không đổi.)

* Các phương trình cân bằng mô-men

Gọi tổng các mô-men ngoại lực tác động lên máy bay là: với các thành phần Mx, My, M­z xác định bởi hệ tọa độ vật.

Theo định luật 2 Newton ta có:

Trong đó:

= (p, q, r) là vec-tơ vận tốc góc tuyệt đối của máy bay xét trong hệ tọa độ vật.

I= , tensor mô-men quán tính của máy bay.

Theo định luật Transport [12,12], ta có:

Trong đó: = (p, q , r )

Coi máy bay đối xứng qua các mặt phẳng Obxbzb, Obybzb nên ta có , do đó phương trình được rút gọn thành:

Mô-men ngoại lực gây ra chủ yếu do lực đẩy tạo ra bởi cánh quạt và lực cản của không khí. Một cách đơn giản, ta có thể coi các thành phần của vec-tơ mô-men ngoại lực như sau:

Trong đó:

l: chiều dài cánh tay đòn, tính bằng khoảng cách từ điểm đặt lực trên cánh tay đòn đến vị trí trọng tâm máy bay.

b: hệ số lực đẩy

d: hệ số mô-men cản. (Theo lý thuyết phần tử cánh, mô-men cản do lực khí động gây ra trên cánh quạt tỉ lệ với tốc độ góc của cánh theo một hằng số nếu coi biên dạng cánh và điều kiện môi trường xung quanh là không đổi)

Kí hiệu:

Cuối cùng, ta thu được hệ phương trình cân bằng mô-men:

* Bổ sung hiệu ứng gyroscope của cánh quạt

Việc xây dựng hệ phương trình cân bằng mô-men cho quadcopter dựa trên giả thiết toàn bộ máy bay là vật rắn tuyệt đối và bỏ qua các chuyển động tương đối giữa các thành phần trong hệ thống. Trong thực tế, các cánh quạt khi quay tạo ra các mô-men gyroscope trên các trục còn lại của máy bay. Vì vậy, ta xét thêm các thành phần này vào tổng mô-men ngoại lực của quadcopter. Hệ phương trình cân bằng mô-men cuối cùng thu được là [5,11]:

Trong đó:

tốc độ quay của cánh quạt.

mô-men quán tính của cánh quạt so với trục quay.

Mô hình tuyến tính của quacopter:

Để xây dựng phương trình trạng thái cho 12 biến trạng thái của Quadcopter, bên cạnh 3 phương trình cân bằng lực và 3 phương trình cân bằng mô-men, ta cần bổ sung thêm 6 phương trình, được xây dựng từ các ma trận chuyển hệ trục tọa độ:

Với mục đích giới hạn các chuyển động của máy bay chỉ trong phạm vi các di chuyển đơn giản và xác lập, bằng cách xác định các chế độ hoạt động cơ bản của Quadcopter, ta có thể tuyến tính hóa hệ phương trình động học tổng quát về hệ tuyến tính quanh các điểm làm việc chính. Việc đưa hệ phi tuyến thay đổi theo thời gian về hệ tuyến tính không đổi theo thời gian cho phép ta tiếp cận hệ thống bằng nhiều công cụ khảo sát ổn định trong miền tần số, giúp người thiết kế hiểu sâu hơn bản chất của hệ thống. Đồng thời cách tiến hành này cho phép tách bài toán điều khiển hệ MIMO thành các bài toán điều khiển các kênh SISO riêng rẽ. Trên mỗi hệ SISO, ta có thể quan sát sự tác động của các thông số động học và tín hiệu điều khiển đến từng trạng thái một cách độc lập. Tóm lại, việc tuyến tính hóa hệ phương trình mô tả động học của Quadcopter giúp giảm độ phức tạp và khối lượng tính toán trong việc thiết kế bộ điều khiển cho hệ thống, cho phép người thiết kế có cái nhìn sâu sắc hơn vào bản chất của hệ thống. Tuy nhiên, phương pháp chỉ ứng dụng được hiệu quả trong các môi trường không có nhiễu khí động lớn...

Việc tiến hành tuyến tính hóa hệ phương trình phi tuyến dựa trên lý thuyết xấp xỉ một hàm số bằng khai triển Taylor và lí thuyết nhiễu nhỏ. Dựa trên thực tế rằng các yếu tố khí động tác động chính tới máy bay là hàm tuyến tính của các nhiễu nhỏ, người ta giả thiết máy bay chịu tác động bởi những thay đổi nhỏ quanh trạng thái dừng ổn định. Phương pháp cho độ chính xác tốt trong các ứng dụng cho máy bay chuyển động đều.

Các trạng thái và tín hiệu điều khiển ở điểm cân bằng là các hằng số, được kí hiệu:

Mối liên hệ giữa các biến trạng thái, các tín hiệu điều khiển với các biến nhiễu nhỏ:

|  |  |
| --- | --- |
|  |  |

Do giả sử rằng các đại lượng nhiễu nhỏ và đạo hàm của chúng là bé, ta coi tích của hai đại lượng hoặc bình phương của một đại lượng xấp xỉ bằng 0. Ngoài ra, đối với các thành phần lượng giác, ta coi Giả thiết áp dụng tương tự với các góc và .

Xét khai triển Taylor của một hàm gồm biến trạng thái v và biến điều khiển U f(v, U) xung quanh trạng thái ổn định (vo, Uo) (tại đây, việc xét hàm số cho hai biến vẫn đảm bảo tính tổng quát của khai triển đối với hàm có số biến bất kì):

Áp dụng khai triển Taylor cho các hàm trên, bỏ qua các đại lượng có bậc ≥ 2, ta có:

Từ giả thiết của phương pháp nhiễu nhỏ, ta có:

và

suy ra

Mặt khác, do

Nên

Vậy, dựa trên lý thuyết nhiễu nhỏ, ta thu được hàm Taylor khai triển cho hàm f (v, U) như sau:

Áp dụng lý thuyết nhiễu nhỏ và khai triển Taylor, ta có thể tuyến tính hóa 12 phương trạng thái của quadcopter về dạng tuyến tính:

* 1. Mô hình động cơ và cánh quạt

Thiết bị tạo lực đẩy được sử dụng cho mô hình là động cơ một chiều không chổi than (Brushless DC motor - BLDC). Mặc dù việc chế tạo phức tạp và có giá thành cao hơn so với động cơ một chiều có chổi than, động cơ BLDC vẫn thường được lựa chọn để chế tạo mô hình máy bay bởi nhiều ưu điểm vượt trội như sau:

* + Cho mô-men xoắn lớn hơn so với động cơ DC cùng khối lượng.
  + Cho hiệu suất mô-men xoắn so với công suất tiêu thụ lớn hơn.
  + Độ tin cậy cao, tuổi thọ cao và ít phải bảo trì do không phải hoạt động thông qua cơ cấu tiếp xúc trực tiếp chổi than.
  + Dễ tích hợp với hệ thống điều khiển số.

Để đơn giản cho việc điều khiển động cơ, ta sử dụng bộ điều tốc điện từ (Electronic Speed Controller - ESC), cho phép điều khiển động cơ bằng vi điều khiển thông qua điều chỉnh độ rộng xung (Pulse width modulation/PWM). Thông thường, các ESC nhận các xung điều khiển có tần số 50Hz với các độ rộng xung từ 1ms đến 2ms để điều khiển động cơ lần lượt ứng với mức tốc độ từ thấp nhất tới cao nhất.

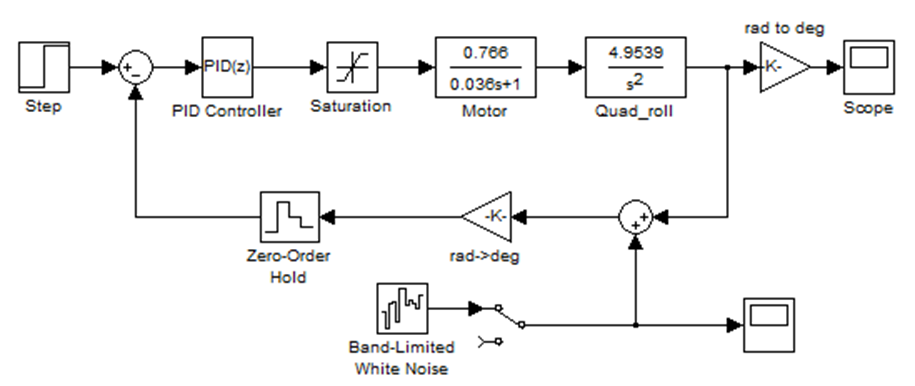
Tổ hợp ESC, động cơ và cánh quạt cho phép điều khiển lực đẩy tác dụng lên máy bay theo tín hiệu xung cấp từ bộ điều khiển thông qua việc biến đổi tốc độ của động cơ.

