**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**



**KHÓA LUẬN TỐT NGHIỆP**

**HỆ THỐNG CHẤM CÔNG CHO VĂN PHÒNG**

**KỸ SƯ NGÀNH KỸ THUẬT MÁY TÍNH**

Giảng viên hướng dẫn: **TS. Hà Lê Hoài Trung**

Sinh viên thực hiện: **Lê Ngọc Huy**

**Liễu Hoàng Anh**

Lớp: **KTMT2013**

***TP. Hồ Chí Minh, tháng 01 năm 2018***

**NHẬN XÉT**

**(Của giảng viên hướng dẫn)**

TP.HCM, ngày … tháng … năm 2018

TS. Hà Lê Hoài Trung

**NHẬN XÉT**

**(Của giảng viên phản biện)**

TP.HCM, ngày … tháng … năm 2018

# **LỜI CẢM ƠN**

Với những gì đã học được tại khoa Kĩ Thuật Máy Tính - trường Đại học Công Nghệ Thông Tin và cùng với sự cho phép của nhà trường, của khoa, chúng em đã vinh dự được thực hiện đề tài cho luận văn tốt nghiệp này.

Do định hướng đề tài từ trước và có kế hoạch phù hợp từ đầu nên đề tài của nhóm vẫn hoàn thành đúng mục tiêu đặt ra. Tuy nhiên, trong quá trình thực hiện, nhóm gặp phải một số vấn đề khó khăn, dẫn đến mất nhiều thời gian. Được sự giúp đỡ của các giảng viên trong khoa, trong trường, nhóm đã tìm ra được cách giải quyết để cho đề tài trên được tiến hành như đã định.

Chúng em xin gửi lời cảm ơn đến Thạc sĩ **Hà Lê Hoài Trung** – Giảng viên hướng dẫn đề tài khóa luận tốt nghiệp, đã giúp nhóm định hướng và thực hiện đề tài này. Đồng thời, nhóm xin cảm ơn các thầy cô trong khoa **Kĩ Thuật Máy Tính** đã tận tâm giúp đỡ nhóm khi đề tài gặp khó khăn. Chúng em hi vọng vẫn nhận được sự giúp đỡ của các thầy, cô trong những quá trình học tập và nghiên cứu về sau nữa.

Cảm ơn tất cả các bạn trong khoa đã chia sẻ kinh nghiệm và có những ý kiến đóng góp để đề tài được thực hiện suôn sẻ.

Mặc dù chúng em đã cố gắng hết sức mình nhưng sẽ không tránh khỏi những thiếu sót. Kính mong quý thầy cô và các bạn tiếp tục đóng góp và giúp đỡ nhóm hoàn thành luận văn tốt nghiệp.

Cuối cùng, chúng em xin gửi đến quý thầy cô và các bạn lời chúc sức khỏe và lời cảm ơn chân thành nhất!

Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 2 tháng 2 năm 2018

Nhóm thực hiện đề tài

**Lê Ngọc Huy**

**Liễu Hoàng Anh**

Khoa Kĩ Thuật Máy Tính. Lớp KTMT2013

# **MỤC LỤC**

[TÓM TẮT BÁO CÁO i](#_Toc474362434)

[CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI 1](#_Toc474362435)

[1.1 Lý do chọn đề tài 1](#_Toc474362436)

[1.2 Giới thiệu đề tài 1](#_Toc474362437)

[1.3 Tình hình nghiên cứu trong và ngoài nước 2](#_Toc474362438)

[1.3.1 Trong nước 2](#_Toc474362439)

[1.3.2 Ngoài nước 3](#_Toc474362443)

[1.4 Mục đích, đối tượng và phạm vi nghiên cứu 5](#_Toc474362448)

[1.4.1 Mục đích nghiên cứu 5](#_Toc474362449)

[1.4.2 Đối tượng nghiên cứu 6](#_Toc474362450)

[1.4.3 Phạm vi nghiên cứu 6](#_Toc474362451)

[1.5 Phương pháp nghiên cứu 7](#_Toc474362452)

[1.5.1 Phương pháp nghiên cứu trực tiếp 7](#_Toc474362453)

[1.5.2 Phương pháp nghiên cứu gián tiếp 7](#_Toc474362454)

[1.6 Ý nghĩa lý luận và thực tiễn của đề tài 7](#_Toc474362455)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 8](#_Toc474362456)

[2.1 Giới thiệu mô hình máy chấm công 8](#_Toc474362457)

[2.1.1 Sơ lược về cấu tạo 8](#_Toc474362458)

[2.1.2 Sơ lược về nguyên lý bay 9](#_Toc474362461)

[2.2 Tìm hiểu giải thuật PID cho Quadcopter 10](#_Toc474362462)

[2.3 Tìm hiểu về PWM 10](#_Toc474362464)

[2.4 Bộ lọc Kalman 11](#_Toc474362465)

[2.4.1 Khái niệm chung về bộ lọc 11](#_Toc474362466)

[2.4.2 Bộ lọc Kalman 11](#_Toc474362467)

[2.5 Giao thức HTTP 13](#_Toc474362471)

[2.6 Uniform Resource Locator 13](#_Toc474362473)

[2.7 Mã Hash MD5 14](#_Toc474362474)

[2.8 Định dạng JSON 14](#_Toc474362475)

[2.9 Chuẩn truyền thông SPI 15](#_Toc474362477)

[2.9.1 Khái niệm 15](#_Toc474362478)

[2.9.2 Hoạt động 16](#_Toc474362480)

[2.10 Chuỗi NMEA và RMC 17](#_Toc474362482)

[2.11 Linh kiện được sử dụng trong quá trình thực hiện đồ án 18](#_Toc474362484)

[2.11.1 Sơ lược về Raspberry Pi 2 Model B 18](#_Toc474362485)

[2.11.2 Motor 1000kV 19](#_Toc474362488)

[2.11.3 Bộ điều tốc ESC 20](#_Toc474362490)

[2.11.4 Khung máy bay 20](#_Toc474362492)

[2.11.5 Pin Lipo 21](#_Toc474362494)

[2.11.6 Cảm biến MPU 6050 21](#_Toc474362496)

[2.11.7 Kit Tiva Launchpad 123 22](#_Toc474362497)

[2.11.9 Mạch thu phát RF NRF24L01 24](#_Toc474362500)

[2.11.10 Mạch giảm áp DC LM2596 25](#_Toc474362502)

[2.11.11 Mạch giảm áp AMS1117 26](#_Toc474362504)

[CHƯƠNG 3. THIẾT KẾ VÀ HIỆN THỰC PHẦN MỀM ĐIỀU KHIỂN 27](#_Toc474362506)

[3.1 Nội dung hiện thực 27](#_Toc474362507)

[3.2 Cấu hình RASPI trở thành Access Point 27](#_Toc474362509)

[3.3 Hiện thực giao thức HTTP 30](#_Toc474362510)

[3.3.1 Client 30](#_Toc474362511)

[3.3.2 Server 31](#_Toc474362512)

[3.3.3 Lập trình HTTP Server 33](#_Toc474362514)

[3.4 Hiện thực trên điện thoại Android 36](#_Toc474362518)

[3.4.1 Giao tiếp giữa RASPI và điện thoại Android 36](#_Toc474362519)

[3.4.2 Các hàm xử lý chính 38](#_Toc474362524)

[3.5 Hiện thực trên Board Tiva và mô hình Quadcopter 39](#_Toc474362525)

[3.5.1 Giao tiếp giữa RASPI và Board Tiva 39](#_Toc474362526)

[3.5.2 Các hàm xử lý chính 41](#_Toc474362528)

[3.5.3 Các hàm xử lý dữ liệu và điều khiển Quadcopter 42](#_Toc474362530)

[3.5.4 Cân chỉnh PID để giữ cho máy bay cân bằng 45](#_Toc474362533)

[3.6 Đọc dữ liệu GPS 50](#_Toc474362542)

[3.7 Giao tiếp RF 51](#_Toc474362544)

[3.8 Lập trình bay thẳng cho máy bay 55](#_Toc474362549)

[CHƯƠNG 4. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ 57](#_Toc474362551)

[4.1. Lắp ráp và thiết kế khung cho Quadcopter 57](#_Toc474362552)

[4.2 Kết quả thử nghiệm và đánh giá 58](#_Toc474362556)

[4.2.1 Kiểm tra xung và tốc độ các động cơ 58](#_Toc474362557)

[4.2.2 Kalmal filter 59](#_Toc474362559)

[4.2.3 PID controller 59](#_Toc474362561)

[CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 61](#_Toc474362564)

[5.1 Thuận lợi và khó khăn 61](#_Toc474362565)

[5.1.1 Thuận lợi 61](#_Toc474362566)

[5.1.2 Khó khăn 61](#_Toc474362567)

[5.1.3 Hướng giải quyết 61](#_Toc474362568)

[5.2 Kết quả đạt được và chưa đạt được 62](#_Toc474362569)

[5.2.1 Kết quả đạt được 62](#_Toc474362570)

[5.2.2 Kết quả chưa đạt được 62](#_Toc474362571)

[5.3 Hướng phát triển 63](#_Toc474362572)

[TÀI LIỆU THAM KHẢO TRONG BÁO CÁO 64](#_Toc474362573)

# **DANH MỤC HÌNH ẢNH**

[Hình 1.1 Quadcopter của Ban Công nghệ FPT 2](#_Toc474362585)

[Hình 1.2 Quadcopter của Đại học Công Nghệ 3](#_Toc474362586)

[Hình 1.3 Quadcopter của Đại học Bách khoa Hà Nội 3](#_Toc474362587)

[Hình 1.4 Quadcopter của công ty Urban Aeronautics 4](#_Toc474362589)

[Hình 1.5 Quadcopter của hải quân Nga 4](#_Toc474362590)

[Hình 1.6 AR Drone của Parrot() 5](#_Toc474362591)

[Hình 1.7 Sự phát triển máy bay không người lái theo quốc gia 5](#_Toc474362592)

[Hình 2.1 Mô hình Quadcopter 8](#_Toc474362604)

[Hình 2.2 Các lực tác dụng lên Quadcopter 9](#_Toc474362605)

[Hình 2.3 Sơ đồ giải thuật điều khiển kinh điển PID 10](#_Toc474362608)

[Hình 2.4 Sơ đồ bộ lọc Kalman 12](#_Toc474362613)

[Hình 2.5 Bản chất của bộ lọc Kalman 12](#_Toc474362614)

[Hình 2.6 Hoạt động của bộ lọc Kalman 12](#_Toc474362615)

[Hình 2.7 Giao thức HTTP 13](#_Toc474362617)

[Hình 2.8 Chuỗi JSON 14](#_Toc474362621)

[Hình 2.9 Giao diện SPI 15](#_Toc474362624)

[Hình 2.10 Truyền dữ liệu SPI 16](#_Toc474362626)

[Hình 2.11 Cấu trúc bản tin theo giao thức NMEA 17](#_Toc474362628)

[Hình 2.12 Kit Raspberry Pi 2 model B 18](#_Toc474362631)

[Hình 2.13 Sơ đồ cấu tạo 19](#_Toc474362632)

[Hình 2.14 Motor 1000kV 20](#_Toc474362634)

[Hình 2.15 Bộ điều tốc ESC 20](#_Toc474362636)

[Hình 2.16 Khung máy bay 21](#_Toc474362638)

[Hình 2.17 Pin Lipo 21](#_Toc474362640)

[Hình 2.20 Dạng đóng gói của Kit Tiva 23](#_Toc474362643)

[Hình 2.21 Mạch GPS SiRF tích hợp Angten 24](#_Toc474362644)

[Hình 2.22 Mạch thu phát RF NRF24L01 25](#_Toc474362646)

[Hình 2.23 Mạch giảm áp DC LM2596 25](#_Toc474362648)

[Hình 2.24 Mạch giảm áp AMS1117 26](#_Toc474362650)

[Hình 3.1 Sơ đồ hệ thống 27](#_Toc474362653)

[Hình 3.2 Chuyển command sang format MD5 33](#_Toc474362660)

[Hình 3.3 Quá trình gửi và nhận command 34](#_Toc474362661)

[Hình 3.4 Hiện thực http server 35](#_Toc474362662)

[Hình 3.5 Giao diện điện thoại Android khi chưa có kết nối 36](#_Toc474362665)

[Hình 3.6 Giao diện điện thoại Android App khi kết nối server 37](#_Toc474362666)

[Hình 3.7 Điện thoại mất kết nối server 37](#_Toc474362667)

[Hình 3.8 Giao diện điện thoại Android chế độ auto 38](#_Toc474362668)

[Hình 3.9 Mô hình tổng quát phần cứng của máy bay 40](#_Toc474362672)

[Hình 3.10 Sơ đồ giải thuật điều khiển máy bay 41](#_Toc474362674)

[Hình 3.11 Tín hiệu sử dụng mạch lọc thông thấp 42](#_Toc474362676)

[Hình 3.12 Tín hiệu đã được lọc nhiễu 44](#_Toc474362677)

[Hình 3.13 Cân chỉnh PID theo frame “**+**” 46](#_Toc474362681)

[Hình 3.14 Cân chỉnh PID frame “X” 47](#_Toc474362682)

[Hình 3.15 Chỉnh P 48](#_Toc474362683)

[Hình 3.16 Thuật toán phân tích dữ liệu GPS 51](#_Toc474362688)

[Hình 3.17 Format message chung cho giao tiếp client – server 53](#_Toc474362691)

[Hình 3.18 Format message khi lấy location 53](#_Toc474362692)

[Hình 3.19 Format message khi máy bay di chuyển 54](#_Toc474362693)

[Hình 3.20 Góc giữa phương nối hai điểm so với kinh tuyến 55](#_Toc474362695)

[Hình 4.1 Lắp ráp ESC và motor 57](#_Toc474362698)

[Hình 4.2 Vị trí đặt cảm biến 57](#_Toc474362699)

[Hình 4.3 Mô hình Quadcopter hoàn chỉnh 58](#_Toc474362700)

[Hình 4.4 Thử nghiệm đo xung PWM và tốc độ động cơ motor 58](#_Toc474362703)

[Hình 4.5 Kết quả bộ lọc Kalman 59](#_Toc474362705)

[Hình 4.6 Đáp ứng biên độ PID 59](#_Toc474362707)

[Hình 4.7 Căn chỉnh PID thực tế 60](#_Toc474362708)

# 

# **DANH MỤC BẢNG**

[Bảng 3.1 Cấu hình chân cho RASPI khi config HTTP server 32](#_Toc474362658)

[Bảng 3.2 Bảng tham chiếu Ziegler – Nichols 45](#_Toc474362679)

[Bảng 3.3 Tác động khi điều chỉnh các thông số PID 45](#_Toc474362680)

[Bảng 3.4 Căn chỉnh trục Roll 49](#_Toc474362684)

[Bảng 3.5 Căn chỉnh trục Pitch 50](#_Toc474362685)

[Bảng 3.6 Căn chỉnh trục Yaw 50](#_Toc474362686)

[Bảng 3.7 Kết nối PIN giữa RASPI và RF 52](#_Toc474362690)

# **DANH MỤC CHỮ VIẾT TẮT**

|  |  |
| --- | --- |
| GPS | Global Positioning System |
| PID | Propotional-Integral-Derivative |
| PWM | Pulse Width Modulation |
| SPI | Serial Peripheral Interface |
| I2C | Inter Integrated Circuit |
| HTTP | Hyper Text Transfer Protocol |
| RASPI | Raspberry Pi 2 Model B |
| ROLL, PITCH, YAW | Euler angles |

# **TÓM TẮT BÁO CÁO**

Ngày nay từ khóa “*Quadcopter*” hay “*Drone*” không còn xa lạ với giới khoa học, công nghệ viễn thông trong và ngoài nước. Rất nhiều công trình nghiên cứu và phát triển trước đó đã thành công vì tính thực tiễn mà nó đem lại. Quadcopter đang là niềm hi vọng trong một số lĩnh vực dân dụng, quân sự hay khoa học vũ trụ, khi mà con người đang dần thay thế các phương tiện bay có người lái bằng các thiết bị bay không người, bởi các tính năng ưu việt của nó như có thể điều khiển từ xa hoặc có khả năng hoạt động ở những nơi mà con người khó tiếp cận được. Để có thể tối ưu hóa được các hoạt động của nó thì tính năng tự động là một yếu tố không thể thiếu đối với loại thiết bị này.

Khóa luận tốt nghiệp với đề tài **Thuật toán điều khiển Drone theo bản đồ lưu sẵn** là một quá trình tìm hiểu và thực hiện đầy thách thức của nhóm. Đề tài là một lĩnh vực khá rộng, cần am hiểu nhiều kiến thức liên quan như điện - điện tử, điều khiển tự động, lập trình Android, cấu trúc dữ liệu và giải thuật, đòi hỏi quá trình thực hiện tương đối dài và khó khăn. Với mục đích tự động hóa quá trình hoạt động của Drone và mang lại một sản phẩm có tính ứng dụng cao, nhóm lựa chọn đề tài này không chỉ vì tính thực tiễn của đề tài mà còn tạo cơ hội cho bản thân thử sức với những lĩnh vực rộng hơn.

