

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KỸ THUẬT GIAO THÔNG
BỘ MÔN KỸ THUẬT Ô TÔ - MÁY ĐỘNG LỰC



ĐỒ ÁN TỐT NGHIỆP
THIẾT KẾ HỆ THỐNG ĐÁNH LỬA CDI – DC
CHO XE MÁY

GVHD: TS.Trần Đăng Long

SVTH: Phạm Toàn Văn Võ 1912453

Trương Quốc Trung 1912328

**TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KỸ THUẬT GIAO THÔNG
BỘ MÔN KỸ THUẬT Ô TÔ – MÁY ĐỘNG LỰC**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập – Tự Do – Hạnh Phúc**

NHIỆM VỤ LVTN

Họ và tên SV: Phạm Toàn Văn Võ.....**MSSV:** 1912453.....

Trương Quốc Trung.....**MSSV:** 1912328.....

Ngành: Kỹ thuật Ô tô – Máy động lực **Lớp:** GT19OTO3

1. Tên đề tài : THIẾT KẾ HỆ THỐNG ĐÁNH LỬA CDI-DC CHO XE MÁY.

.....
.....
.....

2. Yêu cầu về nội dung:

- Thiết kế hệ thống đánh lửa CDI-DC cho xe máy với điện áp cao từ (450 – 500V). Yêu cầu kỹ thuật là hệ thống hoạt động chính xác với tần suất và năng lượng đánh lửa cao.
- Cải tiến giải thuật điều khiển hệ thống đánh lửa CDI-DC đánh lửa một lần với hai vòng quay trực khuỷu, đánh lửa nhiều lần khi khởi động.

.....

3. Yêu cầu về sản phẩm:

- | | | |
|--|---|--|
| <input checked="" type="checkbox"/> Thuyết minh báo cáo | <input checked="" type="checkbox"/> Poster tóm tắt | <input type="checkbox"/> Bài báo khoa học |
| <input type="checkbox"/> Chương trình máy tính | <input checked="" type="checkbox"/> Chương trình vi xử lý | <input type="checkbox"/> Mô hình mô phỏng |
| <input type="checkbox"/> Bản vẽ bố trí chung khổ A3 | <input type="checkbox"/> Bản vẽ lắp khổ A3 | <input type="checkbox"/> Bản vẽ chi tiết khổ A4. |
| <input checked="" type="checkbox"/> Khác: Mạch điều khiển đánh lửa CDI-DC. | | |

.....

4. Ngày giao nhiệm vụ: *ngày tháng năm 20.....*

5. Ngày hoàn thành: *ngày tháng năm 20.....*

Nội dung và yêu cầu đã được Bộ môn Kỹ thuật Ô tô – Máy động lực thông qua.

Ngày tháng năm 20.....

Chủ nhiệm Bộ môn

Ngày tháng năm 20.....

GV hướng dẫn chính



MỤC LỤC

DANH MỤC HÌNH ẢNH.....	4
LỜI NÓI ĐẦU	7
CHƯƠNG I: GIỚI THIỆU	8
1.1. Hệ thống đánh lửa CDI:	8
1.2. Tổng quan đề tài:	8
1.2.1. Thể loại đề tài:	8
1.2.2. Đối tượng thiết kế:.....	8
1.2.3. Mục tiêu đề tài:	9
1.2.4. Điều kiện làm việc:	9
1.2.5. Yêu cầu kỹ thuật:	9
1.2.6. Ý tưởng thực hiện và các bài toán chính cần giải quyết:.....	9
1.2.7. Giới hạn nội dung thực hiện:.....	10
CHƯƠNG II: CƠ SỞ LÝ THUYẾT	11
2.1. Cơ sở lý thuyết điều khiển đánh lửa sớm:	11
2.1.1. Sự cần thiết của việc điều khiển đánh lửa sớm:	11
2.1.2. Điều khiển thời điểm đánh lửa:.....	13
Điều khiển theo tốc độ động cơ:	13
2.2. Cơ sở lý thuyết tính toán thời gian nạp và xả tụ:	15
2.3. Cơ sở lý thuyết điều khiển đánh lửa nhiều lần khi khởi động:	18
2.3.1. Ưu điểm của điều khiển đánh lửa nhiều lần khi khởi động:	18
2.4. Cơ sở lý thuyết điều khiển đánh lửa 1 lần/ 2 vòng quay:.....	18
2.4.1. Ưu điểm của điều khiển đánh lửa 1 lần/2 vòng quay;	18
CHƯƠNG III: THIẾT KẾ BỐ TRÍ CHUNG	19
3.1. Phương án thiết kế bố trí chung:.....	19
3.2. Sơ đồ bố trí chung:	19
CHƯƠNG IV: THIẾT KẾ KỸ THUẬT	20
4.1. Thiết kế giải thuật điều khiển mạch CDI-DC:	20



