ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Phạm Tiến Thành

NGHIÊN CỨU, THIẾT KẾ XE TỰ CÂN BẰNG SỬ DỤNG CẢM BIẾN GIA TỐC 3 TRỤC VÀ VI ĐIỀU KHIỂN 32 BIT

KHÓA LUẬN TỐT NGHIỆP HỆ ĐẠI HỌC CHÍNH QUY

Ngành: Công nghệ kỹ thuật cơ điện tử

HÀ NỘI - 2018

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Phạm Tiến Thành

NGHIÊN CỨU, THIẾT KẾ XE TỰ CÂN BẰNG SỬ DỤNG CẢM BIẾN GIA TỐC 3 TRỤC VÀ VI ĐIỀU KHIỂN 32 BIT

KHÓA LUẬN TỐT NGHIỆP HỆ ĐẠI HỌC CHÍNH QUY

Ngành: Công nghệ kỹ thuật cơ điện tử

Cán bộ hướng dẫn: ThS. Hoàng Văn Mạnh

HÀ NỘI - 2018

NGHIÊN CỨU, THIẾT KẾ XE TỰ CÂN BẰNG SỬ DỤNG CẢM BIẾN GIA TỐC 3 TRỤC VÀ VI ĐIỀU KHIỂN 32 BIT

Phạm Tiến Thành

Khóa QH-2014-I/CQ, ngành Công nghệ kỹ thuật Cơ điện tử

Tóm tắt khóa luận: Nội dung nghiên cứu của đề tài bao gồm 2 phần chính:

Thứ nhất, xe tự cân bằng có khả năng tự đứng thăng bằng trên hai bánh tại mặt phẳng và mặt nghiêng. Xe tự cân bằng này hoàn toàn có thể được coi là một hệ thống cơ điện tử hoàn chỉnh bao gồm cảm biến tiếp nhận thay đổi môi trường, vi điều khiển tiếp nhận thông tin từ cảm biến và gửi tín hiệu điều khiển một hệ cơ học. Vi điều khiển 32 bit, một dòng vi điều khiển mạnh mẽ so với vi điều khiển 8 bit truyền thống, được sử dụng làm phần vi điều khiển xử lý cho xe hai bánh tự cân bằng.

Thứ hai, phần mềm điều khiển, theo dõi, cài đặt cho xe sử dụng được trên thiết bị di động. Có khả năng điều khiển xe di chuyển và ghi lại giao động của xe tự cân bằng. Việc xây dựng phần mềm di động để quản lý thiết bị là nhu cầu tất yếu trong thời đại công nghiệp 4.0 cùng với sự phát triển của IoT. Phần mềm đi kèm với thiết bị sẽ giúp tăng giá trị thiết bị lên rất nhiều.

Từ khóa: *Xe tự cân bằng, cảm biến gia tốc 3 trục, vi điều khiển 32 bit, hệ cơ điện tử.*

LỜI CẢM ƠN

“Không thầy đố mày làm lên” – Đúng như câu tục ngữ, em đã không thể hoàn thành đề tài của luận văn này nếu không có sự giúp đỡ của các thầy các cô cũng như người thân và bạn bè xung quanh. Do vậy với sự trân trọng và lòng cảm kích em xin gửi lời cảm ơn chân thành đến ThS. Hoàng Văn Mạnh, thầy không chỉ là người đã hướng dẫn em trong suốt quá trình hoàn thành khóa luận tốt nghiệp mà còn là người cho em những động lực rất lớn để em không ngừng sáng tạo và phát triển sản phẩm, nâng cao chất lượng nghiên cứu.

Xin cảm ơn thầy cô trong khoa Cơ học kỹ thuật và cũng như thầy cô trong toàn trường Đại học Công nghệ - ĐHQGHN đã truyền đạt nhiều kiến thức quý báu trong suốt 4 năm học, những kiến thức đó là chìa khóa giúp em thực hiện được những hoài bão trong cuộc sống và là hành trang cho em bước vào thời kỳ phát triển mới của đất nước.

Cảm ơn Nguyễn Xuân Tiến, người bạn đã sát cánh cùng em qua nhiều khó khăn trong quá trình học tập và tìm hiểu kiến thức khoa học. Tiến cũng là người nhiệt tình giúp đỡ em trong nhiều phần khó khăn của đề tài.

Ngoài ra, em xin cảm ơn bố, mẹ và em trai của em những người đã, đang và luôn là chỗ dựa tinh thần cho em mọi lúc mọi nơi.

Trong quá trình làm luận văn và thực hiện đề tài, dù đã cố gắng tuy nhiên không thể tránh được sai sót không đáng có. Em rất mong thầy cô chỉ bao thêm giúp em hoàn thành và đạt kết quả tốt hơn nữa. Em xin chân thành cảm ơn

Hà Nội, ngày 23 tháng 4 năm 2018

Sinh viên

LỜI CAM ĐOAN

Em xin cam đoan đây là công trình tìm hiểu, nghiên cứu và chế tạo độc lập của riêng em, phục vụ cho khóa luận tốt nghiệp, dưới sự hướng dẫn của ThS. Hoàng Văn Mạnh. Các số liệu sử dụng phân tích trong luận án đều có nguồn gốc rõ ràng, đã công bố theo đúng quy định. Các kết quả, thiết kế trong luận án do em tự tìm hiểu phân tích một cách trung thực, khách quan và chưa được công bố trong các công trình khác. Nếu không đúng như đã nếu trên, em xin hoàn toàn chịu trách nhiệm về đề tài của mình.

Hà Nội ngày 23, tháng 04, năm 2018

Sinh viên

PHẠM TIẾN THÀNH

MỤC LỤC

DANH MỤC HÌNH ẢNH

CÁC KÝ HIỆU VIẾT TẮT

DMP	Digital Motion Processor
ADC	Analog Digital Converter
DAC	Digital Analog Converter
dps	degree per second
FIFO	First in first out
CMOS	Complementary Metal-Oxide-Semiconductor
MEMS	Micro Electro Mechanical Systems
APB	all-points bulletin
PWM	Pulse Width Modulation
RPM	Revolutions Per Minute

MỞ ĐẦU

❖ *Tính cần thiết của đề tài*

Ngày nay hệ thống cơ điện tử đã và đang xuất hiện trong mọi mặt của cuộc sống. Các công ty đang hướng đến trải nghiệm của người dùng bằng cách sử dụng các hệ thống cơ điện tử trong các ứng dụng của họ. Máy giặt có thể thay đổi nhiệt độ nước để phù hợp với từng loại vải, điều hòa có thể tự tìm kiếm người sử dụng và hướng dòng khí tươi mát đến họ,... tất cả đều là kết quả của việc ứng dụng hệ thống cơ điện tử vào sản phẩm.

Nhu cầu của xã hội và người sử dụng ngày càng cao và tinh vi, đòi hỏi người kỹ sư cũng như các nhà sản xuất ngày càng phải sử dụng nhiều công nghệ mới hơn, nhanh hơn, mạnh mẽ hơn vào quá trình phát triển sản phẩm. Vì lý do này mà hiện này các cảm biến không ngừng phát triển và có thêm nhiều loại cảm biến mới, các vi xử lý 32 bit cũng được phát triển và cải thiện đáng kể về tốc độ so với dòng vi điều khiển 8 bit truyền thống. Điều này là cơ hội đối với các kỹ sư phát triển phần mềm giúp họ có thể tạo ra được các sản phẩm đáp ứng được người sử dụng, nhưng đó cũng là thách thức đối với các kỹ sư bởi tính mới và phương pháp sử dụng.

Do đó đề tài "Nghiên cứu, thiết kế xe tự cân bằng sử dụng cảm biến gia tốc 3 trục và vi điều khiển 32 bit" bằng việc sử dụng cảm biến gia tốc 3 trục và dòng vi điều khiển mới 32 bit sẽ là một ví dụ thiết thực về việc thiết kế hệ thống cơ điện tử và có thể ứng dụng cho các hệ thống cơ điện tử khác.

❖ *Ý nghĩa khoa học và thực tiễn*

Ý nghĩa thực tiễn: Khóa luận mang tính chất nghiên cứu và thiết kế một hệ thống cơ điện tử điển hình.

Ý nghĩa khoa học: Việc phát triển xe hai bánh tự cân bằng là nền tảng để phát triển các loại robot cân bằng di chuyển bằng hai chân trong tương lại.

❖ *Đối tượng và phương pháp nghiên cứu, thiết kế*

Xe tự cân bằng chính là một hệ thống cơ điện tử, các đối tượng trong hệ thống cơ điện tử cơ bản gồm có: cảm biến, vi điều khiển và phần cơ học chấp hành[3]. Xoay quanh một hệ thống cơ điện tử thì sẽ có rất nhiều vấn đề có thể mổ xẻ để giải quyết Nhưng trong khuôn khổ khóa luận và nội dung đề tài em tập chung chính vào 2 đối tượng của hệ thống cơ điện tử và 1 đối tượng tổng quan đó là:

- Cảm biến gia tốc 3 trục

- Vi điều khiển 32 bit
- Lý thuyết xe tự cân bằng

❖ ***Nội dung nghiên cứu, thiết kế***

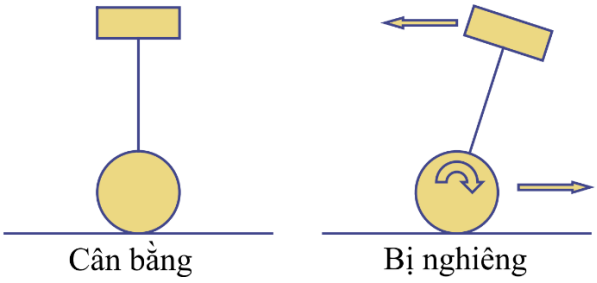
CHƯƠNG 1. TÌM HIỂU VỀ HỆ THỐNG XE HAI BÁNH TỰ CÂN BẰNG

CHƯƠNG 2: NGHIÊN CỨU CẢM BIẾN GIA TỐC 3 TRỤC MPU6050 VÀ VI ĐIỀU KHIẾN 32 BIT STM32F103C8T6

CHƯƠNG 3: THIẾT KẾ XE TỰ CÂN BẰNG

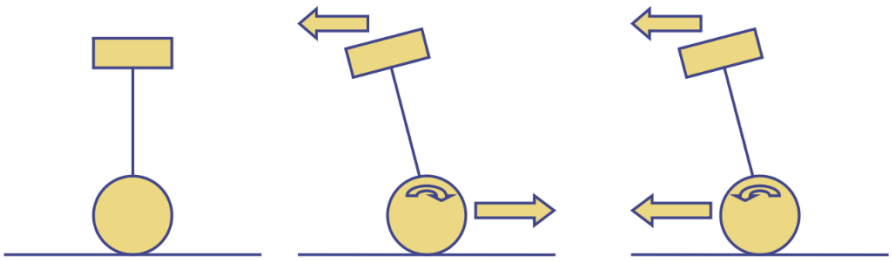
CHƯƠNG 1. TÌM HIỂU VỀ HỆ THỐNG XE HAI BÁNH TỰ CÂN BẰNG

1.1. Thế nào là xe hai bánh tự cân bằng



Hình 1.1. Nguyên lý giữ thăng bằng

Đối với các loại xe ba hay bốn bánh thì việc giữ thăng bằng và ổn định là nhờ trọng tâm chủa chúng nằm trong bề mặt chân đế do các bánh xe tạo nên. Còn đối với xe hai bánh tự cân bằng (hai bánh song song) để giữ thăng bằng cho xe trọng tâm của xe phải cần được giữ nằm cân bằng ngay giữa hai bánh xe[2]. Giống như việc chúng ta giữ một cây gậy thẳng đúng cân bằng trong lòng bàn tay vậy.



Hình 1.2. Mô tả cách thức di chuyển

Tuy nhiên việc xác định trọng tâm của xe là tương đối khó khăn do hình dạng xe và các vật gắng trên xe có phần phức tạp. Tuy nhiên chúng ta hoàn toàn có thể xác định được góc giữa mặt phẳng sàn nhà và xe cũng như trọng lực. Do đó, thay vì đi tìm trọng tâm nằm giữa hai bánh ta cần giữ cho xe luôn thẳng đứng, vương góc với mặt sàn.

Đi kèm với đó là việc khi ta giữ một góc giữa xe và mặt sàn khác 90^0 thì xe có xu hướng tiến hoặc lùi (tùy theo quy ước) về phía trước hoặc sau. Đây chính là nguyên lý để vân hành xe di chuyển. Để dừng lại thì ta chỉ cần điều chỉnh lại góc về giá trị 90^0.

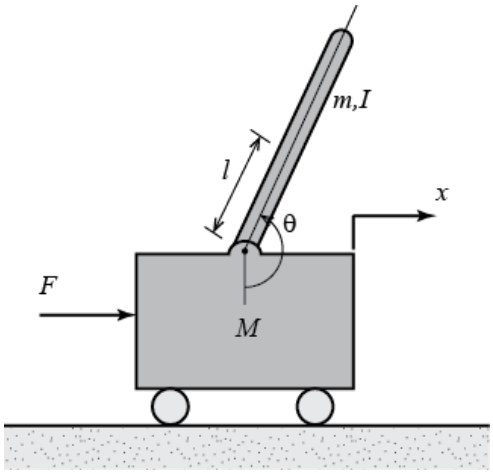
1.2. Phương trình toán học, phương pháp tính toán động lực học của xe hai bánh tự cân bằng

Có nhiều phương pháp dùng để tính động lực học, chẳng hạn: phương pháp Newton, phương pháp Lagrange, phương pháp theo năng lượng…Nhưng trong đề tài này, phương pháp Newton được sử dụng với các ưu điểm của nó. Thứ nhất, nó sử dụng

các phương pháp tính cơ học thông thường. Thứ hai, các công thức và hệ phương trình trong quá trình tính không quá phức tạp. Thứ ba, kết quả tính động lực học của mô hình con lắc ngược được phổ biến hiện nay ở các tài liệu tham khảo được sử dụng để kiểm tra sự sai sót trong quá trình tính toán động lực học của mô hình xe hai bánh tự cân bằng.

Bên cạnh các ưu điểm này, nó vẫn có nhược điểm là phải tuyến tính hóa tính toán tại vị trí góc $\alpha = 0^0$. Tuy nhiên việc này không trầm trọng trong mô hình của đề tài, vì mô hình chỉ hoạt động xung quanh vị trí $0^0(\pm 10^0)$.

❖ *Nền tảng mô hình toán học, phương trình con lắc ngược*[5]



Hình 1.3. Mô hình con lắc ngược

Ta xem xét mô hình toán học của con lắc ngược với các tham số như sau:

(M) Khối lượng xe (kg)

(m) Khối lượng con lắc (kg)

(b) Ma sát của xe (N)

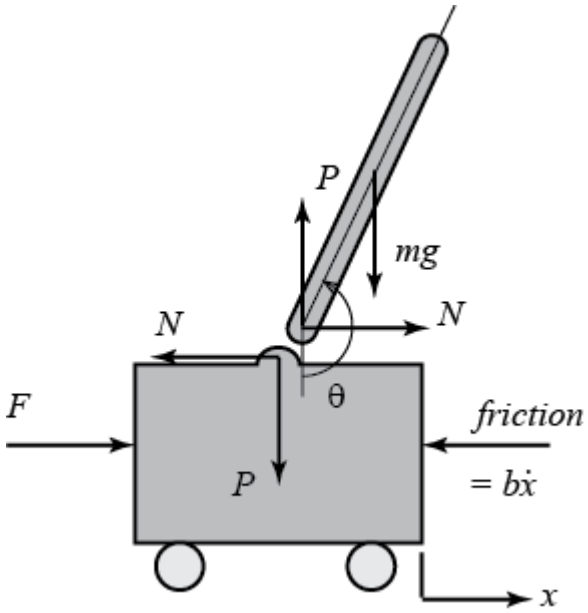
(l) chiều cài ½ con lắc (m)

(I) Momen quan tính của con lắc (Nm)

(F) Lực tác độn vào xe (N)

(x) Vị trí của xe (m)

(θ) Góc của con lắc so với phương thẳng đứng (rad)



Hình 1.4. Phân tích lực trên xe và trên con lắc

Tổng hợp các lực trong sơ đồ tác dụng lên xe theo hướng ngang, ta sẽ có được phương trình chuyển động sau đây.

$$M\ddot{x} + b\dot{x} + N = F \tag{1}$$

Tổng hợp các lực trong sơ đồ tác dụng lên con lắc theo hướng ngang, ta sẽ có biểu hiện sau đây cho lực phản ứng N.