Báo cáo khóa luận được chia thành năm chương với các nội dung như sau:

* **Chương 1.** Tổng quan đề tài.
* **Chương 2.** Cơ sở lý thuyết.
* **Chương 3.** Thiết kế và hiện thực phần mềm điều khiển
* **Chương 4.** Kết quả thử nghiệm và đánh giá.
* **Chương 5.** Kết luận và kiến nghị.

# **CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI**

## **1.1 Lý do chọn đề tài**

Mang lại hứng thú cho giới trẻ cũng như niềm đam mê cho các nhà khoa học, máy bay không người lái đang là xu hướng nghiên cứu và hiện thực trên thế giới. Việc nghiên cứu và chế tạo máy bay không người lái đã được nhiều nước trên thế giới thực hiện từ rất lâu, chủ yếu phục vụ cho các mục đích quốc phòng, quân sự.

Trong thời kì phát triển của công nghiệp hóa, hiện đại hóa, việc tự động hóa một thiết bị hay máy móc là việc vô cùng cần thiết. Quá trình hoạt động của máy bay được điều khiển một cách tự động, cùng với đó sẽ có sự theo dõi chặt chẽ trên hệ thống, góp phần phục vụ cho tính thực tiễn của nó chính là mục đích thôi thúc nhóm thực hiện đề tài này.

Bên cạnh đó, đề tài thuật toán điều khiển drone theo bản đồ lưu sẵn là một đề tài đòi hỏi kiến thức tổng hợp của nhiều lĩnh vực như điện - điện tử, điều khiển tự động, lập trình Android, cấu trúc dữ liệu và giải thuật, thiết kế cơ khí, toán học, truyền nhận tín hiệu, xử lý tín hiệu số… Qua quá trình thực hiện đề tài, nhóm sẽ có cơ hội và thời gian để tìm hiểu nhiều hơn về các kiến thức bổ ích, góp phần nâng cao vốn kiến thức chuyên môn về Kỹ thuật Máy tính cũng như các kỹ năng khác trong qua trình học tập tại trường.

## **1.2 Giới thiệu đề tài**

Đề tài thuật toán điều khiển drone theo bản đồ lưu sẵn sử dụng 2 module chính là board Raspberry Pi 2 Model B và board Tiva Launchpah, kết hợp thêm nhiều module với nhau, thiết kế nên một mô hình máy bay để thực hiện các chức năng cơ bản. Việc kết nối thông tin giữa người sử dụng và máy bay thông qua một server – client được thiết lập trên RASPI và điện thoại Android. Việc truyền tín hiệu đến máy bay được diễn ra thông qua RF giữa RASPI và Tiva. Điện thoại Android sẽ điều khiển máy bay ở hai chế độ là điều khiển bằng tay và tự động. Trên điện thoại sẽ lưu sẵn một bản đồ hiển thị khu vực khảo sát, từ đó ấn định điểm xuất phát và điểm đến cho máy bay trong mỗi quá trình bay.

Việc xác định vị trí của máy bay tại mỗi thời điểm được GPS gửi tín hiệu về hệ thống mỗi giây, từ đó thông tin này được xử lý để xác định đường đi cho máy bay.

## **1.3 Tình hình nghiên cứu trong và ngoài nước**

### **1.3.1 Trong nước**

Có rất nhiều công trình nghiên cứu của các công ty, tập đoàn và các trường đại học trên cả nước: Ban công nghệ FPT, Đại học bách khoa, Đại học FPT, Đại học sư phạm kỹ thuật, Đại học Cần Thơ…

* Mô hình máy bay của Ban Công nghệ FPT



# Hình 1.1 Quadcopter của Ban Công nghệ FPT

Mô hình máy bay này có thể mang vật nặng tối đa khoảng 2 kg, bay được 5 km.

Ban nghiên cứu loại máy này hướng đến việc máy bay có thể giao hàng một

cách nhanh chóng và tiện lợi.

* Mô hình máy bay của Đại học Công Nghệ: được thiết kế nhỏ gọn tối ưu với bộ điều khiển phù hợp, có hiển thị tốc độ của máy bay trong quá trình di chuyển.



# Hình 1.2 Quadcopter của Đại học Công Nghệ

* Máy bay QuadRotor do Đại học Bách khoa Hà Nội



# Hình 1.3 Quadcopter của Đại học Bách khoa Hà Nội

Nhìn chung, các mô hình máy bay đều hoạt động tốt nhưng các đề tài vẫn còn mang tính “đóng”, chỉ dừng lại ở mức điều khiển manual, khả năng ứng dụng chưa cao và khó có thể nâng cấp, phát triển lên.

### **1.3.2 Ngoài nước**

Rất nhiều nghiên cứu trên thế giới đã và đang được phát triển thành sản phẩm để phục vụ cho nhiều mục đích khác nhau. Đáng chú ý khi đây cũng chính là ngành công nghiệp tiềm năng, mang lại nhiều lợi nhuận cho các quốc gia và nhà sản xuất: khoảng

11.3 tỷ USD năm 2013 và ước tính khoảng 140 tỷ đô năm 2020 ([[1]](#footnote-1)).

Một số sản phẩm nổi bật:

* Drone của công ty Urban Aeronautics của Israel



# Hình 1.4 Quadcopter của công ty Urban Aeronautics

Chiếc drone này có kích thước giống như một chiếc xe đã được phát triển 15 năm với mục đích sử dụng cho tìm kiếm cứu nạn, cứu thương trên chiến trường hoặc chuyên chở hàng hóa, binh lính tới những nơi trực thăng khó tiếp cận ([[2]](#footnote-2)).

* Drone của hải quân Nga ([[3]](#footnote-3))



# Hình 1.5 Quadcopter của hải quân Nga

Chiếc máy bay hoạt động trên tàu chiến với thiết kế thông minh đã trợ giúp đắc

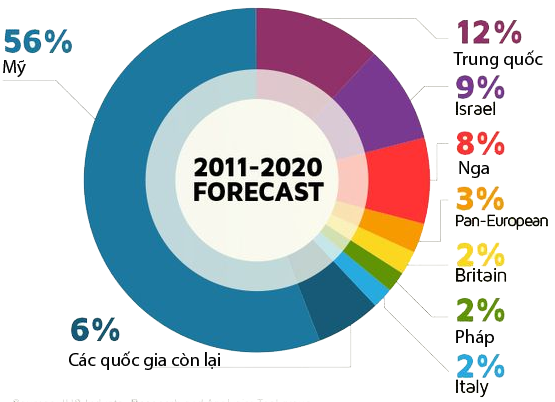
lực cho hải quân Nga trong quá trình tìm kiếm cứu nạn và dò xét an ninh hàng hải.

* AR.Drone được điều khiển từ xa qua sóng [WiFi](https://tinhte.vn/tags/wifi/) do hãng Parrot phát triển.



# Hình 1.6 AR Drone của Parrot([[4]](#footnote-4))

Máy bay không người lái vẫn đang được nhiều quốc gia chú trọng và đầu tư. Ưu điểm của loại thiết bị này là mục đích hướng đến của các nhà phát triển. Theo nghiên cứu của trang Cloud Tweaks thì sự phát triển drone phân theo quốc gia như sau ([[5]](#footnote-5)):



# Hình 1.7 Sự phát triển máy bay không người lái theo quốc gia

## **1.4 Mục đích, đối tượng và phạm vi nghiên cứu**

### **1.4.1 Mục đích nghiên cứu**

Khóa luận tập trung nghiên cứu và điều khiển quadcopter hoạt động ổn định với hai chế độ là điều khiển bằng tay và tự động.

Khóa luận sử dụng board Raspberry Pi 2 Model B và board Tiva Launchpad làm

Board điều khiển trung tâm, cùng với nhiều module hỗ trợ khác như GPS, ESC,

motor, cánh quạt,… để tạo nên một hệ thống hoàn chỉnh. Cụ thể đề tài khóa luận

thực hiện:

* Xây dựng một client – server để thực hiện giao tiếp giữa người dùng và các board trung tâm.
* Xây dựng ứng dụng cho điện thoại Android để điều khiển máy bay
* Ở chế độ manual, máy bay di chuyển theo sự điều khiển của người dùng thông qua điện thoại Android.
* Ở chế độ auto, drone sẽ bay theo một bản đồ được lưu sẵn trên điện thoại Android.
* Tìm hiểu và ứng dụng bộ lọc số trong việc lọc nhiễu tín hiệu đọc về từ con quay hồi chuyển.
* Ứng dụng giải thuật PID vào hệ thống để xây dựng thuật toán điều khiển máy bay.
* Tìm hiểu và giao tiếp các module MPU 6050, GPS, RF với hệ thống.
* Xác định năng lượng cần thiết cho quá trình bay.
* Xây dựng giải thuật bay thẳng cho máy bay.

### **1.4.2 Đối tượng nghiên cứu**

Các thiết bị lắp ráp, mô hình máy bay và hệ thống điều khiển qua điện thoại để cân bằng máy bay, đồng thời bay theo mục đích người dùng.

### **1.4.3 Phạm vi nghiên cứu**

Trong giới hạn của đề tài, nhóm giới hạn nghiên cứu thiết kế hệ thống máy bay và lập trình điều khiển quá trình bay thẳng cho máy bay theo một bản đồ lưu sẵn. Cụ thể là bay qua 2 điểm hoặc nhiều điểm, thời gian bay phụ thuộc vào năng lượng được cung cấp trên máy bay, với phạm vi điều khiển trong tầm bán kính khoảng 500m. Máy bay có thể bay ở độ cao tối đa là 100m trong điều khiển thời tiết bình thường và ít vật cản hoặc môi trường ít biến động.

## **1.5 Phương pháp nghiên cứu**

### **1.5.1 Phương pháp nghiên cứu trực tiếp**

Đặc điểm của phương pháp này chính là xác định trực tiếp được lời giải qua một thủ tục tính toán (công thức, hệ luật, định luật…) hoặc qua các bước căn bản để có được lời giải. Ngoài ra, việc giải quyết vấn đề trên máy tính chỉ là thao tác lập trình hay là sự chuyển đổi lời giải từ ngôn ngữ bên ngoài sang các ngôn ngữ được sử dụng trong máy tính. Ở đây, nhóm đã sử dụng ngôn ngữ lập trình Cđể hiện thực các chức năng cần có cho hệ thống.

### **1.5.2 Phương pháp nghiên cứu gián tiếp**

Phương pháp thử - sai được sử dụng để tìm ra kết quả của bài toán. Các nguyên lý áp dụng trong phương pháp này bao gồm:

Nguyên lý vét cạn toàn bộ: Đây là phương pháp đơn giản nhất, liệt kê tất cả các trường hợp có thể xảy ra.

Nguyên lý mắt lưới: Vận dụng nguyên lý này để sàn lọc ra những trường hợp nào phù hợp với các điều kiện đưa ra, giống như lưới bắt cá chỉ bắt được những con các có kích thước lớn hơn kích thước của mắt lưới.

Nguyên lý thu gọn không gian tìm kiếm: Loại bỏ những trường hợp hoặc nhóm trường hợp chắc chắn không dẫn đến lời giải.

## **1.6 Ý nghĩa lý luận và thực tiễn của đề tài**

Tình hình nghiên cứu drone trong một đất nước còn khó khăn lạc hậu đã và đang gặp nhiều khó khăn. Vì vậy, việc điều khiển thành công một máy bay không người lái, hoạt động tự động thành công sẽ giải quyết được nhiều vấn đề trong xã hội và đón đầu được công nghệ mới. Nước ta sẽ không còn phải nhập khẩu các drone từ nước ngoài, cũng như có thể phát triển trong các hệ thống lớn hơn.

Kết quả của đề tài cho thấy tính khả thi trong nghiên cứu, mở ra nhiều định hướng cho sinh viên khoa Kĩ Thuật Máy Tính nói riêng cũng như sinh viên ngành kĩ thuật nói chung, góp phần thúc đẩy nhiều đề tài nghiên cứu hơn nữa cho khoa học.

# **CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

## **Giới thiệu mô hình máy bay Quadcopter**

### **2.1.1 Sơ lược về cấu tạo**

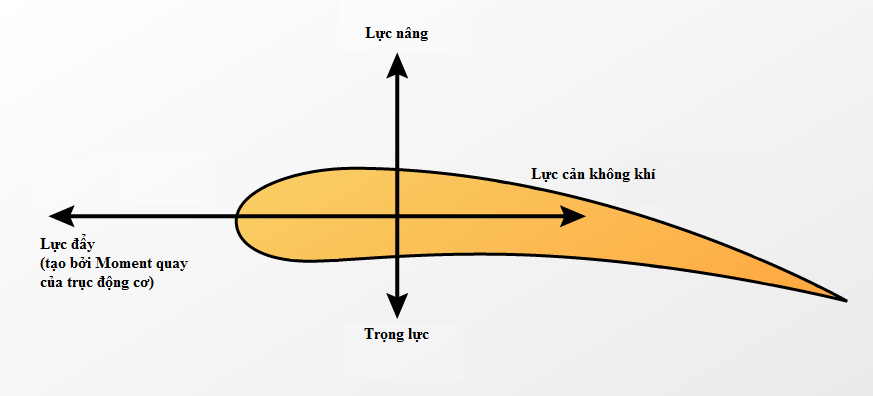
Quadcopter là một thiết bị bay lên theo phương thẳng đứng, với bốn động cơ có gắn cánh quạt được bố trí lần lượt tại bốn đỉnh của một khung chữ thập đối xứng.



# Hình 2.1 Mô hình Quadcopter

Cơ chế bay của Quadcopter đơn giản hơn các loại thiết bị bay khác như máy bay cánh bằng, máy bay trực thăng…Ở Quadcopter, bốn cánh quạt quay với tốc độ cao sẽ tạo ra lực nâng, khi lực nâng này lớn hơn trọng lực tác dụng lên Quadcopter thì nó sẽ bay lên, vận tốc góc của cánh quạt càng lớn, lực nâng cành mạnh thì Quadcopter bay lên nhanh hơn. Khi lực nâng được tạo ra cân bằng với trọng lực tác dụng lên Quadcopter thì nó sẽ giữ độ cao hiện tại.

Lực nâng được tạo ra từ quạt khi quay còn gọi là lực nâng Joukowski [3], là kết quả của sự chênh lệch áp suất không khí tại mặt trên và mặt dưới của vật thể (cánh quạt) khi dòng khí chuyển động bao vật thể. Bốn cánh quạt được chia làm hai loại với chiều quay khác nhau – cùng hoặc ngược chiều kim đồng hồ, trong đó hai cánh đối diện cùng chiều quay, hai cánh cận nhau ngược chiều quay.



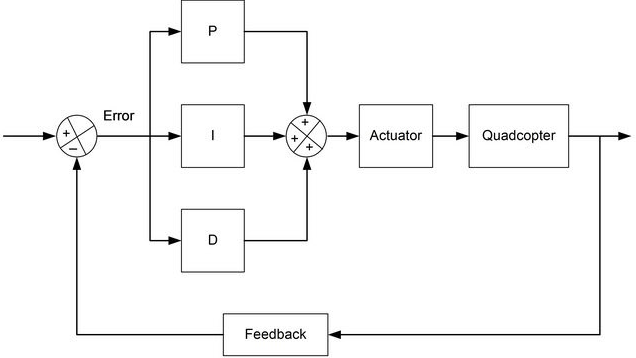
# Hình 2.2 Các lực tác dụng lên Quadcopter

### **2.1.2 Sơ lược về nguyên lý bay**

Do thiết kế đối xứng, với trọng tâm đặt tại chính giữa mô hình, cách đều các cánh quạt nên khoảng cách cánh tay đòn từ các động cơ đến trọng tâm là như nhau, do đó cơ chế thăng bằng của Quadcopter giống như bập bênh, góc nghiêng theo một trục phụ thuộc độ chênh lệch vận tốc giữa hai động cơ đối diện được bố trí trên trục đó. Nguyên lý di chuyển trong không gian của Quadcopter được giải thích như sau:

* Giữ nguyên vị trí hiện tại của Quadcopter: phân bố lực nâng trên bốn cánh quạt bằng nhau và tổng hợp lực cân bằng với trọng lực tác dụng lên Quadcopter.
* Quadcopter bay lên – xuống theo phương thẳng đứng: khi muốn bay lên ta cần tăng đồng thời lực nâng của cả bốn cánh quạt, lực nâng được phân bố đều nhau trên bốn cánh quạt và ngược lại trong trường hợp hạ xuống.
* Thực hiện di chuyển tới, lui, qua trái, qua phải trong mặt phẳng ngang: nếu Quadcopter bay về phía trước, khi đó cần giữ nguyên lực nâng của cặp cánh quạt trái phải, tăng lực nâng của cánh quạt sau, đồng thời giảm lực nâng của cánh quạt trước nhưng vẫn đảm bảo tổng lực nâng của bốn cánh quạt không thay đổi và cân bằng với trọng lực để chuyển trạng thái cân bằng cho Quadcopter.
* Thực hiện xoay tại chỗ: khi xoay Quadcopter trong mặt phẳng ngang theo chiều kim đồng hồ thì tổng moment lực theo chiều kim đồng hồ lớn hơn tổng moment ngược chiều kim đồng hồ, đồng thời vẫn đảm bảo lực nâng của hai cánh quạt đối diện bằng nhau và tổng lực nâng của bốn cánh quạt cân bằng với trọng lực tác dụng lên Quadcopter.