4.1.1. Thiết lập vi điều khiển:	20
4.1.2. Giản đồ thời gian:	21
4.1.3. Lưu đồ giải thuật:	22
4.1.4. Chức năng chương trình:	25
4.2. Thiết kế giải thuật điều khiển thay đổi góc đánh lửa theo tốc độ:	26
4.2.1. Thiết lập vi điều khiển:	26
4.2.2. Giản đồ thời gian:	28
4.2.3. Chức năng chương trình:	28
4.3. Thiết kế giải thuật điều khiển đánh lửa 2 lần/ vòng quay:	29
4.3.1. Thiết lập vi điều khiển:	29
4.3.2. Giản đồ thời gian:	31
4.3.3. Chức năng chương trình:	32
4.4. Thiết kế giải thuật đánh lửa nhiều lần lúc khởi động:	33
4.4.1. Thiết lập vi điều khiển:	33
4.4.2. Giản đồ thời gian:	35
4.4.3. Chức năng chương trình:	35
4.4. Thiết kế mạch điện – điện tử điều khiển đánh lửa	37
4.4.1. Mạch tín hiệu điều khiển	38
4.4.2. Mạch ổn áp	41
4.4.3. Mạch DC-AC converter	43
4.4.4. Mạch CDI - AC	59
4.4.5. Mạch in PCB	62
CHƯƠNG V: THIẾT KẾ CÔNG NGHỆ	66
5.1. Phương án công nghệ	66
5.2. Quy trình chế tạo	66
5.3. Quy trình kiểm tra	68
CHƯƠNG VI: KẾT QUẢ ĐẠT ĐƯỢC	69
PHỤ LỤC	74



Code tham khảo điều khiển góc đánh lửa sớm (thay đổi góc đánh lửa theo tốc độ động cơ, đánh lửa 1 lần/ 2 vòng quay, đánh lửa nhiều lần lúc khởi động)	74
Code tham khảo điều khiển mạch CDI-DC	104
Code tham khảo giả lập tốc độ động cơ.....	113
TÀI LIỆU THAM KHẢO	119



DANH MỤC HÌNH ẢNH

Hình 2. 1: Thời điểm góc đánh lửa sớm.....	11
Hình 2. 2: Giai đoạn cháy trễ	12
Hình 2. 3: Giai đoạn lan truyền ngọn lửa	13
Hình 2. 4: Điều khiển theo tốc độ	13
Hình 2. 5: Điều khiển theo tải của động cơ.....	14
Hình 2. 6: Điều khiển theo tiếng gõ động cơ	14
Hình 2. 7: Mạch nạp RC	15
Hình 2. 8: Đường cong mạch nạp RC	16
Hình 2. 9: Mạch xả RC	17
Hình 2. 10: Đường cong mạch xả RC	17
Hình 3. 1: Sơ đồ bố trí chung	19
Hình 4. 1: Giản đồ thời gian Timer 4.....	21
Hình 4. 2: Lưu đồ giải thuật Timer 3	22
Hình 4. 3: Lưu đồ giải thuật chương trình chính	22
Hình 4. 4: Lưu đồ giải thuật Timer 1	23
Hình 4. 5: Lưu đồ giải thuật Timer 4	24
Hình 4. 6: Lưu đồ giải thuật Timer 3	25
Hình 4. 7: Giản đồ thời gian thay đổi góc đánh lửa sớm theo tốc độ	28
Hình 4. 8: Giản đồ thời gian giản thuật đánh lửa 1 lần 2 vòng quay	31
Hình 4. 9: Giản đồ thời gian đánh lửa nhiều lần lúc khởi động	35
Hình 4. 10: Tổng quan của mạch điện đánh lửa	38
Hình 4. 11: Board vi điều khiển STM8S-DISCOVERY	39
Hình 4. 12: Board vi điều khiển Arduino Uno R3	40
Hình 4. 13: Kết nối giữa board vi điều khiển.....	40
Hình 4. 14: Sơ đồ mạch ổn áp	41
Hình 4. 15: Tản nhiệt cho IC ổn áp.....	42

Hình 4. 16: Tín hiệu điện áp thực tế của mạch ổn áp	43
Hình 4. 17: Sơ đồ minh họa mạch cầu H	44
Hình 4. 18: Minh họa hoạt động khi cặp công tắc Q1, Q4 đóng	45
Hình 4. 19: Minh họa hoạt động khi cặp công tắc Q2, Q3 đóng	46
Hình 4. 20: Bảng trạng thái hoạt động lý thuyết của mạch cầu H	46
Hình 4. 21: Minh họa hoạt động của diode bảo vệ ngược dòng	47
Hình 4. 22: Chiều dòng điện khi cặp công tắc Q1, Q4 cùng đóng, cặp công tắc Q2, Q3 cùng mở.....	48
Hình 4. 23: Chiều dòng điện sau khi công tắc Q1 mở, công tắc Q4 vẫn đóng.....	49
Hình 4. 24: Chiều dòng điện khi công tắc Q2, Q4 cùng đóng.....	49
Hình 4. 25: Chiều dòng điện sau khi công tắc Q4 mở, công tắc Q2 vẫn đóng.....	50
Hình 4. 26: Chiều dòng điện khi công tắc Q2, Q3 đóng, công tắc Q1, Q4 mở	50
Hình 4. 27: Tín hiệu điều khiển mạch cầu H	51
Hình 4. 28: Tín hiệu điều khiển mạch cầu H đo bằng dao động ký.....	52
Hình 4. 29: Đường đặc tính U - t	52
Hình 4. 30: Sơ đồ mạch cầu H của mạch thực tế	53
Hình 4. 31: Mạch cầu H trong board mạch thực tế	54
Hình 4. 32: Sơ đồ chân của IR2110	56
Hình 4. 33: Minh họa cách kết nối của IR2110	58
Hình 4. 34: Sơ đồ thời gian các tín hiệu đầu vào/đầu ra	59
Hình 4. 35: IR2110 trong mạch thực tế	59
Hình 4. 36: Sơ đồ mạch đánh lửa CDI	61
Hình 4. 37: Mạch CDI trên board mạch thực tế.....	62
Hình 4. 38: Schematic của mạch nguồn ổn áp trên Altium	63
Hình 4. 39: Schematic của IC điều khiển MOSFET trên Altium	63
Hình 4. 40: Schematic của mạch CDI-AC trên Altium	64
Hình 4. 41: Sơ đồ đi dây của mạch in PCB	64



