# Chương 1: Giới Thiệu

Đề tài nghiên cứu bộ điều khiển cân bằng máy ảnh 3 trục được lấy ý tưởng từ việc quan sát các máy bay 4 cánh có gắn máy quay phim, việc gắn máy quay phim có thể ghi lại hình ảnh từ trên cao ứng dụng cho việc giám sát hay quay phim, chụp hình. Khi gắn máy quay vào máy bay vì do ảnh hưởng trong quá trình hoạt động: rung động tạo ra từ động cơ, ảnh hương trong khi duy chuyển, ảnh hưởng các yếu tố bên ngoài như gió sẽ làm cho hình ảnh quay về được không được đẹp, và bị rung.

Một điểm nữa là khi quay phim ở những môi trường không ổn định, như quay phim khi đi cano trên biển, quay phim lúc chạy thì chắc chắn hình ảnh quay được sẽ bị lắc vì không người quay phim rất khó để ổn định máy quay trong các trường hợp như thế.

Trên thực tế đã có nhiều thiết bị ổn định máy quay chuyên nghiệp giành cho ngành điện ảnh, các thiết bị này được biết đến nhiều hơn với tên gọi gimbal (từ đây sẽ gọi bộ điều khiển/thiết bị cân bằng máy ảnh 3 trục là gimbal. Một số hình ảnh của những bộ gimbal dùng trong điện ảnh và cho máy bay 4 cánh điều khiển từ xa.

Giá của những bộ cân bằng này giao động từ $1000 đến trên $3000[1], vì giá đắt nên nhóm quyết định tự nghiên cứu cho mình một bộ cân bằng như thế, để có thể quay phim và chụp hình với hình ảnh đẹp nhất.

Một nguyên nhân khác cho việc chọn đề tài này là nhóm mong muốn có thể hiện thực nên 1 hệ thống có thể tự cân bằng và áp dụng những kiến thức này để xây dụng những ứng dụng thú vị khác như là máy bay cân bằng 4 cánh hoặc bàn chơi bida dành cho các thuyền lớn trên biển.

# Chương 2: Cơ sở lý thuyết

## 2.1 Sử dụng cảm biến

Để tìm được sự định hướng (góc roll, pitch) của một đối tượng không chuyển động có thể sử dụng cảm biến gia tốc kế. Đối với một đối tượng tĩnh, cảm biến gia tốc kế đo “lực hấp dẫn” ở 3 trục, sau đó trả về hướng của đối tượng.

Phương pháp này cho kết quá khá chính xác nếu đối tượng đo không duy chuyển (thường được sử dụng nhiều nhất trên các điện thoại thông minh), còn khi duy chuyển vật thể cần đo thì đã tạo một lực tác động lên nó. Điều này làm thay đổi giá trị của gia tốc kế, nên sẽ có một phương pháp khác chính xác hơn

Thêm nữa, cảm biến gia tốc kế rất nhạy với những những lực tác động bên ngoài (ảnh hưởng của rung lắc khi sản phẩm hoạt động) vì vậy để cho hệ thống có thể hoạt động được, nhất thiết phải tìm được chính xác góc (smooth angle) của đối tượng. Đây là bước quan trọng nhất, vì khi góc của đối tượng không chính xác hay không ổn định thì việc điều khiển cân bằng cho đối tượng đố là không thực hiện được.

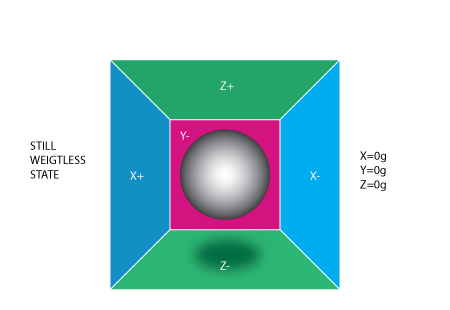
Như đã nói ở trên chỉ sử dụng gia tốc kế để tính góc quay là chưa đủ, cần phải sử dụng thêm một bộ lọc tín hiệu để giảm đi sự rung lắc tác động lên đối tượng. Bộ lọc tín hiệu ở đây là những bộ lọc số (bộ lọc mềm) được nhúng vào trong mã nguồn của hệ thống, có nhiều bộ lọc số khác nhau đã được ứng dụng như Kalman, Complementary, Ballistic.

Các bộ kể trên điều có ưu nhược điểm riêng và được ứng dụng cho các hệ thống khác nhau, Kalman được ứng dụng cho máy bay với sự kết hợp của cảm biến gia tốc và con quay hồi chuyển (hoặc kết hợp thêm cảm biến từ trường) ứng dụng cho máy bay 4 cánh, Ballistic thường kết dùng trong các máy bay 4 cánh có thêm GPS, riêng bộ lọc Complementary được ứng dụng nhiều vào các ứng dụng trong mô hình tự cân bằng (xe cân bằng, glimbal) vì tính dể hiện thực và hiểu quả sử dụng nguồn tài nguyên trong các hệ thống nhúng nhỏ. Phần tiếp theo sẽ trình bày chi tiết hơn về hoạt động của cảm biến gia tốc, con quay hồi chuyển, hoạt động của bộ lọc Complementary, so sánh hiệu quả của nó so với các bộ lọc Kalman.

### 2.1.1 Giới thiệu cảm biến

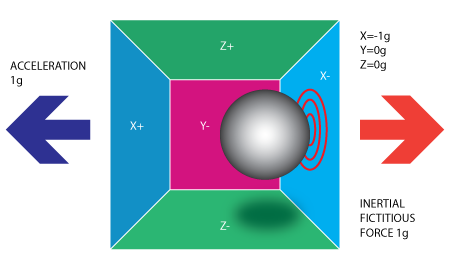
#### 2.1.1.1Gia tốc kế

Để có thể hình dung về gia tốc kế một cách rõ ràng, phương pháp hiệu quả nhất là tưởng tượng một cái hộp hình vuông và một quả bóng nhỏ ở bên trong, như hình mô phỏng bên dưới.