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{2}$$

Thay thế phương trình này vào phương trình đầu tiên, ta sẽ có được một trong hai phương trình điều khiển cho hệ thống này.

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \tag{3}$$

Để có được phương trình thứ hai của chuyển động cho hệ thống này, ta tổng hợp các lực vuông góc với con lắc. Giải quyết hệ thống dọc theo phương này bằng cách đơn giản hóa toán học. Ta sẽ nhận được phương trình sau đây.

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \tag{4}$$

Để thoát khỏi P và N thuật ngữ trong phương trình ở trên, tổng hợp momen tại khối tâm của con lắc để có được phương trình sau đây.

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \tag{5}$$

Kết hợp hai biểu thức cuối cùng này, bạn sẽ có được phương trình điều khiển thứ hai.

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \quad (6)$$

Vì các kỹ thuật phân tích và kiểm soát mà chúng ta sẽ sử dụng trong ví dụ này chỉ áp dụng cho các hệ thống tuyến tính nên tập hợp các phương trình này cần được tuyến tính hóa. Cụ thể, chúng ta sẽ linearize phương trình về vị trí cân bằng trở lên,$\theta = \pi$, và sẽ cho rằng hệ thống nằm trong một khu phố nhỏ của equillbrium này. Giả định này phải có giá trị hợp lý vì được kiểm soát mà chúng tôi mong muốn rằng con lắc không lệch hơn 20 độ so với vị trí đứng lên trên. Để cho ϕ đại diện cho độ lệch của vị trí của bàn đạp từ cân bằng, nghĩa là,$\theta = \pi + \phi$. Một lần nữa giả sử một độ lệch nhỏ (ϕ) từ cân bằng, chúng ta có thể sử dụng các phép tính xấp xỉ nhỏ sau đây của các hàm phi tuyến trong các phương trình hệ của chúng ta:

$$\cos\theta = \cos(\pi + \phi) \approx -1 \quad (7)$$

$$\sin\theta = \sin(\pi + \phi) \approx -\phi \quad (8)$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0 \quad (9)$$

Sau khi thay thế xấp xỉ trên vào phương trình điều phối phi tuyến của chúng ta, chúng ta đi đến hai phương trình tuyến tính của chuyển động. chú thích u đã được thay thế cho đầu vào F

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \quad (10)$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \quad (11)$$

1.3. Tại sao phải thiết kế xe hai bánh tự cân bằng

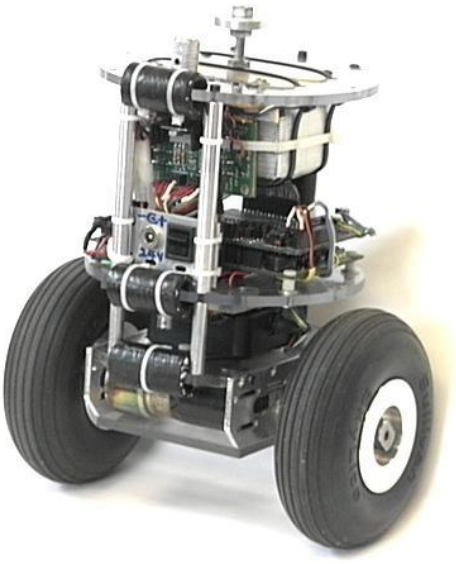
Các dòng xe nhiều hơn 3 bánh trong thực tế là chủ yếu, do việc giữ thăng bằng là đơn giản. Tuy nhiên với nhiều trường hợp như nghiên cứu các robot di chuyển trong địa hình hẹp thì việc sử dụng các robot tự cân bằng 2 bánh mang lại ưu điểm rất lớn ví dụ như xe có thể lập tức quay ngoắt 180 độ trong khoảng không hẹp.

Hơn nữa với địa hình gồ ghề phức tạp như dốc lớn thì việc trọng tâm của các xe lớn hơn 3 bánh lại gặp khó khăn. Ngược lại, các xe hai bánh tự cân bằng lại rất linh động trong việc giữ thăng bằng khi di chuyển trên địa hình phức tạp.

Không chỉ thế việc phát triển xe hai bánh tự cân bằng còn là ý tưởng cho việc thiết kế các robot giống người di chuyển và giữ cân bằng trên hai chân. Vì vậy việc phát triển

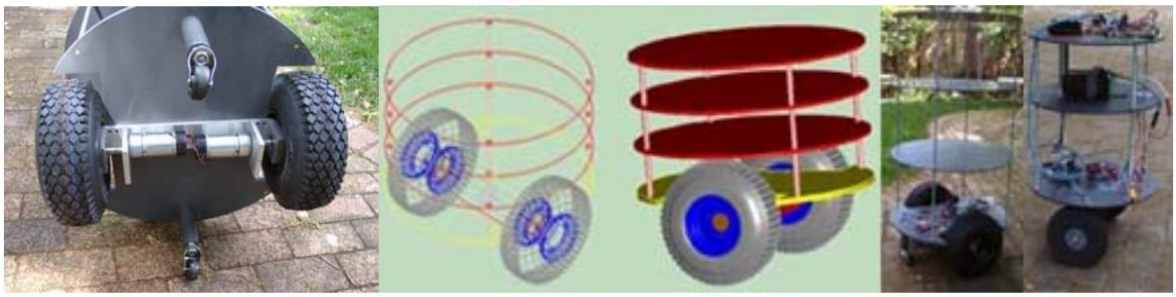
và thiết kế xe hai bánh tự cân bằng có một tầm quan trọng nhất định trong sự phát triển của khoa học công nghệ hiện nay của nước ta cũng như trên thế giới.

1.4. Một số hình ảnh về xe hai bánh tự cân bằng đã được tiến hành



Hình 1.5. Xe tự cân bằng nBot

nBot do David P. Anderson sangs chees. Nbot được lấy ý tưởng cân bằng như sau: các bánh xe sẽ phải chạy theo hướng mà phần trên robot sắp ngã. Nếu bánh xe có thể lái theo cách đứng vững trọng tâm robot, robot sẽ vẫn giữ cân bằng[6].



Hình 1.6. Xe tự cân bằng Bender

Robot cân bằng Bender là đề án do TedLarson, San Francisco thực hiện. Mục tiêu hiện tại của ông là xây dựng robot tự cân bằng trên mặt sàn và từ đó dùng làm nền cơ bản để xây dựng robot tự hành dùng bánh xe.

Hình 1.7. Xe điện Segway

Segway là thương hiệu toàn cầu về cung cấp dòng xe tự hành. Thành lập từ năm 1999 bởi một trong những nhà sáng chế hàng đầu thế giới bấy giờ – ông Dean Kamen.Tính đến nay, với hơn 18 năm kinh nghiệm, Segway đã khẳng định mình là một thương hiệu dẫn đầu về dòng xe điện cân bằng cao cấp. Trên đây là một sản phẩm thương mại hóa của xe tự cân bằng[7].

Hình 1.8. Balancing scooter

Trevor Blackwell chế tạo ra xe scooter dựa theo Segway của hãng Mỹ. Xe scoter tự cân bằng này được xây dựng từ những bộ phận giống động cơ xe lăn và từ các cục pin xe RC. Những bộ phần và module để chế tạo có giá thành thấp hơn phân nửa Segway. Nó không cần phầm mềm thực thi cao hay phức tạp. Phiên bản đầu tiên được viết bằng Python và sử dụng port số để truyền thông tin đến con quay hồi chuyển và mạch điều khiển động cơ[8].

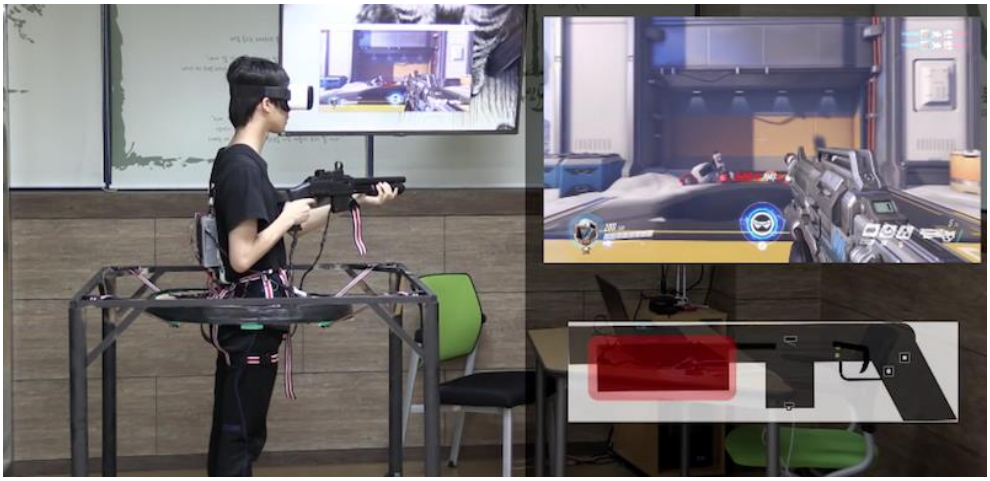
CHƯƠNG 2: NGHIÊN CỨU CẢM BIẾN GIA TỐC 3 TRỤC MPU6050 VÀ VI ĐIỀU KHIỂN 32 BIT STM32F103C8T6

2.1. Cảm biến là gì? Tầm quan trọng của cảm biến

Cảm biến là các phần tử nhạy cảm dùng để biến đổi các đại lượng đo lường, kiểm tra hay điều khiển từ dạng này sang dạng khác thuận tiện hơn cho việc tác động của các phần tử khác. Cảm biến là một thiết bị chịu tác động của đại lượng cần đo m không có tính chất điện và cho một đặc trưng mang bản chất điện (như điện tích, điện áp, dòng điện, trở kháng) kí hiệu là s có $s = F(m)$. Cảm biến thường dùng ở khâu đo lường và kiểm tra. Căn cứ theo dạng đại lượng đầu vào người ta phân ra các loại cảm biến như: cảm biến chuyển dịch thẳng, chuyển dịch góc quay, tốc độ, gia tốc, mô men quay, nhiệt độ, áp suất, quang, bức xạ,...

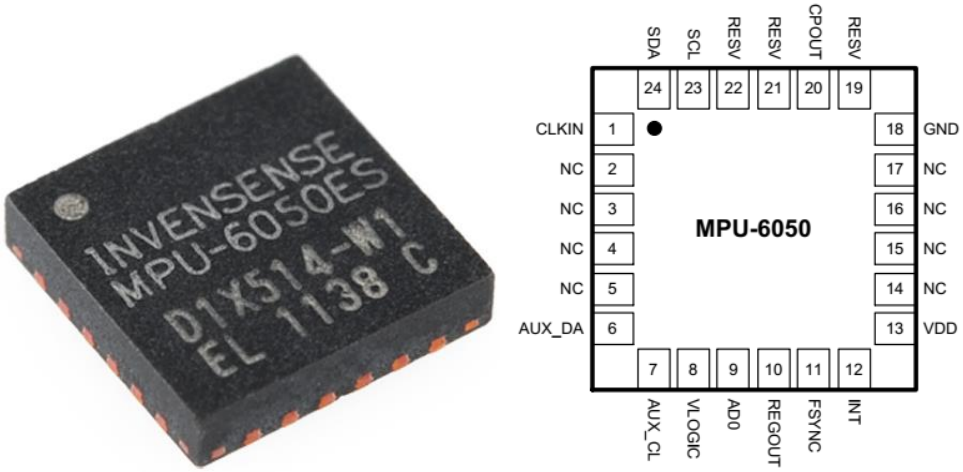
Hiện nay công nghệ về cảm biến được ứng dụng rất nhiều trong khoa học kỹ thuật, cảm biến chính là công cụ không thể thiếu để số hóa các đại lượng đo lường trong cuộc sống, thay thế cho các thiết bị đo tương tự truyền thống ví dụ như: Cảm biến nhiệt độ, độ ẩm (DHT11, DHT22) thay thế cho nhiệt kế thủy ngân, nhiệt kế lò xo; Cảm biến siêu âm (SRF04, SRF05) giúp xác định độ xa thay thế cho một số thước truyền thống; Cảm biến nhịp tim sử dụng ánh sáng thay thế cho việc đo nhịp tim thông thường,... Không những thế các cảm biến còn giúp xác định được thêm những đại lượng tương đối khó đo được trong tự nhiên như cảm biến ánh sáng (LS6b), cảm biến vân tay,... Và ưu điểm mà cảm biến mang lại là rất lớn không những giúp số hóa các đại lượng để thu thập phân tích một cách dễ dàng mà tốc độ làm việc và độ chính xác của các cảm biến hiện nay là rất cao. Điều này khiến tầm quan trọng của cảm biến trong các ngành khoa học kỹ thuật nói chung và ngành cơ điện tử nói riêng là không thể không nhắc đến. Việc nghiên cứu về cảm biến để hiểu và sử dụng thành thạo một cảm biến đóng vai trò rất lớn trong thành công của một đề tài cơ điện tử.

2.2. Cảm biến gia tốc 3 trục MPU6050



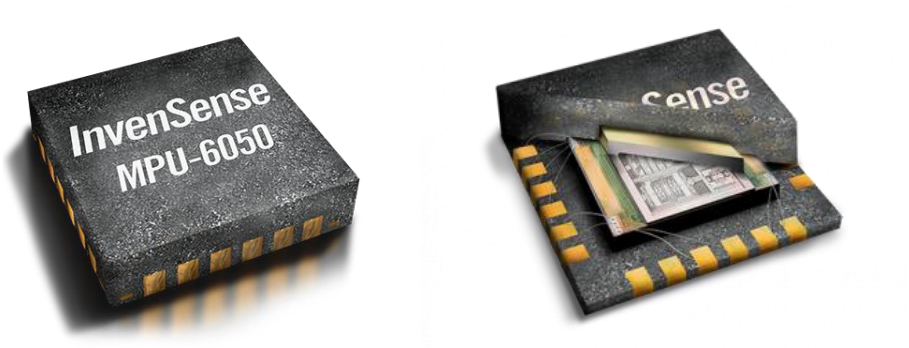
Hình 2.1. Ứng dụng cảm biến chuyện động trong trò chơi thực tế ảo

Cảm biến chuyển động từ lâu nay được các nhà sáng chế smartphone, table,... coi là phần không thể thiếu trong các thiết bị của họ bởi giá trị to lớn mà nó mang lại đến trải nghiệm của người sử dụng. Trong smartphone, cảm biến chuyển động được các ứng dụng với các chức năng như là nhận biển cử chỉ và điều khiển thiết bị, tăng cường điều khiển cho các trò chơi chuyển động thực, chụp ảnh toàn cảnh, ứng dụng đi bộ và chỉ đường phương tiện,... Với khả năng theo dõi chính xác chuyển động người dùng của nó, công nghệ theo dõi chuyển động có thể biến các thiết bị di động cầm tay hay máy tính bảng trở thành thiết bị 3D thông minh được sử dụng trong nhiều ứng dụng từ sức khỏe, thể dục đến các dịch vụ dựa trên vị trí. Các yêu cầu chính cho cảm biến chuyển động là nhỏ gọn, tiêu thụ năng lượng thấp, độ chính xác cao trong nhiều lần sử dụng, chống chịu được sự va đập và đơn giản trong việc lập trình – tất cả những điều này đều cần nằm trong một mức giá tiêu dùng thấp.