Để mô phỏng các ứng xử động học của quadrotor, đầu tiên cần xây dựng mô hình toán học của cụm ESC, động cơ và cánh quạt. Hàm truyền của động cơ một chiều bao gồm một khâu tỉ lệ và một khâu quán tính bậc hai với hai điểm cực lần lượt liên quan tới yếu tố cơ khí và điện của động cơ. Do yếu tố điện có ảnh hưởng không đáng kể (nhỏ hơn so với ảnh hưởng của mô-men quán tính của động cơ), ta có thể loại bỏ một điểm cực và coi gần đúng hàm truyền động cơ là khâu quán tính bậc nhất. Như vậy, hàm truyền cần xác định có dạng:

Việc nhận dạng mô hình động cơ được tiến hành thông qua phương pháp thực nghiệm. Các thí nghiệm được tiến hành để xác định các hằng số K và τ của hàm truyền G. Trong đó K cho biết tỷ lệ giữa lực đẩy sinh ra do cánh quạt và độ rộng xung tín hiệu cấp cho ESC, hằng số thời gian τ cho biết tốc độ đáp ứng của cụm ESC và động cơ, là khoảng thời gian để đạt được tín hiệu đầu ra mong muốn từ khi có tín hiệu điều khiển.

* 1. Mô phỏng tìm hệ số PID

Sau khi đã tìm được các hàm truyền đạt, ta có thể thiết lập mô phỏng để tìm hệ số PID theo sơ đồ sau:



Hình 1.4 Sơ đồ mô phỏng để tìm hệ số PID theo sơ đồ sau:

1. Tìm hiểu bộ điều khiển PID
   1. Lý thuyết điều khiển PID
      1. Khâu tỉ lệ

Khâu tỉ lệ (độ lợi) làm thay đổi giá trị đầu ra, tỷ lệ với giá trị sai số hiện tại. Đáp ứng tỉ lệ có thể được điều chỉnh bằng cách nhân sai số đó với một hằng số Kp, được gọi là độ lợi tỉ lệ.

Khâu tỉ lệ được cho bởi

**Pout = Kp e(t)**

Trong đó

**Pout**: Là thừa số tỉ lệ đầu ra

**Kp**: Là độ lợi tỉ lệ, thông số điều chỉnh

**e**: Là sai số

**t**: Là thời gian hay thời gian tức thời (hiện tại)

Độ lợi của khâu tỉ lệ càng lơn là do thay đổi lớn ở đầy ra mà sai số thay đổi nhỏ. Nếu độ lợi của khâu tỉ lệ quá cao, hệ thông sẽ không ổn định. Ngược lại, độ lợi nhỏ là do đáp ứng đầu ra nhỏ trong khi sai số đầu vào lớn, làm cho bộ điều khiển kém nhạy, hoặc đáp ứng chậm. Nếu độ lợi quá thấp, tác động điều khiển có thể sẽ quá bé khi đáp ứng với các nhiễu của hệ thống.

* + 1. Khâu tích phân

Phân phối của khâu tích phân tỉ lệ thuận với cả biên độ sai số lẫn quãng thời gian xảy ra sai số. Tổng sai số tức thời theo thời gian (tích phân sai số) cho t tích lũy bù dẫ được hiệu chỉnh trước đó. Tích lũy sai số sau đó được nhân với độ lợi tích phân và cộng với tín hiệu đầu ra của bộ điều khiển. Biên độ phân phối của khâu tích phân trên tất cả tác động điều chỉnh được xác định bởi độ lợi tích phân, Ki

Thừa số tích phân được cho bởi:

**Iout = Ki**

Trong đó

**Iout**: Thừa số tích phân của đầu ra

**Ki**: Độ lợi tích phân, 1 thông số điều chỉnh

**e**: sai số

**t**: thời gian hoặc thời gian tức thời (hiện tại)

Khâu tích phân (khi cộng thêm khâu tỉ lệ) sẽ tăng tốc chuyển động của quá trình tới điểm đặt và khử số dư sai số ổn định với một tỉ lệ phụ thuộc vào bộ điều khiển. Tuy nhiên, vì khâu tích phân là đáp ứng của sai số tích lũy trong quá khứ, nó có thể khiến giá trị hiện tại vọt lố qua giá trị đặt (ngang qua điểm đặt và tạo ra một độ lệch với các hướng khác).

* + 1. Khâu vi phân

Tốc độ thay đổi của sai số quá trình được tính toán bằng cách xác định độ dốc của sai số theo thời gian (tức là đạo hàm bậc một theo thời gian) và nhân tốc độ này với độ lợi tỉ lệ Kd. Biên độ của phân phối khâu vi phân (đôi khi được gọi là tốc độ) trên tất cả các hành vi điều khiển được giới hạn bởi độ lợi vi phân, Kd.

Thừa số vi phân được cho bởi:

**Dout = Kd**

Trong đó

**Dout**:Thừa số vi phân của đầu ra

**Kd**: Độ lợi vi phân, một thông số điều chỉnh

**e**: Sai số

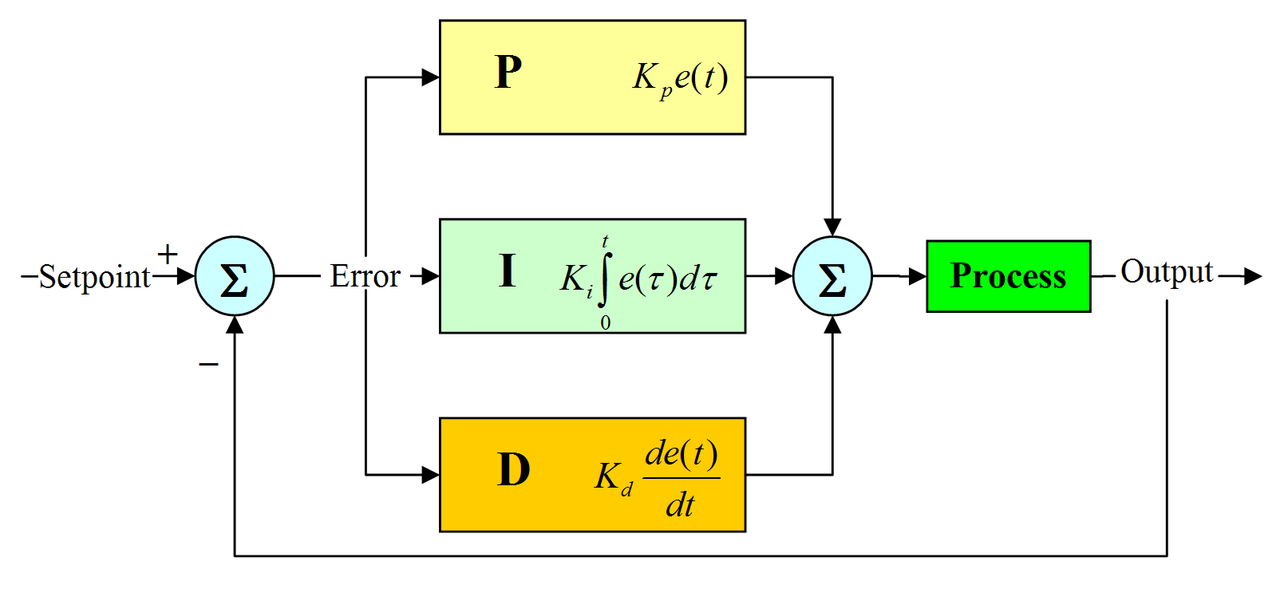
**t**: Thời gian hoặc thời gian tức thời (hiện tại)

Khâu vi phân làm chậm tốc độ thay đổi của đầu ra bộ điều khiển và đặc tính này là đáng chú ý nhất để đạt tới điểm đặt của bộ điều khiển. Từ đó, điều khiển vi phân được sử dụng để làm giảm biên độ vọt lố được tạo ra bởi thành phần tích phân và tăng cường độ ổn định của bộ điều khiển hỗn hợp. Tuy nhiên, phép vi phân của một tín hiệu sẽ khuếch đại nhiễu và do đó khâu này sẽ nhạy hơn đối với nhiễu trong sai số và có thể khiến quá trình trở nên không ổn định nếu nhiễu và độ lợi vi phân đủ lớn. Do đó một xấp xỉ của bộ vi sai với bang thông giới hạn thường được sử dụng hơn. Chẳng hạn như mạch bù sớm pha.

* + 1. Tổng hợp 3 khâu

Khâu tỉ lệ, tích phân, vi phân được cộng lại với nhâu để tính toán đầu ra của bộ điều khiển PID. Định nghĩa rằng u(t) là đầu ra của bộ điều khiển, biểu thức cuối cùng của giải thuật PID là:

**u(t) = Kpe(t) +** **Ki + Kd**



* 1. Điều chỉnh vòng lặp

Điều chỉnh một vòng điều khiển là điều chỉnh các thông số điều khiển của nó tới giá trị đáp ứng điều khiển tối ưu. Độ ổn định là một yêu cầu căn bản, nhưng ngoài ra, các hệ thống khác nhau, có những hành vi khác nhau, những ứng dụng khác nhau có những yêu cầu khác nhau, và vài yêu cầu lại mâu thuẫn với nhau. Hơn nữa, vài quá trình có một mức độ phi tuyến nà`o đấy khiến các thông số làm việc tốt ở điều kiện đầy tải sẽ không làm việc khi quá trình khởi động từ không tải. Các bộ điều khiển PID thường cung cấp các điều khiển chấp nhận được thậm chí không cần điều chỉnh, nhưng kết quả nói chung có thể được cải thiện bằng cách điều chỉnh kỹ lưỡng, và kết quả có thể không chấp nhận được nếu điều chỉnh kém.