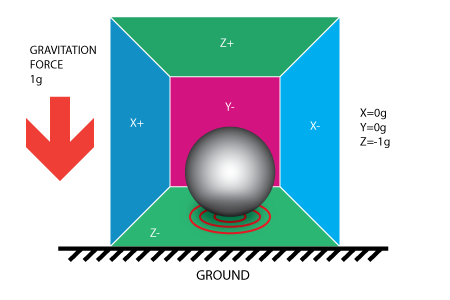
Hình 2 – 17: Mô phỏng gia tốc kế

Nếu để cái hộp đó ở một nơi không có trọng lực, có nghĩa là không có bất cứ “trường” nào tác động vào quả bóng, lúc đó quả bóng sẽ ở vị trí giữa hộp. Hình trên đã bỏ đi mặt phẳng Y+, tưởng tượng mỗi mặt phẳng (vách của chiếc hộp) có thể đo được lực va chạm của quả bóng vào nó (đo được độ nhạy – sensitive). Nếu bất thình lình duy chuyển cái hộp về phía bên trái (giả sử với gia tốc 1g = 9.8 m/s2), lúc này quả bóng sẽ duy chuyển rồi chạm vào vách X-, bằng cách đo lực tác dụng của quả bóng vào vách của hộp ta được kết quả lực (như giả sử là -1g) trên trục X. Hình tiếp theo sẽ mô tả cho quá trình trên.



Hình 2 – 18: Gia tốc kế duy chuyển sang trái

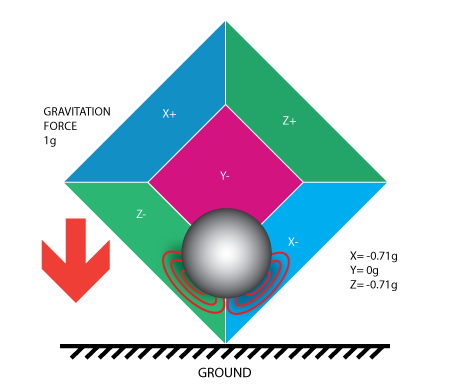
Chú ý rằng gia tốc kế sẽ lấy lực tác dụng có hướng ngược lại với hướng vector gia tốc, lực này gọi là lực khả dĩ (Fictitious Force). Gia tốc kế đo được gia tốc thông qua lực gián tiếp tác dụng vào vách ngăn của nó, lực này có thể được gây ra bởi gia tốc nhưng trong thực tế thì không phải chỉ có lực gia tốc. Nguyên nhân vì gia tốc kế được sử dụng ở Trái Đất, nghĩa là nó có lực hấp dẫn tác dụng lên nó, bình thường thì quả bóng sẽ chạm vào một phía nào đó của vách ngăn (mặt phẳng Z- chẳng hạn).



Hình 2 – 19: Ảnh hưởng của lực hấp dẫn

Đó là lí do tại sao chiếc hộp mặc dù không có di chuyển nhưng ta vẫn đọc được giá trị -1g ở trục Z. Kết quả đo này là do lực hấp dẫn tác dụng vào quả bóng chứ không phải ta duy chuyển chiếc hộp. Trong thực tế có thể có nhiều lực khác nhau tác dụng lên gia tốc kế, ví dụ, nếu quả bóng trong chiếc hộp được làm từ kim loại, vậy thì trong môi trường có từ trường, chắc chắn kết quả đo sẽ bị ảnh hưởng.

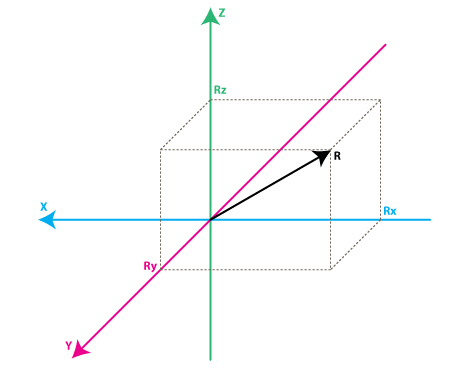
Giá trị thực của gia tốc kế đến từ phép đo của cả 3 trục, ví dụ nếu xoay chiếc hộp một góc 45 độ, cùng chiều kim đồng hồ, lúc này quả bóng chạm vào 2 vách ngăn là Z- và X- như hình bên dưới.



Hình 2 – 20: Tác dụng vào 2 hướng

Kết quả đo được là: X = -0.71g, Y = 0g và Z = -0.71g

Bây giờ chuyển chiếc hộp thành hệ trục tọa độ và bắt đầu tìm hiểu mối liên hệ giữa các đại lượng.



Hình 2 – 21: Hệ trục tọa độ gia tốc kế trong không gian

Mỗi trục tọa độ tương ứng với bề mặt của chiếc hộp và vector R thể hiện giá trị gia tốc mà gia tốc kế đo được. Rx, Ry, Rz là hình chiếu của vector R lên trục X, Y và Z. Mối liên hệ của R và Rx, Ry, Rz như sau:

 Eq.1

Áp dụng Eq. 1 với R = 1, RX = 1/2, RY = 1/2 và RZ = 1/2 ta được:



=> 

Trước khi trình bày tiếp về hình học 3 chiều cần phải nói đến một chút về cảm biển nghiêng, làm thế nào thu được giá trị của cảm biến rồi sau đó chuyển đổi nó thành đại lượng hình học. Cảm biến gia tốc chủ yếu chia làm 2 loại, một loại là digital và một loại là analog, Đối với cảm biến gia tốc analog, giá trị ngõ ra của cảm biến là mức của điện áp, giá trị điện áp này cần được tham chiếu và đọc về bằng bộ ADC của vi điều khiển. Trong khi đó các loại cảm biến gia tốc digital cho ra đại lượng nhị phân trực tiếp mà không cần bộ chuyển đổi ADC, tuy nhiên cần phải sử dụng các chuẩn giao tiếp như I2C, SPI hay UART (tùy theo cảm biến hỗ trợ chuẩn giao tiếp nào) để đọc giá các trị này về.

Ví dụ với bộ ADC 10 bit (khoảng giá trị ADC là 0 đến 210 -1), giá trị ADC đọc được từ các trục của cảm biến lần lượt là:

valADC\_Rx = 586

valADC\_Ry = 630

valADC\_Rz = 561

Mỗi bộ ADC đều có điện áp tham chiếu, giả sử điện áp tham chiếu trong ví dụ này là Vref = 3.3V. Việc chuyển đổi giữa giá trị điện áp và giá trị ADC đọc được ta dùng phương trình sau:

VoltsRx = valADC\_Rx \* Vref/1023

Vậy ta được:

VoltsRx = 586\*3.3V/1023 = ~1.89V

VoltsRy = 630\*3.3V/1023 = ~2.03V

VoltsRz = 561\*3.3V/1023 = ~1.81V

Cảm biến giá tốc đều có giá trị điện áp ở zero-g, giá trị này thường có trong datasheet của chip, giả sử ở 0g có VzeroG = 1.65V ta được:

DeltaVoltsRx = 1.89 – 1.65 = 0.24V

DeltaVoltsRy = 2.03 – 1.65 = 0.38V

DeltaVoltsRz = 1.81 – 1.65 = 0.16V

Một giá trị khác cần được biết đến ở các chip cảm biến nghiêng là độ nhạy của nó, độ nhạy là giá trị thay đổi điện áp trên mỗi thay đổi của g, thường đơn vị này là mV. Giả sử với độ nhạy của chip là: Sensitivity = 478.5mV/g = 0.4785V/g Giá trị cuối cùng cần tìm là lực tác dụng có đơn vị g là:

R = DeltaVoltsR/Sensitivity

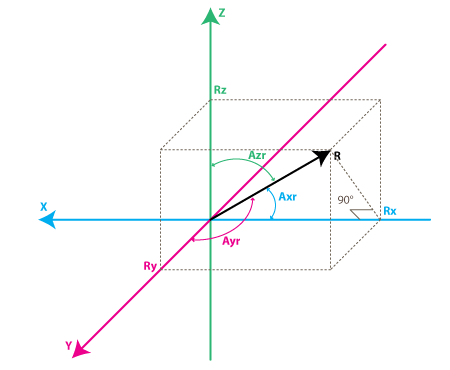
Rx = 0.24V/0.4785V/g = ~0.5g

Ry = 0.38V/0.4785V/g = ~0.79g

Rz = 0.16V/0.4785V/g = ~ 0.33g

Công thức cuối cùng thu được là:

 Eq.2



Hình 2 – 22: Hệ trục tọa độ và góc nghiêng

Với các giá trị R vừa tìm được, có thể tìm được độ dốc của cảm biến so với mặt đất, theo như công thức trên thì chỉ cần tìm góc lệch của vector R với trục Z.

Gọi góc Axr là góc giữa R và trục X, góc Ayr là góc giữa R và trục Y và góc Azr là góc giữa R và trục Z. Ta được:



#### 2.1.1.2 Con quay hồi chuyển

Hệ trục để diễn ta con quay hồi chuyển như hình sau:



Hình 2 – 23: Hệ trục biểu diễn con quay hồi chuyển

Mỗi kênh của con quay hồi chuyển đo góc quay quanh trục 1 trục của kênh đó. Với con quay hồi chuyển 2 trục ta sẽ có được góc quay của cảm biến tương ứng với cả 2 trục là X, Y.

Gọi Rxz là hình chiếu của vector R lên mặt phảng XZ.

Ryz là hình chiếu của vector R lên mặt phẳng YZ.

Sử dụng định lý Pitago trong tam giác vuông ta được:

 và 

Vì con quay hồi chuyển đo tỉ lệ thay đổi của góc theo thời gian nên ta ví dụ ở thời điểm t0 có được góc và tại thời điểm t1 có góc, tỉ lệ thay đổi đó là:



Nếu có giá trị là độ và thời gian t là giây, thì giá trị của  sẽ là deg/s. Đây là những gì mà con quay hồi chuyển đo được cho chúng ta.

Trong thực tế (nếu không phải là con quay hồi chuyển digital) thì giá trị đọc được từ cảm biến là giá trị điện áp, vì vậy cần chuyển thiết để chuyển nó về giá trị deg/s theo mong muốn. Tương tự như gia tốc kế ta được phương trình chuyển đổi của con quay hồi chuyển như sau:

#### 2.1.1.3 Áp dụng gia tốc kế và con quay hồi chuyển để tính góc lệch

### 2.1.2 Giới thiệu bộ lọc Complementary

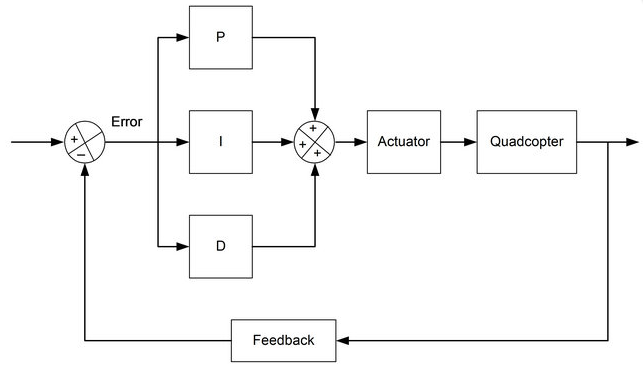
#### 2.1.2.1 Mô hình của bộ lọc Complementary

#### 2.1.2.2 Hoạt động của bộ lọc Complementary

#### 2.1.2.3 So sánh với bộ lọc Kalman

## 2.2 Điều khiển cân bằng

PID (Propotional-Integral-Derivative) là hệ thống điều khiển vòng kín làm cho kết quả thực tế gần hơn với kết quả mong muốn bằng cách điều chỉnh đầu vào



Hình 2 – 25: PID

Có ba thuật toán trong PID tương ứng là P (tỉ lệ), I (tích phân), D (đạo hàm). P phụ thuộc vào sai lệch hiện tại. I dựa trên sự tích luỹ sai lệch trong quá khứ. D là dự đoán sai lệch tương lai dựa trên tỉ lệ sai lệch hiện tại.

Để điều khiển Quadcopter, cần đo đầu ra từ các cảm biến (như góc nghiêng), để có thể ước lượng sai lệch (thực tế chênh lệch bao nhiêu so với góc nghiêng mong muốn)

Ảnh hưởng của ba tham số:

* **Hệ số P:** Hệ số này được xác định bởi người điều khiển hoặc con quay hồi chuyển. Khi P cao Quadcopter sẽ nhạy cảm hơn với sự thay đổi góc và khi P quá cao Quadcopter bắt đầu dao dộng. Khi P quá thấp Quadcopter sẽ chậm và khó khăn hơn để giữ ổn định
* **Hệ số I:** Hệ số này tăng độ chính xác của vị trí góc. Ví dụ khi Quadcopter bị nhiễu và lệch đi một góc 20 độ, trên lý thuyết nó sẽ nhớ được góc đã thay đổi và quay lại một góc đúng bằng góc đã bị lệch. Điều này hữu ích đối với những cơn gió bất thường và bất ổn trong động cơ. Tuy nhiên khi hệ số I quá cao, Quadcopter sẽ phản ứng chậm và giảm tác dụng của hệ số P, nó cũng bắt đầu dao động như khi hệ số P quá cao nhưng với tần số thấp hơn
* **Hệ số D:** Hệ số này cho Quadcopter nhanh chóng đạt được trạng thái mong muốn. Một số người gọi nó là thông số gia tốc vì nó khuếch đại đầu vào của người sử dụng. Nó cũng giảm kiểm soát hành động khi sai lệch giảm xuống. Trên thực tế nó tăng tốc độ phản ứng và tăng hiệu quả của hệ số P

## 2.3 Tối ưu hiệu năng hệ thống nhúng

//code size, performance

Trong một hệ thống nhúng thì hiệu năng có ý nghĩa như là sự thành công, một hệ thống nhúng sẽ phải hoạt động với nguồn tài nguyên hạn chế và thời gian hoạt động dài. Khác với các chương trình trên máy tính, chương trình trên hệ thống nhúng cần đáp ứng các yếu tố về bộ nhớ RAM, ROM, EPPROM, tốc độ xung nhịp cấp vào và các yếu tố này có giới hạn, vì vậy việc lập trình cần chú ý đến hiệu quả sử dụng bộ nhớ, tránh tình trạng rõ rĩ bộ nhớ (memory leak). Đối với tốc độ xử lý, một nguyên tắc cần theo là trong một lệnh C/C++ sinh ra càng ích mã máy càng tốt.