Hình 4. 42: Hình 3d mạch in PCB	65
Hình 4. 43: Mạch in thực tế	65
Hình 5. 1: Sơ đồ quy trình chế tạo testboard mạch thực tế	67
Hình 5. 2: Sơ đồ quy trình chế tạo mạch in PCB thực tế	68
Hình 6. 1: Tín hiệu đánh lửa nhiều lần lúc khởi động	69
Hình 6. 2: Tín hiệu đánh lửa nhiều 1 lần 2 vòng quay trực khuỷu	69
Hình 6. 3: Đánh lửa nhiều lần ở tốc độ động cơ 800 vòng/phút	70
Hình 6. 4: Đánh lửa 1 lần/2 vòng quay tốc độ động cơ 3000 vòng/phút	70
Hình 6. 5: Đánh lửa 1 lần/2 vòng quay ở tốc độ động cơ 6500 vòng/phút	71
Hình 6. 6: Đánh lửa 1 lần/2 vòng quay ở tốc độ động cơ 10000 vòng/phút	71
Hình 6. 7: Đánh lửa nhiều lần ở tốc độ động cơ 1700 vòng/phút	72
Hình 6. 8. Hệ thống hoạt động thực tế	73
Hình 6. 9: Hình ảnh tia lửa thực tế	73



LỜI NÓI ĐẦU

Ngày nay với sự phát triển của khoa học công nghệ thì điều khiển tự động được áp dụng rất nhiều vào các lĩnh vực của đời sống và sản xuất, đặc biệt ngành ô tô nói chung và động cơ đốt trong nói riêng khi ứng dụng các giải pháp khoa học kỹ thuật nhằm đạt được những tính năng ưu việt đang có xu hướng phát triển mạnh.

Đối với động cơ đốt trong thì việc đánh lửa ảnh hưởng rất lớn đến hiệu suất làm việc của động cơ. Hiện nay, các hệ thống đánh lửa ngày càng được cải tiến về năng lượng đánh lửa cũng như là các trạng thái đánh lửa khác nhau tùy theo trạng thái làm việc của động cơ nhằm đạt được hiệu quả tối ưu. Qua quá trình học tập và nghiên cứu, chúng em nhận thấy rằng hệ thống đánh lửa CDI - DC đang được ứng dụng rộng rãi trên nhiều dòng động cơ đặc biệt là những động cơ hoạt động ở tốc độ cao như động cơ xe gắn máy, nhờ có năng lượng đánh lửa cao và kết cấu nhỏ gọn. Bộ điều khiển (ECU) cho phép điều chỉnh góc đánh lửa sớm tùy vào trạng thái vận hành của động cơ (tốc độ động cơ, chế độ tải của động cơ), nhờ đó cải thiện được đặc tính mô men cũng như tăng tính kinh tế của động cơ và giảm lượng khí thải độc hại phát ra môi trường. Hơn nữa, khi cải tiến đặc tính điều khiển của hệ thống đánh lửa CDI-DC sẽ mang lại những tính năng giúp xe vận hành tốt hơn. Đồng thời, quá trình thiết kế hệ thống đánh lửa CDI - DC sẽ giúp chúng em có thêm nhiều kiến thức và kỹ năng như: kỹ năng lập trình, sử dụng vi điều khiển, đọc hiểu datasheet của các loại linh kiện điện tử, thiết kế và chế tạo được một số board mạch đơn giản,...Do đó, chúng em quyết định chọn đề tài “Thiết kế hệ thống đánh lửa CDI - DC” cho Đề án tốt nghiệp.

Chúng em xin chân thành cảm ơn sự hướng dẫn và chỉ bảo tận tình của TS.Trần Đăng Long, sự hỗ trợ của thầy Phạm Trần Đăng Quang và anh Hồ Hoa Nam đã giúp em hoàn thành đề tài này. Em rất mong nhận được những ý kiến đóng góp của các thầy để có thể rút ra nhiều kinh nghiệm để có thể phát triển đề tài trong tương lai.



CHƯƠNG I: GIỚI THIỆU

Một động cơ muốn hoạt động tốt sẽ cần phụ thuộc vào nhiều yếu tố, một trong những yếu tố quan trọng nhất là đánh lửa mạnh (tia lửa mạnh, liên tục, đúng thời điểm). Chính vì vai trò quan trọng của hệ thống đánh lửa, nên các nhà sản xuất đã không ngừng cải tiến để hoàn thiện tính năng của hệ thống này. Hiện nay, hệ thống đánh lửa bằng má vít không còn phổ biến do có quá nhiều khuyết điểm (tia lửa không mạnh, phải thường xuyên bảo trì, hiệu chỉnh,...). Thay vào đó là hệ thống đánh lửa CDI.

1.1. Hệ thống đánh lửa CDI:

CDI (Capacitor Discharge Ignition) là hệ thống đánh lửa điện tử được sử dụng phổ biến trên động cơ xe máy, ô tô, máy cắt cỏ, máy chạy bằng tuabin,... CDI được nâng cấp từ hệ thống đánh lửa phóng điện cảm ứng IDI, phù hợp với động cơ vận hành ở tốc độ cao. Theo đó, CDI sử dụng dòng phóng điện của tụ điện qua bobin để phát lửa cho bugi.