Điều chỉnh PID là một bài toán khó, ngay cả khi chỉ có 3 thông số và về nguyên tắc là dễ miêu tả, bởi vì nó phải thỏa mãn các tiếu chuẩn phức tạp nằm trong những hạn chế của điều khiển PID.

* + 1. Độ ổn định

Nếu các thông số của bộ điều khiển PID được chọn sai, đầu vào quá trình điều khiển có thể mất ổn định, vì các khác biệt đầu ra của nó, có hoặc không có dao động, và được giới hạn chỉ bởi sự bão hòa hoặc đứt gãy cơ khí. Sự không ổn định được gay ra bởi sự dư thừa độ lợi, nhất là khi xuất hiện độ trễ lớn.

Nói chung, độ ổn định của đáp ứng phải thỏa mã và quá trình phải không được tạo dao động vì bất kỳ sự kết hợp nào giữa các điều kiện quá trình và điểm đặt, mặc dù đôi khi ổn định biên có thể được chấp nhận hoặc yêu cầu.

* + 1. Tối ưu hóa hành vi

Tối ưu hóa hành vi trong thay đổi quá trình hoặc thay đổi điểm đặt khác nhau tùy thuộc và ứng dụng.

Hai yêu cầu cơ bản là ổn định và tự hiệu chỉnh lệnh, hai yêu cầu này tùy thuộc vào việc các biến điều khiển theo dõi giá trị mong muốn có tốt hay không. Các yêu chuẩn đặc biệt về tự hiệu chỉnh lệnh bao gồm thời gian khởi động và thời gian xác lập.

* 1. Các phương pháp điều chỉnh vòng lặp

Có nhiều phương pháp khác nhau để điều chỉnh vòng lặp PID. Nhưng phương pháp hữu hiệu nhất thường bao gồm những triển khai của vài dạng mô hình xử lý, sau đó chọn thông số PID dựa trên các thông số của mô hình học. Các Phương pháp điều chỉnh thủ công tương đối không hiệu quả lắm, đặc biệt nếu vòng lặp có thời gian đáp ứng được tính bằng phút hoặc lâu hơn.

* + 1. Phương pháp thủ công

Nếu hệ thống phải duy trì trạng thái online, một phương pháp điều chỉnh là thiết đặt giá trị đầu tiên của Ki và Kd bằng không. Tăng dần Kp cho đến khi đầu ra của vòng điều khiển dao động, sau đó Kp có thể được đặt tới xấp xỉ một nửa giá trị đó để đạt được đáp ứng. Sau đó tang Ki đến giá trị phù hợp sao cho đủ thời gian xử lý. Tuy nhiên, Ki quá lớn sẽ gây mất ổn định. Cuối cùng, tang Kd cho đến khi vòng điều khiển nhanh có thể chấp nhận được nhanh chóng lấy lại được giá trị đặt sau bị nhiễu. Tuy nhiên, Kd quá lớn, sẽ gây đáp ứng dư và vọt lố. Một điều chỉnh cấp tốc của vòng điều khiển PID thường hơi vọt lố một ít khi tiến tới điểm đặt nhanh chóng. Tuy nhiên, vài hệ thống không chấp nhận xảy ra vọt lố, trong trường hợp đó, ta cần một hệ thống vòng kín giảm lố, thiết đặt một giá trị Kp nhỏ hơn một nửa giá trị Kp gây ra dao động.

**Tác động của việc tăng một thông số độc lập**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thông số** | **Thời gian khởi động** | **Quá độ** | **Thời gian xác lập** | **Sai số ổn định** | **Độ ổn định** |
| **Kp** | Giảm | Tăng | Thay đổi nhỏ | Giảm | Giảm cấp |
| **Ki** | Giảm | Tăng | Tăng | Giảm đáng kể | Giảm cấp |
| **Kd** | Giảm ít | Giảm ít | Giảm ít | Không tác động | Cải thiện nếu Kd nhỏ |

* + 1. Phương pháp Ziegler-Nichols

Là phương pháp được đưa ra bởi John G. Ziegler và Nathaniel B. Nichols vào những năm 1940. Giống phương pháp trên, độ lợi Ki và Kd lúc đầu được gán bằng không. Độ lợi P được tang cho đến khi ní tiến tới độ lợi tới hạn Ku ở đầu ra của vòng điều khiển bắt đầu dao động. Ku và thời gian dao động Pu được dùng để gán độ lợi như sau:

**Phương pháp Ziegler-Nichols**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dạng điều khiển** | **Kp** | **Ki** | **Kd** |
| **P** | **0.5 Ku** | **-** | **-** |
| **PI** | **0.45Ku** | **1.2Kp/Pu** | **-** |
| **PID** | **0.6Ku** | **2Kp/Pu** | **KpPu/8** |

* + 1. Sử dụng phần mềm điều chỉnh PID

Hầu hết các ứng dụng công nghiệp hiện đại không còn điều chỉnh vòng điều khiển sử dụng các phương pháp tính toán thủ công như trên nữa. Thay vào đó, phần mềm điều chỉnh PID và tối ưu hóa vòng lặp được dùng để đảm bảo kết quả chắc chắn. Những gói phần mềm này sẽ tập hợp dữ liệu, phát triển các mô hình xử lý. Và đề xuất phương pháp điều chỉnh tối ưu. Gói phần mềm thậm chí còn có thể phát triển việc điềuc hỉnh bằng cách thu thập dữ liệu từ các thay đổi tham khảo.

Điều chỉnh PID bằng toán học tạo ra một xung trong hệ thống, và sau đó sử dụng đáp ứng tần số của hệ thống điều khiển để thiết kế các giá trị của vòng điều khiển PID. Trong những vòng lặp có thời gian đáp ứng kéo dài nhiều phút, nên chọn điều chỉnh bằng toán học, bởi vì việc thử sai thực tế có thể kéo dài nhiều ngày để tìm điểm ổn định cho vòng lặp. Giá trị tối ưu khó tìm hơn. Vài bộ điều khiển số còn có chức năng tự điều chỉnh, trong đó có những thay đổi rất nhỏ của điểm đặt cũng được gửi tới quá trình, cho phép bộ điều chỉnh tự tính toán giá trị điều chỉnh tối ưu.

* 1. Các cải tiến đối với thuật toán PID
     1. Tích phân khởi động

Một vấn đề phổ biến của bộ PID lý tưởng là tích phân khởi động, nơi xảy ra thay đổi điểm đặt lớn (tức là thay đổi dương) và khâu tích phân tích lũy một sai số đáng kể lúc tăng (khởi động), vì vậy làm vọt lố và duy trì liên tục việc tăng sai số đáng kể lúc tăng (khởi động), vì vậy làm vọt lố và duy trì liên tục việc tăng sai số tích lũy bị gián đoạn. Có thể khắng phục điều này bằng cách:

* Thiết đặt giá trị tích phân ban đầu cho bộ điều khiển với giá trị mong muốn
* Tăng điểm đặt với độ dốc thích hợp
* Không cho phép chức năng tích phân cho đến khi PV đi vào vùng điều khiển được
* Giới hạn khoảng thời gian vượt quá sai số tích phân được tính toán
* Ngăn không cho khâu tích phân tích lũy trên hoặc dưới biên xác định trước
  + 1. Đóng băng chức năng tích phân trong trường hợp bị nhiễu

Nếu một vòng PID được sử dụng để điều khiển nhiệt độ của một lò nung điện trở, hệ thống đã ổn định và sau đó cửa lò mở và nhiệt độ thấp luồn vào lò nung, làm cho nhiệt độ rơi xuống dưới điểm đặt. Chức năng tích phân của bộ điều khiển có xu hướng bù sai số này bằng cách đưa ra một sai số khác theo hướng dương. Điều này có thể tránh được bằng cách đóng bang chức năng tích phân sau khi mở cửa lò để đủ thời gian cho vòng điều khiển nung lại lò nung.