Một yếu tố về hiệu măng là thời gian mà hệ thống hoạt động. Một chương trình nhúng cần phải hoạt động ổn định trong một thời gian dài. Để tối ưu được hiệu năng, nhóm thực hiện 2 cách sau đó là dựa vào trình biên dịch và dựa vào chính các thành viên trong nhóm.

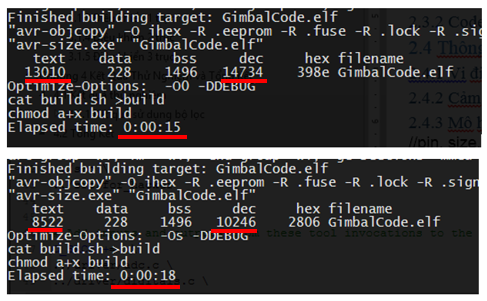
### 2.3.1 Compiler optimize

Trình biên dịch mà nhóm sử dụng là avr-gcc, đây là trình biên dịch C được viết cho AVR. Bộ biên dịch này miễn phí nằm ở dự án AVR libs (không nằm trong dự án GNU[2]) bao gồm các công cụ hỗ trợ như avr-binutils, avr-gcc và avr-libc là trái tim của công cụ lập trình cho avr.

Trong các tùy chọn của avr-gcc có tùy chọn tối ưu lúc biên dịch, bao gồm các cờ -O0, -O1, -O2, -O3 và –Os[3, Ch. 2.2] tương ứng với các mức tối ưu khác nhau.Nội dung của các cờ như sau:

* -O0 không sử dụng tối ưu lúc biên dịch
* -O1 trình biên dịch cố gắng giảm bớt kích thước mã nguồn và thời gian thực thi nhưng không thực hiện bất kỳ tối ưu nào phải mất nhiều thời gian biên dịch.
* -O2 tối ưu nhiều hơn –O1, thực hiện tất cả các tối ưu được hỗ trợ mà không liên hệ đến space-speed, tùy chọn này làm tăng thời gian biên dịch.
* -O3 mở tất cả các tối ưu của –O2 đồng thời mở thêm các tối ưu đặc biệt.
* -Os tối ưu làm giảm kích thước code được sinh ra.

Dưới đây là kết quả tối ưu và thời gian biên dịch với với 2 tùy chọn (qua một số lần chạy thử kết quả dưới đây là kết quả trung bình):



Ta thấy rõ thời gian biên dịch của tùy chọn không bật cờ tối ưu sẽ nhanh hơn thời gian biên dịch bằng cờ -Os, đổi lại là kích thước text lại cao hơn so với tùy chọn –Os. Tổng kết lại ta có kết quả như sau.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tùy chọn –O0 | | Tùy chọn –Os | |
| Thời gian biên dịch | 15s | | 18s | |
| Bộ nhớ Flash | Text: 13010 | Tổng: 13238 bytes | Text: 8522 | Tổng: 8750 bytes |
| Data: 228 | Data: 228 |
| Bộ nhớ RAM | BSS: 1496 | Tổng: 1724 bytes | BSS: 1496 | Tổng: 1724 bytes |
| Data: 228 | Data: 228 |

Nội dung của vùng nhớ text, data, bss, hex như sau[4]:

* text: chứa mã và dữ liệu chỉ đọc (biến constant) trong ứng dụng (bytes).
* data: dữ liệu đọ và ghi trong chương trình (các biến bình thường) (bytes).
* bss: chỉ ra dữ liệu khởi tạo ban đầu là zero.
* dec: tổng số byte của text + data + bss
* hex: dạng hệ 16 của dec

## 2.4 Thông số các thành phần trong hệ thống

Đề tài nghiên cứu sử dụng các thành phần sau: vi điều khiển atmega644p, cảm biến gia tốc và con quay hồi chuyển mpu6050, động cơ rc servo KST DS215

### 2.4.1 Vi điều khiển Atmega644p

Phần này sẽ trình bày các tính năng và thành phần ngoại vi sẽ được sử dụng, một số tính năng của chip atmega644p[5]:

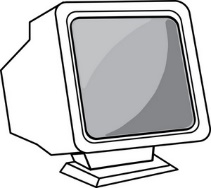
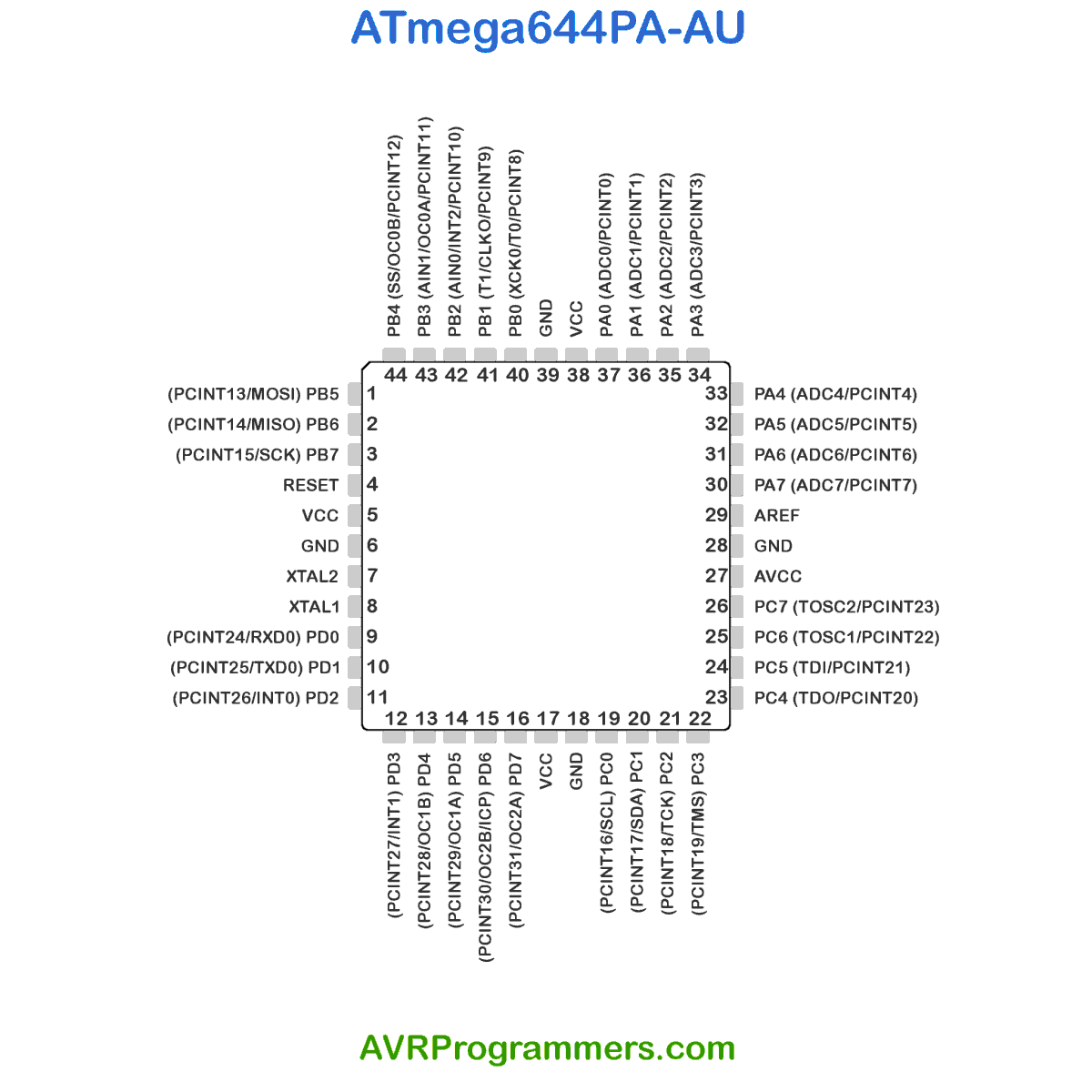
* Chip vi điều khiển 8 bits của hãng atmel.
* Kiến trúc Harvard, 32x8 thanh ghi mục đích chung.
* Bộ nhớ Flash: 64 Kbytes
* Số chân: 44
* Tần số tối đa: 20Mhz
* Timer: 2 timer 8 bit, 1 timer 16 bit
* PWM: 6 kênh
* Max I/O: 32
* Ext interrupts: 32
* TWI(I2C): 1
* UART: 2
* ADC: 8 kênh, 10 bit, tốc độ 15 ksps
* SRAM: 4 Kbytes
* EPPROM: 32 Bytes
* Debug: JTAG
* Điện áp hoạt động: 1.8 – 5.5v

Các ngoại vi nhóm sẽ sử dùng: UART, PWM, I2C và GPIO. Phần tiếp theo sẽ giới thiệu chi tiết các chuẩn giao tiếp và các thông số cài đặt cho từng ngoại vi.

#### 2.4.1.1 UART

Chuẩn truyền nhận nối tiếp RS232 được giới thiệu vào năm 1962 bởi Radio Sector của EIA[6], mục đích ban đầu của chuẩn giao tiếp này là để kết nối giữa terminal với các modem và hiện nay vẫn được dùng rất phổ biến trong các hệ thống nhúng, một số đặc điểm của chuẩn giao tiếp RS232:

* Có 9 chân kết nối, gọi là DB-9, đối với các máy tính cũ hơn thì sử dụng 25 chân kết nối gọi là DB-25.
* Có thể cho phép truyền song công, nghĩa là việc truyền và nhận dữ liệu được thực hiện cùng một lúc.
* Tốc độ giao tiếp tối đa: 10Kbytes/s.
* Nhóm sử dụng 3 chân là GND, Rx và Tx để thực hiện kết nối. Tx của thiết bị này sẽ được nối với Rx của thiết bị kia, hình sao mô tả cách kết nối giữa vi điều khiển atmega644p với máy tính.



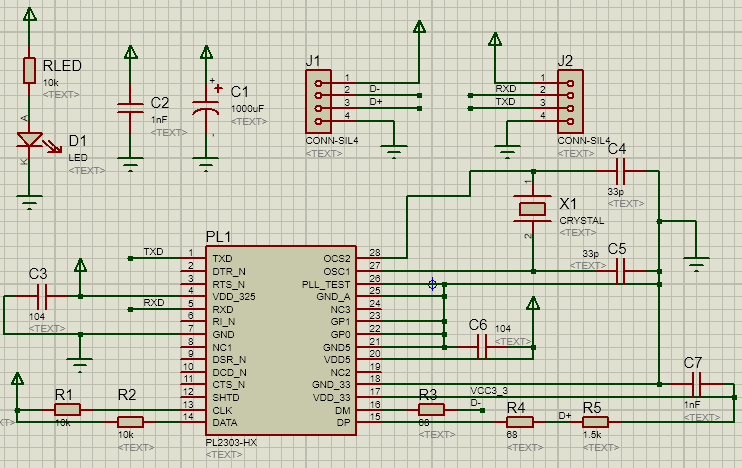
* Dữ liệu được đóng gòi thành 1 byte và truyền đi tuần tự, bit có trọng số thấp nhất sẽ được gửi đi sau cùng, và bit có trọng số lớn được truyền đầu tiên.
* Nhóm sử dụng UART không đồng bộ, nghĩa là không có xung clock điều khiển việc truyền nhận dữ liệu, các bước để thực hiện việc truyền và nhận không đồng bộ như sau:

1. Cả 2 bên truyền và nhận thỏa thuận với nhau trước các quy tắc (tốc độ truyền, định dạng truyền) đây là bước bắt tay trước khi quá trình truyền bắt đầu.
2. Bộ truyền gửi “idle” ( bằng “1”) trong khoảng thời gian nghĩ.
3. Bộ truyền gửi “start” ( = “0”) trước mỗi byte được truyền, vì vậy bộ nhận có thể biết được khi nào 1 byte dữ liệu sẽ được gửi đến.
4. 8 bit của byte dữ liệu được truyền đi.
5. Bộ truyền gửi “stop” ( = “1”) sau mỗi byte được truyền