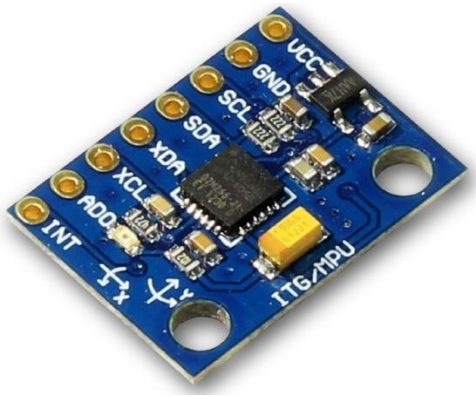
Hình 2.2. Cảm biến MPU6050 và sơ đồ chân

Cảm biến MPU6050 một cảm biến của hãng InvenSense là thiết bị tích hợp 6 trục cảm biến chuyển động đầu tiên trên thế giới nó phối hợp 3 trục con quay hồi chuyển, 3 trục gia tốc kế và bộ xử lý chuyển động số (DMP) trong một chíp nhỏ $4 \times 4 \times 0.9\ mm$. Với đường truyền I^2C chuyên dụng của nó, nó trực tiếp chấp nhận đầu vào từ la bàn 3 trục bên ngoài để cung cấp đầu ra 6 trục hoàn chỉnh. Tích hợp phần hiệu chỉnh thời gian chạy cho phép nhà sản xuất loại bỏ được sự phức tạp, chi phí và mức độ tích hợp của các thiết bị rời rạc, đảm bảo chuyển động tối ưu hiệu suất cho người sử dụng. Cảm biến MPU6050 cũng được thiết kế để giao tiếp với nhiều cảm biến kỹ thuật số phi quán tính như cảm biến áp suất, trên cổng I^2C phụ trợ của nó.



Hình 2.3. Mô hình thiết kế MPU6050

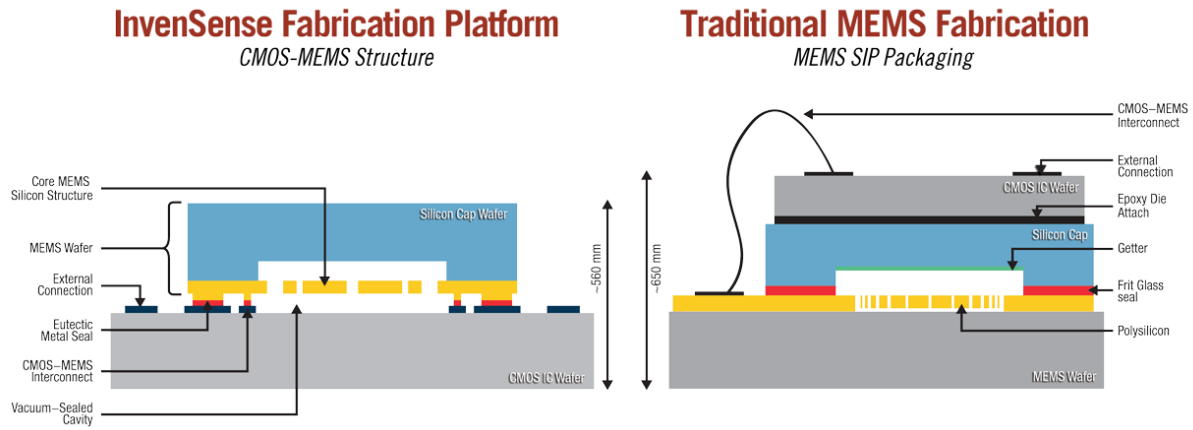
Cảm biến MPU6050 có 3 bộ chuyển đổi tương tự sang số (ADCs) dùng cho số hóa tín hiệu ra con quay và 3 bộ ADCs dùng cho số hóa tín hiệu ra gia tốc kế. Để việc theo dõi cả chuyển động nhanh và chuyển động chậm đều được chính xác, lập trình viên có thể đặt được khoảng giá trị lấy mẫu của con quay hồi chuyển trong khoảng các giá trị $\pm 250, \pm 500, \pm 1000$ và $\pm 2000^0/sec$ (dps), tương tự lập trình viên có thể đặt được khoảng giá trị vận tốc của gia tốc kế trong khoảng giá trị $\pm 2g, \pm 4g, \pm 8g$ và $\pm 16g$.



Hình 2.4. Cảm biến MPU6050 trong module đã ra chân

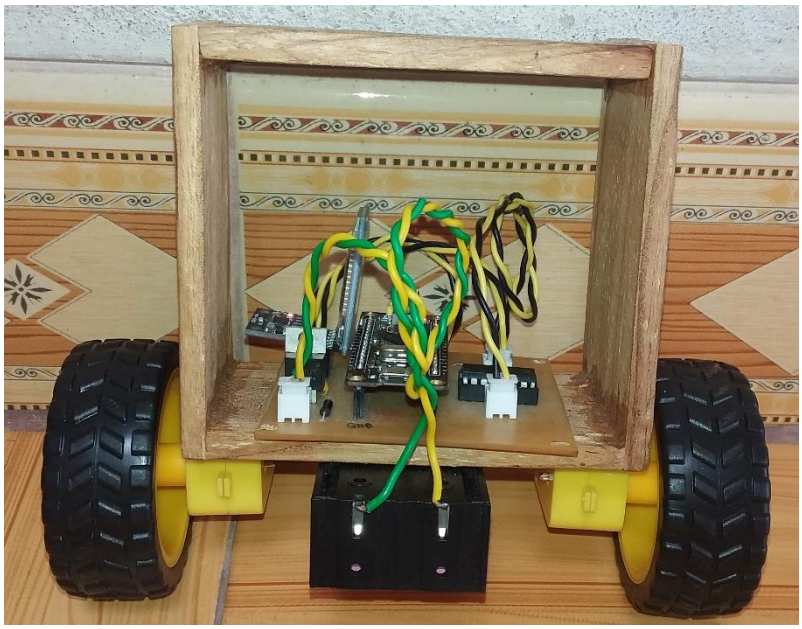
Một bộ đệm 1024 Byte FIFO trên cảm biến giúp mức tiêu thụ điện của hệ thống thấp hơn bằng cách cho phép bộ vi điều khiển đọc dữ liệu cảm biến trên các trong các

chuyển động và sau đó vào chế độ năng lượng thấp khi đó cảm biến thu thập được nhiều dữ liệu. Với toàn bộ tiến trình và các bộ phận cần thiết trên cảm biến để hỗ trợ rất nhiều chuyển động dựa trên các trường hợp sử dụng, cảm biến MPU6050 chỉ cho phép duy nhất các ứng ứng dụng năng lượng thấp truy cập với yêu cầu xử lý đã được giảm thiểu cho bộ xử lý hệ thống. Băng việc cung cấp một ngõ ra tích hợp chuyển động dung hợp, bộ DMP trong cảm biến xuất ra yêu cầu tính toán xử lý chuyên sâu từ hệ thống vi điều khiển, giảm thiểu nhu cầu lấy dữ liệu thường xuyên của đầu ra cảm biến chuyển động. Cảm biến còn được gắn thêm cảm biến nhiệt độ với sai số 1% nhiệt độ trên nhiệt độ hoạt động của chip.



Hình 2.5. Mô hình nền tảng InvenSense và mô hình MEMS truyền thống

Giao tiếp với toàn bộ thanh ghi của vi cảm biến bằng việc sử dụng đường truyền I^2C ở tần số $400kHz$. Tận dụng nền tảng Nasiri-Fabrication đã được cấp bằng sáng chế và đã được chứng minh, nó tích hợp các tấm silic NEMS với CMOS xuyên qua lớp silic, invenSense đã làm cảm biến MPU6050 giảm kích thước mang tính đột phá về công nghệ $4 \times 4 \times 0.9mm$, trong khi nó mang đến hiệu quả cao nhất, mức độ ồn nhỏ nhất và chi phí nhỏ nhất về lượng chất bán dẫn cho các thiết bị điện tử. Bộ phận này có khả năng chống sốc $10000g$ mạnh mẽ và có bộ lọc thông thấp có thể lập trình cho các con quay hồi chuyển, gia tốc kế và cảm biến nhiệt độ trên chip.



Hình 2.6. Sản phẩm xe hai bánh tự cân bằng

Cảm biến được sử dụng trong một số ứng dụng như: công nghệ BlurFree (làm mở trong các hình ảnh/ video), công nghệ AirSign (AirSign như một khinh khí cầu sử dụng trong bảo mật và đánh dấu), công nghệ touchAnywhere, công nghệ MotionCommand (Điều khiển bằng chuyển động), cho phép chuyển động trong các game, nhận biết tư thế, dịch vụ dựa trên vị trí, thiết bị sức khỏe, thể dục, thể thao đeo trên người, đồ chơi...

2.3. Lịch sử vi điều khiển 32 bit

2.3.1. Vi xử lý

Vi xử lý được chế tạo từ các tranzito tích hợp trên một vi mạch tích hợp đơn. Xuất hiện lần đầu tiên vào những năm đầu của thập kỷ 70 của thế kỷ 20. Sử dụng mã BCD trên nền 4 bit. Các vi xử lý 4 bit và 8 bit được sử dụng trong các thiết bị đầu cuối, máy in, các hệ thống tự động...Đến giữa những năm 1970 thì lần đầu tiên các vi xử lý 8 bit với 16 bit địa chỉ được sử dụng như máy tính đa mục đích. Các hãng sản xuất vi xử lý đầu tiên ở thời điểm này là Intel, Texas Instruments và Garrett AiResearch với ba dòng chip tương ứng: Intel 4004, TMS 1000 và Central Air Data Computer. Đây là những vi xử lý 4 bit. Sau sự ra đời của các vi xử lý 4 bit thì các hãng cho ra đời các dòng 8 bit, 12 bit, 16 bit, 32 bit, 64 bit. Intel 8008 là vi xử lý 8 bit đầu tiên trên thế giới được sản xuất năm 1972. Tiếp sau thành công của 8008 là các phiên bản như 8080 (1974), Zilog Z80 (1976). Các vi xử lý của Motorola 6800 được phát hành tháng 8 năm 1974 và MOS technology ra đời năm 1975. Intersil 6100 là vi xử lý 12 bit, từ khi được sản xuất bởi công ty Harris nó được biết đến với tên HM-6100 được sử dụng trong quân đội suốt thập niên 1980. Vi xử lý 16 bit đầu tiên được giới thiệu bởi hãng National Semiconductor IMP-16 vào năm 1973 đây là vi xử lý đa chip. Đến năm 1975 hãng này giới thiệp vi xử lý đơn chip đầu tiên. Hãng Texas Instruments ra đời vi xử lý 16 bit đơn chip TI-990 sử dụng như một máy tính mini. Intel cũng cho ra đời dòng vi xử lý 16 bit lấy tên 8086. Vi xử lý 16 bit chỉ xuất hiện trên thị trường một thời gian ngắn thì dòng 32 bit đã bắt đầu xuất hiện. MC6800 là vi xử lý 32 bit đầu tiên của hãng Motorola, họ 68k có 32 bit thanh ghi nhưng sử dụng đường dẫn dữ liệu 16 bit bên trong và 16 bit dữ liệu bên ngoài để giảm số lượng pin, hỗ trợ 24 bit địa chỉ. Motorola thường được biết đến như vi xử lý 16 bit mặc dù nó có cấu trúc 32 bit. Vi xử lý 32 bit đầy đủ đầu tiên là AT&T Bell Labs BELLMAC-32A với mẫu đầu tiên vào năm 1980 và sản xuất năm 1982. Vi xử lý 32 bit đầu tiên của Intel là dòng iAPX 432 được giới thiệu năm 1981 nhưng không thu được thành công. Vi xử lý ARM đầu tiên ra đời năm 1985 với thiết kế RISC viết tắt của reduced instruction set computer máy tính có tập lệnh rút gọn, các vi xử lý ARM được sử dụng chủ yếu trong các điện thoại di động. Vi xử lý 64 bit được thiết kế cho các máy tính cá nhân. Nó được thiết kế vào đầu những năm 1990 đến đầu những năm 2000 chứng kiến vi xử lý 64 bit nhằm vào thị trường máy tính. Vi xử lý AMD 64 bit tương thích ngược với x86, x86-64 còn gọi là AMD64 trong tháng 9 năm 2003, tiếp sau thành công của Intel64. Kỷ nguyên của máy tính 64 bit đã bắt đầu.

2.3.2. Vi điều khiển

Vi điều khiển là một máy tính được tích hợp trên một chíp, nó thường được sử dụng để điều khiển các thiết bị điện tử. Vi điều khiển thực chất gồm một vi xử lý có hiệu suất đủ cao và giá thành thấp (so với các vi xử lý đa năng dùng trong máy tính) kết hợp với các thiết bị ngoại vi như các bộ nhớ, các mô đun vào/ra, các mô đun biến đổi từ số sang tương tự và từ tương tự sang số, mô đun điều chế độ rộng xung (PWM)... Vi điều khiển thường được dùng để xây dựng hệ thống nhúng. Nó xuất hiện nhiều trong các dụng cụ điện tử, thiết bị điện, máy giặt, lò vi sóng, điện thoại, dây truyền tự động... Hầu hết các loại vi điều khiển hiện nay có cấu trúc Harvard là loại cấu trúc mà bộ nhớ chương trình và bộ nhớ dữ liệu được phân biệt riêng. Cấu trúc của một vi điều khiển gồm CPU, bộ nhớ chương trình (thường là bộ nhớ ROM hoặc bộ nhớ Flash), bộ nhớ dữ liệu (RAM), các bộ định thời, các cổng vào/ra để giao tiếp với các thiết bị bên ngoài, tất cả các khối này được tích hợp trên một vi mạch. Các loại vi điều khiển trên thị trường hiện nay:

- Freescale 68HC11 (8-bit)
- Intel 8051
- STMicroelectronics STM8S (8-bit), ST10 (16-bit) và STM32 (32-bit)
- Atmel AVR (8-bit), AVR32 (32-bit), và AT91SAM (32-bit)
- Freescale ColdFire (32-bit) và S08 (8-bit)
- Hitachi H8 (8-bit), Hitachi SuperH (32-bit)
- MIPS (32-bit PIC32)
- PIC (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24)
- PowerPC ISE
- PSoC (Programmable System-on-Chip)
- Texas Instruments Microcontrollers MSP430 (16-bit), C2000 (32-bit), và Stellaris (32-bit)
- Toshiba TLCS-870 (8-bit/16-bit)
- Zilog eZ8 (16-bit), eZ80 (8-bit)
- Philips Semiconductors LPC2000, LPC900, LPC700

2.3.3. Vi điều khiển ARM

Cấu trúc ARM (viết tắt từ tên gốc là Acorn RISC Machine) là một loại cấu trúc vi xử lý 32-bit kiểu RISC được sử dụng rộng rãi trong các thiết kế nhúng. Được phát triển lần đầu trong một dự án của công ty máy tính Acorn. Do có đặc điểm tiết kiệm năng lượng, các bộ CPU ARM chiếm ưu thế trong các sản phẩm điện tử di động, mà với các sản phẩm này việc tiêu tán công suất thấp là một mục tiêu thiết kế quan trọng hàng đầu.

Ngày nay, hơn 75% CPU nhúng 32-bit là thuộc họ ARM, điều này khiến ARM trở thành cấu trúc 32-bit được sản xuất nhiều nhất trên thế giới. CPU ARM được tìm thấy khắp nơi trong các sản phẩm thương mại điện tử, từ thiết bị cầm tay (PDA, điện thoại di động, máy đa phương tiện, máy trò chơi cầm tay, và máy tính cầm tay) cho đến các thiết bị ngoại vi máy tính (ổ đĩa cứng, bộ định tuyến để bàn.). Một nhánh nổi tiếng của họ ARM là các vi xử lý Xscale của Intel. Giới thiệu về vi điều khiển LPC2148: Là dòng vi điều khiển ARM được sản xuất bởi hãng Philips. Tính năng:

- Vi điều khiển 16/32-bit ARM7TDMI-S
- 40k RAM tĩnh (8k +32k), 512k flash
- Tích hợp USB 2.0
- Hộ trợ hai bộ ADC 10 bit
- Một bộ DAC 10 bit
- 2 bộ timer 32 bit, 6 ngõ điều chế độ rộng xung
- Đồng hồ thời gian thực hỗ trợ tần số 32kHz
- Khả năng thiết lập chế độ ưu tiên và định địa chỉ cho ngắt
- 45 chân GPIO vào ra đa dụng
- 9 chân ngắt ngoài (tích cực cạnh hoặc tích cực mức)
- CPU clock đạt tối đa 60MHz thông qua bộ PLL lập trình được
- Xung PLCK hoạt động độc lập.