Ưu điểm của hệ thống CDI:

- Có khả năng sản sinh ra dòng điện khỏe và ổn định.
- Không cần tiến hành điều chỉnh tầm điện như hệ thống đánh lửa tự cảm dùng tiếp điểm.

Hệ thống đánh lửa CDI gồm có 2 dạng: CDI-DC (hệ thống đánh lửa sử dụng nguồn điện một chiều) và CDI-AC (hệ thống đánh lửa sử dụng nguồn điện một chiều). Hầu hết xe máy ngày nay đều sử dụng hệ thống đánh lửa CDI-DC.

1.2. Tổng quan đề tài:

1.2.1. Thể loại đề tài:

Thể loại: Thiết kế sản phẩm

1.2.2. Đối tượng thiết kế:

Hệ thống điều khiển đánh lửa CDI-DC trên xe gắn máy.



1.2.3. Mục tiêu đề tài:

Thiết kế hệ thống đánh lửa thông minh CDI-DC trên xe gắn máy.

1.2.4. Điều kiện làm việc:

Môi trường làm việc của hệ thống đánh lửa:

- Làm việc với tần suất cao.
- Chịu rung động và va đập.
- Chịu nhiệt độ cao.

1.2.5. Yêu cầu kỹ thuật:

Về chức năng:

- Điều chỉnh điện áp nạp tụ.
- Điều chỉnh góc đánh lửa sớm.
- Đánh lửa 1 lần/ 2 vòng quay trực khuỷu.
- Đánh lửa nhiều lần lúc khởi động và tốc độ thấp.

Về hiệu suất, hiệu quả, độ bền:

- Nhận biết nhạy với các tín hiệu đầu vào.
- Có độ bền, tuổi thọ cao.
- Tín hiệu đầu ra, ổn định, độ tin cậy cao.

Về công nghệ:

- Dễ dàng thay thế, sửa chữa khi hư hỏng.
- Giá thành rẻ, phù hợp với công nghệ hiện có.

1.2.6. Ý tưởng thực hiện và các bài toán chính cần giải quyết:

1.2.6.1. Ý tưởng thực hiện:

- Giả lập tín hiệu động cơ bằng vi điều khiển.
- Xây dựng bộ mạch điều khiển đánh lửa sớm.
- Thay đổi góc đánh lửa sớm và thời gian nạp tụ bằng biến trở.
- Xây dựng bộ mạch điều khiển mạch CDI-DC.

1.2.6.2. Các bài toán chính cần giải quyết:



- Thiết lập kết nối giữa các vi điều khiển.
- Tính toán các thông số cần thiết dựa trên sự thay đổi biến trớ.
- Xử lý các tín hiệu đầu vào để xuất ra các tín hiệu đầu ra đúng với mục tiêu điều khiển.

1.2.7. Giới hạn nội dung thực hiện:

- Thiết kế giải thuật thay đổi góc đánh lửa sớm.
- Thiết kế giải thuật thay đổi thời gian nạp tụ.
- Thiết kế giải thuật điều khiển đánh lửa 1 lần / 2 vòng quay trực khuỷu.
- Thiết kế giải thuật điều khiển đánh lửa nhiều lần lúc khởi động và tốc độ chậm.
- Thiết kế giải thuật điều khiển mạch CDI-DC.

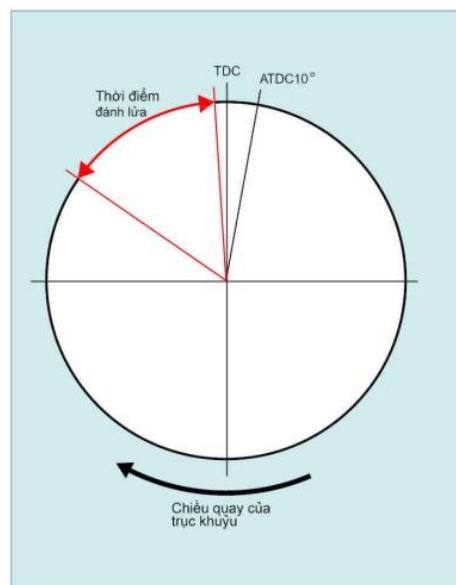
CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. Cơ sở lý thuyết điều khiển đánh lửa sớm:

2.1.1. Sự cần thiết của việc điều khiển đánh lửa sớm:

Trong động cơ xăng, hỗn hợp không khí và nhiên liệu được đánh lửa để đốt cháy (nổ) và áp lực sinh ra từ sự bốc cháy sẽ đẩy piston đi xuống.

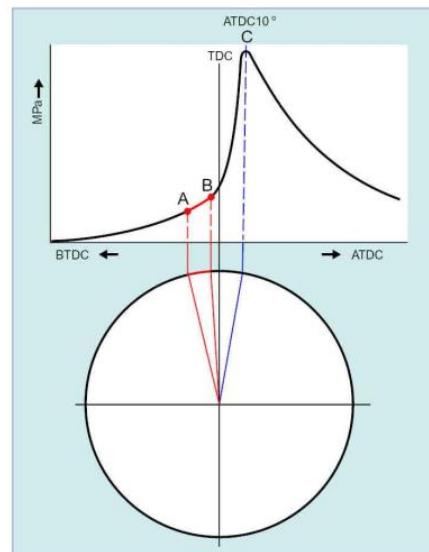
Năng lượng nhiệt được biến thành động lực có hiệu quả cao nhất khi áp lực nổ cực đại được phát sinh vào thời điểm trực khuỷu ở vị trí 10° sau điểm chết trên. Động cơ không tạo ra áp lực nổ cực đại vào thời điểm đánh lửa mà nó phát ra áp lực nổ cực đại chậm một chút sau khi đánh lửa.