* Ví dụ có một byte dữ liệu là 0x55, thì quá trình truyền diễn ra như hình bên dưới:

http://www.fpga4fun.com/images/SerialCommunication55.gif

* Tốc độ truyền của RS-232 quy định bởi giá trị baud (baud rate). Baud rate là giá trị chỉ ra bao bit được truyền trong 1 giây. Ví dụ: baud rate bằng 1000 nghĩa là sẽ có 1000 bit được truyền đi trong 1 giây, hay có 1 bit được truyền sau 1 mili giây.
* Tốc độ baud không phải là 1 giá trị được chọn bất kỳ, nó được chọn theo một số tốc độ chuẩn. Vi dụ: 1200, 9600, 38400 hay 115200. Ở tốc độ baud 115200, mỗi bit được truyền sau (1/115200) = 8.7us, nếu có 8-bits dữ liệu thì dữ liệu sẽ được truyền đi sau 8x8.7us = 69us. Nếu bao gồm các bit biểu thị start và stop thì tốc độ sẽ là 87us. Nghĩa là dữ liệu được truyền đi với tốc độ 11.5Kbytes trên 1 giây.
* Ở một số máy tính có thể yêu cầu 1 stop bit dài (1.5 hay 2 bit), lúc này tốc độ thực ở baud 115200 giảm xuống còn 10.5 Kbytes trên 1 giây.
* Tín hiệu truyền trên dây không theo mức TTL, giá trị “1”-mức cao ở điện áp -10v (ở khoảng -5v đến -15v) vag giá trị “0”-mức thấp ở điện áp + 10v (trong khoảng 5v đến 15v). Do đó để giao tiếp giữa chuẩn RS232 với các thiết bị ngoại vi khác sử dụng mức điện áp 0V và 5V người ta phải sử dụng một mạch chuyển đổi mức điện áp.
* Mạch chuyển đổi thông dụng nhất hiện nay là mạch sử dụng IC PL2303, đây là IC chạy ổn định nhất hiện nay. Dưới đây là mạch giao tiếp sử dụng PL2303.



* Thiết lập UART cho vi điều khiển:
  + Baud rate 115200
  + Stop bit 1
  + None parity.
  + Giá trị các thanh ghi điều khiển cho uart

|  |
| --- |
| UBRR1 = 10;  UCSR1A = 0;  UCSR1B = \_BV(TXEN1) | \_BV(RXEN1) | \_BV(RXCIE1);  UCSR1C = \_BV(UCSZ11) | \_BV(UCSZ10); |

#### 2.4.1.2 PWM

PWM là từ viết tắt của cụm từ Pulse Width Modulation, tiếng việt là phương pháp điều chế độ rộng xung. Đây là phương pháp điều khiển các thiết bị khác bằng cách thay đổi độ rộng của xung cao. Một số định nghĩa cần biết khi sử dụng PWM:

* Period: Là chu kỳ của xung
* Pulse Width: là độ rộng của xung cao
* Duty: là tỉ lệ phần trăm của Pulse Width với Period.

Thiết bị nhận được tín hiệu PWM sẽ dựa vào Duty khác nhau để thực hiện các lệnh khác nhau. Đối với trong ứng dụng điều khiển tốc độ thì Duty tỉ lệ thuận với tốc độ quay của thiết bị, ví dụ như Duty 70% sẽ cho ra tốc độ 70% so với tốc độ tối đa. Áp dụng PWM để điều khiển RC servo cần phải biết thông số kỹ thuật của servo để việc điều khiển được chính xác, thông số kỹ thuật của servo được trình bày ở phần sau, mục 2.4.3.

Vi điều khiển Atmega644p có 6 kênh PWM dư để điều khiển 3 động cơ. Giá trị thiết lập cho các thanh ghi của AVR.

|  |
| --- |
| OCR1A = TCNT1;  OCR1B = TCNT1;  TIFR1 |= \_BV(OCF1A) | \_BV(OCF1B); // clear OCA and OCB interrupt  TIMSK1 |= \_BV(OCIE1A) | \_BV(OCIE1B); // enable OCA and OCB interrupt on timer1 |

#### 2.4.1.3 I2C

#### 2.4.1.4 GPIO

### 2.4.2 Cảm biến MPU6050

Thông số của cảm biến:

* Bao gồm 3 trục gia tốc kế, 3 trục con quay hồi chuyển
* Điện áp hoạt động: 2.375-3.46V
* Gyro Full Scale Range: ±250 ±500 ±1000 ±2000 °/sec
* Gyro Sensitivity: 131 65.5 32.8 16.4 LSB/°/sec
* Accel Full Scale Range: ±2 ±4 ±8 ±16 g
* Accel Sensitivity: 16834 8192 4096 2048
* Chuẩn giao tiếp: I2C
* Kích thước: 4x4x0.9mm

### 2.4.3 Servo KST DS521MG

Thông số hoạt động của servo:

* Điện áp hoạt động: 4.5 – 6V
* Lực nâng: 3.18kg/cm ở điện áp 6V
* Tốc độ: 0.06s/600 ở điện áp 6v
* Kích thước: 22.90 x 12.27 x 27.30 mm
* Tần số hoạt động: 1520us/333Hz

### 2.4.4 Mô hình

// Todo pin, size, cân nặng

# Chương 3 Hiện Thực Hệ Thống

Chương 3 sẽ trình bày chi tiết quá trình hiện thực hệ thống, bao gồm: Liệt kê những gì cần thiết để thực hiện nghiên cứu, phần này được trình bày ở phần 3.1 Tài nguyên sử dụng. Thời gian biểu để thực hiện nghiên cứu ở phần 3.2, thời gian biểu giúp nhóm có thể giám sát được quá trình thực hiện nghiên cứu, đồng thời biết được những vẫn đề sẽ nảy sinh trong tương lai.

Các hiện thực 3.3 được chia thành các phần nhỏ khác nhau, phần này chính yếu trình bày chi tiết giải thuật để điều khiển cân bằng cho gimbal.

## 3.1 Tài nguyên sử dụng

//Todo resource for project

Để hoàn thành nghiên cứu này, nhóm đồng thời sử dụng các nguồn tài nguyên có sẵng và miễn phí, đồng thời sử dụng tối ưu kinh phí hỗ trợ từ trường. Các tài nguyên bao gồm:

* Mạch vi điều khiển Atmega664p bao gồm cảm biến gia tốc con và quay hồi chuyển mpu6050. Mạch được sử dụng lại từ đồ án tốt nghiệp (do bị hỏng module LCD, nhóm sử dụng UART để thay thế phần hiện thị thông tin).
* Động cơ điều khiển, sử dụng 3 động cơ digital RC servo, với góc điều khiển nhỏ, mua mới.
* Mica để làm mô hình (sẽ trình bày ở phần 3.1.1), mua mới.
* Trình biên dịch avg-gcc (tích hợp trong IDE miễn phí Atmel studio).
* IDE microsoft visual studio (miễn phí qua tài khoản DreamSpark).
* Phần mền thiết kế Inventor (crack?? hay sao m?)
* Quản lý qua Mercurial cung cấp bởi bitbucket (miễn phí với với dự án nhỏ hơn 5 thành viên)

## 3.2 Lịch trình nghiên cứu

Thời gian biểu để hoàn thành nghiên cứu như sau:

//Todo add screenshot of timesheet when project finished.