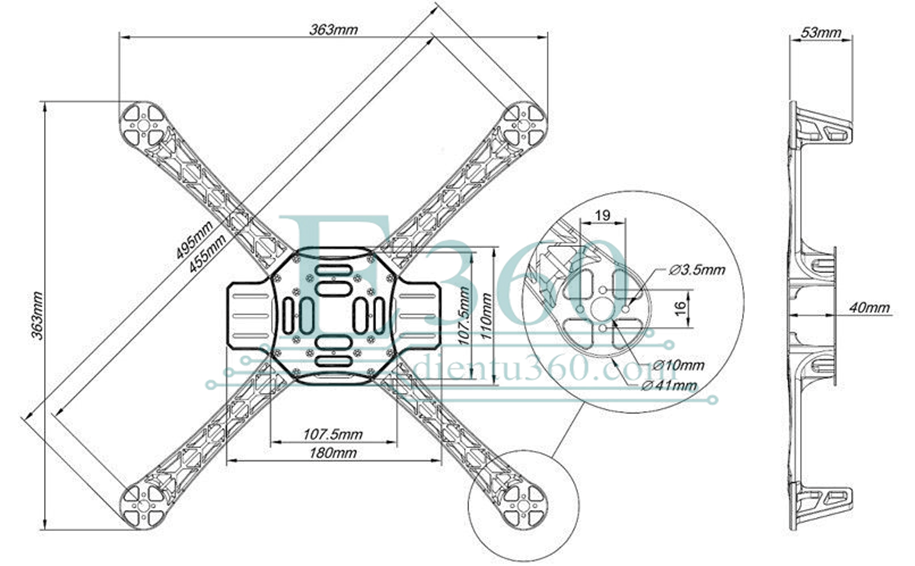
* + 1. Thay chức năng tích phân bằng một phần dựa trên mô hình

Thường thì thời gian đáp ứng của hệ thống được biết trước. Do đó rất tiện lợi để mô phỏng thời gian đáp ứng này với một mô hình và tính toán vài thông số chưa biết từ đáp ứng của hệ thống. Nếu giả sử hệ thống là một lò nung điện thì đáp của hiệu số giữa nhiệt độ lò và nhiệt độ môi trường để thay đổi lượng điện năng sẽ tương tự với một bộ lọc thông thấp RC đơn giản được nhân bởi một hệ số tỉ lệ chưa biết. Điện năng thực tế cung cấp cho lò nung được làm trễ bởi một bộ lọc thông thấp để mô phỏng đáp ứng của nhiệt độ lò nung và sau đó lấy nhiệt độ thực trừ đi nhiệt độ môi trường rồi chia cho thành phần điện năng thông thấp này. Sau đó, kết quả trên sẽ được ổn định bởi một bộ lọc thông thấp khác dẫn đến việc tìm ra hệ số tỉ lệ. Với việc tính toán này, ta có thể tính toán lượng điện yêu cầu bời cách chi điểm đặt nhiệt đọ trừ đi nhiệt độ môi trường bởi hệ số này. Kết quả này sau đó có thể được dùng để thay thế chức năng tích phân. Điều này còn có thể đạt được một sai số điều khiển bằng '0' trong chế độ xác lập, bằng cách tránh tích phân khởi động và có thể cải tiến đáng kể tác động điều khiển so với một bộ điều khiển PID tối ưu. Bộ điều khiển dạng này làm việc rất chính xác trong trường hợp vòng hở, tạo ra tích phân khởi động với một hàm tích phân. Đây là một ưu điểm, nếu thí dụ, nhiệt cấp cho lò nung bị giảm một lúc vì hư hỏng trong bộ nung, hoặc nếu bộ điều khiển được dùng như một hệ thống tham vấn cho nhân viên điều khiển, và người này không thể đưa nó về làm việc trong vòng kín. Cũng sẽ rất hiệu quả nếu bộ điều khiển nằm trong một nhánh của một hệ thống điều khiển phức tạp mà có thể tạm thời không làm việc.

1. Các bước thiết kế quadcopter
   1. Thiết kế cơ khí và mạch điện tử
      1. Thiết kế cơ khí

Khung máy bay được thiết kế với hình dạng chữ “X”, cân đối với nhau để trọng tâm của máy bay ở chính giữa khung. Bên cạnh đó, một thông số thiết kế quan trọng khác là khoảng cách từ trục động cơ tới trọng tâm máy bay cũng cần được thiết kế chính xác. Giúp cảm biến có thể gửi dữ liệu trạng thái máy bay với sai số nhỏ nhất.

Do vậy chúng em lựa chọn khung máy bay đã được thiết kế sẵn. Với nhiểu ưu điểm như: Chất liệu bền, nhẹ, sai số nhỏ, biến dạng lực nhỏ.



Hình 2.1 Thiết kế khung cơ khí

Bỏ qua các sai số do chế tạo, ta coi lực đẩy của động cơ vuông góc với mặt phẳng xy của máy bay và có phương trùng với trục động cơ. Khi đó mô-men do lực đẩy động cơ tác động lên máy bay có thể tính toán được và bằng tích của trị số lực đẩy nhân với khoảng cách từ trục động cơ tới trọng tâm máy bay.

* + 1. Thiết kế điện tử

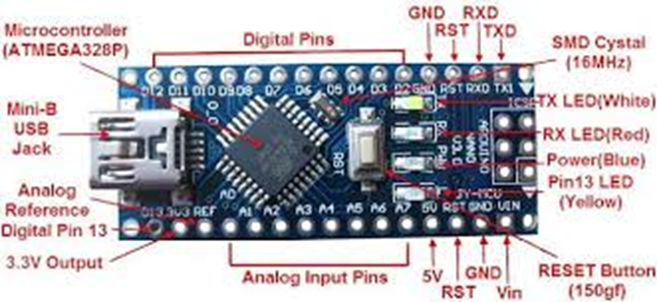
Các thiết bị điện tử được lựa chọn sử dụng trên mô hình:

* + - * + **Arduino nano**

Việc cân bằng quadcopter trong không gian rất khó vì nó trôi nổi trong không gian không có điểm tựa vì vậy luôn có xu hướng mất ổn định, vì vậy máy bay cần một hệ thống tự động cân bằng để ổn định.

Vi điều Atmega 328P là vi điều khiển 8 bit có 4x16 ngõ vào, và có 4 bộ phát xung PWM, cho phép xử lý các tác vụ nhanh chóng. Đồng thời, nó cho phép giao tiếp qua các giao thức phổ biến như UART, SPI, I2C, Serial… cho phép kết nối với các cảm biến và giao tiếp với máy tính và các vi điều khiển khác một cách thuận tiện.

Arduino nano là sự tiện dụng, đơn giản có thể lập trình trực tiếp bằng máy tính, tích hợp sẵn vi điều khiển Atmega 328P và đặc biệt hơn cả là kích thước của nó. Kích thước arduino cực kì nhỏ (1.85cm x4.3 cm), giá thành rẻ hơn arduino uno nhưng dùng được tất cả các thư viện của mạch này.



Hình 2.2 Arduino nano

|  |  |
| --- | --- |
| Vi điều khiển | Atmega 328 (họ 8 bit) |
| Điện áp hoạt động | 5V-DC |
| Tần số hoạt động | 16 MHz |
| Dòng điện tiêu thụ | 30mA |
| Điện áp khuyên áp khuyên dùng | 7-12 V DC |
| Điện áp giới hạn | 6-20 V DC |
| Số chân digital | 14 chân (6 chân PWN ) |
| Số chân analog | 8 chân (độ phân giải 10bit) |
| Dòng điện tối đa trên mỗi chân I/O | 40 mA |
| Dòng ra tối đa 5V | 500mA |
| Dòng ra tối đa (3,3 v) | 50mA |
| Bộ nhớ flash | 32 KB (ATmega328) với 2KB dùng bởi bootloade |
| SRAM | 2KB (Atmega328) |
| EEPROM | 1KB (Atmega328) |
| Kích thước | 1.85x4.3 cm |

Bảng 2.1 Thông số của kỹ thuật của Arduino nano

* + **Cảm biến IMU MPU-6050**

Trong hệ thống điều khiển, khối đo lường quán tính (IMU) bao gồm các cảm biến: gyro, gia tốc kế, từ kế… đóng vai trò đặc biệt quan trọng. Chúng cho phép xác định trạng thái (vị trí, độ cao, vận tốc, gia tốc, góc nghiêng…) của máy bay theo thời gian và không gian. Độ chính xác của các cảm biến đóng vai trò quan trọng bởi đây là nguồn tín hiệu đầu vào cho bộ vi xử lý. Công nghệ MEMS (Hệ thống vi cơ điện tử) cho ra đời các cảm biến quán tính có kích thước nhỏ, với giá thành rẻ, dễ dàng tích hợp với các UAV nói chung và quadcopter nói riêng.

Cảm biến MPU-6050 có 6 bậc tự do gồm một gyroscope 3 trục và một gia tốc kế 3 trục), cho phép xác định 6 thông số quán tính độc lập bao gồm: các gia tốc dài theo các phương và các thành phần vận tốc góc xoay quanh các trục của máy bay. Bên trong cảm biến có một vi xử lý, thực hiện việc tính toán và chuyển đổi ngay từ giá trị đo tương tự sang giá trị số, làm giảm khối lượng công việc cho vi điều khiển. Các giá trị tính toán được truyền về vi điều khuyển thông qua giao tiếp I2C.