Hình 2. 1: Thời điểm góc đánh lửa sớm

Vì vậy, phải đánh lửa sớm, sao cho áp lực nổ cực đại được tạo ra vào thời điểm 10° sau điểm chết trên. Thời điểm đánh lửa để động cơ đạt được áp lực nổ cực đại thường xuyên thay đổi, tùy thuộc vào điều kiện làm việc của động cơ. Vì vậy, hệ thống đánh lửa phải có khả năng đánh lửa vào thời điểm để động cơ tạo ra áp lực nổ một cách có hiệu quả nhất, phù hợp với điều kiện làm việc của động cơ.

Sự bốc cháy (nổ) của hỗn hợp không khí và nhiên liệu không phải xuất hiện ngay sau khi đánh lửa. Giai đoạn đầu, một khu vực nhỏ (hạt nhân) ở sát ngay tia lửa bắt đầu cháy, và quá trình bắt cháy này lan ra khu vực xung quanh.



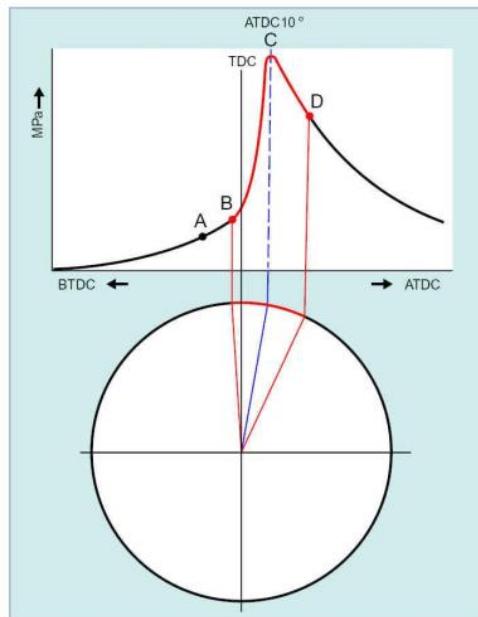
Hình 2. 2: Giai đoạn cháy trễ

Quãng thời gian từ khi hỗn hợp không khí và nhiên liệu được đánh lửa cho đến khi nó bốc cháy được gọi là giai đoạn cháy trễ (khoảng A đến B trong hình 4.2). Giai đoạn cháy trễ đo gần như không thay đổi và nó không bị ảnh hưởng của điều kiện làm việc của động cơ.

Sau khi hạt nhân ngọn lửa hình thành, ngọn lửa nhanh chóng lan truyền ra xung quanh. Tốc độ lan truyền này được gọi là tốc độ lan truyền ngọn lửa, và thời kỳ này được gọi là thời kỳ lan truyền ngọn lửa ($B \rightarrow C \rightarrow D$ trong hình 4.4).

Khi có một lượng lớn không khí nạp vào, hỗn hợp không khí và nhiên liệu trở nên có mật độ cao hơn. Vì thế, khoảng cách giữa các hạt trong hỗn hợp không khí và nhiên liệu giảm xuống, nhờ đó, tốc độ lan truyền ngọn lửa tăng lên. Ngoài ra, luồng hỗn hợp không khí – nhiên liệu xoáy lốc càng mạnh thì tốc độ lan truyền ngọn lửa càng cao. Khi

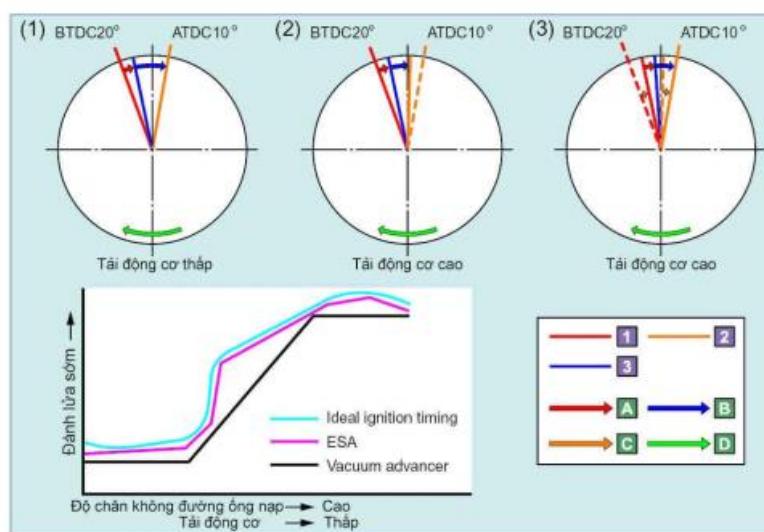
tốc độ lan truyền ngọn lửa cao, cần phải định thời đánh lửa sớm. Do đó vẫn phải điều khiển thời điểm đánh lửa theo điều kiện làm việc của động cơ.