## 3.3 Hiện Thực

Dưới đây là mô hình của toàn bộ hệ thống:

Mạch điều khiển và cảm biến

Máy tính cá nhân

Nguồn điện cho hệ thống (pin DC 12v và điện AC 220v)

Bộ chuyển đổi DC 12v – DC 5v

Bộ chuyển đổi AC 220v – DC 18v

UART

I2C

Động đáp ứng RC servo 1

Động đáp ứng RC servo 2

Động đáp ứng RC servo 3

PWM

**Chi tiết của mô hình:**

* Khối trung tâm là mạch điều khiển và cảm biến. Mạch điều khiển nhóm sử dụng vi điều khiển ATmega664p với cảm biến gia tốc và con quay hồi chuyển mpu6050, vi điều khiển và cảm biến sẽ giáo tiếp với nhau thông qua chuẩn giao tiếp 2 dây I2C, vi điều khiển sẽ là master và cảm biến sẽ là slave. Mạch điều khiển trung tâm được mua ở ngoài thị trường nhằm giảm bớt công việc thiết kế mạch, đồng thời tăng tính ổn định của hệ thống lên.
  + Khối trung tâm có nhiệm vụ đọc giá trị trả về của cảm biến, sau đó lọc đi dữ liệu lỗi, nhiễu và áp dụng giải thuật điều khiển PID để cân bằng cho hệ thống.
  + Tính hiệu điều khiển được xuất là là tín hiệu PWM như đã trình bày ở trên, tín hiệu hiệu này sẽ điều khiển góc quay cho 3 động cơ RC servor.
  + Khối trung tâm còn có nhiệm vụ giao tiếp với máy tính thông qua chuẩn giao tiếp UART, việc giao tiếp với máy tính sẽ dể dàng cho việc kiểm lỗi và giám sát hệ thống.
  + Ngoài ra trên khối trung tâm còn có thể mạch nguồn chuyển đội điện áp 12v từ pin sang điện áp 5v với dòng điện cung cấp tối đa là 3A.
* Máy tính cá nhân sẽ giao tiếp với mạch trung tâm, nhận dữ liệu trả về và gửi dữ liệu điều khiển.
* 3 động cơ RC sẽ là cơ cấp chấp hành chính, sẽ hoạt động theo lệnh điều khiển của mạch trung tâm.
* Khối nguồn bao gồm pin lipo 12v, 2200mAh và điện gia đình 220v (cung cấp cho máy tính). Nhiệm vụ của khối này cực kỳ quan trọng, giúp nuôi sống cho toàn bộ hệ thống.

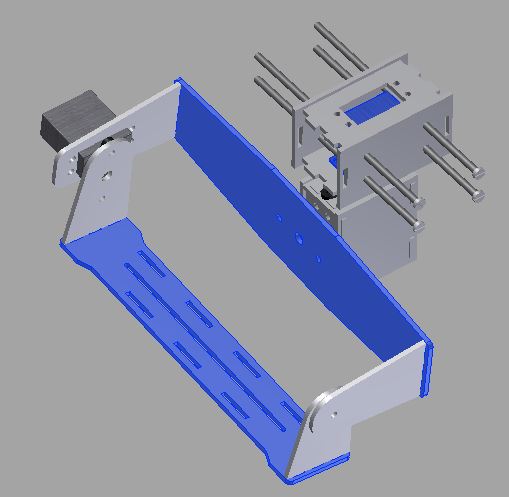
### 3.1.1 Mô hình phần cứng

Mô hình phần cứng bao gồm mạch điều khiển và khung của sản phẩm.

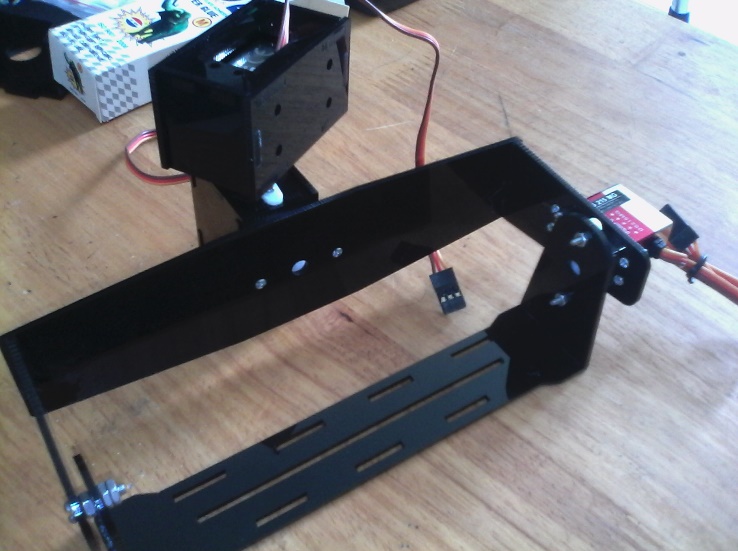
Đối với mạch phần cứng, nhóm sử dụng mạch điều khiển Atmega664p có tích hợp cảm biến MPU6050.

//Todo add picture board.

Khung của sản phẩm được thiết kế qua thông phần mền Inventor để sản phẩm có độ chính và ổn định trong qua trình hoạt động, hình chụp của thiết kế.



Sau khi có bản thiết kế sẽ tiến đi căt lazer, vật liệu sử dụng là mica đen.



Đây là mô hình hoàn chỉnh của thiết kế, mô hình đáp ứng được yêu cầu về hoạt động cơ học, chịu được va chạm trong quá trình thử nghiệm.

### 3.1.2 Mô hình phần mềm

Để thuận tiện cho quá trình phát triển mã nguồn cũng như kiểm lỗi và thu thập kết quả điều khiển, nhóm đã thiết kế mô hình phần mền như sau:

Khối Serial

(serial.c)

Khối I2C

(i2cMaster.c)

Khối PWM

(pwm.c)

Khối system (system.c)

Khối Sensor

(mpu6050.c)

Khối PID

(pid.c)

Khối Filter

(imu.c)

Khối Control

(control.c)

Khối Debug

(monitor.c)

Khối Digital

(digital.c)

Chương trình chính (main.c)

Chi tiết từng khối sẽ được trình bày ngay sau đây. Việc trình bày chỉ mô tả hoạt động và vai trò của từng khối trong mã nguồn, bao gồm liệt kê các hàm chính yếu trong khối (mô tả chi tiết tính năng của hàm sẽ được trình bày ở phần phụ lục).