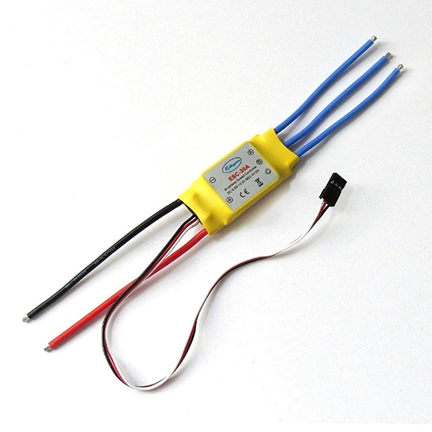
Giao tiếp I2C có tốc độ truyền gửi dữ liệu thấp hơn so với giao tiếp SPI nhưng yêu cầu ít dây hơn để kết nối (một dây SCK giữ xung nhịp để đồng bộ tốc độ giữa các vi xử lý và một dây SDA để truyền dữ liệu) và số dây này không tăng lên khi số lượng cảm biến kết nối với vi điều khiển tăng.

|  |  |
| --- | --- |
| Giao tiếp | I2C |
| Độ phân giải | 16 bits |
| Giải đo gyroscope | ±250o/giây ±500o/giây ±1000 o/giây  ±2000 o/giây |
| Tốc độ lấy mẫu (gyroscope) | 8000 Hz |
| Giải đo gia tốc kế | ±2g, ±4g, ±8g and ±16g |
| Tốc độ lấy mẫu (gia tốc kế) | 1000 Hz |
| Hệ số độ nhạy | 16 384 LSB/g |
| Thay đổi độ nhạy với nhiệt độ (trong khoảng -40 oC tới 85 oC) | ±0,02 %/oC |
| Ảnh hưởng giữa các trục | ±2 % |
| Sai lệch hiệu chỉnh ban đầu | Trục X và Y: ±50 mg  Trục Z :±80 mg |
| Mật độ nhiễu | 400 μg/√Hz |

Bảng 2.2 Thông số kỹ thuật của Cảm biến IMU MPU-6050

* + - * + **ESC (Electronic speed controller)**

ESC không chổi than là một trong những phụ kiện của mô hình máy bay, cần có mạch điều khiển để hoạt động. Chịu trách nhiệm nhận tín hiệu điều khiển và cung cấp năng lượng cho motor.



Hình 2.3 Esc cho động cơ không chổi than

* **Pin Li-Po WildScorpion**

Trong số các loại pin được sử dụng cho mô hình máy bay, pin Li-Po (Lithium-ion Polymer) cho dung lượng lớn với dòng ra ổn định và có kích thước nhẹ, phù hợp với ứng dụn trên các máy bay mô hình cỡ nhỏ.

|  |  |
| --- | --- |
| Kích thước | 23mm x 33mm x 107mm |
| Khối lượng | 184g |
| Dung lượng | 2200mAh |
| Điện áp | 11.1V |
| Tốc độ xả liên tục | 35C |
| Tốc độ sạc Max | 5C |
| Số cell | 3 |

Bảng 2.3 Thông số kỹ thuật của Pin Li-Po WildScorpion



Hình 2.4 Pin Li-Po

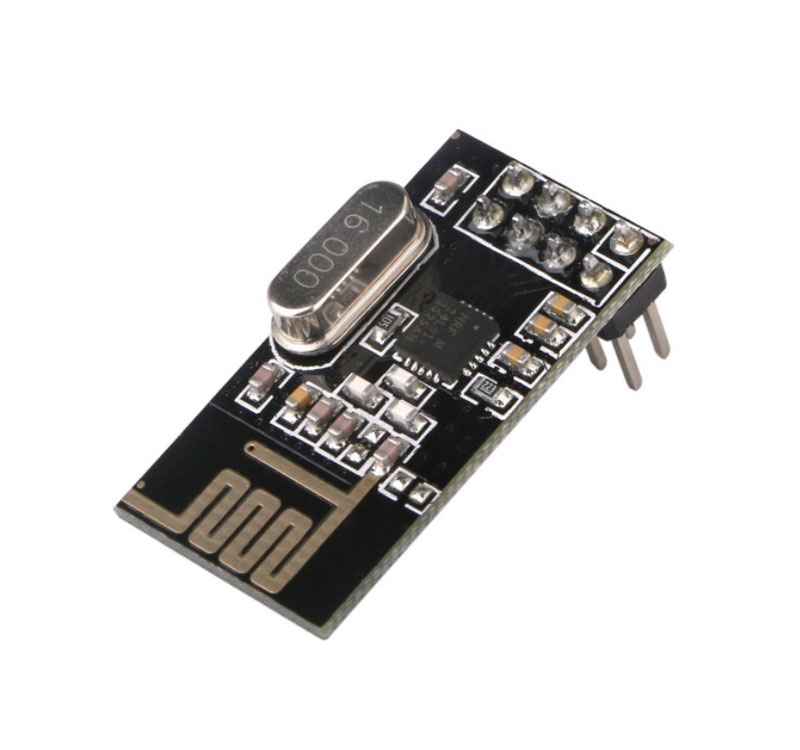
* + **Động cơ 3 pha DC không chổi than**

Thuộc nhóm động cơ đồng bộ nam châm vĩnh cửu. Là loại động cơ song hình thang, những động cơ còn lại là động cơ sóng hình Sin. So với động cơ DC có chổi than, động cơ 3 pha DC không chổi than có kết cấu cơ khí tin cậy hơn, đồng thời cho hiệu suất lực đẩy so với khối lượng cao hơn.

Hình 2.5 Động cơ 3 pha DC không chổi than

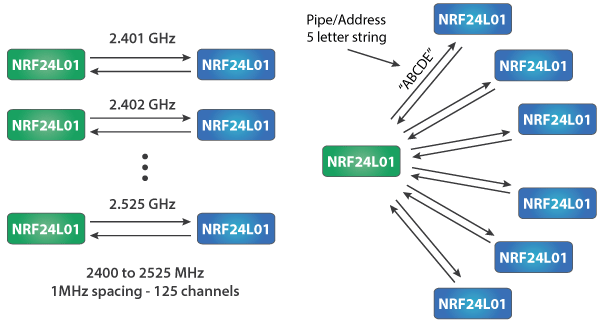


* + - * + **Module NRF24L01**

Là module truyền tải tín hiệu không dây, khoảng cách thu phát khoảng 100m.

Hình 2.6 Module NRF24L01

Thích hợp cho bộ điều khiển cầm tay. Có khả năng truyền tín hiệu 2 chiều.

Module sử dụng 125 kênh khác nhau mang lại khả năng có mạng lưới 125 modem hoạt động độc lập ở một nơi. Mỗi kênh có tối đa 6 địa chỉ hoặc mỗi đơn vị có thể liên lạc với tối đa 6 đơn vị khác cùng một lúc.

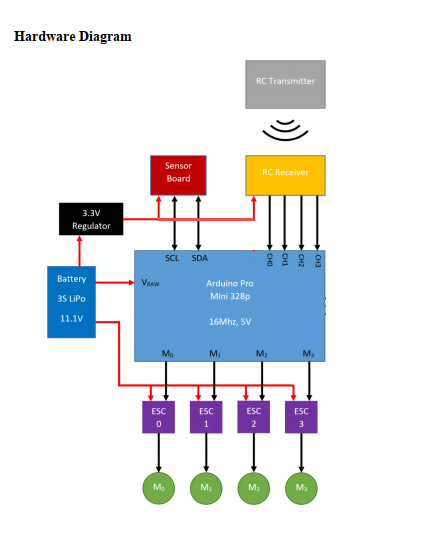
Hình 2.7 Mô hình truyền nhận của Module NRF24L01

* + **Joystick**

Là một thiết bị đầu vào bao gồm một gậy quay và bộ lọc hoặc hướng điều khiển truyền tới thiết bị mà nó đang kiểm soát. Cần điều khiển thường được sử dụng để điều khiển trò chơi điện tử hoặc robot.