## **2.2 Tìm hiểu giải thuật PID cho Quadcopter**



# Hình 2.3 Sơ đồ giải thuật điều khiển kinh điển PID

Trong đó:

* PID là cách viết tắt của các từ Propotional(tỉ lệ), Integral (tích phân), Derivative (đạo hàm) và là giải thuật điều khiển được dùng nhiều nhất trong các ứng dụng điều khiển tự động với yêu cầu chính xác, nhanh, ổn định.
* Gain P: giúp Quadcopter giữ cân bằng tốt, phản ứng chính xác lệnh điều khiển.
* Gain I: giữ cho các thao tác roll, loop ổn định, gain I chỉ cần set vừa đủ đến khi roll, loop ổn định và mượt là ngưng.
* Gain D: Sau khi đã chỉnh xong P và I, đến lúc chỉnh gain D để Quadcopter có độ dừng lệnh như mong muốn.

## **2.3 Tìm hiểu về PWM**

Điều chế độ rộng xung (Pulse Width Modulation) là một kĩ thuật kiểm soát power,

được sử dụng để kiểm soát lượng điện đi vào động cơ và do đó biết được động cơ

quay nhanh hay chậm như thế nào.

Xung là các trạng thái cao/thấp (HIGH/LOW) về mức điện áp được lặp đi lặp lại. Đại lượng đặc trưng cho 1 xung PWM bao gồm tần số (frequency) và chu kì xung (duty cycle). Tần số là số lần lặp lại trong 1 đơn vị thời gian. Đơn vị tần số là Hz, tức là số lần lặp lại dao động trong 1 giây. Dao động được xác định từ trạng thái bắt đầu và kết thúc ngay trước khi trạng thái bắt đầu được lặp lại.

## **2.4 Bộ lọc Kalman**

### **2.4.1 Khái niệm chung về bộ lọc**

Bộ lọc (filter) là một quá trình xử lý nhằm loạị bỏ những gì không có giá trị hoặc không quan tâm đến và giữ lại những gì có giá trị sử dụng.

Trong xử lý tín hiệu, bộ lọc được thiết kế để lọc tín hiệu cần tìm từ trong tínhiệu trộn lẫn giữa tín hiệu sạch và nhiều tín hiệu bẩn (không cần thiết). Tuy nhiên, cũng cần phải chú ý rằng không có bộ lọc nào có thể lọc hoàn toàn các tín hiệu nhiễu. Vậy nên, các bộ lọc cũng chỉ lọc được tương đối các tín hiệu sạch, không thể tuyệt đối được.

Nếu có tín hiệu S (signal) trộn lẫn với nhiễu N (noise) trong một tín hiệu tổng hợp X và ta cần lọc để loại bỏ N ra khỏi X.

|  |  |
| --- | --- |
|  | **(1)** |

Nhiễu N dao động xung quanh 0 và có giá trị trung bình là 0.

|  |  |
| --- | --- |
|  | **(2)** |

Với trường hợpkhi M đủ lớn.

Để loại bỏ N, ta có thể lấy tổng của X trên một cửa sổ có kích thước M.

|  |  |
| --- | --- |
|  | **(3)** |

Từ các bước trên, N đã được loại bỏ.

### **2.4.2 Bộ lọc Kalman**

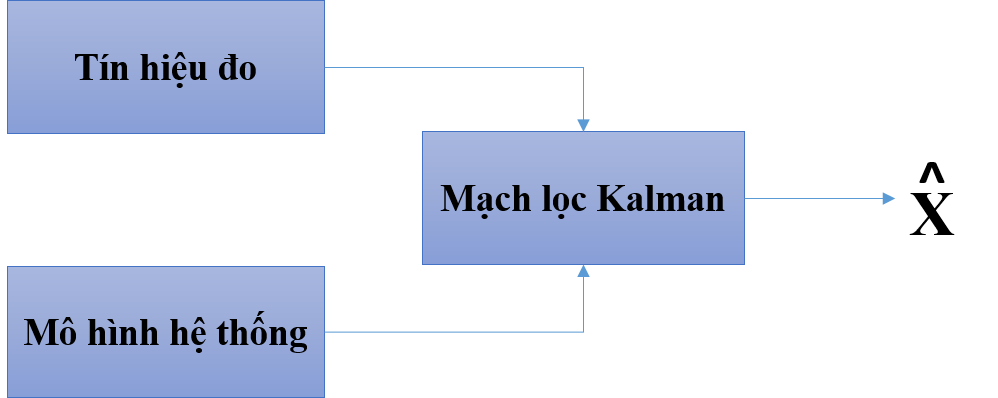
Một cách khái quát, bộ lọc Kalman là một tập hợp các phương trình toán học

mô tả một phương pháp tính toán truy hồi hiệu quả cho phép ước đoán trạng thái của một quá trình (process) sao cho trung bình phương sai của độ lệch.



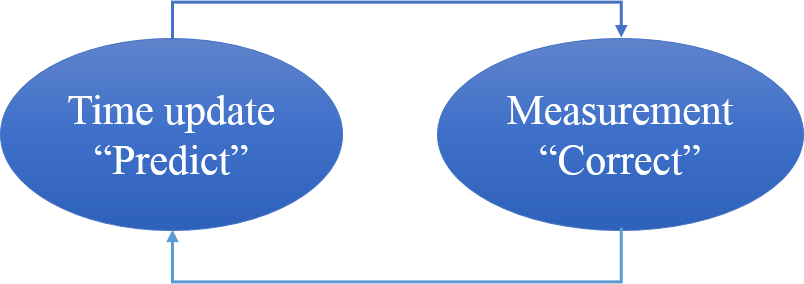
# Hình 2.4 Sơ đồ bộ lọc Kalman

Bộ lọc Kalman rất hiệu quả trong việc ước đoán các trạng thái trong quá khứ, hiện tại và tương lai thậm chí ngay cả khi tính chính xác của hệ thống mô phỏng không được khẳng định.



# Hình 2.5 Bản chất của bộ lọc Kalman

Khi cài đặt bộ lọc Kalman, hiệp phương sai của lỗi đo lường R thường được đo đạc trước để điều khiển bộ lọc hoạt động.



# Hình 2.6 Hoạt động của bộ lọc Kalman

Đo đạc hiệp phương sai của lỗi đo lường R là hoàn toàn khả thi vì chúng ta cần đo

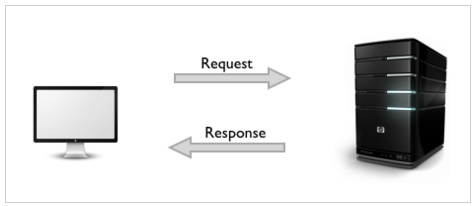
lường quá trình bằng mọi cách (trong khi bộ lọc hoạt động), vì vậy, chúng ta nên đo đạc một số mẫu trước (off-line) để xác định phương sai của sai số đo.

Tóm lại, mạch lọc Kalman bao gồm 2 bước :

* Ước đoán trạng thái tiên nghiệm
* Dựa vào kết quả đo để hiệu chỉnh lại ước đoán.

## **2.5 Giao thức HTTP**

Giao thức HTTP là viết tắt của Hyper Text Transfer Protocol (Giao thức tải siêu văn bản) dựa trên mô hình client - server (request – response). Giao tiếp thường diễn ra qua giao thức TCP/IP với port mặc định là 80, nhưng các port khác cũng có thể được sử dụng.



# Hình 2.7 Giao thức HTTP

## **2.6 Uniform Resource Locator**

* URL được dùng với mục đích tham chiếu đến những tài nguyên trên mạng Internet. Khả năng siêu liên kết của các trang mạng có được là nhờ URL.
* Các thành phần của URL:
* Tên giao thức: http hoặc https, ngoài ra thì còn có ftp, mailto...
  + Tên miền.
  + Port chỉ định (có thể có hoặc không)
  + Đường dẫn trên máy chủ chứa tài nguyên.
* World Wide Web được xây dựng dựa trên HTTP và HTML (HyperText Markup Language - Ngôn ngữ đánh dấu siêu văn bản).
* Thông tin trong HTTP bao gồm:
* Request URL
* Header
* Body

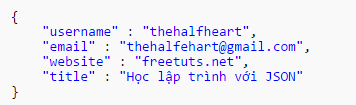
## **2.7 Mã Hash MD5**

MD5 (Message - Digest algorithm 5) là một bộ tạo mảng băm mật mã được sử dụng phổ biến với giá trị dài 128-bit. Chúng thường được biểu diễn bằng một số hệ thập lục phân 32 ký tự. 1 dữ liệu nào đó khi băm MD5 thì sẽ cho ra một mã MD5 duy nhất và đường đi là một chiều (tức là băm từ dữ liệu ra mã MD5 chứ không thể dịch ngược lại).

Mã MD5 thường được dùng trong bảo mật, kiểm tra sự toàn vẹn của dữ liệu khi chia sẻ trên Internet (tránh trường hợp file bị thay đổi, bị hacker chèn mã độc vào file). Nếu kết quả check giữa dữ liệu đưa vào và dữ liệu sau khi đã được tạo ra là giống nhau thì file hoặc data được đảm bảo toàn vẹn, nếu khác thì file đã bị thay đổi, có thể trong quá trình download bị lỗi.

## **2.8 Định dạng JSON**

JSON (Javascript Object Notation) là một dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. Nó có thể được lưu vào một file hoặc một record trong CSDL dễ dàng. JSON có định dạng đơn giản, dễ dàng sử dụng và truy vấn hơn XML rất nhiều nên tính ứng dụng của JSON hiện nay rất phổ biến.



# Hình 2.8 Chuỗi JSON

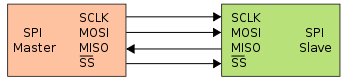
Cấu trúc chuỗi JSON:

* Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}.
* Các *key, value* của JSON bắt buộc phải đặt trong dấu nháy kép {"}, nếu đặt nó trong dấu nháy đơn thì đây không phải là một chuỗi JSON đúng chuẩn.
* Nếu có nhiều dữ liệu (nhiều cặp *key* => *value*) thì dùng dấu phẩy (,) để ngăn cách.
* Các *key* của JSON nên đặt chữ cái không dấu hoặc số, dấu \_ và không có khoảng trắng, ký tự đầu tiên không nên đặt là số.

## **Chuẩn truyền thông SPI**

### **2.9.1 Khái niệm**

SPI ([[6]](#footnote-6)) (Serial Peripheral Bus) được phát triển bởi Motorola. Đây là một chuẩn đồng bộ nối tiếp để truyền dữ liệu ở chế độ song công toàn phần (full- duplex) tức trong cùng một thời điểm có thể xảy ra đồng thời quá trình truyền và nhận. Đôi khi SPI còn được gọi là chuẩn giao tiếp 4 dây (Four-wire).



# Hình 2.9 Giao diện SPI

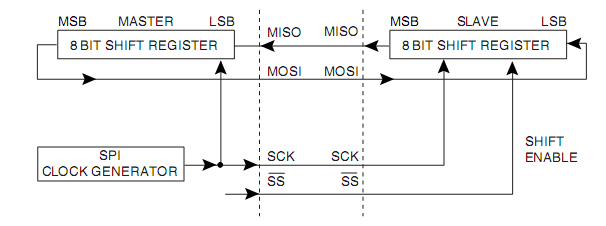
SPI là giao diện đồng bộ, bất cứ quá trình truyền nào cũng được đồng bộ hóa với tín hiệu clock chung. Tín hiệu này sinh ra bởi master.

Trong giao diện SPI có bốn tín hiệu số:

* MOSI hay SI – cổng ra của bên Master (Master Out Slave In). Đây là chân dành cho việc truyền tín hiệu từ thiết bị chủ động đến thiết bị bị động.
* MISO hay SO – Công ra bên Slave (Master IN Slave Out). Đây là chân dành cho việc truyền dữ liệu từ Slave đến Master.
* SCLK hay SCK là tín hiệu clock đồng bộ (Serial Clock). Xung nhịp chỉ được tạo bởi Master.
* CS hay SS là tín hiệu chọn vi mạch (Chip Select hoặc Slave Select). SS sẽ ở mức cao khi không làm việc. Nếu Master kéo SS xuông thấp thì sẽ xảy ra quá trình giao tiếp.

### **2.9.2 Hoạt động**

Mỗi chip Master hay Slave có một thanh ghi dữ liệu 8 bits. Cứ mỗi xung nhịp do Master tạo ra trên đường giữ nhịp SCK, một bit trong thanh ghi dữ liệu của Master được truyền qua Slave trên đường MOSI, đồng thời một bit trong thanh ghi dữ liệu của chip Slave cũng được truyền qua Master trên đường MISO. Do 2 gói dữ liệu trên 2 chip được gởi qua lại đồng thời nên quá trình truyền dữ liệu này được gọi là “song công”.



# Hình 2.10 Truyền dữ liệu SPI

Cực của xung giữ nhịp, phase và các chế độ hoạt động: cực của xung giữ nhịp (Clock Polarity) được gọi tắt là CPOL là khái niệm dùng chỉ trạng thái của chân SCK ở trạng thái nghỉ. Ở trạng thái nghỉ (Idle), chân SCK có thể được giữ ở mức cao (CPOL=1) hoặc thấp (CPOL=0). Phase (CPHA) dùng để chỉ cách mà dữ liệu được lấy mẫu (sample) theo xung giữ nhịp. Dữ liệu có thể được lấy mẫu ở cạnh lên của SCK (CPHA=0) hoặc cạnh xuống (CPHA=1). Sự kết hợp của SPOL và CPHA làm nên 4 chế độ hoạt động của SPI.

## **2.10 Chuỗi NMEA và RMC**

Chuỗi NMEA: là chuẩn cho các chip thu GPS trả ra tọa độ cho các thành phần khác nhau. Có nhiều dạng chuỗi NMEA, tùy vaò từng chuỗi để lọc ra các thông số cần thiết như kinh độ, vĩ độ, độ cao…

**

# Hình 2.11 Cấu trúc bản tin theo giao thức NMEA

Cấu trúc chuỗi NMEA:

* Mỗi chuỗi bắt đầu bằng ký tự $.
* 5 ký tự đầu tiên biểu hiện loại của chuỗi. 2 ký tự đầu định nghĩa thiết bị sử dụng chuỗi. 3 ký tự tiếp theo xác định ý nghĩa của chuỗi trả về.
* Các trường dữ liệu nối tiếp nhau bởi dấu “,”.
* Khi trường dữ liệu cuối cùng kết thúc, ký tự tiếp sau nó là dấu “\*”.
* Sau dấu “\*” là 2 ký tự số checksum miêu tả 1 số hex. Checksum là 1 phép XOR giữa tất cả các ký tự từ dấu “,” đến dấu “\*”. Checksum không bắt buộc đối với hầu hết các câu dữ liệu nhưng bắt buộc đối với RMA, RMB và RMC.
* Chuỗi kết thúc với ký tự.
* Ví dụ:

*$GPGBS,235458.00,1.4,1.3,3.1,03,,-21.4,3.8\*5B*

*$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,8,1.01,499.6,M,48.0,M,,0\*5B*

Trong phạm vi của đề tài này, chúng ta sử dụng RMC theo chuẩn giao thức NMEA. Chuỗi RMC cung cấp dữ liệu về vị trí, vận tốc, thời gian GPS. Ví dụ:

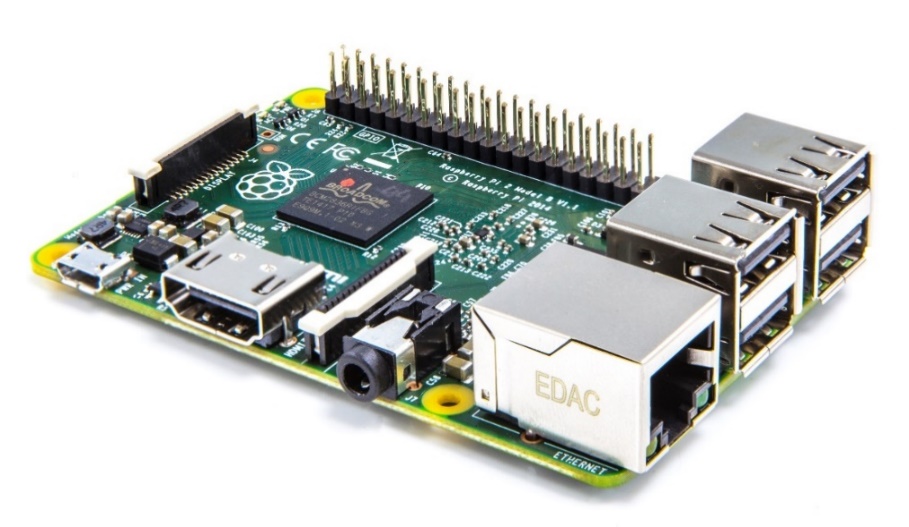
*$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W\*6A*

Trong đó:

* RMC: Recommended Minimum Sentence C
* 123519: nhận tín hiệu lúc 12:5:19 UTC
* A: A=active hoặc V=inValid. Active nghĩa là tại thời điểm hiện tại cường độ tín hiệu của ít nhất 3 vệ tinh đủ mạnh để tính toán được vị trí dưới mặt đất. inValid nghĩa là không tìm tính toán được vị trí hiện tại dưới mặt đất.
* 4807.038,N: Vĩ độ : 48 độ 07.038' N
* 01131.000,E: Kinh độ : 11 độ 31.000' E
* 022.4: vận tốc bề mặt tính theo hải lí (Speed)
* 084.4: (Course) góc chỉ hướng dựa theo vành chân trời thông tin này được biểu diễn là một "azimuth".
* 230394: Ngày tháng - 23 tháng 3 1994
* 003.1,W: độ biến thiên từ trường
* \*6A: dữ liệu tổng hợp "checksum" luôn bắt đầu với kí tự \*.