1. Lớp thấp nhất của hệ thống nằm ở khối system, khối system được hiện thực qua file mã nguồn system.c, Khối system sẽ khai báo các thanh ghi dùng để khởi tạo hệ thống, bao gồm khai báo các port sử dụng trên vi điều khiển, khai báo timer (hàm setup()) và các hàm lấy thời gian của hệ thống ( gồm hàm millis(), micros(), ticks()) khởi tạo các mode cho giao tiếp uart, khởi tạo pwm và khởi tạo i2c để giao tiếp với cảm biến.

Khối system còn cung cấp các macro và define dùng trong cho hệ thống (thông qua include file system.h), ví dụ như MIN, MAX, ABS, FILTER, F\_CPU, DEBUG, ROLL, PITH, YALL...

1. Phần cảm biến bao gồm các khối I2C, Sensor và Filter, thứ tự từ thấp đến cao như trong hình vẽ (số mấy), nhiệm vụ của phần cảm biến là trả về góc lệch của đối tượng, góc lệch này cần phải chính xác, ổn định, không bị nhiễu bởi các rung lắc. Để hiện thực nên phần này, nhóm chia thành 3 khối nhỏ để phân chia công việc rõ ràng, cụ thể:
   * Khối I2C (file i2cMaster.c) sẽ hiện thực các hàm I2C để giao tiếp với cảm biến (ví dụ như hàm i2c\_writeReg(),i2c\_getSixRawADC(),i2c\_readReg16()...) các hàm này đọc dữ liệu sẽ được trả về từ cảm biến.
   * Việc đọc các hàm này được thực hiện qua khối Sensor (mpu6050.c), dữ liệu đọc được là dữ liệu thô, chưa qua xử lý và cần được filter bởi khối Filter.
   * Khối Filter hiện thực qua file imu.c, ở đây sẽ dùng hàm imuCalculate() để lọc đi dữ liệu nhiễu, việc lọc dữ liệu này đã được trình bày ở phần 2.1.2
2. Khối digial là khối phụ, dùng để điều khiển đèn led thông báo, việc thông báo bằng đền led sẽ có lợi cho việc giám sát khi cho chạy hệ thống thực tế, nó là tín hiệu báo lỗi cực kỳ hữu ích.
3. Một phần quan trọng của hệ thống nữa là phần Debug, phần này có nhiệm vụ rất nhiều trong quá trình phát triển mã cho hệ thống, để module Debug có thể hoạt động cần thiết lập tùy chọn ở compiler setting là –D DEBUG. Trong phần Debug có các khối nhỏ bao gồm: Khối Serial và khối Debug.
   * Khối Serial hiện thực ở file serial.c, trong file này nhóm hiện thực nên các hàm cơ bản như serialWriteChar() serialReadChar(). Các hàm cơ bản này sẽ ghi hay đọc dữ liệu vào queue, chuẩn giao tiếp uart cho tốc độ truyền nhận không cao vì vậy cần một queue để dữ liệu được buffering trước khi truyền, tránh trường hợp hệ thống bị lỗi khi truyền dữ liệu quá nhiều. Với thiết lập tốc độ truyền (baud rate) 115200 thì mỗi giây ...
     + Khối serial còn override hàm printf() trong C để việc xuất dữ liệu thêm dễ dàng.
   * Ở file monitor.c là khối Debug, khối này sẽ nhận dữ liệu điều khiển từ máy tính và sau đó thực hiện các hàm đã được định nghĩa trước. Ví dụ một số hàm cơ bản như ver, help, wr, Lệnh help sẽ xuất ra một số thông tin hướng dẫn:

//Todo add screenshoot

* + - Thuộc tính của các lệnh điều khiển:
      * Độ dài lệnh bao gồm cả đối số: CMD\_SIZE 32
      * Đối số là kiễu dữ liệu float
      * Số lượng tối đa của đối số: MAX\_PARMS 8
      * Độ dài một lệnh tối đa: PARMS\_SIZE 8

1. File main.c là file chương trình chính chứa hàm main của hệ thống, file này và bao gồm file main.h sẽ chữa các biến extern để sử dụng cho toàn bộ các file khác trong chương trình.

Một phần quan trọng trong việc thiết kế phần mền cho hệ thống là việc tối ưu bộ nhớ cho vi điều khiển, chip atmega664p có bộ nhớ RAM và ROM giới hạn, cần file sắp xếp các biến, các struct, các constant vào các vùng nhớ hợp lý, nội dung này như đã được trình bày ở phần 2.3 Tối ưu hiệu năng hệ thống nhúng. Tiếp theo là số đồ khối của code, bao gồm sơ đồ khối của chương trình chính, bộ lọc Complementary, bộ điều khiển PID, Serial debug và Digital.

//Todo add code diagram.

### 3.1.3 Điều khiển 1 trục

### 3.1.4 Điều khiển 2 trục

### 3.1.5 Điều khiển 3 trục

# Chương 4 Kết Quả Thử Nghiệm và Tổng Kết

## 4.1 Kết quả thử nghiệm

### 4.1.1 Kết quả sử dụng bộ lọc

#### 4.1.1.1 Không sử dụng bộ lọc

#### 4.1.1.2 Sử dụng bộ lọc Kalamn

#### 4.1.2 Kết quả hoạt động của toàn hộ thống

## 4.2 Tổng Kết

### 4.2.1 Đạt được

### 4.2.2 Không đạt được

### 4.2.3 Hướng phát triển

# Tham khảo

[1] “Ghost Gimbals | Spaulding International Cinema.” [Online]. Available: http://ghostgimbals.com/collections/ghost-gimbals. [Accessed: 07-Nov-2015].

[2] “AVR C Runtime Library - Summary [Savannah].” [Online]. Available: http://savannah.nongnu.org/projects/avr-libc/. [Accessed: 07-Nov-2015].

[3] Atmel Corporation, “Tips and Tricks to Optimize Your C Code for 8-bit AVR Microcontrollers,” p. 18, 2011.

[4] “FlashRamSize - \*\* Code Red Support Site \*\*.” [Online]. Available: http://www.support.code-red-tech.com/CodeRedWiki/FlashRamSize. [Accessed: 07-Nov-2015].

[5] ATmel, “ATmega 6444/V,” pp. 1–376, 2012.

[6] *EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange*. Washington: Electronic Industries Association. Engineering Dept, 1969.