On-chip Flash Memory: LPC 2148 có 512K bộ nhớ Flash có thể được dùng để lưu trữ code và dữ liệu. Trong khi thực thi ứng dụng, vẫn có thể xóa hoặc lập trình Flash thông qua IAP (In Application Programming). Khi đó trình loader trên chip được sử dụng, bộ nhớ trống còn lại là 500K. Bộ nhớ Flash có thể ghi xóa được ít nhất 100000 lần, lưu trữ dữ liệu đến 20 năm. On-chip Static RAM: LPC 2148 có 32K RAM tĩnh, có thể được truy xuất theo đơn vị byte, half word & word. Bộ điều khiển SRAM sử dụng phương thức write-back buffer để ngăn chặn tình trạng treo CPU khi có thao tác ghi. Bộ đệm luôn giữ dữ liệu cuối cùng từ chương trình gửi tới bộ nhớ. Dữ liệu chỉ được ghi vào SRAM khi có 1 thao tác ghi khác từ chương trình. Lập trình cho ARM: Ngôn ngữ lập trình chính cho ARM hiện nay là ngôn ngữ C. Các trình biên dịch cho ARM thường được dùng:

- Keil ARM.
- IAR.
- HTPICC for ARM.
- ImageCraft ICCV7 for ARM

2.3.4. STMicroelectronics

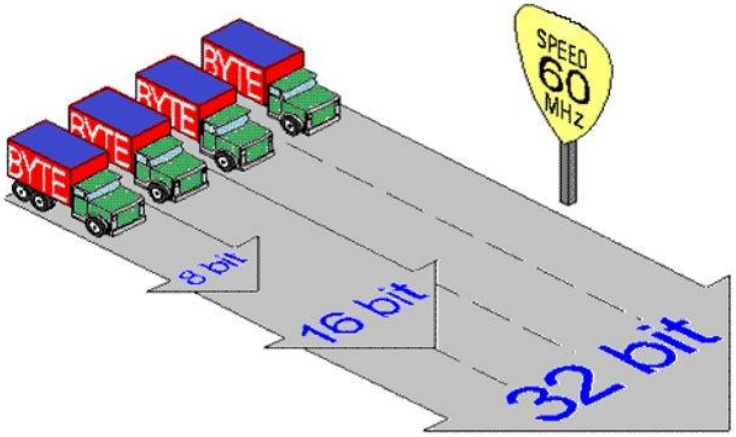
Hình 2.7. Logo hãng STMicroelectronics

Đề tài sử dụng vi điều khiển STM32F103C8T6 một trong những dòng vi điều khiển của hãng STMicroelectronics. Hãng STMicroelectronics là công ty hàng đầu trên thế giới trong việc cung cấp các giải pháp bán dẫn có đóng góp tích cực cho cuộc sống của con người, ngày nay và tương lai. Đặc biệt là các giải pháp cho Smart Driving và IoT. Các chuyên gia của công ty nhận định rằng, trong thời gian không xa 80% xe hơi sẽ được khởi động bởi hệ thống chip điện tử. Do đó việc phát triển các dòng vi điều khiển hiện đại sẽ hoàn toàn phù hợp trong thời gian tới. Dòng vi điều khiển 32 bit có nhiều ưu điểm và chức năng mới, thuận tiện cho người kỹ sư lập trình. Tốc độ xử lý nhanh (lên đến 72Mhz đối với dòng STM32F1, 168Mhz đối với STM32F4), có ưu điểm cao hơn hẳn so với dòng vi điều khiển 8 bit thông thường. Mặc dù vi điều khiển 8 bit hoàn toàn đáp ứng đề tài, nhưng em vẫn quyết định ứng dụng dòng vi điều khiển 32 bit vào đề tài nhằm tìm hiểu công nghệ cho sự phát triển sau này. Cũng như góp một phần nhỏ tài liệu cho những bạn muốn tìm hiểu về các dòng vi điều khiển 32 bit hiện nay.

2.4. So sánh vi điều khiển 32 bit và vi điều khiển 8 bit

Công nghệ vi điều khiển 8 bit đã tồn tại khoảng nửa thế kỉ. Các thiết bị này cho phép triển khai nhanh chóng và dễ dàng các dự án nhúng nhỏ. Bên cạnh đó, chúng ta được biết đến công nghệ vi điều khiển 32 bit mới hơn, đã đạt được rất nhiều sự nổi tiếng. Các thiết bị này cung cấp gần như tất cả mọi thứ từ sức mạnh xử lý cao đến các thiết bị ngoại vi phong phú cùng với các công cụ lập trình phát triển dễ dàng và nhanh chóng giúp cho người kỹ sư lập trình rất nhiều.

2.4.1. Kiến trúc



Hình 2.8. Mô tả 8 bit, 16 bit, 32 bit

8, 16 hay 32 bit về cơ bản là để xác định kích thước của đường truyền nội bộ và kích thước thanh ghi của vi điều khiển. Vì vậy một vi điều khiển 8 bit sẽ có $2^8 = 258$ byte không gian địa chỉ trong khi vi điều khiển 32 bit sẽ có $2^{32} = 4{,}294{,}967{,}296$ byte hay tính ra là 4GB không gian địa chỉ. Hầu hết các dòng vi điều khiển 8 bit có dòng địa chỉ 16 bit (8 bit thấp và 8 bit cao) cho phép chúng có $2^{16} = 65536$ nghĩa là 64KB bộ nhớ. Do đó, với các ứng dụng cần lớn hơn 64KB thì vi điều khiển 32 bit là một thiên đường cho các kỹ sư cơ điện tử.

Mỗi cổng vào ra cho một vi điều khiển 32 có 16/32 chân mỗi cổng, không giống như 8 chân mỗi cổng cho vi điều khiển 8 bit. Điều này thì có thể không ảnh hưởng nhiều. Nhưng nếu một ứng dụng mà chúng ta chần nhiều chân vào ra thì một vi điều khiển 32 bit sẽ phục vụ rất tốt. Một vi điều khiển 8 bit điển hình sẽ không có số pin trên 64 chân trong khi đó số pin 100-144 là rất phổ biến trong dòng vi điều khiển 32 bit.

2.4.2. Cách thức phát triển phần mềm

Với các vi điều khiển 8 bit thì việc phát triển phần mềm là tương đối dễ dàng. Chúng có vài thanh ghi để tinh chỉnh với quá trình cài đặt dễ dàng. Ví dụ, một vi điều khiển 32 bit điển hình có nhiều cài đặt xung nhịp nội bộ. Các cài đặt này được yêu cầu cài đặt trước khi khởi động vi điều khiển. Điều này thì rất đơn giản với các vi điều khiển 8 bit, chúng ta có thể thực hiện tại nhiều vị trí với việc khoier tạo con trỏ ngăn xếp, mất khoảng 4 dòng lệnh (ví dụ như tinyAVR và megaAVR của Atmel). Nhưng các nhà sản xuất vi điều khiển 32 bit đang cố gắng để phân tích khuân mẫu này và đưa ra các mẫu và thư viện mặc định, làm cho quá trình phát triển đơn giản và nhanh hơn. Chỉ cần sử dụng các thư viện này trong dự án của chúng ta là các khởi tạo sẽ được thực hiện.

2.4.3. Hiệu suất

Vi điều khiển 32 bít cung cấp khả năng tính toán đang kinh ngạc khi chúng có thanh ghi 32 bit. Nếu ta lập trình bằng ngôn ngữ assemble, thì việc giới hạn kích thước thanh ghi là điều hiển nhiên sẽ xảy ra. Ví như khi ta muốn cộng hai số 32 bit trên một vi điều khiển 8 bit thì ta phải chia số đó ra thành nhiều phần sau đó cộng các phần lại với nhau, điều này làm cho việc lập trình tính toán rất phức tạp. Khi ta lập trình bằng ngôn ngữ C việc tính toán trở nên mờ mịt, tất nhiên là vẫn có cách tính toán nhưng sẽ mất nhiều thời gian khi sư dụng vi điều khiển 8 bit. Do đó vi điều khiển 8 bit không phải một lựa chọn hoàn hảo khi lập trình các ứng dụng liên quan đến tính toán cao.

2.4.4. Thiết bị ngoại vi

Hiển nhiên vi điều khiển 8 bit truyền thống sẽ bị thiếu rất nhiều thiết bị ngoại vi. Do các thiết bị ngoại vi ngày càng phát triển. Không những thế nó còn bị giới hạn không gian địa chỉ bộ nhớ. Khi ta muốn sử dụng thêm các thiết bị ngoại vi, các thanh ghi dùng điều khiển chúng sẽ cần đặt vào trong cùng một địa chỉ có sẵn. Bên cạnh đó các vi điều khiển 32 bit có rất nhiều các thiết bị ngoại vi, được cung cấp ADCs, DACs, nhiều timer, phần cứng chuyên dụng cho kết nối,...

Tuy vậy vẫn có một số ít vi điều khiển 8 bit được cung cấp ngoại vi phong phú

2.4.5. Sự tiêu thụ năng lượng

Kiến trúc của vi điều khiển 8 bit được biết đến là tiêu thụ năng lượng ít vì sử dụng ít phần bán dẫn hơn. Hơn nữa vi điều khiển 8 bit cũng đã có lịch sử phát triển lâu dài nên sự tiết kiệm về năng lượng được cải thiện đáng kể. Các vi điều khiển 32 bit thì tiêu thụ năng lượng hơn, mặc dù dòng điện là chỉ khoảng vài mA.

Nhưng với sự ra đời của dòng vi điều khiển 32 bit Cortex M0, mọi thứ đã thay đổi ất nhiều. Công nghệ này được giới thiệu vào năm 2012 và nó được xem như là bộ vi điều khiển ARM tiết kiệm năng lượng nhất kể từ năm 2016.

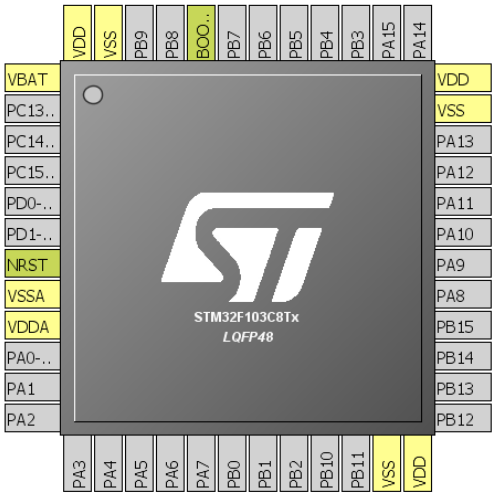
Tóm lại, dòng vi điều khiển 32 bit phù hợp với các ứng dụng cần sự tính toán cường độ cao. Còn vi điều khiển 8 bit thì sử dụng trong các ứng dụng đơn giản với giá thành thấp, mức độ tính toán đơn giản. Hiện nay trên thế giới các kỹ sư và các lập trình viên nhúng đang dần chuyển sang sử dụng vi điều khiển 32 bit vì những ưu điểm của chúng.

2.5. Một số đặc tính của vi điều khiển STM32F103C8T6

Hình 2.9. Vi điều khiển STM32F103C8T6

STM32F103C8T6 là một vi điều kiển sử dụng lõi vi điều khiển ARM Cortex-M3 32 bit (CortexM3 thế hệ sau của CortexM0) hoạt động tại xung nhịp cao 72Mhz, truy xuất bộ nhớ nhúng tốc độ cao (bộ nhớ Flash lên đến 128 Kbytes và bộ nhớ truy cập ngẫu nhiên tĩnh (SRAM) lên đến 20 Kbytes), các cổng được cải tiến hàng loạt và các thiết bị ngoại vi đã được kết nối với đường kết nối ngoại vi nâng cao (APB). Vi điều khiển có 2 bộ chuyển đổi ADC 12 bit, 3 timer thường và một timer PWM, cũng như cổng giao tiếp cơ bản và nâng cao lên đến 2 cổng I^2C và cổng SPI, ba cổng giao tiếp UART, một cổng USB và một cổng CAN.

Thiết bị hoạt động ở mức điện áp $2.0V - 3.6V$. Hoạt động được cả dải nhiệt độ -40^0C đến $+85^0C$ và khoảng rộng nhất là -40^0C đến $+105^0C$. Một bộ điều chế tiết kiệm năng lượng toàn diện cho phép thiết kế các ứng dụng công suất thấp.



Hình 2.10. Sơ đồ chân của vi điều khiển STM32F103C8T6

Dòng vi điều kiển này còn cung cấp nhiều chuẩn chân VFQFPN, UFQFPN, LFBGA, LQFP, UFBGA, LQFP, TFBGA, LQFP cho phép người kỹ sư thiết kế đa dạng sử dụng chip trong nhiều ứng dụng.

Thông số kỹ thuật của vi điều khiển STM32F103C8T6:

- Thể loại: Mạch tích hợp (ICs)
- Dòng vi điều kiển: STM32 F1
- Lõi vi xử lý: ARM® Cortex™-M3
- Kích thước lõi; 32-Bit
- Xung nhịp hoạt động: 72MHz
- Chuẩn giao tiếp: CAN, I²C, IrDA, LIN, SPI, UART/USART, USB
- Thiết bị ngoại vi: DMA, Motor Control PWM, PDR, POR, PVD, PWM, Temp Sensor, WDT
- Số chân I/ O: 37 chân
- Dung lượng bộ nhớ chương trình: 64KB (64K x 8)
- Loại bộ nhớ chương trình: FLASH
- Dung lượng RAM: 20K x 8
- Điện áp sử dụng (Vcc/Vdd): 2 V ~ 3.6 V
- Trình chuyển đổi dữ liệu: A/D 10x12b
- Loại giao động: Giao động nội
- Nhiệt độ hoạt động: -40°C ~ 85°C
- Kiểu chân: 48-LQFP

CHƯƠNG 3: THIẾT KẾ XE TỰ CÂN BẰNG

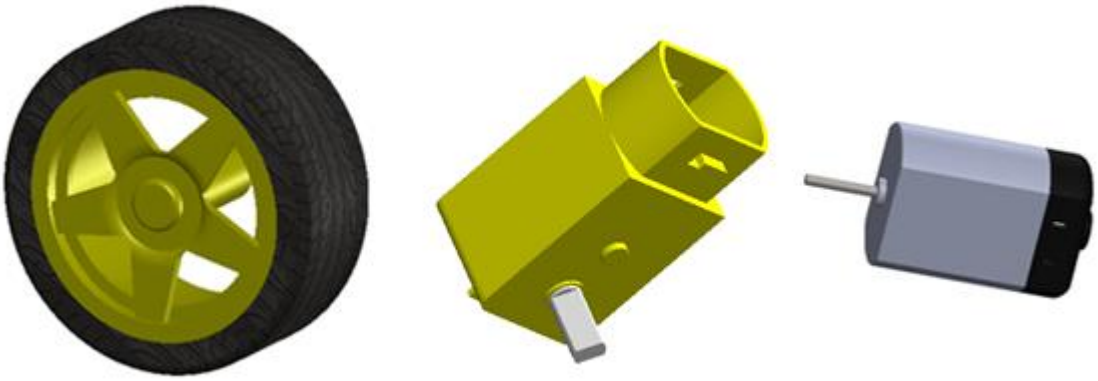
3.1. Thiết kế phần cứng xe tự cân bằng

3.1.1. Thiết kế mô hình

Hình 3.1. Logo phần mềm SolidWorks

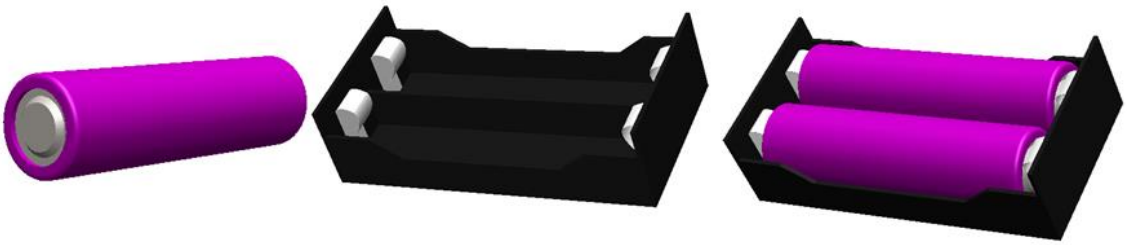
Mô hình xe tự cân bằng được thiết kế trên phần mềm SolidWorks. Phần mềm Solidworks là một trong những phần mềm chuyên về thiết kế 3D do hãng Dassault System phát hành dành cho những xí nghiệp vừa và nhỏ, đáp ứng hầu hết các nhu cầu thiết kế cơ khí hiện nay. Solidworks được biết đến từ phiên bản Solidworks 1998 và được du nhập vào nước ta với phiên bản 2003 và cho đến nay với nhiều phiên bản và phần mềm này đã phát triển đồ sộ về thư viện cơ khí và phần mềm này không những dành cho những xí nghiệp cơ khí nữa mà còn dành cho các ngành khác như: đường ống, kiến trúc, trang trí nội thất, mỹ thuật …