## **2.11 Linh kiện được sử dụng trong quá trình thực hiện đồ án**

### **2.11.1 Sơ lược về Raspberry Pi 2 Model B**

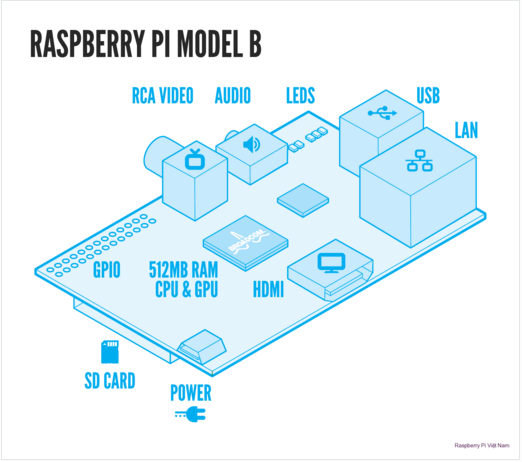


# Hình 2.12 Kit Raspberry Pi 2 model B

RASPI được trang bị một bộ xử lý Broadcom lõi tứ ARM Cortex A7 900MHz, với 1GB RAM, do đó mang lại hiệu xuất cao hơn gấp 6 lần so với Raspberry Pi Model B+. Ngoài ra, RASPI còn có khả năng tương tác với nhiều thiết bị, được sử dụng trong hàng loạt các dự án kỹ thuật số như thiết bị nghe nhạc, parent detectors, trạm thời tiết…

Thông số kĩ thuật của RASPI:

* Broadcom BCM2836 ARMv7 Quad Core Processor trợ Single Board
* CPU 900MHz, 1GB RAM
* 40 chân GPIO mở rộng
* 4 cổng USB
* Cổng ra Audio và cổng Composite video 4 cực
* 01 cổng HDMI
* CSI – cổng camera kết nối Raspberry Pi Camera
* DSI – cổng hiển kết nối màn hình cảm ứng Raspberry Pi
* 01 cổng Micro SD cho hệ điều hành, nguồn điện Micro USB

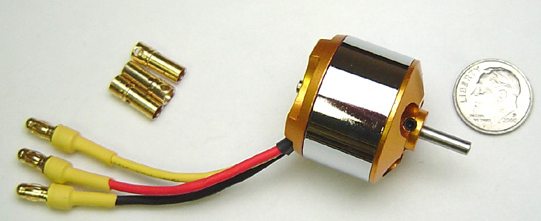


# Hình 2.13 Sơ đồ cấu tạo

### **2.11.2 Motor 1000kV**

Thông số kĩ thuật:

* Kv: 1000
* Tối đa hiệu quả: 80%
* Hiệu quả hiện tại: 4-10a (> 75%)
* Công suất: 12a/60s
* Động cơ kích thước: φ27.5 x 27mm
* Đường kính trục: φ3.17mm



# Hình 2.14 Motor 1000kV

### **2.11.3 Bộ điều tốc ESC**

Thông số kĩ thuật:

* Output: 30A
* Điện áp đầu vào: 5.6V - 16.8V / 2S-4S (7.4V-14.8V) BEC (Battery Loại bỏ Circuit) Output: 2A, Linear Mode BEC
* Tốc độ tối đa: 210,000rpm 35,000rpm 12 cực BLM. (BLM = Brushless Motors)
* Kích thước: 24mmx45mmx11mm (width x length x height)

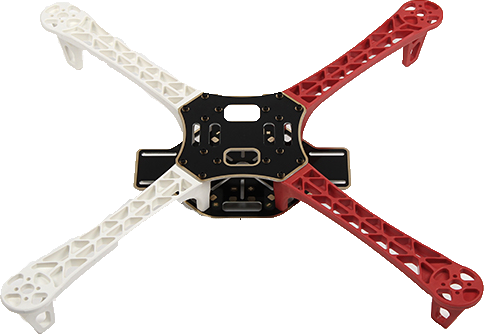


# Hình 2.15 Bộ điều tốc ESC

### **2.11.4 Khung máy bay**

Thông số kĩ thuật:

* Chiều dài cơ sở: 330mm
* Sản xuất từ sợi thủy tinh chất lượng và polyamide nylon
* Trọng lượng: 155g.



# Hình 2.16 Khung máy bay

### **2.11.5 Pin Lipo**

Thông số kĩ thuật:

* Công suất: 1500 mAh
* Tỷ lệ xả liên tục: 25c
* Tốc độ bùng nổ: 40c
* Điện áp: 22.2V



# Hình 2.17 Pin Lipo

### **Cảm biến MPU 6050**

MPU-6050 là cảm biến của hãng InvenSense. MPU-6050 là một trong những cảm

biến chuyển động đầu tiên trên thế giới có tới 6 (mở rộng tới 9) trục cảm biến tích

hợp trong 1 chip duy nhất.

MPU-6050 tích hợp 6 trục cảm biến bao gồm:

* Con quay hồi chuyển 3 trục (3-axis MEMS gyroscope)
* Cảm biến gia tốc 3 chiều (3-axis MEMS accelerometer)

Ngoài ra, MPU-6050 còn có 1 đơn vị tăng tốc phần cứng chuyên xử lý tín hiệu (Digital Motion Processor - DSP) do cảm biến thu thập và thực hiện các tính toán cần thiết. Điều này giúp giảm bớt đáng kể phần xử lý tính toán của vi điều khiển, cải thiện tốc độ xử lý và cho ra phản hồi nhanh hơn. Đây chính là 1 điểm khác biệt đáng kể của MPU-6050 so với các cảm biến gia tốc và gyro khác.

Các thông số kĩ thuật khác của module MPU-6050:

* Nguồn: 3-5V, trên module MPU-6050 đã có sẵn LDO chuyển nguồn.
* Giao tiếp I2C ở mức 3V
* Khoảng cách chân cắm: 2.54mm
* Địa chỉ: 0x68, có thể cấp mức cao vào chân AD0 để chuyển địa chỉ thành 0x69



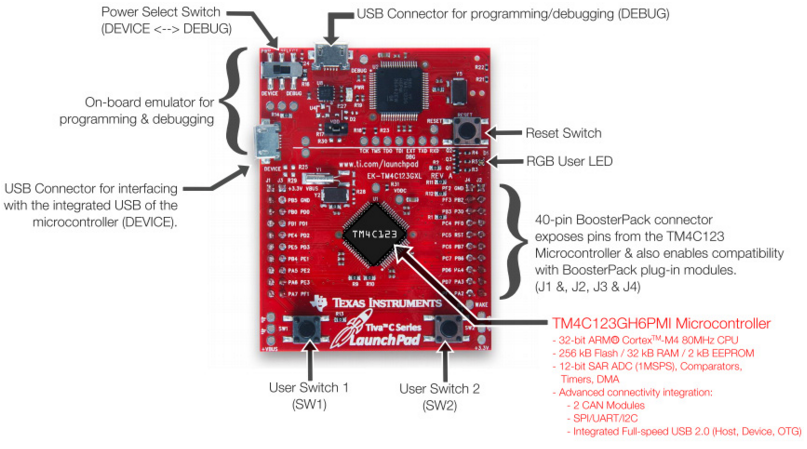
Hình 2.18 Cảm biến MPU 6050

### **Kit Tiva Launchpad 123**

Kit Tiva Launchpad 123sử dụng chip ARM TM4C123GH6PM với lõi ARM Cortex M4 mạnh mẽ. Trên KIT đã bao gồm mạch nạp và MCU và một số ngoại vi đơn giản như nút bấm, LED, giao tiếp USB.

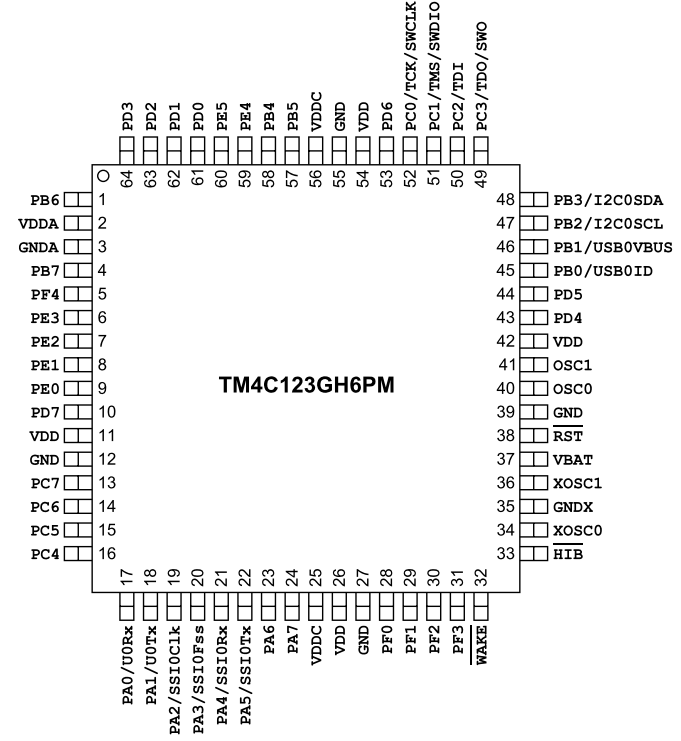
Thông số kĩ thuật của **Chip TM4C123GH6PM:**

* CPU: 32bits ARM Cortex M4
* Thumb2 16/32-bit code
* Flash: 256KB, SRAM: 32KB, EEPROM: 2KB
* Tốc độ: Up to 80MHz
* PWM: 16 chanels
* ADC: 12bits – 12 chanels
* Giao tiếp: SSI/SPI, I2C, UART, USB, CAN



Hình 2.19 Tổng quan về Kit Tiva LaunchPad 123

Kit Tivi hỗ trợ băm xung tốt, thường được sử dụng để điều khiển các động cơ vừa và nhỏ.



# Hình 2.20 Dạng đóng gói của Kit Tiva

* + 1. **Mạch GPS SiRF III tích hợp Angten**

Mạch GPS SiRF III tích hợp Angten có kích thước nhỏ gọn, giá thành hợp lý.

Mạch có thiết kế chắc chắn, vỏ sắt chống nhiễu cho độ ổn định và độ bền cao nhất.

Mạch GPS SiRF III sử dụng dòng chip SiRF III của hãng SiRF Technology cho độ ổn định cao, khả năng bắt sóng mạnh và ổn định, thích hợp cho các ứng dụng định vị với chi phí thấp.

Thông số kĩ thuật:

* Nguồn điện sử dụng: 3.3 – 5VDC
* Dòng tiêu thụ: 50 mA
* Chuẩn giao tiếp UART TTL, baudrate default 9600, optional
* Xung clock chuẩn: 0.25 Hz – 1 KHz



# Hình 2.21 Mạch GPS SiRF tích hợp Angten

### **2.11.9 Mạch thu phát RF NRF24L01**

Module RF24L01([[7]](#footnote-7)) hoạt động ở tần số sóng ngắn 2.4G, khả năng truyền dữ liệu tốc độ cao và truyền nhận dữ liệu trong điều kiện môi trường có vật cản. Modul nRF24L01 có 126 kênh truyền giúp việc truyền nhận dữ liệu trên nhiều kênh khác nhau. Modul có khả năng thay đổi công suất phát bằng chương trình, vì vậy có thể hoạt động trong chế độ tiết kiệm năng lượng. Điện áp thường cung cấp là 3.3V nhưng các chân IO tương thích với chuẩn 5V.

Thông số kĩ thuật:

* Điện áp hoạt động: 1.9 – 3.6V
* Công suất phát: 0, -6, -12, -18dBm
* RF hoạt động: Dải tần 2.4G, 126 kênh. Truyền tốc độ cao 1Mbps hoặc 2Mbps.
* Giao tiếp: 4 pin SPI, tốc độ tối đa 8Mbps. 3 – 12 byte trên một khung truyền.



# Hình 2.22 Mạch thu phát RF NRF24L01

### **Mạch giảm áp DC LM2596**

Mạch giảm áp DC nhỏ gọn, có khả năng giảm áp từ 35V xuống 1.5V mà vẫn đạt hiệu suất cao (92%), thích hợp cho các ứng dụng chia nguồn, hạ áp, cấp cho các thiết bị như camera, motor, robot…



# Hình 2.23 Mạch giảm áp DC LM2596

Thông số kĩ thuật:

* Điện áp đầu vào: 3 – 40V
* Điện áp đầu ra: 1.5 – 35V
* Dòng đáp ứng: 3A
* Công suất: 15W
* Hiệu suất: 92%

### **Mạch giảm áp AMS1117**



# Hình 2.24 Mạch giảm áp AMS1117

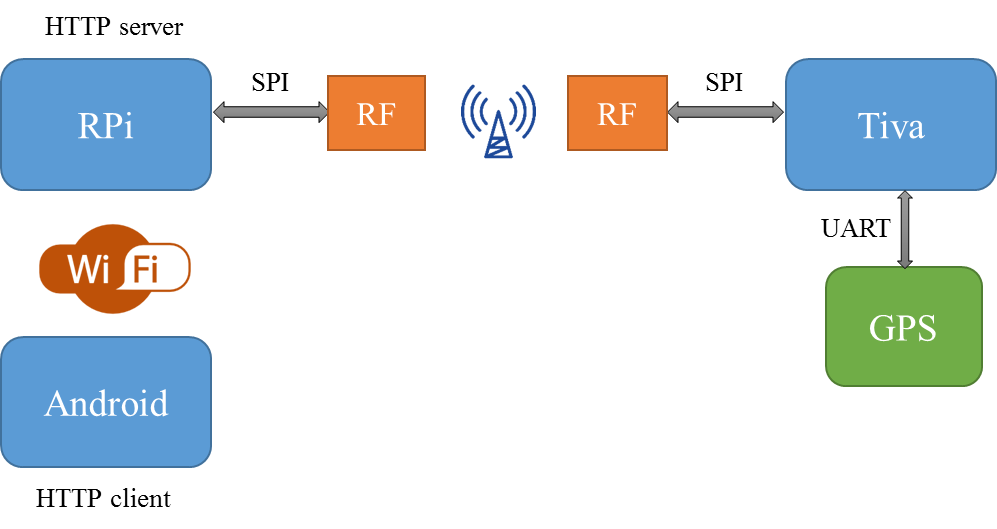
Thông số kĩ thuật:

* Điện áp đầu vào: 4.5 – 7V
* Điện áp đầu ra: 3.3 – 6V
* Dòng đáp ứng: 800mA
* Kích thước: 2.5x1.1 cm

# 

# **CHƯƠNG 3. THIẾT KẾ VÀ HIỆN THỰC PHẦN MỀM ĐIỀU KHIỂN**

## **3.1 Nội dung hiện thực**



# Hình 3.1 Sơ đồ hệ thống

Hệ thống phần mềm điều khiển mà nhóm thực hiện gồm 3 khối chính (màu xanh dương trong hình 3.1):

* Điện thoại Android: đóng vai trò là HTTP client
* RASPI: đóng vai trò HTTP server
* Board Tiva – máy bay: cùng với các module khác như MPU 6050, mạch giảm áp DC LM2596, mạch giảm áp AMS1117, pin Lipo, ESC…, là phần cứng của hệ thống, đảm nhận vai trò vô cùng quan trọng là điều khiển các động cơ của máy bay.

## **3.2** **Cấu hình RASPI trở thành Access Point**

Access Point (điểm truy cập) ([[8]](#footnote-8)) cung cấp 1 ngõ truy cập cho client khi muốn kết nối vào WLAN. Đây là 1 thiết bị [half - duplex](https://vi.wikipedia.org/w/index.php?title=Half-duplex&action=edit&redlink=1), hoạt động tương đương như một [Switch](https://vi.wikipedia.org/wiki/Switch) Ethernet thông minh.