Hình 2. 3: Giai đoạn lan truyền ngọn lửa

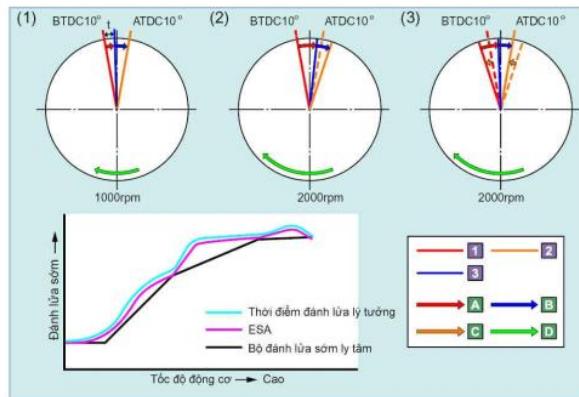
2.1.2. Điều khiển thời điểm đánh lửa:

Điều khiển theo tốc độ động cơ:



Hình 2. 4: Điều khiển theo tốc độ

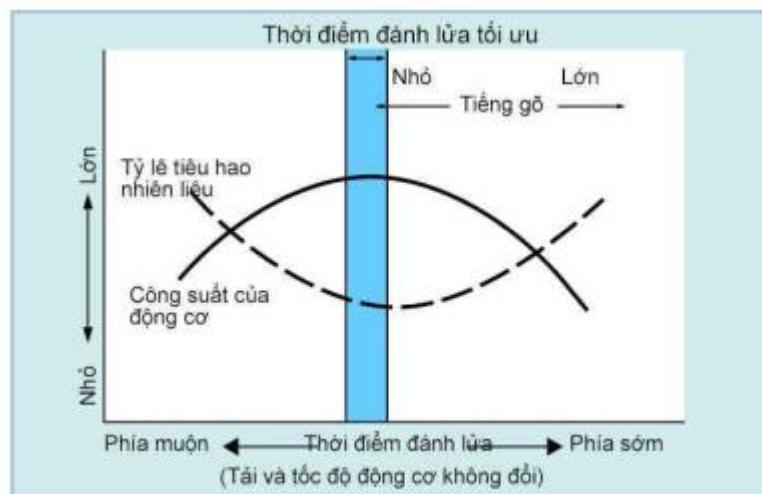
Điều khiển theo tải của động cơ:



- t : Khoảng cháy trễ
- ▀ Thời điểm đánh lửa
 - ▀ Thời điểm đánh lửa để có áp lực nổ cực đại
 - ▀ Ranh giới giữa giai đoạn cháy trễ và tốc độ lan truyền ngọn lửa
 - ▀ A Giai đoạn cháy trễ
 - ▀ B Giai đoạn lan truyền ngọn lửa
 - ▀ C Đánh lửa muộn
 - ▀ D Góc quay của trực khuỷu

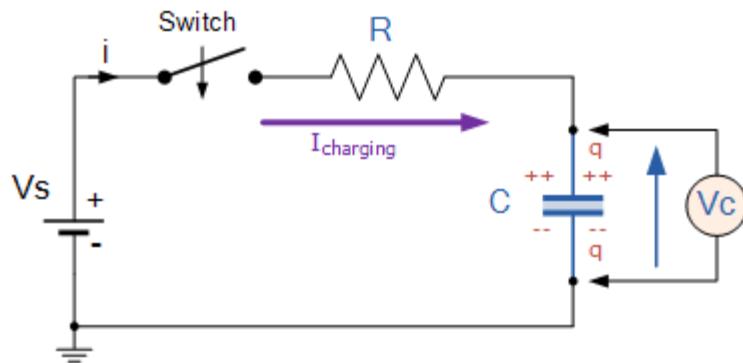
Hình 2. 5: Điều khiển theo tải của động cơ

Điều khiển theo tiếng gõ động cơ:



Hình 2. 6: Điều khiển theo tiếng gõ động cơ

2.2. Cơ sở lý thuyết tính toán thời gian nạp và xả tụ:



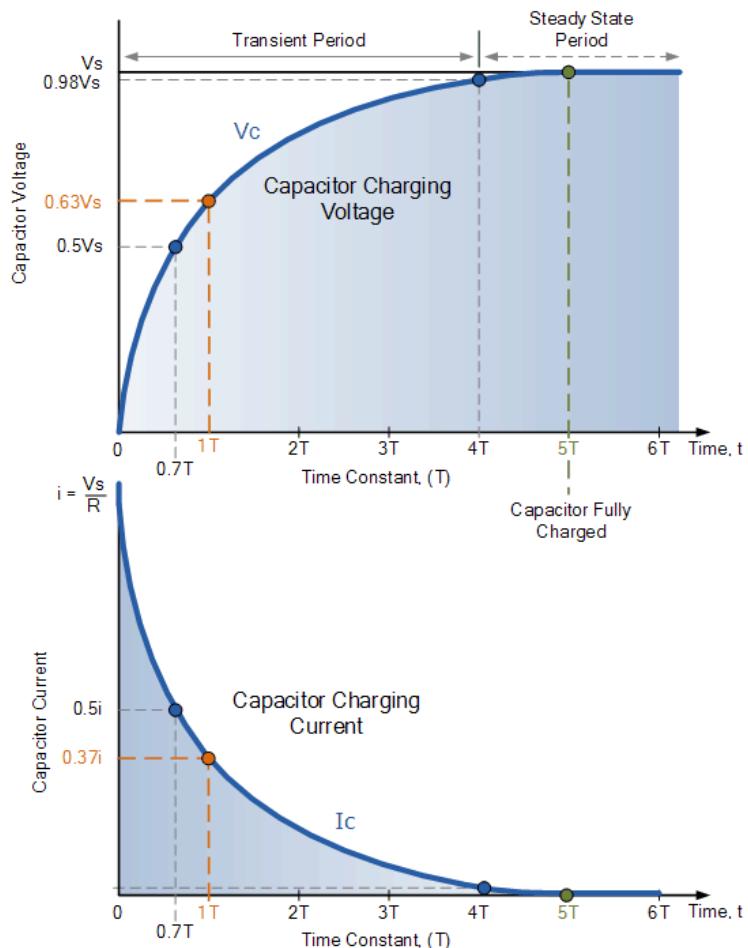
Hình 2.7: Mạch nạp RC

Thời gian nạp tụ được xác định theo công thức:

$$V_C = V_S (1 - e^{\left(\frac{-t}{RC}\right)})$$

Trong đó:

- V_C là hiệu điện thế trên tụ điện.
- V_S là điện áp cung cấp.
- e là một số vô tỉ bằng 2,7182.
- t là lời gian trôi qua kể từ khi đặt điện áp nguồn.
- RC là hằng số thời gian của mạch nạp RC.