Hình 3.2. Thanh xe và khung xe

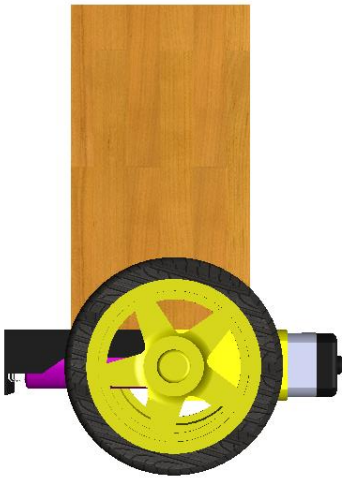


Hình 3.3. Bánh xe, hộp số bánh xe và động cơ rời rạc

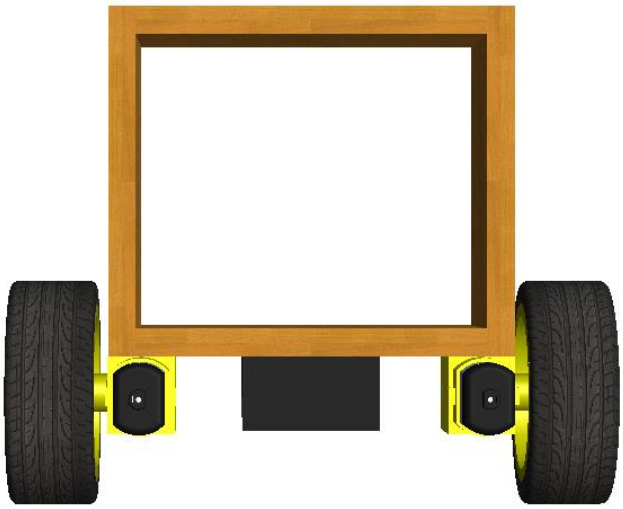
Hình 3.4. Hộp số bánh xe và động cơ lắp hoàn chỉnh



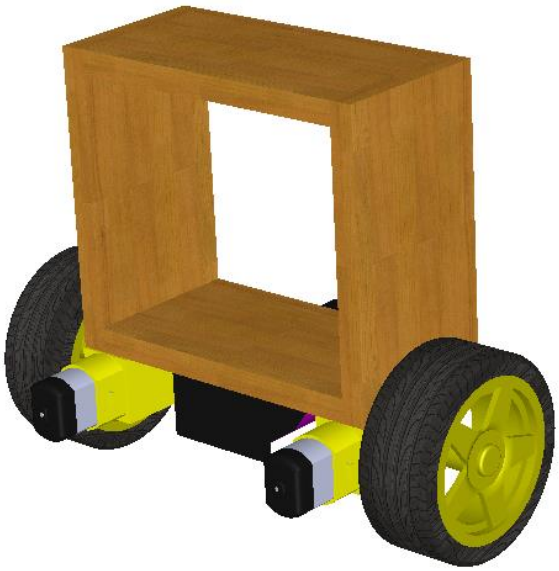
Hình 3.5. Pin và khay pin



Hình 3.6. Mặt nghiêng xe



Hình 3.7. Mặt trước xe

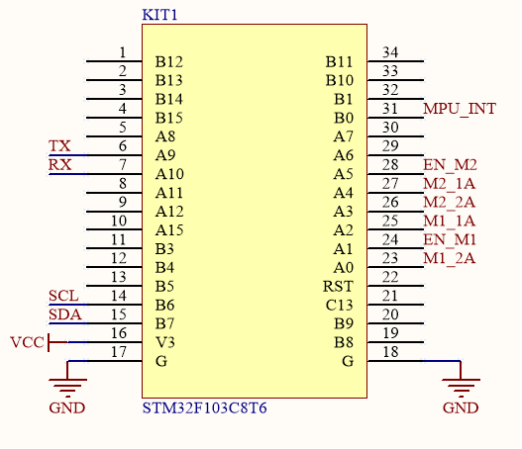


Hình 3.8. Góc nghiêng xe

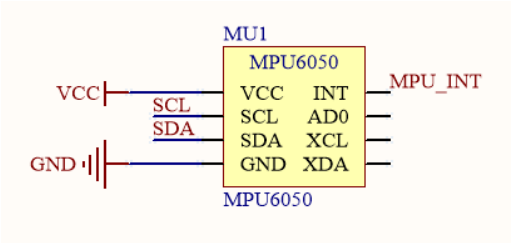
3.1.2. Thiết kế mạch điện

Hình 3.9. Logo phần mềm Altium

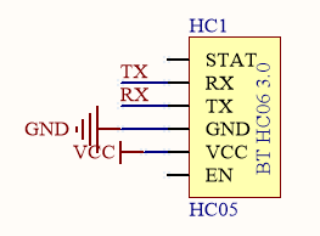
Mạch điện của xe tự cân bằng được thiết kế bằng phần mềm vẽ mạch chuyên dụng Altium. Altium ngày nay đang là một trong những phần mềm vẽ mạch điện tử mạnh và được ưa chuộng ở Việt Nam.Ngoài việc hỗ trợ tốt cho hoạt động vẽ mạch,Altium còn hỗ trợ tốt trong việc quản lý mạch,trích xuất file thống kê linh kiện. Altium Designer cung cấp một ứng dụng kết hợp tất cả công nghệ và chức năng cần thiết cho việc phát triển sản phẩm điện tử hoàn chỉnh, như thiết kế hệ thống ở mức bo mạch và FPGA, phát triển phần mềm nhúng cho FPGA và các bộ xử lý rời rạc, bố trí mạch in (PCB)… Altium Designer thống nhất toàn bộ các quá trình lại và cho phép bạn quản lý được mọi mặt quá trình phát triển hệ thống trong môi trường tích hợp duy nhất. Khả năng đó kết hợp với khả năng quản lý dữ liệu thiết kế hiện đại cho phép người sử dụng Altium Designer tạo ra nhiều hơn những sản phẩm điện tử thông minh, với chi phí sản phẩm thấp hơn và thời gian phát triển ngắn hơn.



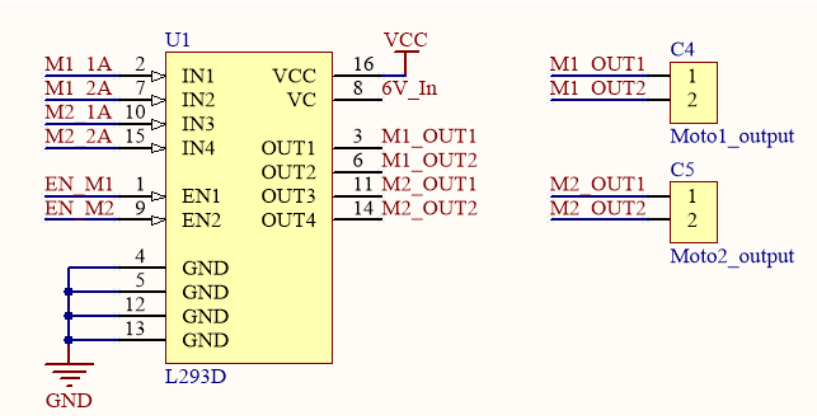
Hình 3.10. Khối vi điều khiển



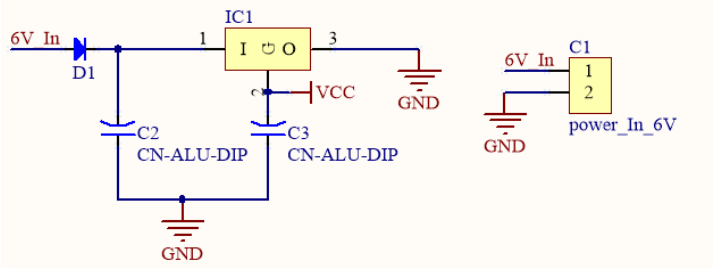
Hình 3.11. Khối cảm biến



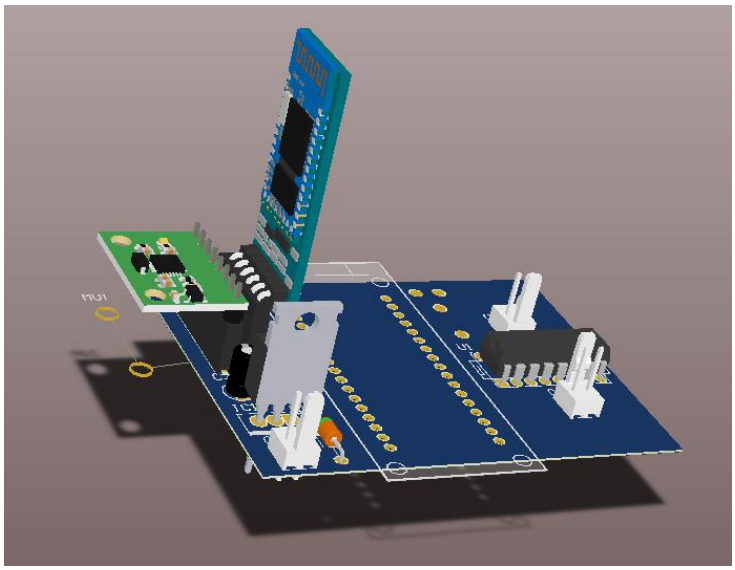
Hình 3.12. Khối giao tiếp Bluetooth



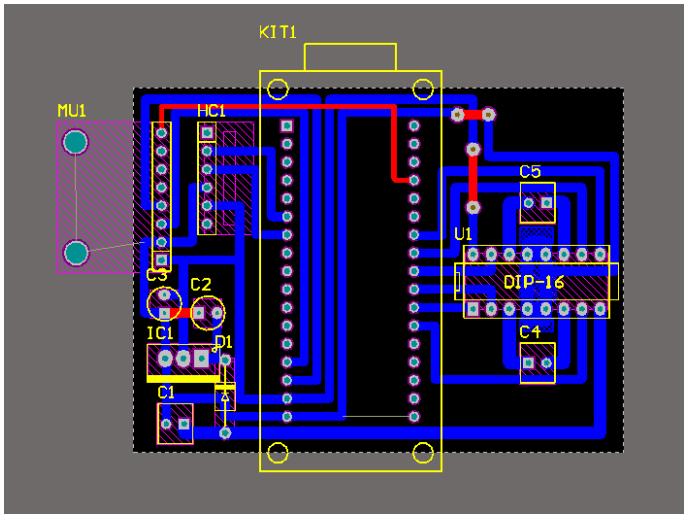
Hình 3.13. Khối điều khiển động cơ



Hình 3.14. Khối nguồn



Hình 3.15. Mạch 3D

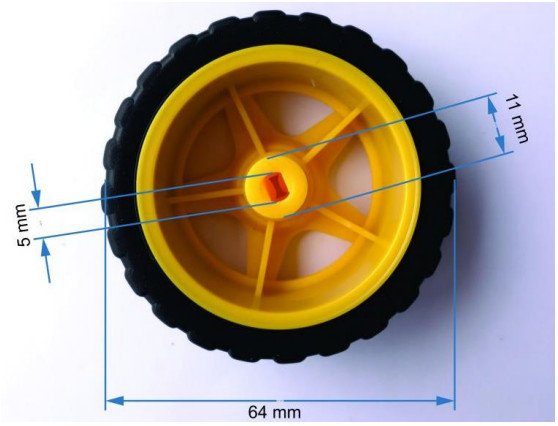


Hình 3.16. Mạch PCB

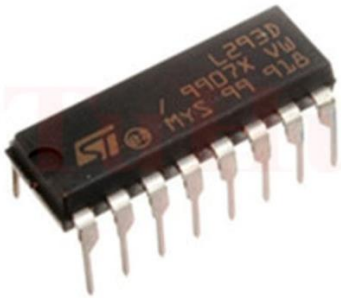
3.2. Chế tạo phần cứng của xe tự cân bằng

Hình 3.17. Động cơ giảm tốc $3V - 6V$

Xe hai bánh tự cân bằng sử dụng 2 động cơ giảm tốc $3V - 6V$. Tốc độ quay với ứng với điện áp $3V - 150mA$ và $6V - 200mA$ là $90RPM \pm 10RPM$ và $200RPM \pm 10RPM$. Mô men xoắn cực đại $0.8kg.cm$.



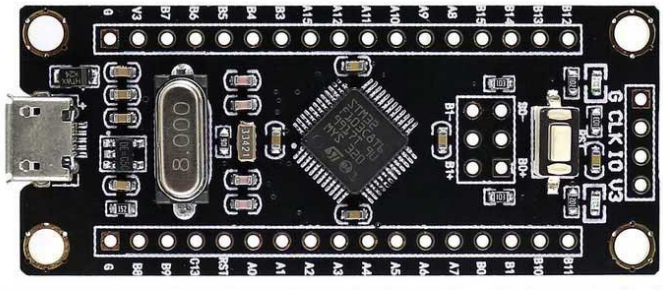
Hình 3.18. Bánh xe động cơ giảm tốc



Hình 3.19. IC điều khiển động cơ L293

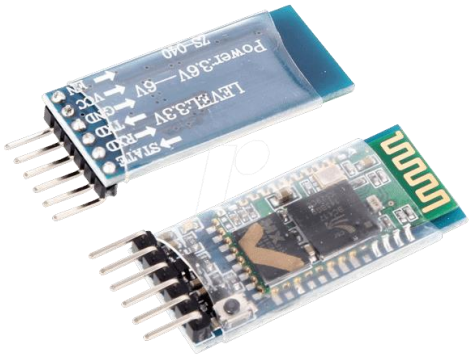
Bánh xe được điều khiển bằng IC L293, IC có một thông số kỹ thuật đặc trưng sau:

- Loại: Half H- Bridge (Quad)
- Loại đầu vào: TTL Logic
- Đầu vào Enable: Có
- Dòng ra max: 600 mA
- Số đầu ra: 4
- Điện áp ra tải max: 36 V
- Điện áp nguồn: 36 V
- Đóng vỏ: PDIP-16



Hình 3.20. Kit STM32F103C8T6

Kit STM32F103C8T6 được sử dụng cho xe tự cân bằng, trên kit chứa vi điều khiển 32 bit STM32F103C8T6 và các thành phần cơ bản để sử dụng vi điều khiển như thạch anh, tụ điện, điện trở,...