Cách cấu hình RASPI trở thành Access Point: trên Tera Term hoặc Terminal trên

Linux, thực hiện cấu hình cho RASPI trở thành access point thông qua các bước sau:

* Bước 1: Cài đặt các gói phần mềm cần thiết

sudo apt-get install hostapd udhcpd

* Bước 2. Cấu hình DHCP. Copy dòng lệnh sau vào file /etc/udhcpd.conf:

start 10.10.0.2 # This is the range of IPs that the hostspot will give to client devices.

end 10.10.0.50

interface wlan0 # The device uDHCP listens on.

remaining yes

opt dns 8.8.8.8 4.2.2.2 # The DNS servers client devices will use.

opt subnet 255.255.255.0

opt router 10.10.0.1 # The Pi's IP address on wlan0 which we will set up shortly.opt

lease 864000 # 10 day DHCP lease time in seconds

* Bước 3: Trong file /etc/default/udhcpd, comment dòng sau:

DHCPD\_ENABLED="no"

* Bước 4: Để boot một cách tự động thì trong file /etc/network/interfaces, tìm và thay thế dòng "iface wlan0 inet dhcp" thành dòng:

iface wlan0 inet static

address 10.10.0.1

netmask 255.255.255.0

Nếu không có dòng "iface wlan0 inet dhcp" thì thêm các dòng trên vào cuối file.

* Bước 5: Comment các dòng sau:

allow-hotplug wlan0

wpa-roam /etc/wpa\_supplicant/wpa\_supplicant.conf

iface default inet manual

* Bước 6: Configure thông tin Access Point, thêm các dòng sau vào file /etc/hostapd/hostapd.conf:

interface=wlan0

driver=nl80211

ssid=your\_access\_point\_name

hw\_mode=g

channel=6

macaddr\_acl=0

auth\_algs=1

ignore\_broadcast\_ssid=0

wpa=2

wpa\_passphrase=your\_access\_point\_password

wpa\_key\_mgmt=WPA-PSK

wpa\_pairwise=TKIP

rsn\_pairwise=CCMP

Sửa “ssid” để đặt tên cho Access point, sửa “wpa\_passphrase” để đặt password cho AP.

* Bước 7: trong file /etc/default/hostapd, sửa dòng sau:

#DAEMON\_CONF=""

Thành:

DAEMON\_CONF="/etc/hostapd/hostapd.conf"

* Bước 8: Configure NAT (Network Address Translation). Thêm các dòng sau vào cuối file /etc/sysctl.conf:

net.ipv4.ip\_forward=1

* Bước 9: Thêm các dòng sau vào cuối file /etc/network/interfaces:

up iptables-restore < /etc/iptables.ipv4.nat

* Bước 10: Khởi động access point:

sudo service hostapd start

sudo service udhcpd start

sudo update-rc.d hostapd enable

sudo update-rc.d udhcpd enable

* Bước 11: Khởi động lại Kit Raspberry Pi 2 Model B.

## **3.3 Hiện thực giao thức HTTP**

### **3.3.1 Client**

#### **a. Lấy thông tin của drone**

Drone có các thông tin cơ bản như: Drone Name, Total Flight Time, Motors Info, Network Info, Battery Info, Location Info…Client sẽ luôn gửi lên server theo giao thức HTTP Post với định dạng như sau:

* URI: http://<ip\_address>/api/info
* Request params: sig <string>

#### **b. Điều khiển drone thay đổi vị trí**

* URI: http://<ip\_address>/api/move
* Request params:
* x <integer>,
* y <integer>,
* z <integer>,
* sig <string>

Trong đó:

* *sig* (signature) là mã hash MD5 của tất cả các tham số (theo thứ tự alphabet) kèm với secret key (được xem như password, nhằm đảm bảo chỉ có thiết bị được cài đặt bởi app của mình mới có thể điều khiển drone). Trong trường hợp không có tham số nào thì sig chính là mã hash MD5 của secret key. Secret key sẽ luôn là:

"drone\_hung\_hue@uit.edu.vn".

* x là tốc độ (%) theo tọa độ x
* y là tốc độ (%) theo tọa độ y
* z là tốc độ (%) theo tọa độ y

### **3.3.2 Server**

#### **a. Định nghĩa giao thức cho server**

Bên server sẽ luôn response với định dạng JSON:

{

"error": <integer>,

"data": <JSONObject>

}

Trong đó:

* error là mã lỗi trả về, với các giá trị sau:
* 0: success
* -1: invalid params, được trả về khi client truyền thiếu tham số hoặc truyền giá trị tham số bị sai. Trường hợp client gửi nhiều tham số hơn quy định vẫn không sao. Vì sever sẽ check từng tham số được yêu cầu, nếu có thì xử lý tham số đó, không thì sẽ báo lỗi.
* -2: invalid signature, được trả về khi client truyền mã sig không đúng (server sẽ tính ra mã hash MD5 của các tham số được yêu cầu và so sánh giá trị đó với mã sig do client truyền lên nhằm đảm bảo tính an ninh).
* -3: no support, được trả về khi server không đáp ứng được yêu cầu từ client, có thể là do server gặp phải lỗi trong quá trình xử lý yêu cầu, cũng có thể server chưa cung cấp chức năng đó (ví dụ, server chưa cung cấp chức năng lấy thông tin của drone).
* data là dữ liệu trả về theo yêu cầu của client.
* Trong trường hợp “điều khiển drone”, server chỉ cần trả về mã lỗi, không có dữ liệu nào khác cần trả về nên data sẽ là null.
* Trong trường hợp “lấy thông tin của drone”, server trả về các thông số của drone. Tên từng thông số sẽ do bên server tự định nghĩa. Thông số trả về cũng do bên server quyết định nhưng phải đảm bảo là được phải trả về dưới dạng mã JSON. Khi đó, bên client sẽ đọc nội dung của field này và sẽ tự biết lấy thông tin cần thiết.

#### **b. Hiện thực HTTP server**

*Libmicrohttpd* là thư viện cung cấp api để hiện thực HTTP server trên linux.

Các bước cài đặt như sau:

* Download: <http://ftpmirror.gnu.org/libmicrohttpd/libmicrohttpd-0.9.49.tar.gz>
* Giải nén: *tar –xvf libmicrohttpd-0.9.49.tar.gz*
* Vào trong thư mục: *cd libmicrohttpd-0.9.49*
* Run trên terminal các command:

./configure

make

sudo make install

* Export library path:

LD\_LIBRARY\_PATH = /usr/local/lib/

Export $LD\_LIBRARY\_PATH

*OpenSSL* là thư viện dùng để tạo chuỗi MD5 trong project này.

sudo apt-get install libssl-dev

# Bảng 3.1 Cấu hình chân cho RASPI khi config HTTP server

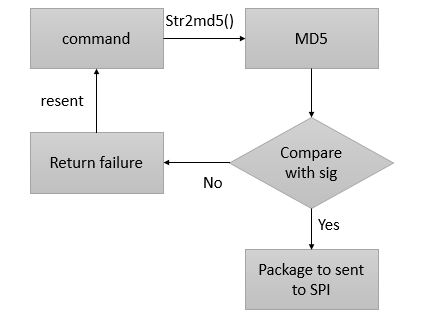
|  |  |
| --- | --- |
| UART | RASPI |
| GND | Port 6 |
| TX | Port 8 |
| RX | Port 10 |

### **3.3.3 Lập trình HTTP Server**

**a. Các hàm xử lí chính**

Hàm *spi\_data\_to\_send()*đưa ra dữ liệu cần thiết để gửi tới SPI slave. Trong hàm

này, data sẽ được convert từ các số sang mảng, sau đó sử dụng hàm *memset()* để lưu trữ data này rồi gửi đến client thông qua SPI. Quá trình truyền nhận được liên tục cho đến khi hết chuỗi.

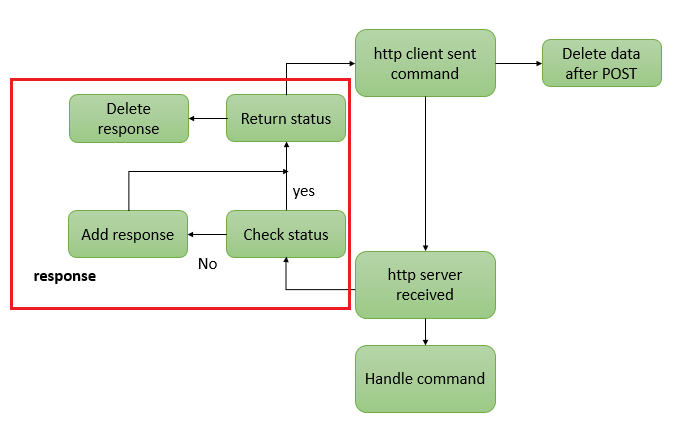


# Hình 3.2 Chuyển command sang format MD5

Hàm *Str2md5()* chuyển một chuỗi dữ liệu bất kì sang định dạng MD5, dùng trong quá trình kiểm tra thông tin truyền nhận có đầy đủ hay không. Nếu chuỗi data được gửi lên giống với chuỗi data sau khi đã qua convert thì dữ liệu nhận đã chính xác. Ngược lại, quá trình truyền dữ liệu không thành công.

Nhiệm vụ chính của server là phân tích thông tin cụ thể nhận từ request của client thông qua hàm *parse\_request()* bằng cách search các key là x, y, z, sig và &. Hàm *strstr()* trong C chuẩn sẽ trả về vị trí của key được tìm thấy lần đầu tiên. Từ đó sẽ đọc được giá trị của x, y, z và sig chính là chuỗi sau key.

Trong phạm vi của đồ án này, HTTP server chỉ support phương thức POST, vậy nên server sẽ không hỗ trợ cho những phương thức khác. Dữ liệu nhận từ client sẽ được xử lí và đóng gói trong hàm *process\_post\_data().*



# Hình 3.3 Quá trình gửi và nhận command

Trong hàm này, server sẽ kiểm tra coi command mà client gửi lên thuộc method gì, nếu là POST thì server tiếp tục xử lí, ngược lại sẽ báo về là phương thức trên không được hỗ trợ. Hàm này sẽ pasre tiếp data được lấy ra từ hàm *parse\_request()*, kiểm tra xem các thông số x, y, z, sig từ command gửi tới có đầy đủ không, so sánh với chuỗi data được phân tích ở *parse\_request()*, sau đó đóng gói dữ liệu để gửi xuống cho Tiva thông qua RF.

Để phân biệt các thông số x, y và z một cách dễ dàng, ở byte đầu của x\_buf[ ], tức là x\_buf[0] sẽ được gắn giá trị là 0x11, y\_buf[0] = 0x22 và z\_buf[0] = 0x33.

#### **b. Build và Run HTTP Server**

* Di chuyển vào thư mục chứa app server:

cd workspace/Quadcopter/app

* Tạo một HTTP server:

make

* Chuyển port server từ 80 sang 8080:

sudo iptables –A PREROUTING –t nat –p tcp –dport 80 –j REDIRECT –to-port 8080

* Run app:

./Quadcopter

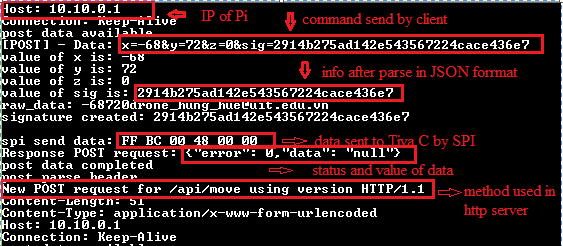
Hoặc sử dụng file script *startup.sh* để thay thế cho việc thực hiện tất cả các thao tác trên.

* Di chuyển vào file /etc/profile để chạy file startup.sh:

vi /etc/profile

* Thêm dòng sau vào cuối file:

/home/pi/startup.sh



# Hình 3.4 Hiện thực http server

Hình trên cho biết các thông tin cơ bản khi có một giao tiếp giữa client và server, đó là mã lỗi error và data. Chuỗi sig được xây dựng từ các thông số x, y, z và key để nhận biết như đã đề cập trước đó.

## **3.4 Hiện thực trên điện thoại Android**

### **Giao tiếp giữa RASPI và điện thoại Android**

* Khi có giao tiếp giữa RASPI và điện thoại Android thì RASPI sẽ luôn luôn ở chế độ Access Point.
* HTTP server boot và chờ để nhận command từ client gửi xuống.
* RASPI có nhiệm vụ là xử lý command, sau đó phản hồi status cho client biết là đã xử lý thành công hay chưa.
* Command từ client sẽ là data bao gồm các thông số x, y, z hoặc x, y, z, t, m, Lat, Long, tùy thuộc vào loại request. HTTP server sẽ parse thông tin, sau đó gửi cho Tiva MCU thông qua RF.

****

# Hình 3.5 Giao diện điện thoại Android khi chưa có kết nối

Trong đó:

1: Button để điều khiển hướng cho máy bay

2: Hiển thị thông tin tọa độ

3: Button kết nối với server

4: Button điều khiển độ cao

5: Button chuyển mode auto/manual

* Điều khiển động cơ: di chuyển button 1 để thay đổi thông số x, y; kéo button 4 để thay đổi thông số z (không cần giữ cố định).
* Sự thay đổi của 3 thông số x, y, z được thể hiện ở (2).
* Sau khi nhấn button 5, màn hình điện thoại chuyển sang chế độ auto.



# Hình 3.6 Giao diện điện thoại Android App khi kết nối server

* Nhấn button 3 để kết nối với HTTP server. Phím Start ban đầu là màu xanh lá, sau khi có kết nối sẽ chuyển thành phím Stop (màu đỏ).

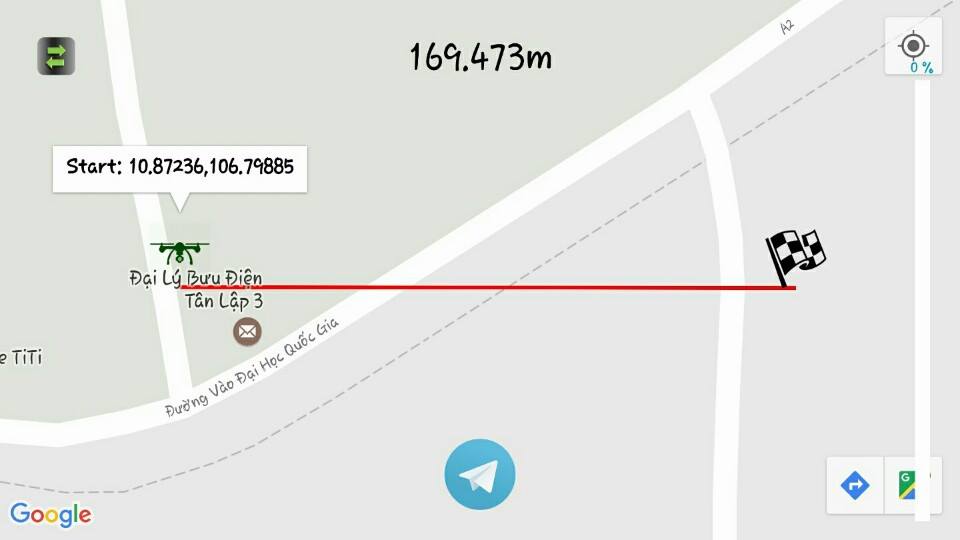


# Hình 3.7 Điện thoại mất kết nối server

* Nếu xảy ra việc mất kết nối giữa điện thoại và server, một hộp thông báo sẽ hiện lên, yêu cầu kiểm tra việc kết nối. Khi xảy ra hiện tượng này, hệ thống trên board Tiva sẽ nhanh chóng để máy bay reset các giá trị x, y, z để chờ request tiếp theo.
* Nếu các thông số x, y, z đưa ra không đúng hoặc việc chuỗi sig trả về không

phù hợp thì việc kết nối client – server cũng không thành do thông tin không chính xác.

* Khi chuyển sang chế độ auto, đầu tiên cần bật GPS trên điện thoại để máy bay có thể xác định được vị trí hiện tại. Sau đó, vị trí của máy bay được cập nhật liên tục sau mỗi giây tại từng vị trí bay.
* Khi chọn điểm đến trên bản đồ, tọa độ của điểm này sẽ được hiển thị ra.

****

# Hình 3.8 Giao diện điện thoại Android chế độ auto

* Ở giao diện này, màn hình thể hiện hai địa điểm của máy bay, đó chính là vị trí xuất phát (mô hình máy bay màu xanh) và vị trí cần bay tới (lá cờ). Trên điện thoại đồng thời cũng hiển thị khoảng cách giữa hai điểm này.