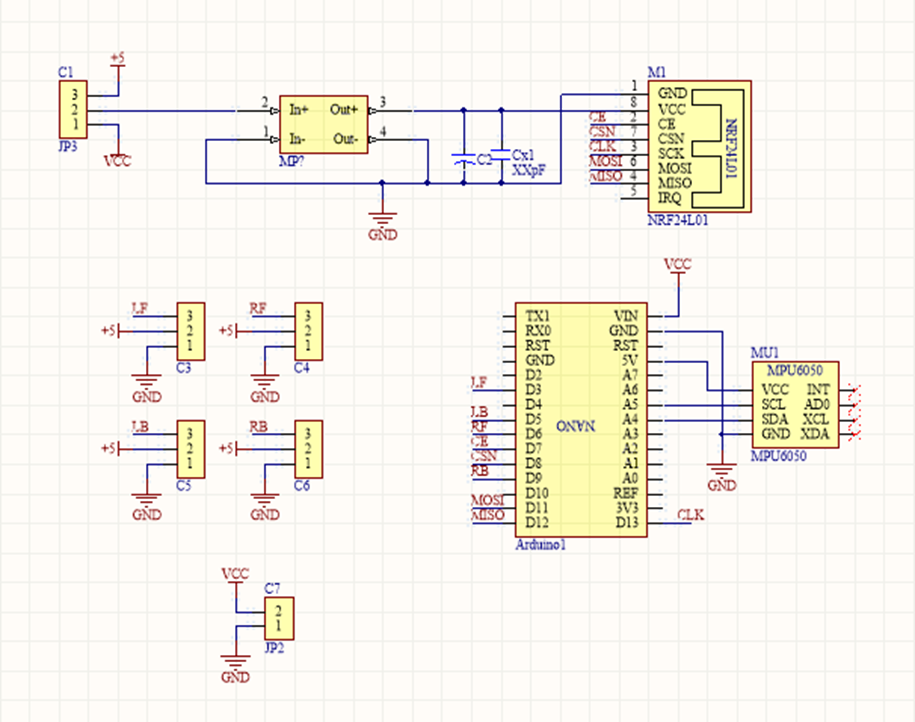
Hình 2.8 Joystick

* 1. Sơ đồ các module và mạch nguyên lý
     1. Các khối module

Hình 2.9 Hardware Diagram

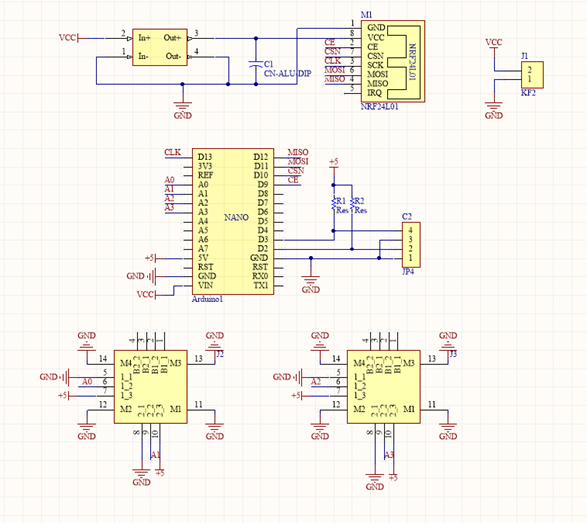
Các module chính gồm có:

* + - * + Pin: Cung cấp năng lượng cho toàn bộ hệ thống
        + RC Transmiter: Gửi tín hiệu điều khiển từ người dùng đến vi điều khiển
        + RC Receiver: Nhận tín hiệu không dây từ RC Transmiter và gửi về vi điều khiển
        + Arduino Nano: Nhận giữ liệu từ cảm biến và xử lý qua thuật toán PID để đưa ra tín hiệu cho các ESC
        + Cảm biến MPU: Gửi dữ liệu giúp vi điều khiển nhận biết được trạng thái máy bay
        + ESC: Nhận tín hiệu xung từ vi điều khiển và cung cấp năng lượng cho motor
    1. Mạch nguyên lý khối nhận và xử lý tín hiệu



Hình 2.10 Mạch nguyên lý khối nhận và xử lý tín hiệu

* + 1. Mạch nguyên lý khối truyền tín hiệu



Hình 2.11 Mạch nguyên lý khối truyền tín hiệu

* 1. Thiết kế bộ điều khiển
     1. Tìm hiểu về và các phương pháp thiết kế bộ điều khiển PID

Trong phần này, ta tiến hành xây dựng các bộ điều khiển cho UAV. Các công trình được nghiên cứu trên thế giới trước đây đã áp dụng thành công nhiều bộ điều khiển khác nhau cho việc ổn định quadcopter

Trong số đó, thực nghiệm đã chứng tỏ bộ điều khiển PID (Proportional – Integral – Derivative) cho chất lượng điều khiển tương đối tốt trong môi trường nhiễu khí động không quá lớn.

Một trong số các ưu điểm của bộ điều khiển PID là khả năng tích hợp dễ dàng với chi phí thấp lên các phần cứng thương mại được bán phổ biến trên thị trường. Giảm sai số xác lập đến mức tối thiểu nhất, hạn chế dao động, giảm thời gian xác lập và độ vọt lố. Tuy nhiên, việc tính toán và dò thực nghiệm các thông số PID trong thực tế cần nhiều kinh nghiệm và có thể mất khá nhiều thời gian. Việc đạt được một yêu cầu này thường kéo theo việc phải hạ thấp các tiêu chuẩn khác. Ví dụ: để tăng tốc độ phản hồi thường kéo theo xuất hiện vọt lố ở đầu ra so với tín hiệu đáp ứng mong muốn. Người thiết kế phải căn cứ vào các yêu cầu của hệ thống (về tốc độ phản hồi, độ bền vững của hệ thống...) để cân đối các đặc tính của hệ thống.

* + 1. Phương pháp tuyến tính kết hợp mô phỏng

Để thiết kế bộ điều khiển PID, ta có thể sử dụng mô hình quadcopter đơn giản, chỉ xét đến những yếu tố tác động chính lên ứng xử động học của máy bay. Ở đây, ta bỏ qua hiện tượng gyro của các cánh quạt do các mô-men này không đáng kể so với mô-men gây ra bởi lực đẩy từ các động cơ. Bằng cách đó, để đơn giản bài toán, ta có thể xét riêng các bài toán thiết kế bộ điều khiển cho từng chuyển động và coi chúng độc lập với nhau. Các bài toán xét đến bao gồm:

* Ổn định góc roll
* Ổn định góc pitch
* Ổn định góc yaw

Sau khi áp dụng giả thiết để đơn giản mô hình toán học, ta tiến hành mô phỏng và thiết kế bộ điều khiển bằng phần mềm MATLAB-Simulink. Để đánh giá chất lượng của các bộ điều khiển, ta căn cứ vào các tiêu chuẩn: Vọt lố xuất hiện trên đáp ứng trong miền thời gian cần không vượt quá 10% tín hiệu mong muốn.

Độ bền vững đánh giá thông qua độ dư pha và độ dư biên độ, thông qua đánh giá biểu đồ Bode của hệ hở. Hệ thống cần đạt độ dự trữ biên độ tối thiểu là 3dB và độ dự trữ pha tối thiểu là 30olà yêu cầu chung đối với phần lớn các hệ thống điều khiển trong công nghiệp công nghiệp.

* + 1. Phương pháp dò thực nghiệm kết hợp Zieler-Nichols

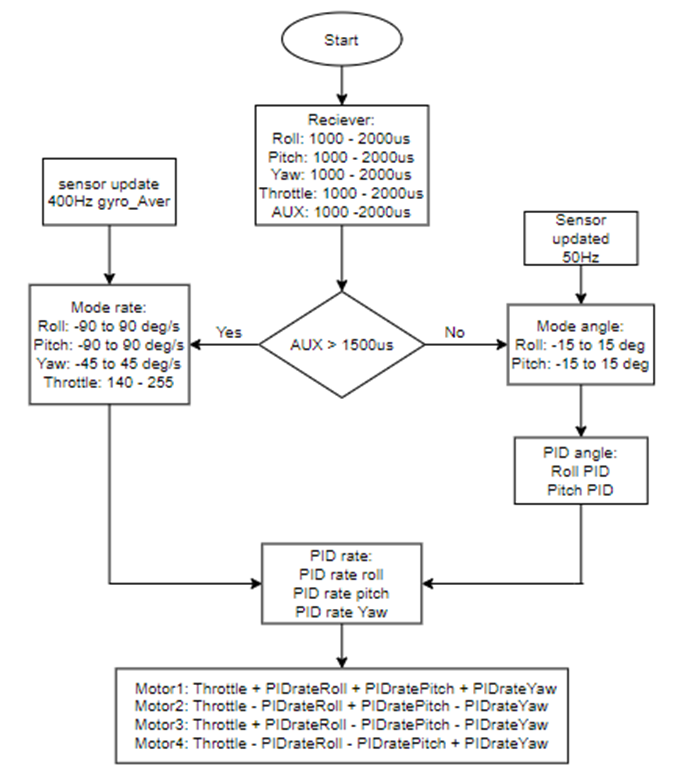
Việc thiết kế bộ điều khiển PID theo phương pháp tuyến tính có nhiều hạn chế như các tác động nhiễu, hiệu ứng vật lý tác động máy bay sẽ không sát với quá trình mô phỏng nên dẫn đến bộ PID không sát với yêu cầu thực tế. Vì vậy phương pháp dò thực nghiệm tuy khó khăn hơn nhưng cho kết quả sát với thực tế, tính ổn định tốt hơn.

Trong đề tài này, chúng em tìm các thông số của PID bằng cách kết hợp phương pháp Ziegler-Nichols và thực nghiệm.

* + 1. Lưu đồ dải thuật cân bằng

Đầu tiên, bộ xử lý nhận giá trị điều khiển từ bộ điều khiển từ xa với giá trị ban đầu từ 1000us – 2000us, sau đó so sánh các giá trị nhận được với giá trị mẫu là 1500us.