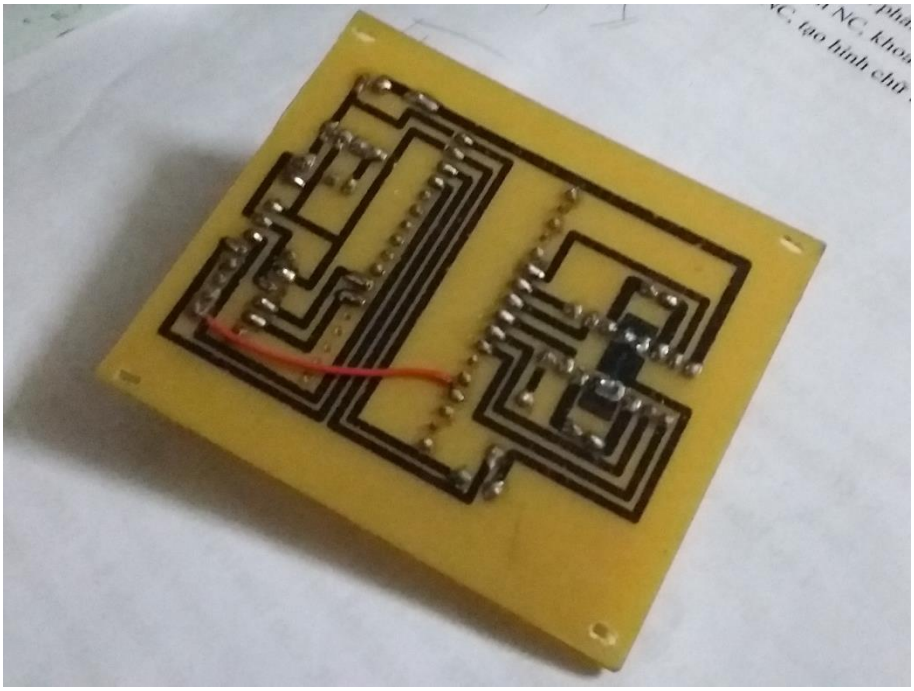


Hình 3.21. Module bluetooth HC-05

Để kết nối giữa xe tự cân bằng và thiết bị điều khiển di động, đề tài sử dụng module bluetooth HC-05. Một số thông số kỹ thuật chung của HC-05:

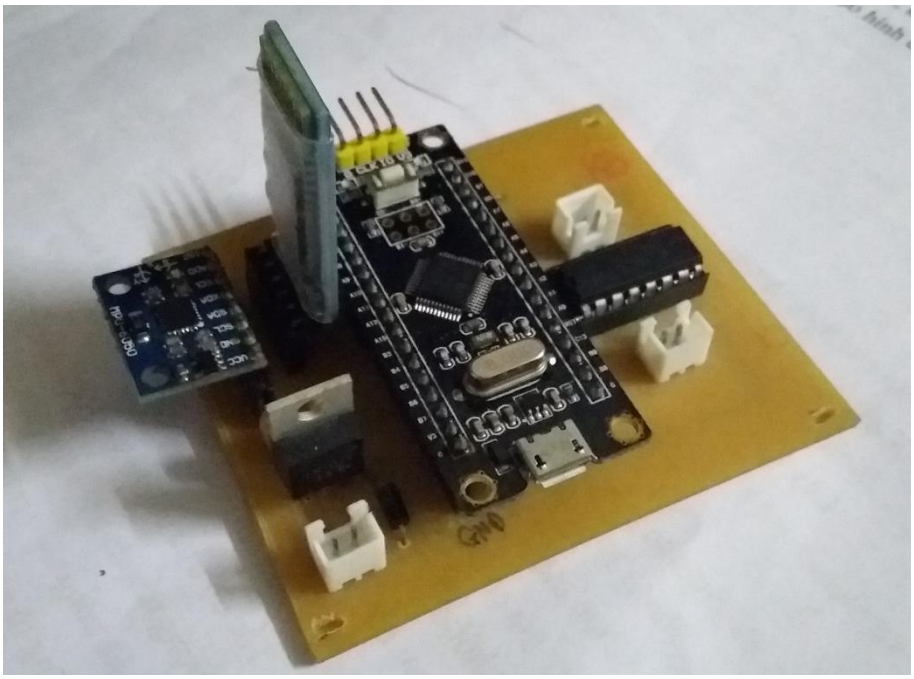
- Điện áp hoạt động: 3.3 ~ 5VDC
- Mức điện áp chân giao tiếp: TTL tương thích 3.3VDC và 5VDC.
- Dòng điện khi hoạt động: khi Pairing 30 mA, sau khi pairing hoạt động truyền nhận bình thường 8 mA.
- Baudrate UART có thể chọn được: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Support profiles: Bluetooth serial port (master and slave)
- Bluetooth protocol: Bluetooth specification v2.0 + EDR
- Frequency: 2.4 GHz ISM band
- Modulation: GFSK (Gaussian frequency shift keying)
- Transmit power: =4 dBm, class 2
- Sensitivity: =-84 dBm at 0.1% BER
- Rate: Asynchronous: 2.1 Mbps (max.)/160 kbps

- Synchronous: 1 Mbps/1 Mbps
- Security features: authentication and encryption
- Kích thước: 15.2 x 35.7 x 5.6mm



Hình 3.22. Mặt sau mạch điện

Mạch điện được thiết kế bằng phần mềm Altium sau đó mạch được làm bằng phương pháp thủ công ủi nhiệt, ăn mòn $FeCl_3$.

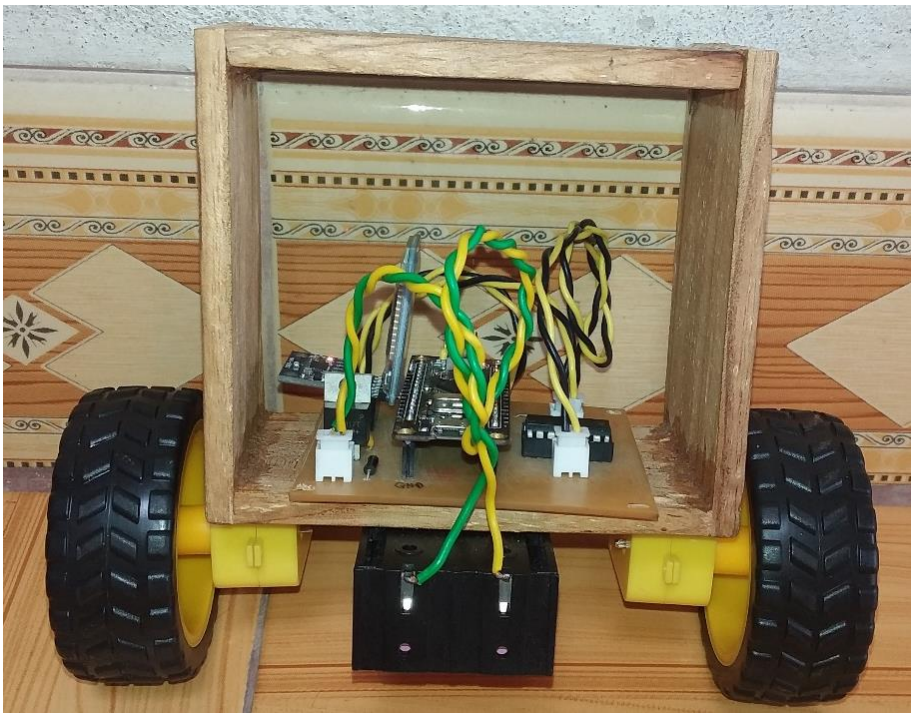


Hình 3.23. Mạch điện được gắn đủ linh kiện

Hình 3.24. Pin Li-ion NCR18650A

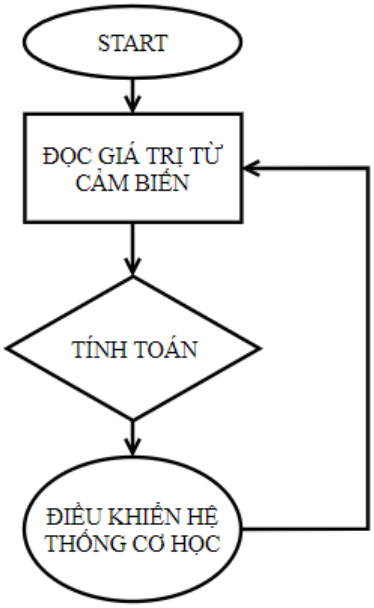
Mạch điện và động cơ được nuôi bởi nguồn 2 pin Li-ion 3.7V nối tiếp. Một số thông số kỹ thuật của pin:

- Dung lượng : 3100mAh
- Dòng xả : 10A
- Size : 18650
- Kích thước khoảng : 18mmx65mm/ viên
- Điện thế : 3.7V, khi sạc đầy có thể đạt đến 4.2v

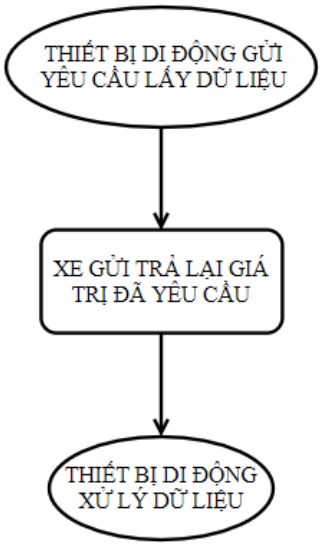


Hình 3.25. Xe tự cân bằng

3.3. Thuật toán sử dụng trong hệ thống xe tự cân bằng



Hình 3.26. Thuật toán giữ xe tự cân bằng



Hình 3.27. Thuật toán thiết bị di động lấy dữ liệu từ xe cân bằng

Hình 3.28. Thuật toán thiết bị di động điều khiển xe tự cân bằng

3.4. Thiết kế phần mềm điều khiển cho xe tự cân bằng

3.4.1. Phần mềm lập trình đa nền tảng Qt

Hình 3.29. Logo phần mềm Qt

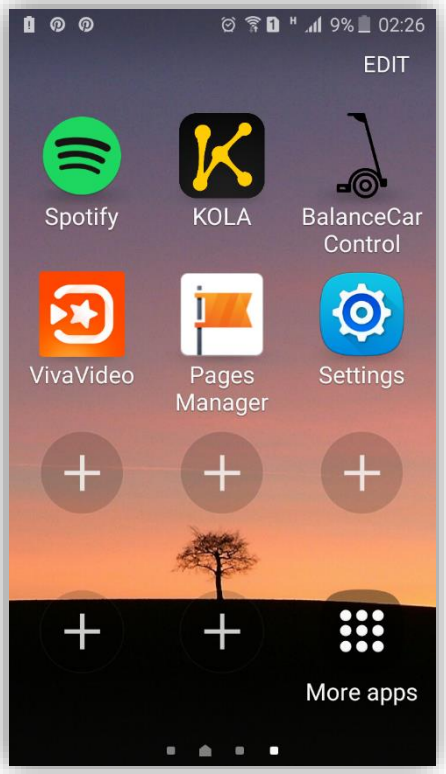
Qt được phát triển bởi Trolltech, một công ty phần mềm của Nauy. Năm 2008 công ty này được mua lại bởi Nokia. Tháng 8 năm 2012 một công ty phần mềm của Phần Lan là Digia đã mua lại công nghệ Qt từ Nokia. Từ đó Qt được phát triển thêm một phiên bản mã nguồn mở. Trang web mã nguồn mở của Qt là http://www.qt.io. Hiện tại Qt đang được phát triển bởi 2 công ty, một là công ty Qt Company – một nhánh con của Digia, nhắm tới các ứng dụng dành cho doanh nghiệp, hai là Qt Project dành cho các dự án mã nguồn mở.

Qt là một framework đa nền tảng. Một số ứng dụng phổ biến được viết từ Qt có thể kể đến như KDE, Opera, Google Earth, và Skype. Qt lần đầu tiên được giới thiệu vào tháng 5 năm 1995. Qt có thể được dùng để phát triển ứng dụng mã nguồn mở lẫn

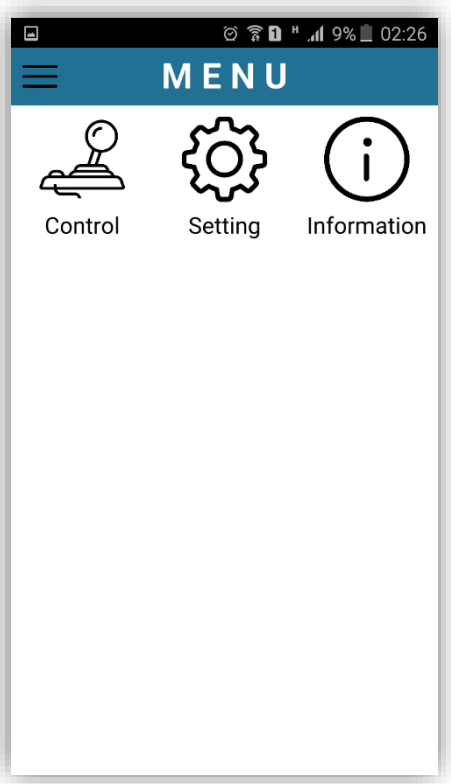
các ứng dụng dành cho doanh nghiệp. Bộ công cụ phát triển Qt rất mạnh mẽ vì nó được cả một cộng đồng mã nguồn mở hỗ trợ. Có đến hàng ngàn các nhà phát triển mã nguồn mở sử dụng Qt trên toàn thế giới.

Ưu điểm lớn nhất của Qt có thể nói đó là ngôn ngữ lập trình đa nền tảng. Lập trình viên chỉ cần viết mã nguồn một lần cho ứng dụng trên bất kỳ môi trường nào. Mã nguồn đó có thể được dịch và chạy trên nhiều môi trường khác nhau như Windows, Android, IOS, linux, QnX,... Ngoài ra cũng có thể nó Qt là một ngôn ngữ lập trình giao diện dễ sử dụng và được hỗ trợ nhiều phần cứng khác nhau.

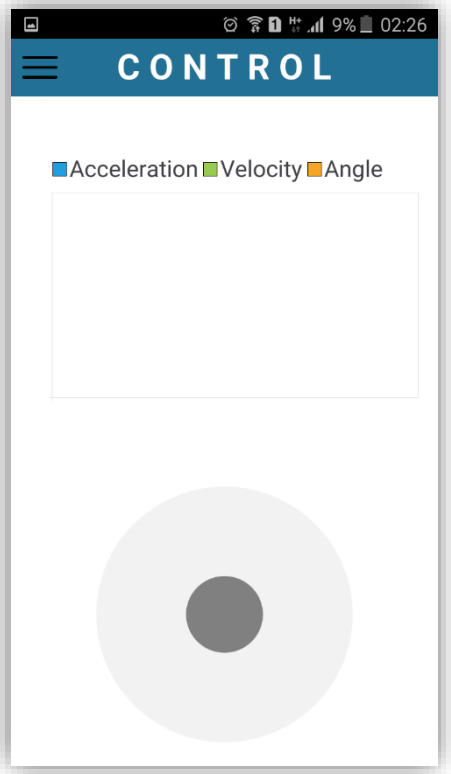
3.4.2. Phần mềm điều khiển xe tự cân bằng



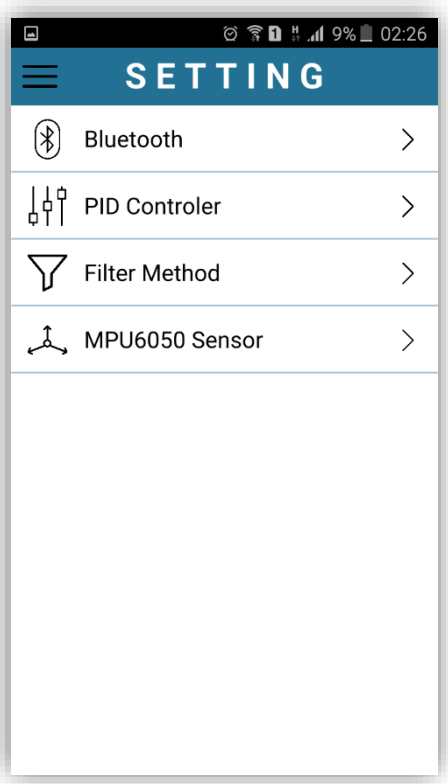
Hình 3.30. Ứng dụng điều khiển xe cân bằng "BalanceCarControl" trong menu



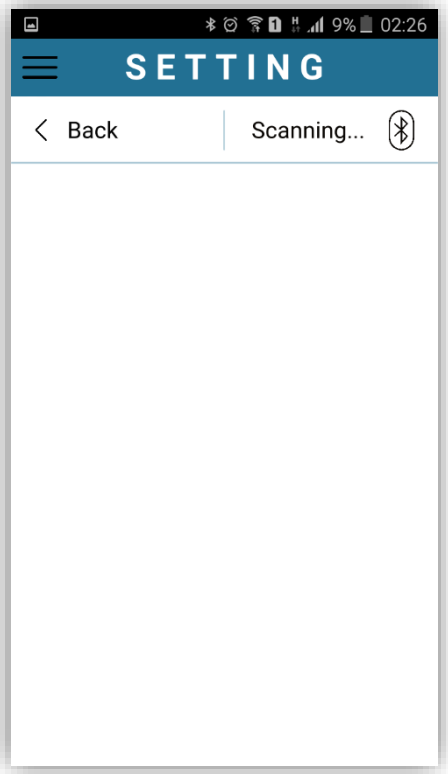
Hình 3.31. Màn hình menu lựa chọn của ứng dụng