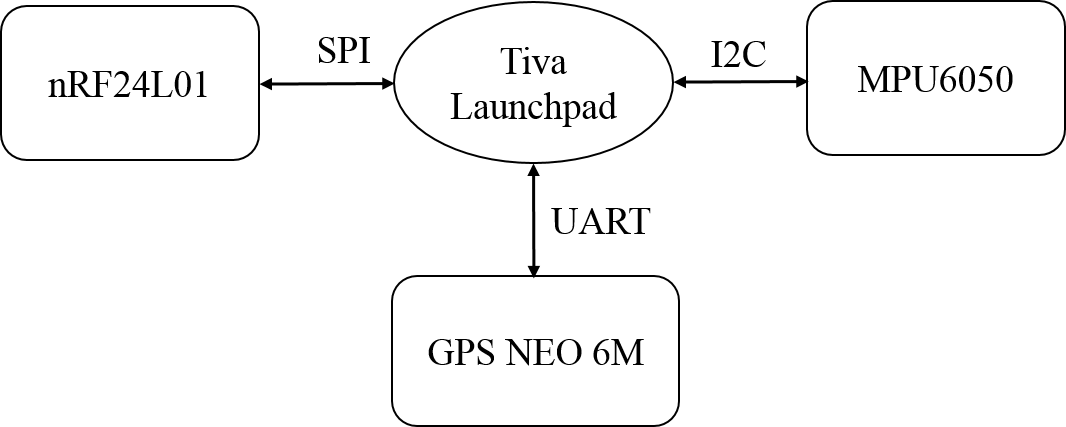
### **Các hàm xử lý chính**

* Một trong những phần quan trọng mà điện thoại Android xử lý chính là quá trình cho phép truyền/nhận dữ liệu giữa server và client. HTTP client kiểm tra trạng thái gửi command trước đó có thành công hay không. Cứ mỗi 10ms, HTTP client sẽ kiểm tra trạng thái của hai cờ *previousCommandSent* (trạng thái gửi trước đó có thành công hay không) và cờ *controlValueChanged* (giá trị conntrol hiện tại có thay đồi gì so với lần gửi trước hay không). Nếu lần gửi ngay trước đó là thành công (kiểm tra status của command trước đó) thì cờ *previousCommandSent* được bật lên là *true*, đồng thời nếu ở lần gửi sau, các thông số x hoặc y hoặc z được thay đổi giá trị thì cờ *controlValueChanged* cũng được bật lên *true*.
* Khi cả hai cờ trên đều được bật thì command sẽ được gửi tới server. Ngược lại, client bỏ qua việc gửi các giá trị control đã thay đổi lên server vào thời điểm đó cho đến khi đáp ứng đủ hai điều kiện trên. Một khi command được gửi thành công thì lập tức hai cờ trên cũng bị reset. Quá trình kiểm tra và gửi command là liên tục.
* Sau mỗi lần client gửi thành công thì đồng thời server cũng phải trả về trạng thái của command nhận được là thành công (*Successful)*, hay thông tin không chính xác (*IncorrectInfo),* hay là không thể kết nối được (*UnableToConnect).*
* Việc cập nhật các thông số x, y và z được thực hiện qua hàm *updateControl()*. Hàm này sẽ so sánh giá trị được đưa ra bởi client so với giá trị được nhóm mặc định trước để đưa ra con số phù hợp. Giá trị của x và y sẽ nằm trong khoảng (-99 -> 100). Giá trị z nằm trong khoảng (0->100).
* Việc đưa bản đồ vào App Android được thực hiện qua hàm *onMapReady()*. Nhờ được mở GPS ngay khi chuyển sang chế độ auto mà trên bản đồ sẽ thẻ hiện ngay vị trí hiện tại của máy bay.

## **3.5 Hiện thực trên Board Tiva và mô hình Quadcopter**

### **Giao tiếp giữa RASPI và Board Tiva**

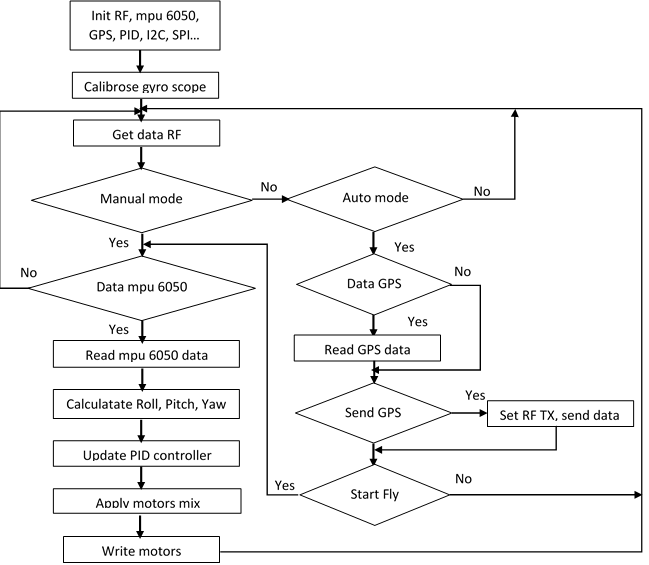
* Do phạm vi đề tài là máy bay có thể bay auto đến một điểm chọn trước trên bản đồ nên yêu cầu khoảng cách giữa server và máy bay là lớn. Nếu chỉ đơn thuẩn sử dụng giao thức SPI thì không được. Để giảm tải cho máy bay và tiện lợi cho quá trình di chuyển, RASPI sẽ được đặt cố định, còn sensor GPS sẽ được gắn trên thân máy bay cùng với board Tiva. Điều đó đồng nghĩa với việc sử dụng 1 board trung gian truyền và nhận dữ liệu giữa server và Tiva khi ở khoảng cách xa.
* Qua tìm hiểu và có sự cố vấn của giảng viên hướng dẫn, nhóm thực hiện đã quyết định sử dụng module RF nRF24L01.



# Hình 3.9 Mô hình tổng quát phần cứng của máy bay

* Khi đó, RASPI và Tiva đều đóng vai trò là SPI master. Sau khi nhận request data từ client thì server sẽ parse và đóng gói data theo format là 14 byte hoặc 10 bytes (tùy thuộc vào từng loại request).
* Ở mô hình tổng quát trên, việc không có board RASPI trên máy bay giúp năng lượng được tiếp kiệm hơn nên máy bay có thể bay được lâu hơn. Hơn nữa, việc cố định sóng wifi sẽ giúp cho các thiết bị smartphone điều khiển máy bay có được lượng sóng luôn ở mức cao và sẽ tránh được việc mất dữ liệu điều khiển.
* Trong mô trình trên, kit tiva Launchpad được xem là đầu não, bởi vì nó đảm nhận việc xử lý giải thuật PID và nhận tín hiệu đầu vào là cảm biến gyro và accelometer MPU6050. Khi có được tín hiệu về các góc nghiêng và con quay hồi chuyển, kit tiva sẽ tính toán số liệu cho giải thuật PID, điều chỉnh các motor để giữ cân bằng cho máy bay.
* Việc kết nối giữa kit tiva Launchpad với sensor MPU 6050 được thực hiện qua giao thức I2C. Kết nối này quan trọng hơn cả vì chỉ cần có sai sót về phần cứng làm cho việc kết nối dẫn đến việc máy bay không giữ được cân bằng và có thể rơi ngay lập tức.
* Ở chế độ tự động, module GPS và nRF24L01 có vai trò vô cùng quan trọng. Khi có dữ liệu từ module GPS NEO-6M, tiva sẽ xử lý và gửi tín hiệu qua module nRF24L01 cho server. Cũng thông qua module nRF24L01, server sẽ nhận tín hiệu và gửi các tín hiệu từ người dùng sử dụng smart phone cho tiva. Có thể thấy các module có vai trò quan trọng như nhau và không thể tách rời.
* Trên máy bay còn có các board mạch chủ và các mạch nguồn nuôi. Board mạch chủ có vai trò điều khiển, kết nối tất cả các module và cấp tín hiệu cho các ESC để điều khiển các motor. Các mạch nguồn được thiết kế có thể chịu được áp lớn lên tới 30V, điều này giúp cho nguồn nuôi board phong phú và dễ dàng hơn cho việc lựa pin cho máy bay. Trên board, nguồn được thiết kế cho các cổng 5V và 3.3V để có thể cung cấp được cho các thiết bị sử dụng nguồn 5V như GPS, board tiva và 3.3V như nRF24L01…

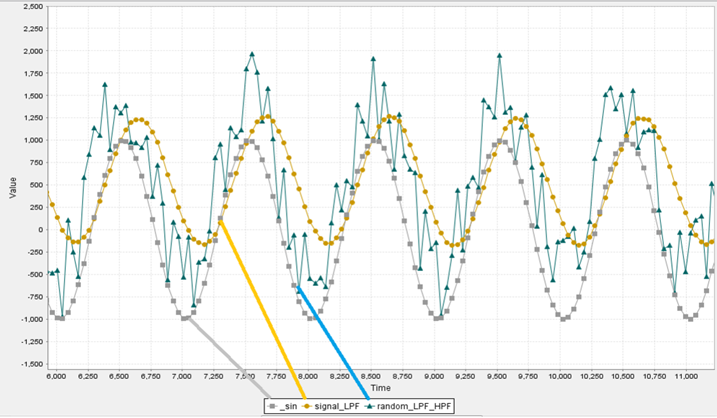
### **3.5.2 Các hàm xử lý chính**



# Hình 3.10 Sơ đồ giải thuật điều khiển máy bay

* Đầu tiên thì board tiva cần khởi tạo các module như RF, MPU6050, UART… để mọi thứ sẵn sàng hoạt động trước khi vào các chương trình chính.
* Thông thường trước khi máy bay được cất cánh, ta cần căn chỉnh lại thiết bị đo góc và gia tốc. Các thiết bị đo góc nghiêng và gia tốc MPU6050 thường có sai số nên khi thiết lập và căn chỉnh sử dụng thêm giải thuật Low Pass Filter để có thể lọc được các tín hiệu nhiễu thấp, đồng thời sử dụng thêm bộ lọc kalman để có thể lọc được các tín hiệu nhiễu cao và trung. Khi thiết bị đo góc và gia tốc đã sẵn sàng làm việc, lúc này máy bay mới được phép cất cánh.
* Chế độ điều khiển bằng tay và chế độ tự động đều sử dụng chung giải thuật PID và đều cho các tín hiệu điều khiển vào bộ điều chế để điều khiển motor. Điểm khác biệt ở đây chính là việc quản lý dữ liệu điều khiển. Trong chế độ điều khiển ta chỉ cần xét và lấy các tín hiệu để điều khiển các trục x, y, z để điều khiển máy bay lên thẳng hay rẽ trái, phải mà không cần quan tâm việc phải gửi các tín hiệu cập nhật GPS. Ở chế độ tự động, sau mỗi giây server sẽ yêu cầu gửi các tín hiệu cập nhật vị trí, đồng thời, server sẽ gửi tín hiệu về độ cao cho phép bay và điểm cần đến. Máy bay sẽ phải tự động dò đường và tới điểm đích đó.

### **Các hàm xử lý dữ liệu và điều khiển Quadcopter**

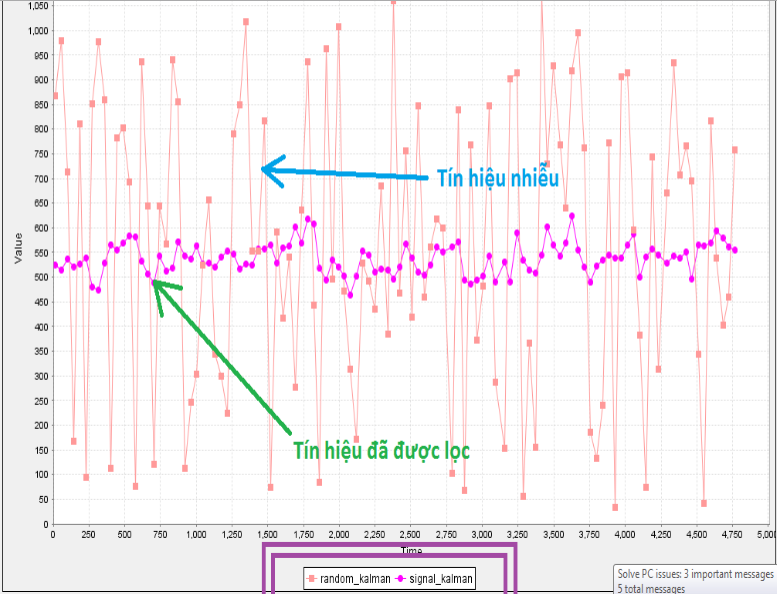


Hình 3.11 Tín hiệu sử dụng mạch lọc thông thấp

* Việc quan trọng trong xử lý điều khiển Quadcopter là giữ được thăng bằng cho Quadcopter khi bay và khử nhiễu từ các sensor (cảm biến góc nghiêng và gia tốc).
* Để có thể khử được tín hiệu nhiễu về góc đầu vào, nhóm đã sử dụng bộ lọc thông thấp (Low pass filter). Bộ lọc thông thấp là bộ lọc chỉ cho phép tín hiệu hoạt động từ 0 Hz đến tần số cắt (cut off) của nó, các tín hiệu tần số cao sẽ được lọc đi.
* Hình trên cho thấy tín hiệu sử dụng mạch lọc thông thấp. Tín hiệu màu xám là tín hiệu đầu vào, màu xanh là tín hiệu tham chiếu lọc nhiễu và màu vàng là tín hiệu đầu ra. Ta có thể thấy, tín hiệu thấp đã được bỏ đi và kết quả ta sẽ được một tín hiệu màu vàng, trơn tru và dễ kiểm soát hơn.
* Trong các quá trình lọc nhiễu, chúng ta không thể tránh được sai lệch và việc xử lý thêm đầu ra dữ liệu là hết sức cần thiết và nhóm đã thêm vào bộ lọc Kalman để có thể giải quyết bài toán trên. Công thức chuẩn của bộ lọc Kalman:

|  |  |
| --- | --- |
|  | **(4)** |
|  | **(5)** |
|  | **(6)** |
|  | **(7)** |
|  | **(8)** |

* Trong thư viện kalman.c, hàm *kalman\_single()* gồm 1 đầu vào và 1 đầu ra; hàm *kalman()* gồm 3 đầu vào và 3 đầu ra, với z là tín hiệu nhiễu cần được lọc nhiễu.
* Sau quá trình sử dụng bộ lọc Kalman, nhóm nhận thấy tín hiệu đầu ra đã ổn định. Quadcopter rất nhạy với nhiễu, chỉ cần có nhiễu nhỏ cũng có thể làm cho nó rung chuyển nên việc thêm bộ lọc làm các chuyến bay của nó mượt mà hơn.
* Các thông số cấu hình cho bộ lọc như sau:
* Cấu hình sample rate là 1 KHz.
* Disable FSYNC sau đó set cho bộ lọc Gyro là 41 Hz.
* Accelerometer Full Scale: + - 8g
* Gyro Full Scale Range: + 2000 deg/s.
* Acc filtering: 41 Hz.



# Hình 3.12 Tín hiệu đã được lọc nhiễu

* Động cơ được điều khiển thông qua các ESC và ta chỉ cần cấp các xung PWM cho chúng. PWM được cấu hình để điều khiển ESC với tần số 400Hz ứng với mỗi chu kì là 2,5 ms. Cấu hình các chân GPIO tương ứng hỗ trợ băm xung: GPIO\_PIN\_4, GPIO\_PIN\_5, GPIO\_PIN\_6 và GPIO\_PIN\_7
* Cấu hình PWM generator:

PWMGenConfigure (PWM0\_BASE, PWM\_GEN\_0,

PWM\_GEN\_MODE\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

PWMGenConfigure (PWM0\_BASE, PWM\_GEN\_1,

PWM\_GEN\_MODE\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

### **3.5.4 Cân chỉnh PID để giữ cho máy bay cân bằng**

* Nếu có sự hỗ trợ từ trang thiết bị hiện đại, ta có thể dùng những giải thuật tiên tiến để căn chỉnh PID cho máy bay như Auto tuner hoặc Ziegler – Nichols.

# Bảng 3.2 Bảng tham chiếu Ziegler – Nichols

|  |  |  |  |
| --- | --- | --- | --- |
| Bộ điều khiển | Kp | Ki | Kd |
| P | 0.50Ku | - | - |
| PI | 0.45Ku | 1.2Kp/Tu | - |
| PID | 0.60Ku | 2Kp/Tu | Kp.Tu/8 |

Các giá trị Kp, Ki, Kd có ảnh hưởng rất lớn với nhau.Trong đó giá trị Kp là có thay

đổi rõ rệt nhất, và Kd làít ảnh hưởng nhất.

# Bảng 3.3 Tác động khi điều chỉnh các thông số PID

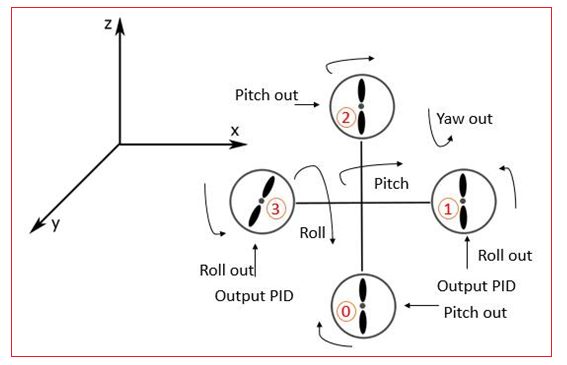
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Thông số | Thời gian khởi động | Quá độ |  | Thời gian xác lập | Sai số ổn định | Độ ổn định |
| Kp | Giảm | Tăng |  | Thay đổi | Giảm | Giảm cấp |
| Ki | Giảm | Tăng |  | Tăng | Giảm đáng kể | Giảm cấp |
| Kd | Giảm ít | Giảm ít |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

Tuy nhiên, do nhiều hạn chế về trang thiết bị nên nhóm đã chọn cho mình các cân chỉnh PID theo phương pháp thủ công. Ở phương pháp này, đòi hỏi nhiều kinh nghiệm thực tế. Khi nào cần tăng P, khi nào cần tăng I hoặc D tất cả phụ thuộc vào kinh nghiệm và sự phán đoán nhanh nhạy.