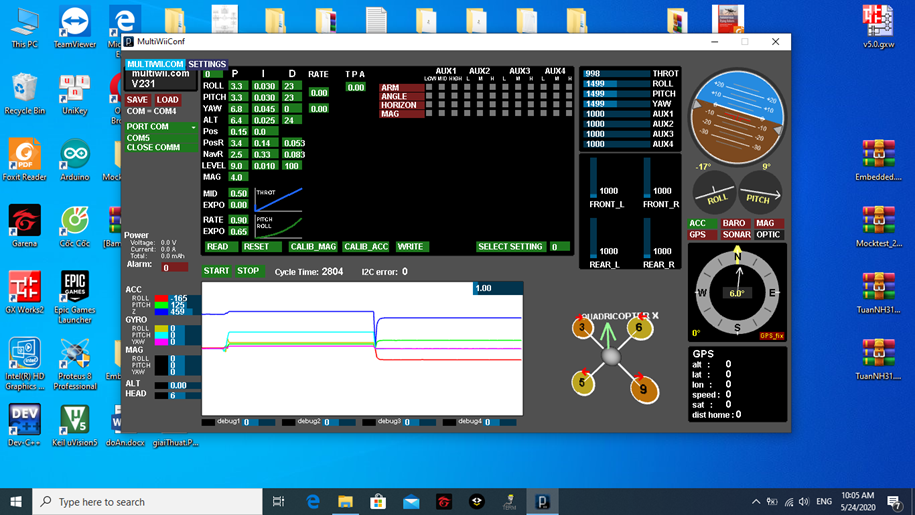
Nếu giá trị nhận được nhỏ hơn giá trị mẫu thì bộ xử lý thực hiện chế độ cân bằng góc với tần số lấy mẫu là 50Hz và giới hạn góc nghiêng trong khoảng -15 đến 15 độ. Sau đó tiếp tục thực hiện xử lý giá trị cảm biến thông qua PID. Giá trị sau khi xử lý trong PID góc sẽ được lấy để làm giá trị đầu vào của PID cân bằng di chuyển.



Hình 2.12 Lưu đồ dải thuật cân bằng

* 1. Quá trình thực nghiệm
     1. Quá trình dò các thông số PID

Từ những phần trên ta tiến hành hành dò PID sử dụng phầm mềm Multiwii hỗ trợ cho việc dò tìm thông số và quan sát trạng thái máy bay.



Hình 2.13 Phần mềm hỗ trợ điều chỉnh PID

Các bước dò Kp, Ki, Kd: Đặt máy bay ở chế độ Mode Rate (chế độ cân bằng di chuyển) sau đó thực hiện bước điều chỉnh thông số Kp, Ki, Kd:

Bước 1: Tăng 50% giá trị điều khiển tốc độ

Bước 2: Tăng Kp cho tới khi xuất hiện dao động. Sau đó giảm 10% giá trị vừa tìm được.

Bước 3: Tăng Kd cho tới khi xuất hiện dao động. Sau đo giảm 20% giá trị vừa tìm được.

Bước 4: Lặp lại bước 1 với những giá trị thay đổi nhỏ cho đến khi xuất hiện dao động.

Bước 5: Tăng Ki cho đến khi máy bay ổn định

Bước 6: Tinh chỉnh các thông số để mô hình đáp ứng tốt khi có sự ảnh hưởng của các yếu tố bên ngoài.

Các thông số Kp, Ki, Kd được lựa chọn bằng cách kết hợp phương pháp thực nghiệm và Ziegler-Nichols. Các thông số thực nghiệm được xác định theo giá được lựa chọn sao cho gần với giá trị được chọn từ phương pháp Ziegler-Nichols.

Thông số PID dò được:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Kp | Ki | Kd |
| Roll | 3.3 | 0.03 | 23 |
| Pitch | 3.3 | 0.03 | 23 |
| Yaw | 6.8 | 0.045 | 5 |

Bảng 2.4 Thông số kỹ thuật PID dò được

* + 1. Bay thử nghiểm sản phẩm

Việc dò các thông số PID rất phức tạp và cần nhiều kinh nghiệm và thời gian, nhất là đối với máy bay không người lái, cần cân bằng 3 trục cùng lúc. Do vậy mô hình chưa thể cân bằng tốt trước những tác động bên ngoài.

Hình 2.14 Bay thử nghiểm sản phẩm

1. Kết luận

Máy bay không người lái là một đề tài hay và khó. Đòi hỏi nhiều kiến thức tổng hợp như: Điện tử, lý thuyết điều khiển tự động, cơ khí… Đề tài đã thực hiện tốt phần thiết kế về điện tử: Gửi, nhận tín hiệu điều khiển không dây và phân tích tín hiệu phản hồi từ cảm biến sử dụng thuật toán PID. Tuy nhiên phần cài đặt thông số PID chưa mang lại hiệu quả cao do còn thiếu kinh nghiệm dẫn đến máy bay chưa có độ ổn định.

**TÀI LIỆU THAM KHẢO**

[1] M. Hehn and R.D’Andrea, “A flying inverted pendulum” in IEEE International Conference on Robotics and Automation (ICRA), 2011

[2] V. Thanh and H. Tung, “Optinizing PID Parameters in Control Accelerometers and Gyroscopes In Self-Balancing Quadrotors” Journal of Science and Technology, 2015

[3] N. H .T. Sơn, V. N. Phúc “Thiết kế và thi công mô hình máy bay” Đại học Lạc Hồng, 2015

[4] <https://en.wikipedia.org/wiki/PID_controller>

**PHỤ LỤC**

#include <Wire.h>

#include <Servo.h>

Servo L\_F\_prop;

Servo L\_B\_prop;

Servo R\_F\_prop;

Servo R\_B\_prop;

unsigned long counter\_1, counter\_2, counter\_3, counter\_4, current\_count;

byte last\_CH1\_state, last\_CH2\_state, last\_CH3\_state, last\_CH4\_state;

int input\_YAW;

int input\_PITCH;

int input\_ROLL;

int input\_THROTTLE;

//Gyro Variables

float elapsedTime, time, timePrev;

int gyro\_error=0;

float Gyr\_rawX, Gyr\_rawY, Gyr\_rawZ;

float Gyro\_angle\_x, Gyro\_angle\_y;

float Gyro\_raw\_error\_x, Gyro\_raw\_error\_y;

//Acc Variables

int acc\_error=0;

float rad\_to\_deg = 180/3.141592654;

float Acc\_rawX, Acc\_rawY, Acc\_rawZ;

float Acc\_angle\_x, Acc\_angle\_y;

float Acc\_angle\_error\_x, Acc\_angle\_error\_y;

float Total\_angle\_x, Total\_angle\_y;

//More variables for the code

int i;

int mot\_activated=0;

long activate\_count=0;

long des\_activate\_count=0;

//////////////////////////////PID FOR ROLL///////////////////////////

float roll\_PID, pwm\_L\_F, pwm\_L\_B, pwm\_R\_F, pwm\_R\_B, roll\_error, roll\_previous\_error;

float roll\_pid\_p=0;

float roll\_pid\_i=0;

float roll\_pid\_d=0;

///////////////////////////////ROLL PID CONSTANTS////////////////////

double roll\_kp=0.3;//3.55

double roll\_ki=0;//0.003

double roll\_kd=0;//2.05

float roll\_desired\_angle = 0;

//////////////////////////////PID FOR PITCH//////////////////////////

float pitch\_PID, pitch\_error, pitch\_previous\_error;

float pitch\_pid\_p=0;

float pitch\_pid\_i=0;

float pitch\_pid\_d=0;

///////////////////////////////PITCH PID CONSTANTS///////////////////

double pitch\_kp=0.3;//3.55

double pitch\_ki=0;//0.003

double pitch\_kd=0;//2.05

float pitch\_desired\_angle = 0;

void setup() {

PCICR |= (1 << PCIE0);

PCMSK0 |= (1 << PCINT0);

PCMSK0 |= (1 << PCINT1);

PCMSK0 |= (1 << PCINT2);

PCMSK0 |= (1 << PCINT4);

DDRB |= B00100000;

PORTB &= B11011111;

L\_F\_prop.attach(4);

L\_B\_prop.attach(5);

R\_F\_prop.attach(7);

R\_B\_prop.attach(6);

L\_F\_prop.writeMicroseconds(1000);

L\_B\_prop.writeMicroseconds(1000);

R\_F\_prop.writeMicroseconds(1000);

R\_B\_prop.writeMicroseconds(1000);

Wire.begin();

Wire.beginTransmission(0x68);

Wire.write(0x6B);

Wire.write(0x00);

Wire.endTransmission(true);

Wire.beginTransmission(0x68);

Wire.write(0x1B);

Wire.write(0x10);

Wire.endTransmission(true);

Wire.beginTransmission(0x68);

Wire.write(0x1C);

Wire.write(0x10);

Wire.endTransmission(true);

Serial.begin(9600);

delay(1000);

time = millis();

if(gyro\_error==0)

{

for(int i=0; i<200; i++)

{

Wire.beginTransmission(0x68);

Wire.write(0x43);

Wire.endTransmission(false);

Wire.requestFrom(0x68,4,true);

Gyr\_rawX=Wire.read()<<8|Wire.read();

Gyr\_rawY=Wire.read()<<8|Wire.read()

/\*---X---\*/

Gyro\_raw\_error\_x = Gyro\_raw\_error\_x + (Gyr\_rawX/32.8);

/\*---Y---\*/

Gyro\_raw\_error\_y = Gyro\_raw\_error\_y + (Gyr\_rawY/32.8);

if(i==199)

{

Gyro\_raw\_error\_x = Gyro\_raw\_error\_x/200;