Hình 3.32. Màn hình điều khiển xe cân bằng và hiển thị đồ thị



Hình 3.33. Màn hình cài đặt thông số cho thiết bị



Hình 3.34. Màn hình cài đặt bluetooth

3.5. Cách vận hành và điều khiển xe tự cân bằng

❖ *Vận hành xe tự cân bằng:*

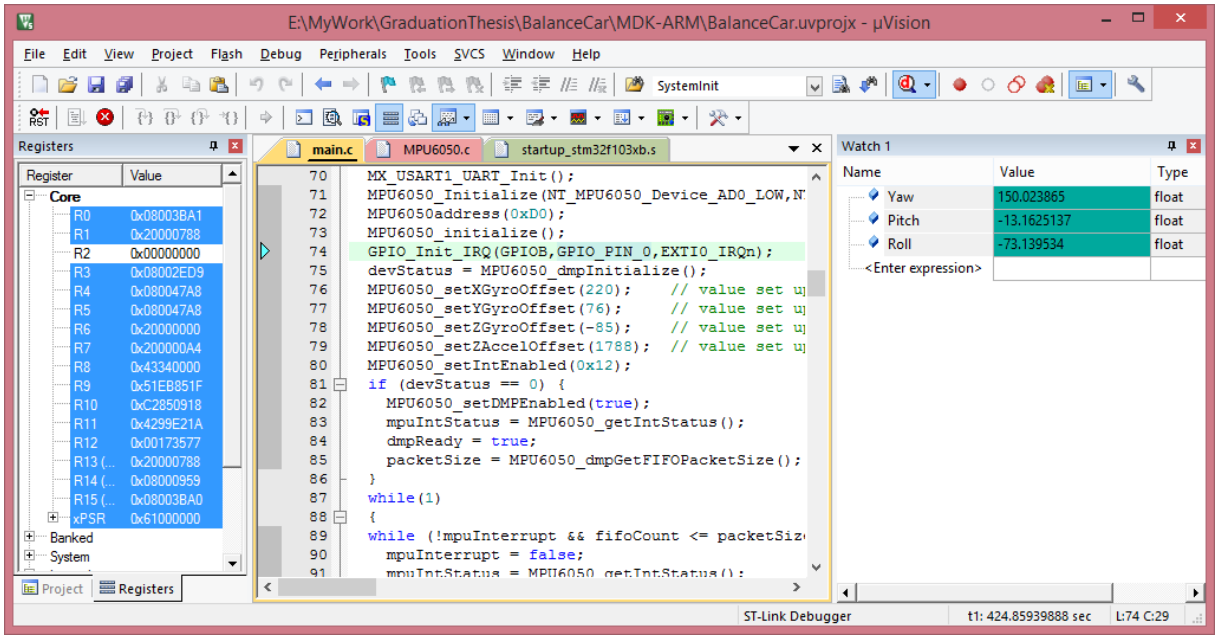
- Bật công tắc nguồn cho xe, xe sẽ phản hồi lại một tiếng beep
- Để xe đứng yên cho cảm biến gia tốc 3 trục ổn định.
- Sau khi đạt giá trị ổn định xe sẽ phản hồi 2 tiếng beep
- Nâng xe đến vị trí cân bằng (vị trí góc 0^0). Khi đã đến được góc 0^0 xe sẽ sáng LED màu xanh và bắt đầu hoạt động.

❖ *Kết nối với xe tự cân bằng:*

- Khởi động bluetooth của thiết bị di động
- Bật ứng dụng trên thiết bị di động
- Từ "MENU" → "Setting" → "Bluetooth" → "Scan Now" chờ ứng dụng tìm kiếm thiết bị bluetooth – HC05 của xe tự cân bằng
- Chọn "BalanceMoto" (tên của thiết bị bluetooth – HC05)
- Quay lại menu và chọn "Control" để điều khiển và theo dõi các thông số của xe

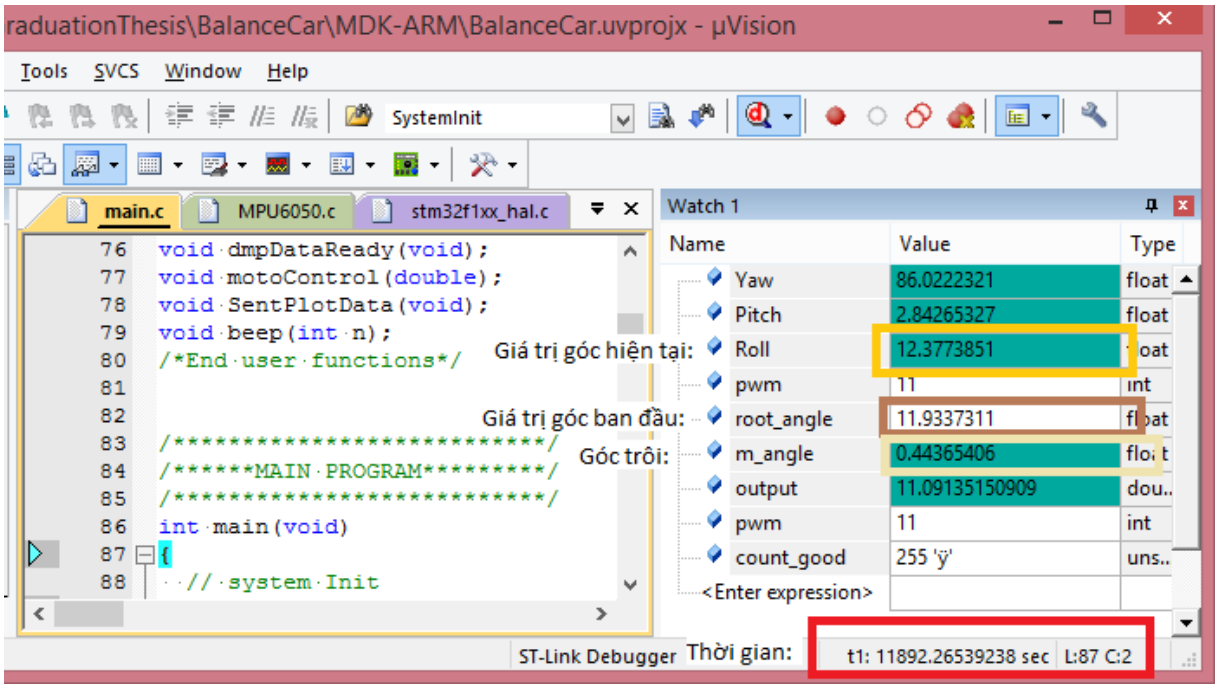
3.6. Các kết quả thực nghiệm

❖ *Giá trị đo lường từ cảm biến:*



Hình 3.35. Giá trị đo lường của cảm biến từ màn hình debug

Cảm biến MPU6050 đo được các gí trị góc nghiêng, góc quay và góc ngã của xe. Cảm biến lấy được giá trị gốc vị trí của cảm biến.



Hình 3.36. Độ trôi góc khi đo đạc

Kết quả đo đạc trong khoảng 19 phút liên tục. Với góc ban đầu là 11.9337311^0, góc khi kết thúc thực nghiệm là 12.3773851^0 góc trôi là 0.44365406^0

❖ *Mức độ cân bằng*

Hình 3.37. Xe tự cân bằng trên mặt phẳng ngang

- Với mặt phẳng cân ngang xe có khả năng cân bằng rất lâu khi không có tác động ngoại lực

- Với mặt phẳng nghiêng, xe chưa thể giữ được vị trí cân bằng.

- Với các ngoại lực tác động quá lớn, xe bị mất cân bằng

❖ *Điều khiển*

- Điều khiển xe di chuyển với tốc độ chậm

❖ *Thời gian sử dụng pin*

- Với nguồn là hai viên pin 3.7V, xe có thể hoạt động liên tục trong 1 giờ. Khi pin có giấu hiệu yếu điện, xe bị mất cân bằng.

KẾT LUẬN

❖ ***Kết quả đạt được***

Cân bằng: Xe có khả năng cân bằng trên mặt phẳng

Điều khiển: Điều khiển được xe di chuyển các hướng bằng phần mềm điều khiển trên di động

Thu thập dữ liệu: Ghi lại được các giá trị giao động của xe góc nghiêng, góc xoay, góc ngã và gửi về cho thiết bị di động.

❖ ***Hạn chế còn sót lại***

Di chuyển: Xe chưa di chuyển được trên dịa hình gồ ghề, sỏi đá, mặt nghiêng

Thẩm mỹ: Sản phẩm chưa được thiết kế gọn gàng và thẩm mỹ

❖ ***Đề xuất cải tiến***

Giảng dạy: Việc xây dụng một hệ cơ điện tử đơn giản là không hề khó, hơn nữa còn mang lại nhiều kinh nghiệm cho mỗi sinh viên tham gia xây dựng. Vì vậy việc đưa các mô hình cơ điện tử cơ bản, điển hình vào giảng dạy và trực tiếp hướng dẫn sinh viên tạo ra sản phẩm sẽ mang lại những kết quả cao trong giảng dạy.

Thiết kế hệ thống: Vi điều khiển 32 bit là một vi điều khiển mạnh mẽ hiện nay, hoàn toàn có thể thay thế các dòng vi điều khiển 8 bit truyền thống hơn nữa còn vượt trội về nhiều mặt. Do đó trong các thiết kế hiện tại và tương lai thì chúng ta nên sử dụng dòng vi điều khiển này.

❖ ***Phương hướng phát triển***

Tận dụng tối đa sức mạnh của vi điều khiển: Xe cân bằng sử dụng vi điều khiển 32 bit mạnh mẽ tuy nhiên chưa có nhiều khai thác sự mạnh mẽ của nó. Cùng với khả năng đi lại linh hoạt của xe cân bằng, sẽ tạo ra được các thế hệ xe thăm dò linh hoạt.

Phát triển mô hình giao tiếp, truyền thông: Bluetooth HC-05 là một module vừa đóng vai trò làm master vừa có thể là một slave, tận dụng tính năng này ta hoàn toàn có thể thiết lập một hệ thống xe có khả năng tự giao tiếp với nhau trong phạm vi sóng bluetooth.

Nâng cao hệ thống cơ khí: Động cơ 6V, 9V, 12V hay động cơ lớn hơn về cơ bản thì đều có nguyên tắc hoạt động như nhau. Do đó chúng ta có thể nâng cấp xe cân bằng để sử dụng trong di chuyển cá nhân.

Thẩm mỹ: Hiện tại xe đang được thiết kế bằng gỗ, trong tương lai có thể sử dụng nhiều loại nguyên liệu khác nhau để tạo ra chiếc xe cân bằng với mẫu mã đẹp mắt hơn.

Chất lượng hệ thống: Tiến hành nâng cao các bộ lọc nhiễu và thêm các encoder để hệ thống được ổn định và chính xác.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Phạm Văn Ất, *C++ và lập trình hướng đối tượng*, Nhà xuất bản giao thông vận tải, 1999, tr 12 – 50.

[2] Mai Tuấn Đạt, *Xe hai bánh tự cân bằng di chuyển trên địa hình phẳng*, Luận văn tốt nghiệp đại học, Trường Đại học bách khoa – Đại học Quốc gia TP Hồ Chí Minh, 2005, tr 2 – 14.

[3] Nguyễn Ngọc Phương, Nguyễn Trường Thịnh, *Sổ tay hệ thống cơ điện tử*, Nhà xuất bản Đại học Quốc gia Hồ Chí Minh, 2016, tr 10 – 15.

[4] Phạm Mạnh Thắng, Hoàng Văn Mạnh, *Vi xử lý và vi điều khiển* – Nguyên lý và ứng dụng, Nhà xuất bản Đại học Quốc gia Hà Nội, 2016, tr. 25 – 73.

Website

[5] Website Control Tutorials for Matlab & Simulink, http://ctms.engin.umich.edu

[6] Website nBot Balancing Robot, http://www.geology.smu.edu/~dpa-www/robo/nbot

[7] Website hãng Segway, http://www.segway.com

[8] Website của Trevorblackwell, http://www.trevorblackwell.com

PHỤ LỤC

1. Hướng dẫn sinh mã nguồn tự động cho dòng vi điều khiển STM32 bằng phần mềm STM32CubeMx

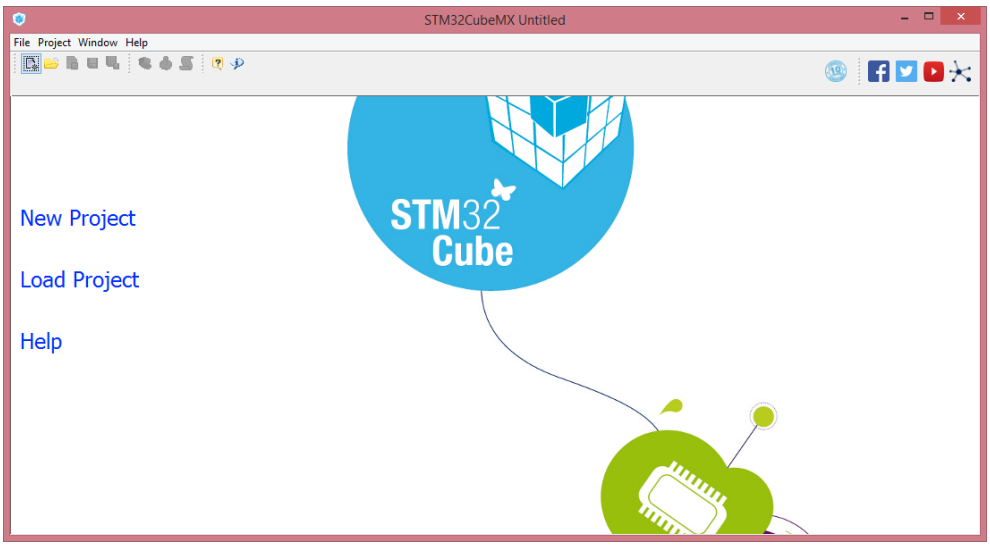
1.1. Chuẩn bị

Phần mềm STM32CubeMx

Phần mềm lập trình ARM 32 bit (ví dụ KeilC ARM V5)

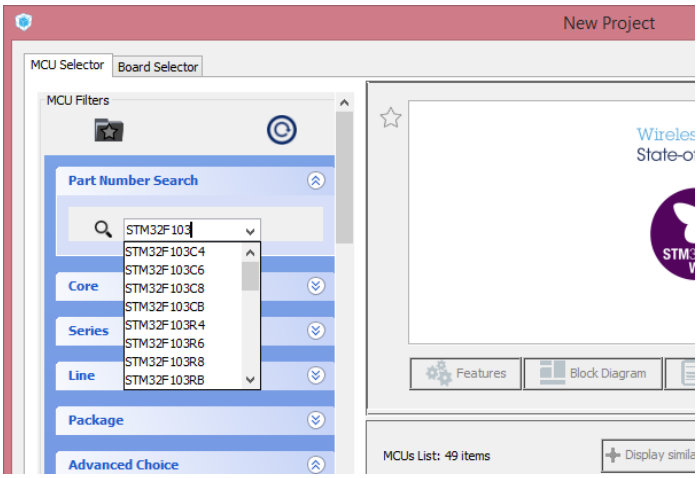
1.2. Hướng dẫn

Như đã giới thiệu từ ban đầu, các vi điều khiển 32 bit tuy mạnh mẽ nhưng việc cài đặt và lập trình bước đầu tương đối vất vả, do số lượng thanh ghi lớn cũng như cấu trúc hoạt động. Tuy nhiên, các nhà sản xuất đã có những nỗ lực rất lớn trong việc tạo ra các công cụ hỗ trợ cho người kỹ sư phần mềm. Đó là phần mềm STM32CubeMX do chính hãng STMicroelectronics phát triển.