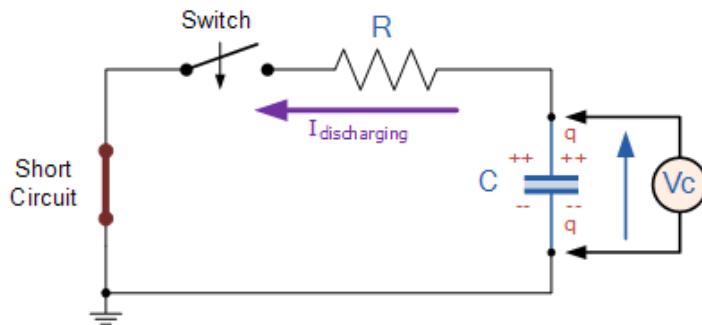
Hình 2. 8: Đường cong mạch nạp RC

Thời gian xả tụ được xác định theo công thức:

$$V_c = V_s (e^{\frac{-t}{RC}})$$

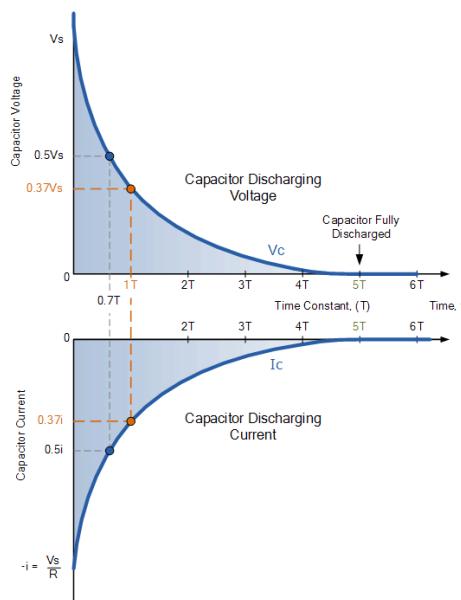
Trong đó:

- V_c là hiệu điện thế trên tụ điện.



Hình 2. 9: Mạch xả RC

- V_s là điện áp cung cấp.
- e là một số vô tỉ bằng 2,7182.
- t là lời gian trôi qua kể từ khi đặt điện áp nguồn.
- RC là hằng số thời gian của mạch nạp RC .



Hình 2. 10: Đường cong mạch xả RC



2.3. Cơ sở lý thuyết điều khiển đánh lửa nhiều lần khi khởi động:

2.3.1. Ưu điểm của điều khiển đánh lửa nhiều lần khi khởi động:

Ngay cả những động cơ hiện đại nhất đều trải qua hiện tượng gọi là COV (Coefficient of Variation) ở một số điều kiện vận hành, ví dụ như quá trình chạy cầm chừng khi hỗn hợp hòa khí có thể nghèo dần đến quá trình cháy bên trong động cơ trở thành ván dề.

Bởi vì sự hiệu quả của quá trình cháy ở động cơ là một vấn đề của nhiệt độ của động cơ ở tốc độ thấp, nên việc đánh lửa nhiều lần vào quá trình nén của động cơ trong một khoảng thời gian ngắn đảm bảo đủ năng lượng để đốt cháy toàn bộ lượng hỗn hợp không khí/nhiên liệu mặc dù hỗn hợp không khí/nhiên liệu có thể không đồng nhất hoàn toàn.

Do đó, lợi ích thực tế lones nhất của việc đánh lửa nhiều lần khi khởi động là cải thiện một cách rõ rệt ở điều kiện mà chuyển động của piston cũng như vòng xoáy tạo ra bởi hệ thống cảm ứng tiên tiến không cung cấp đủ được sự hỗn loạn trong buồng cháy để đảm bảo một hỗn hợp không khí/nhiên liệu đồng nhất trong kỳ nén của động cơ ở chế độ tải nhỏ/tốc độ thấp.

Một lợi ích khác của đánh lửa nhiều lần là cho phép giảm tốc độ cầm chừng của động cơ khi so sánh với sử dụng đánh lửa một lần. Từ đó cải thiện được khả năng tiết kiệm nhiên liệu, giảm khí thải và giảm rung động của động cơ.

2.4. Cơ sở lý thuyết điều khiển đánh lửa 1 lần/ 2 vòng quay:

2.4.1. Ưu điểm của điều khiển đánh lửa 1 lần/2 vòng quay;

Khi đánh lửa 1 lần mỗi vòng quay trong đó sẽ có những lần gọi là “đánh lửa bở” (đánh lửa vào kỳ thải của động cơ). Nhưng khi động cơ hoạt động ở tốc độ cao thì thời gian nạp tụ sẽ bị rút ngắn lại dẫn đến năng lượng đánh lửa thấp ảnh hưởng không nhỏ đến quá trình cháy của động cơ.Thêm vào đó, hệ thống đánh lửa sẽ hoạt động liên tục với tần suất cao dễ dẫn đến quá tải trong quá trình vận hành.