Gyro\_raw\_error\_y = Gyro\_raw\_error\_y/200;

gyro\_error=1;

}

}

}

if(acc\_error==0)

{

for(int a=0; a<200; a++)

{

Wire.beginTransmission(0x68);

Wire.write(0x3B);

Wire.endTransmission(false);

Wire.requestFrom(0x68,6,true);

Acc\_rawX=(Wire.read()<<8|Wire.read())/4096.0 ;

Acc\_rawY=(Wire.read()<<8|Wire.read())/4096.0 ;

Acc\_rawZ=(Wire.read()<<8|Wire.read())/4096.0 ;

/\*---X---\*/

Acc\_angle\_error\_x=Acc\_angle\_error\_x+((atan((Acc\_rawY)/sqrt(pow((Acc\_rawX),2) + pow((Acc\_rawZ),2)))\*rad\_to\_deg));

/\*---Y---\*/

Acc\_angle\_error\_y=Acc\_angle\_error\_y+((atan(1\*(Acc\_rawX)/sqrt(pow((Acc\_rawY),2) + pow((Acc\_rawZ),2)))\*rad\_to\_deg));

if(a==199)

{

Acc\_angle\_error\_x = Acc\_angle\_error\_x/200;

Acc\_angle\_error\_y = Acc\_angle\_error\_y/200;

acc\_error=1;

}

}

}

}

void loop() {

/////////////////////////////I M U/////////////////////////////////////

timePrev = time;

time = millis();

elapsedTime = (time - timePrev) / 1000;

//////////////////////////////////////Gyro read/////////////////////////////////////

Wire.beginTransmission(0x68);

Wire.write(0x43);

Wire.endTransmission(false);

Wire.requestFrom(0x68,4,true);

Gyr\_rawX=Wire.read()<<8|Wire.read();

Gyr\_rawY=Wire.read()<<8|Wire.read();

/\*---X---\*/

Gyr\_rawX = (Gyr\_rawX/32.8) - Gyro\_raw\_error\_x;

/\*---Y---\*/

Gyr\_rawY = (Gyr\_rawY/32.8) - Gyro\_raw\_error\_y;

/\*---X---\*/

Gyro\_angle\_x = Gyr\_rawX\*elapsedTime;

/\*---X---\*/

Gyro\_angle\_y = Gyr\_rawY\*elapsedTime;

//////////////////////////////////////Acc read/////////////////////////////////////

Wire.beginTransmission(0x68);

Wire.write(0x3B);

Wire.endTransmission(false);

Wire.requestFrom(0x68,6,true);

Acc\_rawX=(Wire.read()<<8|Wire.read())/4096.0 ; //each value needs two registres

Acc\_rawY=(Wire.read()<<8|Wire.read())/4096.0 ;

Acc\_rawZ=(Wire.read()<<8|Wire.read())/4096.0 ;

/\*---X---\*/

Acc\_angle\_x=(atan((Acc\_rawY)/sqrt(pow((Acc\_rawX),2)+pow((Acc\_rawZ),2)))\*rad\_to\_deg) - Acc\_angle\_error\_x;

/\*---Y---\*/

Acc\_angle\_y=(atan(-1\*(Acc\_rawX)/sqrt(pow((Acc\_rawY),2)+pow((Acc\_rawZ),2)))\*rad\_to\_deg) - Acc\_angle\_error\_y;

//////////////////////////////////////Total angle and filter/////////////////////////////////////

/\*---X axis angle---\*/

Total\_angle\_x = 0.98 \*(Total\_angle\_x + Gyro\_angle\_x) + 0.02\*Acc\_angle\_x;

/\*---Y axis angle---\*/

Total\_angle\_y = 0.98 \*(Total\_angle\_y + Gyro\_angle\_y) + 0.02\*Acc\_angle\_y;

/\*///////////////////////////P I D///////////////////////////////////\*/

roll\_desired\_angle = map(input\_ROLL,1000,2000,-10,10);

pitch\_desired\_angle = map(input\_PITCH,1000,2000,-10,10);

roll\_error = Total\_angle\_y - roll\_desired\_angle;

pitch\_error = Total\_angle\_x - pitch\_desired\_angle;

roll\_pid\_p = roll\_kp\*roll\_error;

pitch\_pid\_p = pitch\_kp\*pitch\_error

if(-3 < roll\_error <3)

{

roll\_pid\_i = roll\_pid\_i+(roll\_ki\*roll\_error);

}

if(-3 < pitch\_error <3)

{

pitch\_pid\_i = pitch\_pid\_i+(pitch\_ki\*pitch\_error);

}

roll\_pid\_d = roll\_kd\*((roll\_error - roll\_previous\_error)/elapsedTime);

pitch\_pid\_d = pitch\_kd\*((pitch\_error - pitch\_previous\_error)/elapsedTime);

roll\_PID = roll\_pid\_p + roll\_pid\_i + roll\_pid\_d;

pitch\_PID = pitch\_pid\_p + pitch\_pid\_i + pitch\_pid\_d;

if(roll\_PID < -400){roll\_PID=-400;}

if(roll\_PID > 400) {roll\_PID=400; }

if(pitch\_PID < -400){pitch\_PID=-400;}

if(pitch\_PID > 400) {pitch\_PID=400;}

pwm\_R\_F = 115 + input\_THROTTLE - roll\_PID - pitch\_PID;

pwm\_R\_B = 115 + input\_THROTTLE - roll\_PID + pitch\_PID;

pwm\_L\_B = 115 + input\_THROTTLE + roll\_PID + pitch\_PID;

pwm\_L\_F = 115 + input\_THROTTLE + roll\_PID - pitch\_PID;

//Right front

if(pwm\_R\_F < 1100)

{

pwm\_R\_F= 1100;

}

if(pwm\_R\_F > 2000)

{

pwm\_R\_F=2000;

}

//Left front

if(pwm\_L\_F < 1100)

{

pwm\_L\_F= 1100;

}

if(pwm\_L\_F > 2000)

{

pwm\_L\_F=2000;

}

//Right back

if(pwm\_R\_B < 1100)

{

pwm\_R\_B= 1100;

}

if(pwm\_R\_B > 2000)

{

pwm\_R\_B=2000;

}

//Left back

if(pwm\_L\_B < 1100)

{

pwm\_L\_B= 1100;

}

if(pwm\_L\_B > 2000)

{

pwm\_L\_B=2000;

}

roll\_previous\_error = roll\_error;

pitch\_previous\_error = pitch\_error;

if(mot\_activated)

{

L\_F\_prop.writeMicroseconds(pwm\_L\_F);

L\_B\_prop.writeMicroseconds(pwm\_L\_B);

R\_F\_prop.writeMicroseconds(pwm\_R\_F);

R\_B\_prop.writeMicroseconds(pwm\_R\_B);

}

if(!mot\_activated)

{

L\_F\_prop.writeMicroseconds(1000);

L\_B\_prop.writeMicroseconds(1000);

R\_F\_prop.writeMicroseconds(1000);

R\_B\_prop.writeMicroseconds(1000);

}

if(input\_THROTTLE < 1100 && input\_YAW > 1800 && !mot\_activated)

{

if(activate\_count==200)

{

mot\_activated=1;

PORTB |= B00100000;

}

activate\_count=activate\_count+1;

}

if(!(input\_THROTTLE < 1100 && input\_YAW > 1800) && !mot\_activated)

{

activate\_count=0;

}

if(input\_THROTTLE < 1100 && input\_YAW < 1100 && mot\_activated)

{

if(des\_activate\_count==300)

{

mot\_activated=0;

PORTB &= B11011111;

}

des\_activate\_count=des\_activate\_count+1;

}

if(!(input\_THROTTLE < 1100 && input\_YAW < 1100) && mot\_activated)

{

des\_activate\_count=0;

}

input\_THROTTLE = 1200;

mot\_activated = 1;

}

ISR(PCINT0\_vect){

current\_count = micros();

//Channel 1

if(PINB & B00000001){

if(last\_CH1\_state == 0){

last\_CH1\_state = 1;

counter\_1 = current\_count;

}

}

else if(last\_CH1\_state == 1){

last\_CH1\_state = 0;

input\_ROLL = current\_count - counter\_1;

}

//Channel 2

if(PINB & B00000010 ){

if(last\_CH2\_state == 0){

last\_CH2\_state = 1;

counter\_2 = current\_count;

}

}

else if(last\_CH2\_state == 1){

last\_CH2\_state = 0;

input\_PITCH = current\_count - counter\_2;

}

//Channel 3

if(PINB & B00000100 ){

if(last\_CH3\_state == 0){

last\_CH3\_state = 1;

counter\_3 = current\_count;

}

}

else if(last\_CH3\_state == 1){

last\_CH3\_state = 0;

input\_THROTTLE = current\_count - counter\_3;

}

//Channel 4

if(PINB & B00010000 ){

if(last\_CH4\_state == 0){

last\_CH4\_state = 1;

counter\_4 = current\_count;

}

}

else if(last\_CH4\_state == 1){

last\_CH4\_state = 0;

input\_YAW = current\_count - counter\_4;

}

}