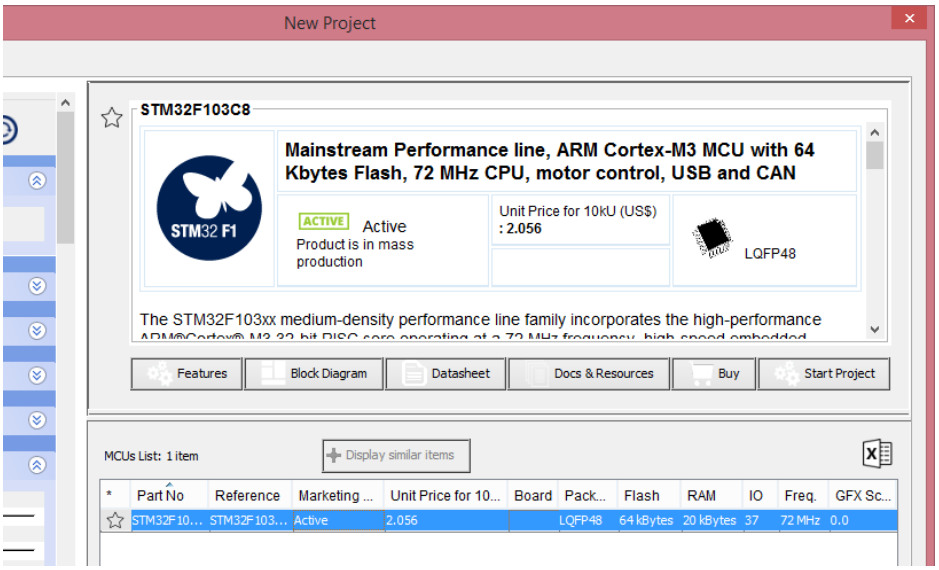
Giao diện ban đầu của phần mềm STM32CubeMX

Phần mềm STM32CubeMX là một phần mềm giao diện cho phép người kỹ sư lập trình lựa chọn, cài đặt và tùy chỉnh các thuộc tính, chức năng trên vi điều khiển một cách dễ dàng. Sau đó phần mềm sẽ tự động sinh ra phần mã nguồn (code) giúp người lập trình tiến nhanh và gần hơn đến các phần xử lý logic của dự án.



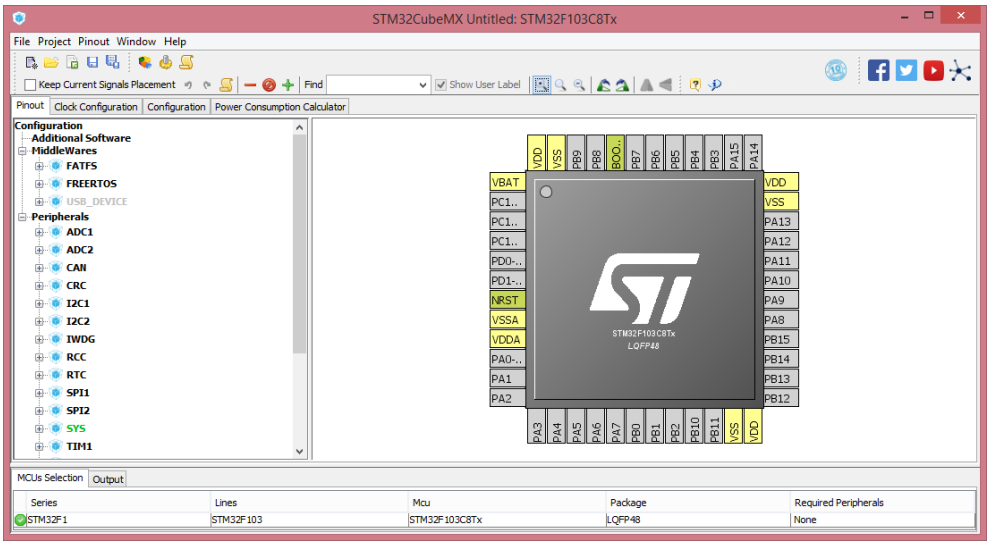
Lựa chọn vi điều khiển

Sau khi click vào "New Project" (Dự án mới), kỹ sư lập trình sẽ cần phải lựa chọn loại vi điều khiển cần phát triển bằng cách tìm kiếm tên vi điều khiển trong ô tìm kiếm (cách đơn giản nhất).



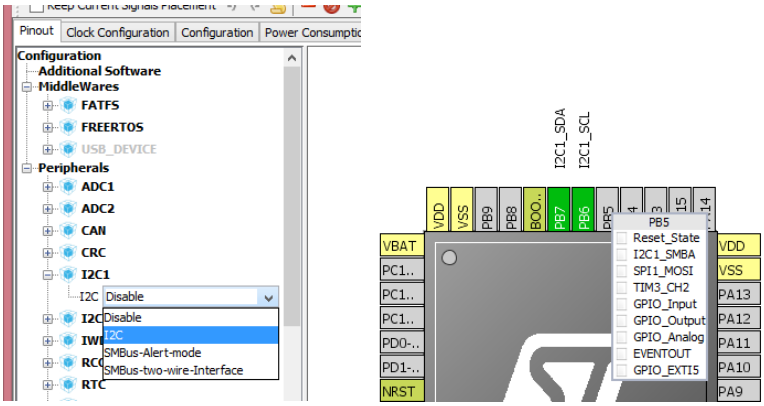
Bắt đầu một project mới

Sau khi tìm kiếm thành công, một danh sách các vi điều khiển sẽ được lọc ra trong bảng. Click chọn vi điều khiển và nhấn vào "Start Project" là chúng ta đã đi được bước đầu tiên trong việc tạo một dự án cho dòng vi điều khiển 32 bit này. Bên cạnh đó phần mềm STM32CubeMx cũng hiển thị nhiều thông tin giúp người kỹ sư lập trình hiểu hơn về vi điều khiển như phần thông tin vi điều khiển, kiểu chân, giá cả,... cùng nhiều thông tin khác.



Giao diện cấu hình cho vi điều khiển

Sau khi đã tạo được dự án mới, giao diện thuận tiện của STM32CubeMx hiện lên vô cùng trực quan với 4 tab chính: cấu hình các chân I/O (pinout), cấu hình xung clock (Clock Configuration), cấu hình các chức năng (Configuration) và tính toán mức độ tiêu thụ năng lượng (Power Consumption Calulator).

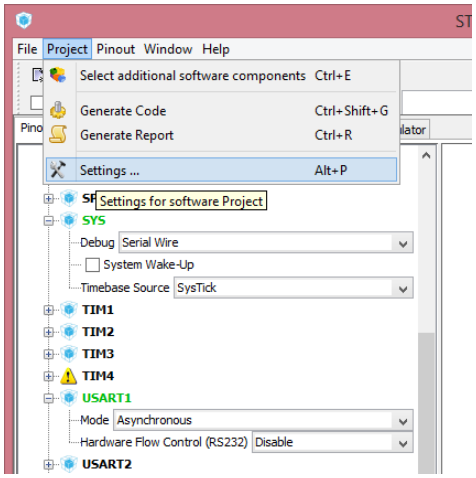


Cấu hình chức năng chân cho vi điều khiển

Khi cấu hình chân I/O cho vi điều khiển, phần mềm phân chia rất rõ ràng thành 2 phần, phần danh sách giao tiếp các thiết bị ngoại vi và phần trực quan vi điều khiển. Đối với phần danh sách giao tiếp các thiết bị ngoại vi, người kỹ sư lập trình chỉ cần lựa chọn loại giao tiếp mình cần sử dụng, ví dụ như I^2C, người kỹ sư lập trình sẽ chọn I2C1 hoặc I2C2 (vì STM32F103C8T6 cung cấp 2 cổng I^2C) và chọn thuộc tính cho nó, chọn "I2C" để sử dụng cổng này, "Disable" để ngắt cổng và một số tùy chọn nâng cao khác. Đối với phần trực quan hiển thị các chân vi điều khiển, kỹ sư click chuột trái vào một chân của vi điều khiển, nó sẽ hiển thị ra các tùy chọn có thể thiết lập đối với chân đó như biến

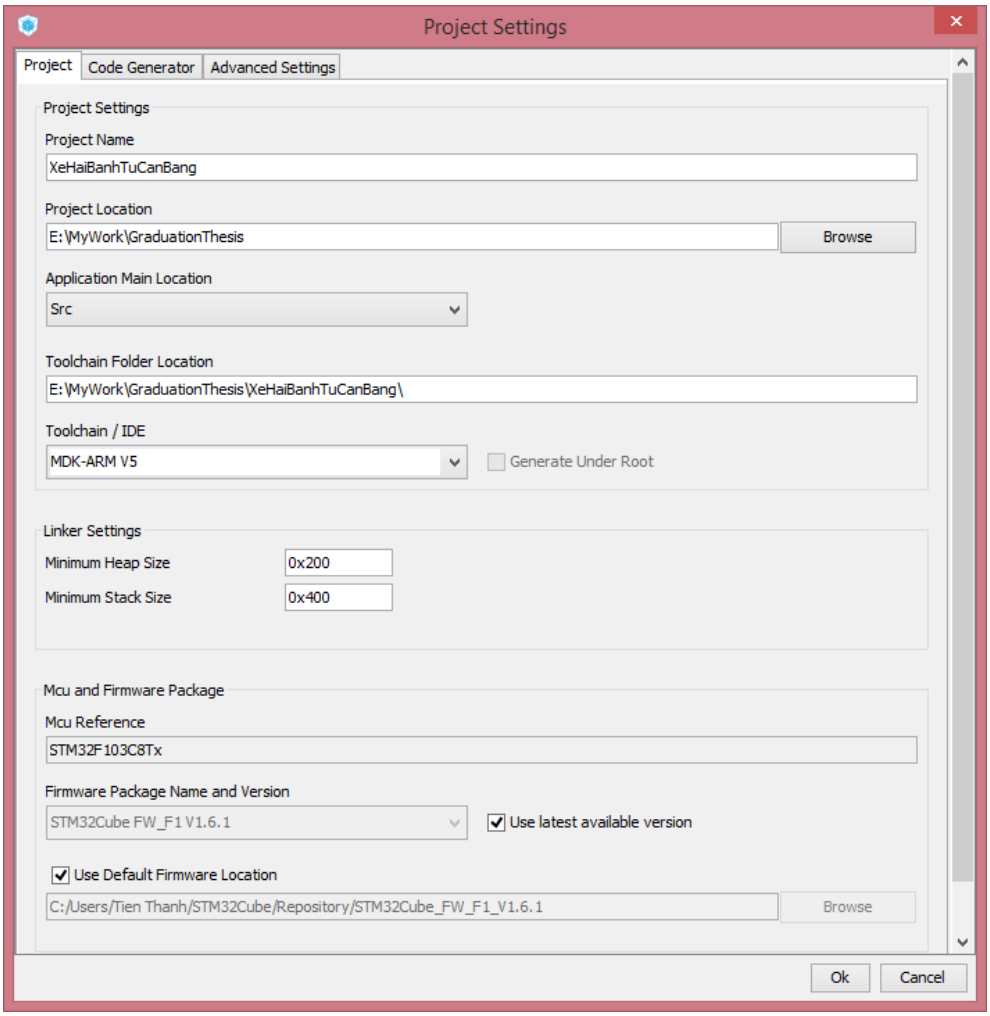
nó thành một chân reset, một chân ngắt Timer, một chân đầu ra,... Hoàn toàn trực quan và dễ dàng sử dụng.

Tương tự các thiết lập khác đều tương đối dễ dàng cho người kỹ sư lập trình sử dụng vào cài đặt cho vi điều khiển của mình.



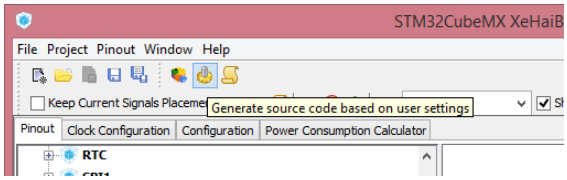
Lựa chọn cài đặt dự án

Sau khi thiết lập xong toàn bộ các thuộc tính cần dùng cho vi điều khiển, kỹ sư lập trình cần vào cài đặt dự án (Settings...) bằng cách nhấn vào "Project" trên thanh menu và lựa chọn "Setting..." hoặc sử dụng tổ hợp phím "Alt+P".



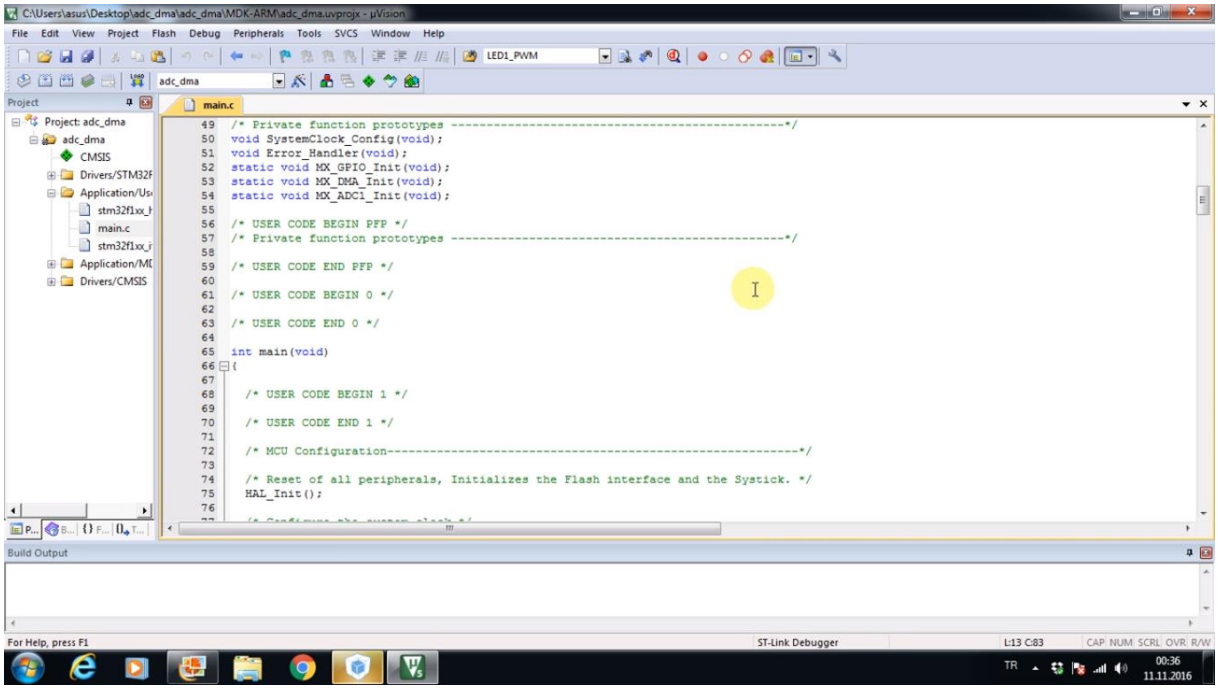
Cài đặt dự án

Tại phần cài đặt dự án, người kỹ sư điền đầy đủ thông tin về dự án. Như tên dự án, địa chỉ thư mục lưu dự án,... Quan trọng nhất là công cụ để phát triển mã nguồn về sau này tại phần "Toolchain/ IDE", có rất nhiều công cụ phù hợp với việc tự động sinh mã nguồn của STM32CubeMx như EWARM, MDK-ARM V5, TrueSTUDIO, SW4STM32,... Sau khi lựa chọn xong công cụ phát triển, chúng ta nhấn OK và trong lần đầu tiên sử dụng STM32CubeMx cho một dòng vi điều khiển mới, công cụ STM32CubeMx sẽ cần download các thư viện hỗ trợ, hỗ trợ ngoại vi từ trên server về máy của chúng ta. Việc này hoàn toàn tự động, tiện ích và do chính nhà sản xuất vi điều khiển cung cấp do đó sẽ vô cùng an toàn cho vi điều khiển cũng như dự án.



Sinh mã nguồn cho vi điều khiển

Cuối cùng là sinh mã nguồn cho vi điều khiển bằng cách nhấn vào biểu tượng răng cưa tại thanh menu. Mã nguồn sẽ được tự động sinh ra theo như những gì mà người kỹ sư lập trình đã cài đặt từ đầu. Sẽ không phải tốn thêm một giây nào cho việc cài đặt xung nhịp hoạt động của vi điều khiển hay thiết lập giao tiếp. Do đó người kỹ sư lập trình có thể bắt tay trực tiếp vào phần xử lý logic của dự án.



Mã nguồn sinh tự động cho trình soạn thảo KeilC5-ARM

2. Mã nguồn và tài liệu khóa luận

Mã nguồn và tài liệu khóa luận được công khai và lưu tại:

https://github.com/tienthanh-pham/GraduationThesis.git

Gồm có: Mã nguồn xe tự cân bằng cho vi điều khiển STM32F103C8T6 được sinh mã tự động bằng STM32CubeMX và chỉnh sửa biên dịch lại bằng KeilC5-ARM; Mã nguồn phần mềm điều khiển xe tự cân bằng được viết bằng phần mềm Qt với ngôn ngữ C++ và QML; Các thiết kế mạch và mô hình của khóa luận. Ngoài ra còn chứa các tài liệu liên quan.