Các bước mà nhóm đã vận dụng để cân chỉnh:

* Bước 1: Chọn Kp trước, thử bộ điều khiển P với đối tượng thật (hoặc mô phỏng), điều chỉnh Kp sao cho thời gian đáp ứng đủ nhanh, chấp nhận overshot nhỏ. Lúc này hệ thống sẽ dao động tuần hoàn.
* Bước 2: Thêm thành phần D để loại overshot, tăng Kd từ từ, thử nghiệm và chọn giá trị thích hợp. Steady state error có thể sẽ xuất hiện.
* Bước 3: Thêm thành phần I để giảm steady state error. Nên tăng Ki từ bé đến lớn để giảm steady state error đồng thời không để cho overshot xuất hiện trở lại.

Với frame “+” thì việc cân chỉnh PID dễ dàng hơn. Việc gây ra lỗi xác lập theo một trục chỉ do một motor nên việc điều chỉnh dễ dàng hơn. Tuy nhiên với frame này thì việc cân bằng lại đơn giản nhưng Quadcopter dễ bị trôi và khó có thể kiểm soát tốt.



# Hình 3.13 Cân chỉnh PID theo frame “**+**”

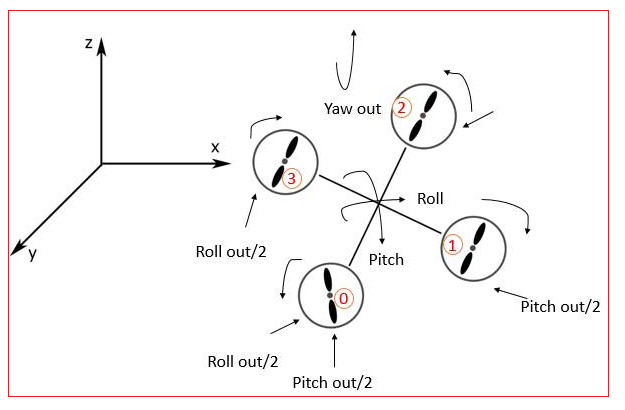
* Công thức cân chỉnh ở frame này như sau:

Gọi M là vận tốc bay được. Ta có: **(9)**

* Tune PID:

|  |  |
| --- | --- |
|  | **(10)** |
|  | **(11)** |
|  | **(12)** |
|  | **(13)** |

Để tránh việc bị trôi và khó có thể kiểm soát Quadcopter thì có một cách khác đó là thiết kế lại khung frame. Ta có thể thiết kế Quadcopter theo frame “X”. Tuy nhiên việc này dẫn tới việc thiết kế bộ PID khó khăn hơn và phức tạp hơn. Việc gây ra lỗi xác lập do cả 2 motor cùng phía gây nên và việc thiết kế bộ PID sẽ phải tác động đến cả 2 motor gây ra lỗi cùng lúc. Tuy nhiên, việc tìm được bộ PID chính xác sẽ rất khó khăn và giá trị của bộ PID sẽ được san đều cho cả 2 motor để tránh vọt lố trong quá trình điều chỉnh. Ở frame này, các momen xoắn được phân bổ đều, chính vì vậy việc giảm trôi tốt hơn so với frame “+” và việc điều khiển hướng cũng dễ dàng hơn.



# Hình 3.14 Cân chỉnh PID frame “X”

* Công thức tính toán PID ở frame này như sau:

Gọi M là vận tốc bay được.

Ta có:

|  |  |
| --- | --- |
|  | **(14)** |

* Tune PID:

|  |  |
| --- | --- |
|  | **(15)** |
|  | **(16)** |
|  | **(17)** |
|  | **(18)** |

1. **Cân chỉnh giá trị tỷ lệ P**

* Để có thể cân chỉnh được, nhóm đã lập trình cho kit Tiva có thể truyền qua UART các thông số của hệ thống từ đó lập biểu đồ để điều chỉnh từ từ. Việc căn chỉnh P ảnh hưởng tới việc đáp ứng mạnh hay chậm của Quadcopter.



# Hình 3.15 Chỉnh P

* Đầu tiên, nhóm đã thiếp lập giá trị P ở mức 0.005 và tăng từ từ giá trị này với hệ số là 0.001. Điều chỉnh thủ công theo từng trục, cần cố định 1 trục và dùng tay để cảm nhận sự thay đổi của Quadcopter sau mỗi lần tăng P. Một giá trị P đẹp khi mà đáp ứng nhanh, vọt lố không quá nhiều. Sau khi thử nghiệm với rất nhiều hệ số và kiểm nghiệm thực tế, nhóm đã chọn ra cho mình những giá trị tối ưu nhất mà P có thể đáp ứng tốt và vọt lố không quá nhiều.

1. **Cân chỉnh giá trị vi phân D**

* Việc chọn giá trị D có vẻ nhẹ nhàng hơn so với P, thậm chí nếu P và I làm việc đủ tốt có thể không cần tới giá trị của D. Tuy nhiên, có thêm D sẽ làm cho hệ thống ổn định cao hơn và thường D chỉ cần thêm vào một chút. Nhóm đã thiết lập giá trị D ban đầu là 0,0001 và tăng với bước nhảy là 0,0001.
* Giá trị D rất khó có thể cảm nhận sự thay đổi nên việc chọn D sao cho hợp lý sẽ rất khó khăn. Nếu D lớn sẽ làm cho Quadcopter đáp ứng chậm vì D kìm hãm lại P. Nhóm đã điều chỉnh D bằng cách chỉ để ý phần vọt lố ban đầu do P gây ra còn các giá trị sau sẽ rất khó phát hiện sự thay đổi của P. Một con số D đẹp sẽ làm giảm vọt lố hết mức có thể.

1. **Cân chỉnh giá trị tích phân I**

* Có thể nói việc cân chỉnh giá trị I và P là quan trọng hơn cả khi căn chỉnh PID. Máy bay sẽ không thể bay trong môi trường khắc nghiệt như nhiều gió nếu không có giá trị I.
* I có vai trò cho việc giữ Quadcopter giảm trôi và Quadcopter sẽ phải được “khóa cứng” trong không khí nếu gặp được một giá trị I tốt. Nếu giá trị I thấp thì Quadcopter không giữ được thăng bằng, giá trị I cao sẽ gây ra sai số xác lập khá lớn nên với bước nhảy của I, nhóm đã thiết lập giá trị ban đầu ở 0,0005 và tăng lên từ từ.

1. **Các con số thực nghiệm**

* Sau quá trình điều chỉnh các thông số thực nghiệm, sử dụng nhiều bộ số khác nhau, nhóm thực hiện đề tài đã đưa ra được các con số tốt nhất cho 3 bộ PID cho trục Roll, Pitch và Yaw như sau:

# Bảng 3.4 Căn chỉnh trục Roll

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thông số | Giá trị đo | | | | | | |
| P | 0.35 | 0.345 | 0.324 | 0.16 | 0.32 | 0.315 | 0.31 |
| I | 1.5 | 1.65 | 1.55 | 1.45 | 1.7 | 1.72 | 1.75 |
| D | 0.001 | 0.005 | 0.0055 | 0.0045 | 0.003 | 0.00035 | 0.0004 |

Qua các lần đo để căn chỉnh giá trị trục Roll, bộ PID phù hợp nhất là

(0.32, 1.7, 0.003).

# Bảng 3.5 Căn chỉnh trục Pitch

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thông số | Giá trị đo | | | | | | |
| P | 0.325 | 0.345 | 0.4 | 0.46 | 0.32 | 0.315 | 0.037 |
| I | 1.45 | 1.65 | 1.505 | 1.75 | 1.7 | 1.72 | 1.65 |
| D | 0.004 | 0.0055 | 0.0053 | 0.0045 | 0.003 | 0.0035 | 0.00375 |

Giá trị căn chỉnh cho trục Pitch phù hợp nhất là (0.037, 1.65, 0.00375).

# Bảng 3.6 Căn chỉnh trục Yaw

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thông số | Giá trị đo | | | | | | |
| P | 0.37 | 0.71 | 0.75 | 0.79 | 0.8 | 0.85 | 0.87 |
| I | 6.0 | 6.2 | 5.7 | 5.6 | 5.61 | 5.65 | 5.67 |
| D | 0.001 | 0.0012 | 0.0025 | 0.003 | 0.0035 | 0.0022 | 0.002 |

Giá trị căn chỉnh cho trục Yaw phù hợp nhất là (0.087, 5.67, 0.002).

## **Đọc dữ liệu GPS**

GPS cho biết giá trị tọa độ của máy bay tại mỗi thời điểm được đo. Trong phạm vi đề tài này, chúng ta quan tâm đến hai giá trị longitude (kinh độ) và latitude (vĩ độ) của máy bay. Hai giá trị này sẽ được gửi vể điện thoại Android để xử lý đường đi trong quá trình bay tự động. Vì vậy, từ data mà GPS gửi về, ta chỉ cần quan tâm chuỗi RMC.

Về tốc độ lấy mẫu của GPS, nhóm đã cố gắng đưa ra biện pháp tối ưu nhất để khi máy bay bay nhanh, giá trị tọa độ lúc nhận là tọa độ thời điểm hiện tại hoặc tọa độ gần nhất đo được trước đó mà sai số không quá lớn. Mặc định cứ mỗi 1s, GPS sẽ gửi data về 1 lần thông qua cổng serial. Khi có tín hiệu request từ điện thoại Android, server sẽ quyết định xem có gửi data về client hay không.

UART trong chip Tiva hỗ trợ việc gửi dữ liệu tối đa 1024 byte và có thể nhận dữ liệu tối đa 128 byte nên cần phải cấu hình buffer cho Tiva.



# Hình 3.16 Thuật toán phân tích dữ liệu GPS

Trong thuật toán trên, khối *GPS GPRMC data analysis* có vai trò quan trọng nhất vì nó chứa thông tin bản tin GPRMC, nếu bản tin sai sẽ dẫn tới việc lấy dữ liệu vị trí bị sai và thuật toán luôn luôn không nhận dạng được vị trí của máy bay khi bay theo chế độ tự động. Khi đã có đầy đủ dữ liệu về vị trí ta sẽ tới thuật toán phân tích góc giữa điểm cần bay tới và vị trí của máy bay. Ở đây ta phân tích số liệu theo kinh độ và vĩ độ.

## **3.7 Giao tiếp RF**

Một yêu cầu cơ bản trong truyền nhận RF là địa chỉ và kênh truyền – nhận ở hai bên phải giống nhau. Nhóm đã tiến hành thiết lập cho hai bên truyền – nhận có cùng kênh 1 thông qua hàm *nrf24\_setChannel(0x00)* cũng như cấu hình cho rf radio qua các hàm:

nrf24\_init();

nrf24\_begin();

nrf24\_setAutoAck(1);

nrf24\_setRetries(15,15);

nrf24\_setDataRate(RF24\_1MBPS);

nrf24\_setPALevel(RF24\_PA\_MAX);

nrf24\_setCRCLength(RF24\_CRC\_16);

Đặt địa chỉ của kênh truyền là:

uint8\_t addresses\_trans[] = {0x05, 0x04, 0x03, 0x02, 0x01}

thì địa chỉ của kênh nhận là:

uint8\_t addresses\_recv[] = {0x01, 0x02, 0x03, 0x04, 0x05}

Kết nối chân cho các module như bảng sau:

# Bảng 3.7 Kết nối PIN giữa RASPI và RF

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RASPI | 1 | 25 | 15 | 19 | 21 | 23 | 24 |
| RF | 2 | 1 | 3 | 6 | 7 | 5 | 4 |

Tiến hành cài đặt thư viện librf24.so.1.0 cho RASPI như sau:

* Sử dụng git để download thư viện RF24 về:

git clone <https://github.com/stanleyseow/RF24.git>

* Di chuyển vào thư mục chứa thư viện librf24:

cd RF24/librf24-rpi/librf24

* Compile file bằng lệnh:

make

* Tiến hành cài đặt thư viện bằng các câu lệnh sau:

sudo make install

sudo ldconfig -v | grep librf librf24.so.1 -> librf24.so.1.0

cd ../examples

make

* Để chạy chương trình thì gõ lệnh:

sudo ./rpi-hub

hoặc lệnh:

sudo ./scanner

* Để truy cập vào spidev0.0 thì gõ lệnh:

sudo/root

Client - server giao tiếp với nhau qua format message chung gồm 2 phần như hình bên dưới:

. 

# Hình 3.17 Format message chung cho giao tiếp client – server

Trong đó:

* msg\_type: byte đầu tiên trong format message, phân biệt loại message là L (location) hay M (move).
* data: dữ liệu đi kèm tương ứng với từng loại message.

Với request dạng “/api/ move”, message là một mảng dài 14 byte, bao gồm 8 thông số.



# Hình 3.18 Format message khi lấy location

* M: quy định loại message là /api/move. Kích thước 1 byte
* x: thông số bay trái. Kích thước 1 byte
* y: thông số bay phải. Kích thước 1 byte
* z: thông số quy định tốc độ bay. Kích thước 1 byte
* t: quy định mode. Máy bay ở chế độ manual nếu t = 0 và sẽ ở chế độ auto nếu t = 1. Kích thước 1 byte
* m: byte quy định tín hiệu bay. Máy bay sẽ start fly nếu m = 1, ngược lại máy bay chưa start khi m = 0. Kích thước 1 byte
* Lat: latitude (vĩ độ). Kích thước 4 byte
* Long: longitude (kinh độ). Kích thước 4 byte

Khi đó, chuỗi JSON trả về có dạng:

{

"error": 0,

"data": {

Lat = xxx;

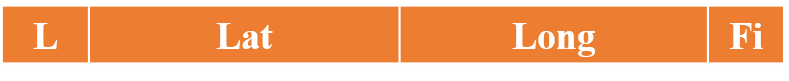
Long = yyy;

}

}

Tọa độ tại mỗi vị trí máy bay đi qua cần được gửi về cho điện thoại lưu lại trên bản đồ. URI này được sử dụng cả khi ở chế độ manual lẫn auto. Khi ở mode manual thì các thông số m, Lat, Long là không cần thiết.

Với request dạng “/api/location” thì format ngắn hơn, chỉ là 10 byte. Tuy nhiên, cuối message này có quy định 1 byte để kết thúc quá trình bay.



# Hình 3.19 Format message khi máy bay di chuyển

* L: quy định loại message là /api/location. Kích thước 1 byte
* Lat: latitude (vĩ độ). Kích thước 4 byte
* Long: longitude (kinh độ). Kích thước 4 byte
* Fi: byte quy định việc kết thúc bay. Với Fi = 0 thì quá trình bay chưa kết thúc, ngược lại thì quá trình bay kết thúc khi Fi = 1. Kích thước 1 byte

Khi đó, chuỗi JSON trả về có dạng:

{

"error": 0,

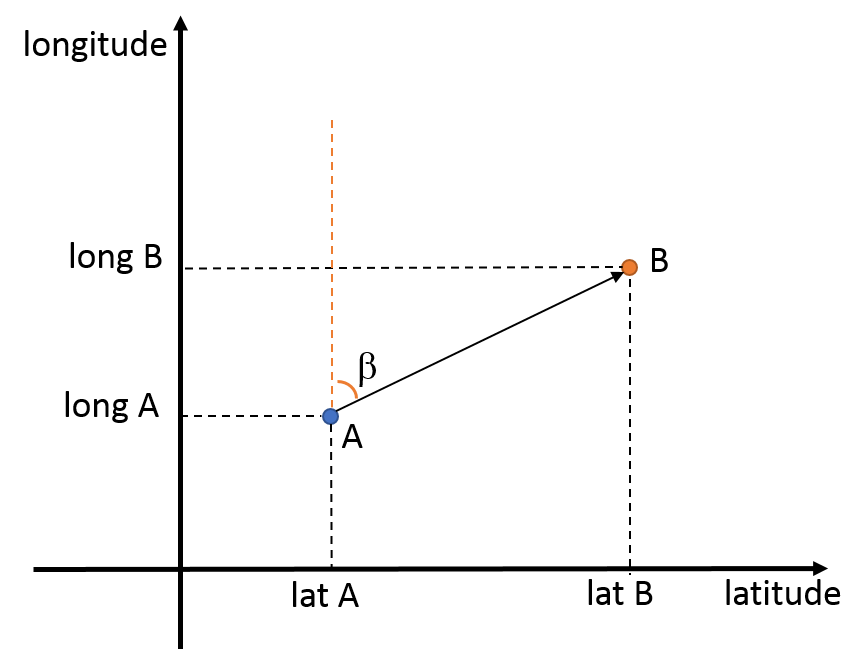
"data": NULL

}

Data của chuỗi NULL là do trong quá trình máy bay di chuyển thì việc hiển thị các thông số khi bay là không cần thiết. URI này cứ mỗi giây sẽ được request một lần. Bằng giao thức HTTP post, yêu cầu được gửi đến cho server sẽ đảm bảo nhất.

## **3.8 Lập trình bay thẳng cho máy bay**

Việc lập trình bay thẳng cho máy bay là công việc quan trọng của đề tài. Sau khi tìm hiểu một số giải thuật tính toán cũng như lập trình của các nghiên cứu trong và ngoài nước, nhóm thực hiện đã quyết định rõ các yếu tố cần thiết trong việc lập trình này, đó là phải xác định được góc giữa phương nối hai điểm xuất phát của máy bay và điểm cần bay tới so với phương bắc. Yếu tố thứ hai đó là cần tìm được khoảng cách giữa hai điểm này.



# Hình 3.20 Góc giữa phương nối hai điểm so với kinh tuyến

Việc đầu tiên, ta cần dời trục và chọn một trục nào đó làm trục cố định để có thể từ đó tính góc một cách chuẩn xác nhất. Ở đây nhóm đã chọn trục longitude làm trục chính vậy góc cần tính sẽ là góc β như hình trên.

Với d là khoảng cách giữa hai điểm (km) thì:

|  |  |
| --- | --- |
|  | **(19)** |

Trong đó:

|  |  |
| --- | --- |
|  | **(20)** |
|  | **(21)** |
|  | **(22)** |
|  | **(23)** |

Với β là góc bắc thật (góc giữa phương nối hai điểm 1 và 2 so với kinh tuyến) thì:

|  |  |
| --- | --- |
|  | **(24)** |

Với X, Y được xác định như sau:

|  |  |
| --- | --- |
|  | **(25)** |
|  | **(26)** |

Trong đó:

* R: bán kính trái đất = 6.371 km
* latA, latB: vĩ độ của điểm A và điểm B
* longA, longB: kinh độ của điểm A và điểm B

Các góc dùng để tính toán ở trên phải được đổi về đơn vị Radian.

Khi có hai thông số là khoảng cách giữa hai điểm đầu - cuối và góc được tạo bởi hai điểm đó so với phương bắc thì thuật toán có thể nói đơn giản là lập trình cho máy bay vừa xoay đầu vừa di chuyển tới điểm cuối với góc xoay đã xác định trước. Khi bay gần đến điểm cuối với khoảng cách tương đối cho phép thì máy bay sẽ tự động hạ cánh. Việc bay thẳng này phụ thuộc nhiều yếu tố như nguồn nuôi máy bay, độ chuẩn xác của GPS hay khả năng truyền nhận của RF. Vì vậy, cần có sự kết hợp hợp lý và chặt chẽ giữa các module với nhau.

# **CHƯƠNG 4. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ**

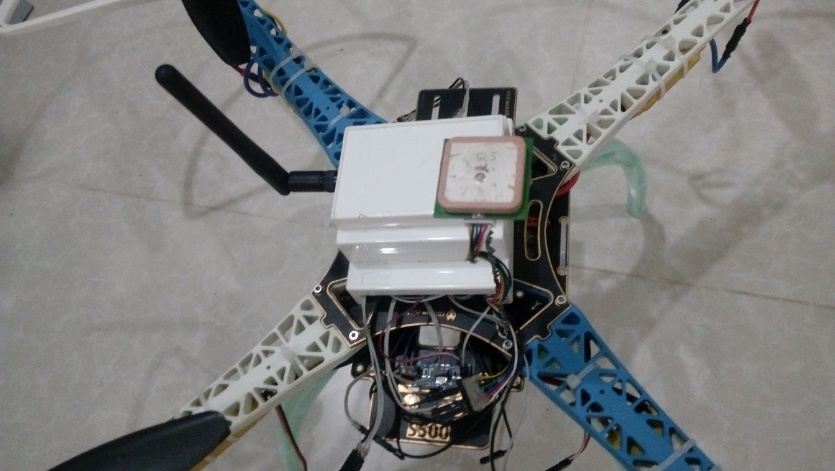
## **4.1. Lắp ráp và thiết kế khung cho Quadcopter**

Trong quá trình lắp ráp mô hình máy bay, hai phần vô cùng quan trọng là ESC và cảm biến.



# Hình 4.1 Lắp ráp ESC và motor

Bộ khung có thể lắp ráp thêm bộ bảo vệ cánh giúp cho việc bay an toàn hơn. ESC được thiết kết bên dưới đòn đỡ rất gọn gàng và đảm bảo việc sẽ không bị rối khi cánh quạt hoạt động.

****

# Hình 4.2 Vị trí đặt cảm biến

Các cảm biến được thiết kế nhỏ gọn và đặt vừa trên Flip đồng chính với 3 tầng chính. Ở giữa của tầng trên cùng được đặt module MPU-6050 để có thể cảm biến được góc nghiêng cân chỉnh PID. Nhóm có thiết kế thêm hộp riêng gắn trên Quadcopter để tránh nước và ánh nắng – nguyên nhân làm hỏng board điều khiển. Đồng thời, module nRF24L01 được đặt phía bên trong hộp và đầu Anten đặt ra ngoài để đảm bảo việc thu sóng. Tầng 2 đặt các board nguồn 5V và 3.3V, đảm bảo không xảy ra hiện tượng đoản mạch xảy ra trong quá trình bay. Tầng dưới cùng được lắp đặt 1 pin Lipo 12,6V 2200 cung cấp điện cho toàn bộ máy bay.



# Hình 4.3 Mô hình Quadcopter hoàn chỉnh

## **4.2 Kết quả thử nghiệm và đánh giá**

### **4.2.1 Kiểm tra xung và tốc độ các động cơ**

Để thử nghiệm hoạt động của động cơ motor và tần số xung PWM mà kit Tiva Launchpad tạo ra, nhóm đã sử dụng máy đo sóng ghi lại các tần số PWM và quan sát sự thay đổi tốc độ của cánh quạt.

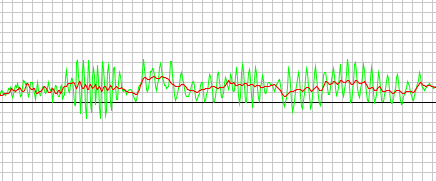


# Hình 4.4 Thử nghiệm đo xung PWM và tốc độ động cơ motor

Hình 4.4 cho thấy xung PWM ổn định, theo đúng số liệu mà nhóm đã tính toán.

### **4.2.2 Kalmal filter**

Bộ lọc Kalman đã lọc được một cách tương đối các nhiễu sinh ra từ cảm biến. Từ đó sai số trong quá trình tính toán của bộ điều khiển PID được giảm thiểu.



# Hình 4.5 Kết quả bộ lọc Kalman

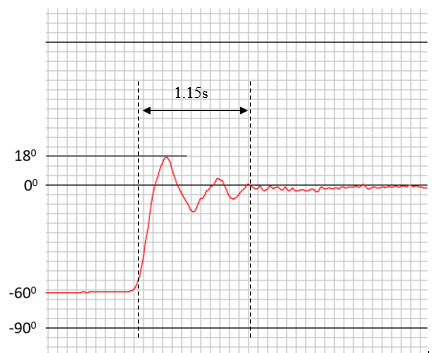
* Chú thích:

xanh: tín hiệu thô đọc về từ cảm biến

đỏ: tín hiệu đã đi qua bộ lọc Kalman

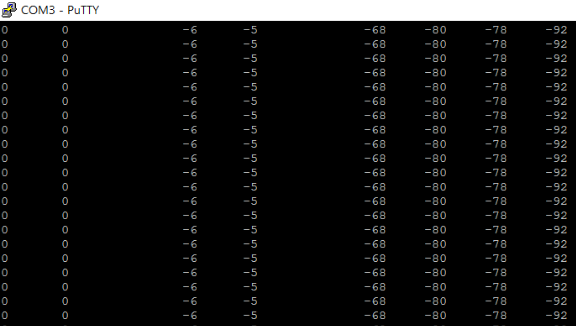
### **4.2.3 PID controller**

Đồ thị đáp ứng về biên độ độ nghiêng của bộ điều khiển PID:



# Hình 4.6 Đáp ứng biên độ PID

Thử nghiệm với điều kiện đầu vào là độ lệch khá lớn: 600. Hệ thống được tự động cân bằng về với trạng thái xác lập với thời gian tương đối nhỏ: khoảng 1.15 giây. Độ vọt lố tương đối thấp, đáp ứng được yêu cầu đề ra.

Để thử nghiệm bộ điểu chỉnh PID trên thực tế nhóm đã sử dụng thêm chức năng truyền dữ liệu căn chỉnh qua UART khi Quadcopter được cân chỉnh trên giá đỡ.

# Hình 4.7 Căn chỉnh PID thực tế

Ở hình trên, 2 cột đầu tiên thể hiện thông số Roll in, Pitch in, 2 cột tiếp theo trong hình trên cho thấy thông số Roll out và Pitch out. Đồng thời, 4 cột còn lại là tốc độ của motor 1, 2, 3 và 4. Các thông số trên ở các thời điểm khác nhau đều ổn định và không có sự sai lệch.

**4.3 Thử nghiệm bay thực tế**

Mô hình Quadcopter của nhóm đã bay thành công ở cả hai chế độ là điều khiển bằng tay và tự động. Ở chế độ điều khiển bằng tay, máy bay nhận thành công các tín hiệu điều khiển, bay đúng theo yêu cầu của người sử dụng. Ở chế độ tự động, máy bay bay đúng theo bản đồ lưu sẵn, đồng thời, phản hồi các thông tin cần thiết trên hệ thống.

# **CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **5.1 Thuận lợi và khó khăn**

### **5.1.1 Thuận lợi**

* Môi trường cài đặt chương trình phong phú, mã nguồn mở, tích hợp sẵn các gói thư viện và device cần thiết; dễ dàng tìm mua được linh kiện phù hợp trên thị trường.
* Là 1 đề tài đang được nhiều cá nhân và tổ chức quan tâm nên nguồn tài liệu tham khảo phong phú.
* Sử dụng các giao thức, thuật toán phổ biến trong quá trình lập trình và điều khiển mô hình máy bay.

### **5.1.2 Khó khăn**

* Do mô hình Quadcopter được thiết kế theo chữ “X” nên quá trình căn chỉnh PID phải thực hiện trên cả 3 trục Roll, Pitch và Yaw.
* Khi truyền dữ liệu qua RF thì tốc độ truyền thấp và không ổn định.
* Nguồn nuôi máy bay không đảm bảo khi máy bay hoạt động lâu.
* Kết hợp nhiều module với nhau nên dễ xảy ra tình trạng hỏng một phần dẫn đến tình trạng hệ thống.
* GPS sai số quá lớn dẫn đến việc sai lệch so với thực tế và làm cho máy bay khó có thể tính toán hướng để căn chỉnh.

### **5.1.3 Hướng giải quyết**

* Đối với việc căn chỉnh PID, nhóm phải thực nghiệm nhiều lần để có kết quả phù hợp và ổn định nhất. Đồng thời, hiệu chỉnh thuật toán PID kết hợp với cảm biến 3 trục MPU 6050.
* Kiểm tra tính ổn định của RF, sau đó thêm các điều kiện kiểm tra trạng thái truyền nhận để đưa ra giải pháp hợp lý.
* Sử dụng các mạch giảm áp để giảm nguồn cần nuôi các module liên quan. Thay thế nguồn hiện tại bằng nguồn có công suất lớn hơn để dự trữ năng lượng cho máy bay.
* Thu gọn các module với nhau bằng cách thiết kế mạch nguồn tích hợp các module đó.

## **5.2 Kết quả đạt được và chưa đạt được**

### **5.2.1 Kết quả đạt được**

* Sau quá trình thực hiện đề tài, nhóm đã cho ra đời một mô hình Quadcopter tương đối hoàn chỉnh, có thể bay được ở hai chế độ: điều khiển bằng tay hoặc tự động.
* Nhóm đã tích lũy thêm được kiến thức về lập trình C, C++. Nhóm cũng đã có cơ hội nghiên cứu và hiểu sâu hơn về các kiến thức về các MCU, CPU, Embedded System…Đồng thời, nhóm đã nghiên cứu và nắm bắt một cách tổng thể về máy bay không người lái nói chung và máy bay quadcopter nói riêng, nắm bắt được nguyên lý bay, các mô hình toán học của hệ thống, khí động lực học…

### **5.2.2 Kết quả chưa đạt được**

Bên cạnh những kết quả đạt được, đề tài vẫn còn tồn tại một số hạn chế cần được khắc phục:

* Bộ khung bằng nhựa nên có khối lượng tương đối lớn. Để khắc phục vấn đề này, nhóm đề ra giải pháp là chế tạo một bộ khung bằng chất liệu sợi carbon hoặc nhôm để có khối lượng nhỏ hơn.
* Thời gian bay tương đối ngắn. Có thể khắc phục được điều này bằng cách dùng pin có dung lượng cao hơn, thay đổi động cơ có công suất tiêu thụ thấp hơn. Việc giảm khối lượng máy bay cũng góp phần không nhỏ trong việc tăng thời gian bay.
* Thỉnh thoảng xảy ra hiện tượng mất dữ liệu và đôi khi kết nối thiếu ổn định do bộ điều khiển được nhóm chế tạo.
* Nhóm đã cố gắng khắc phục sai số GPS nhưng độ chính xác của GPS chưa như mong muốn.

## **5.3 Hướng phát triển**

Phát triển Quadcopter và đưa vào ứng dụng thực tế là một việc làm cấp bách và hoàn toàn có thể thực hiện thành công. Từ mô hình hiện tại, nhóm có hai hướng phát triển như:

* Trang bị thêm camera cho Quadcopter để có thể trở thành flycam, ghi lại hình ảnh trong quá trình di chuyển, phục vụ các công việc tìm kiếm và do thám.
* Phát triển thêm việc tích hợp và định vị GPS, giải quyết bài toán năng lượng trên Quadcopter để phát triển hệ thống giao hàng.

# **TÀI LIỆU THAM KHẢO TRONG BÁO CÁO**

**Tiếng Việt:**

**[1] Hoàng Minh Sơn,** *Một số phương pháp chỉnh định lại các tham số PI/PID trong vòng kín*, Chuyên san Kỹ thuật Điều khiển Tự động (số 6), Tạp chí Tự động hóa ngày nay, 2007.

[2] Nguyễn Thị Phương Hà, Huỳnh Thái Hoàng, *Lý thuyết điều khiển tự động*, Nhà xuất bản Đại học Quốc gia TP. Hồ Chí Minh, 2005.

**Tiếng Anh:**

[1] M.H. Ang, Jr, V.D. Tourassis, *Singularities of Euler and Roll – Pitch - Yaw representations*, Production Automation Project, 2007.

[2] Atheer L. Salih, M. Moghavvemi, Haider A.F. Mohamed and Khalaf Sallom Gaeid, *Flight PID controller design for a UAV quadroto*, 2003.

[3] Jose C. V. Junior, Julio C. De Paula, Gideon V. Learndro, Marilo C. Bonfim,

*Stability Control of a Quad-Rotor Using a PID Controller*, 2009.

[4] Matt Parker, Gerad Bottorff, *Quadcopter design*, Colorado State University, 2012.

[5] Quadcopter overview.

URL: <https://tqkhaicdt.wordpress.com/2016/05/13/mo-hinh-bay-Quadcopter-p1/>

[6] RASPI overview:

URL: <https://www.raspberrypi.org/forums/viewtopic.php?f=32&t=119413>

[7] SPI – I2C protocol.

URL: <https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial>

[8] HTTP server – client.

URL: <http://www.jmarshall.com/easy/http/>

1. () <http://cloudtweaks.com/2014/12/cloud-infographic-drone-consumer-revolution/> [↑](#footnote-ref-1)
2. () <http://soha.vn/israel-tung-san-pham-drone-trieu-do-lam-duoc-dieu-ky-dieu-dau-tien-tren-the-gioi-20170106133418275.htm> [↑](#footnote-ref-2)
3. ()<http://dantri.com.vn/the-gioi/hai-quan-nga-phat-trien-uav-hoat-dong-tren-tau-chien-20170104160239476.htm> [↑](#footnote-ref-3)
4. () [www.ardrone2.parrot.com](http://www.ardrone2.parrot.com) [↑](#footnote-ref-4)
5. () <http://cloudtweaks.com/2014/12/clou d-infographic-drone-consumer-revolution/> [↑](#footnote-ref-5)
6. () <https://kienltb.wordpress.com/2015/04/05/chuan-giao-tiep-spi/> [↑](#footnote-ref-6)
7. () <http://hshop.vn/products/mach-thu-phat-nrf24l01-2-4ghz> [↑](#footnote-ref-7)
8. ()<http://www.vnpro.org/forum/forum/ccent%C2%AE/icnd-1-basic-wireless-lan-config/26548-t%C3%ACm-hi%E1%BB%83u-v%E1%BB%81-access-point> [↑](#footnote-ref-8)