Do đó, đánh lửa 1 lần/ 2 vòng quay trực khuỷu sẽ giúp kéo dài thời gian nạp tụ ở những dải tốc độ cao của động cơ nhờ đó đảm bảo được năng lượng đánh lửa cho quá trình cháy ở động cơ.

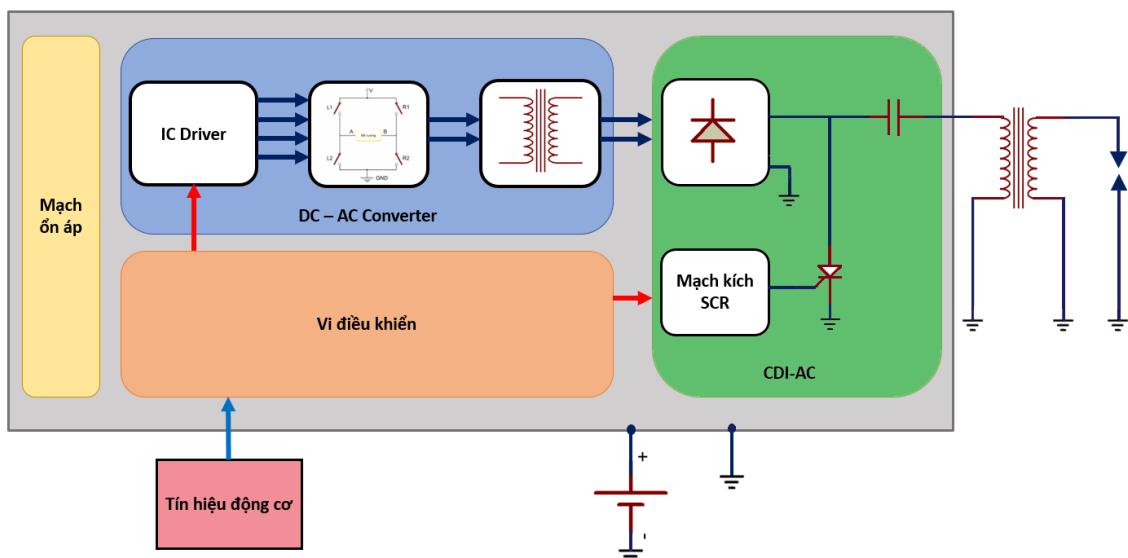
Một ưu điểm nữa là sẽ tạo ra những khoảng thời gian nghỉ cho hệ thống đánh lửa cho hệ thống đánh lửa từ đó tránh tình trạng quá tải ảnh hưởng đến cái linh kiện trong quá trình hoạt động.

CHƯƠNG III: THIẾT KẾ BỘ TRÍ CHUNG

3.1. Phương án thiết kế bộ trí chung:

- Mạch điều khiển đánh lửa sớm được kết nối dây dẫn nhận tín hiệu tốc độ động cơ và được kết nối với hai biến trở, một biến trở để điều khiển góc đánh lửa sớm và biến trở còn lại để điều khiển thời gian nạp tụ.
- Mạch điều khiển khiển mạch được kết nối dây dẫn nhận tín hiệu từ mạch điều khiển đánh lửa sớm để điều khiển mạch DC-AC converter và mạch CDI -AC.

3.2. Sơ đồ bộ trí chung:



Hình 3. 1: Sơ đồ bộ trí chung

Hệ thống bao gồm tín hiệu từ tốc độ động cơ từ trực khuỷu qua mạch gia công tín hiệu đi vào vi điều khiển thứ nhất với nhiệm vụ điều khiển đánh lửa sớm. Tín hiệu tốc độ động cơ được vi điều khiển nhận và xử lý sau đó tạo ra tín hiệu điều khiển đánh lửa sớm với thời gian nạp tụ và góc đánh lửa sớm mong muốn (thay đổi bằng biến trở).

Tín hiệu điều khiển đánh lửa sớm được sớm được vi điều khiển thứ 2 nhận và xử lý để điều khiển mạch DC-AC converter và mạch CDI - AC.

Tuy nhiên ở đồ án này, nội dung thực hiện không bao gồm cảm biến nhận tín hiệu tốc độ động cơ cũng như là mạch gia công tín hiệu, mà thay vào đó là sử dụng vi điều khiển để giả lập tín hiệu tốc độ động cơ.



CHƯƠNG IV: THIẾT KẾ KỸ THUẬT

4.1. Thiết kế giải thuật điều khiển mạch CDI-DC:

4.1.1. Thiết lập vi điều khiển:

Cấu hình xung nhịp cho vi điều khiển STM8S105C6 là 16MHz.

Chân nhận tín hiệu đầu vào Digital:

- Pin PC2 (timer 1 – channel 2), PC4 (timer 1 – channel 4)

Chân tạo tín hiệu đầu ra PWM:

- Pin PD0, PD2, PD3, PD4, PD5.

Timer/Counter 1:

- Prescaler = 1
- Tần số của STM8S = 16 MHz
- Tần số của Timer/Counter = $16\text{MHz}/1 = 16\text{MHz}$
- Mode: Input Capture (channel 2 – cạnh lên, channel 4 – cạnh xuống).
- TOP = ARR = 2499
- Tần số làm việc: $16\text{MHz}/2500 = 6400 \text{ Hz}$
- Kiểu đếm: up-counter
- Bắt cạnh lên và cạnh xuống của tín hiệu đầu vào

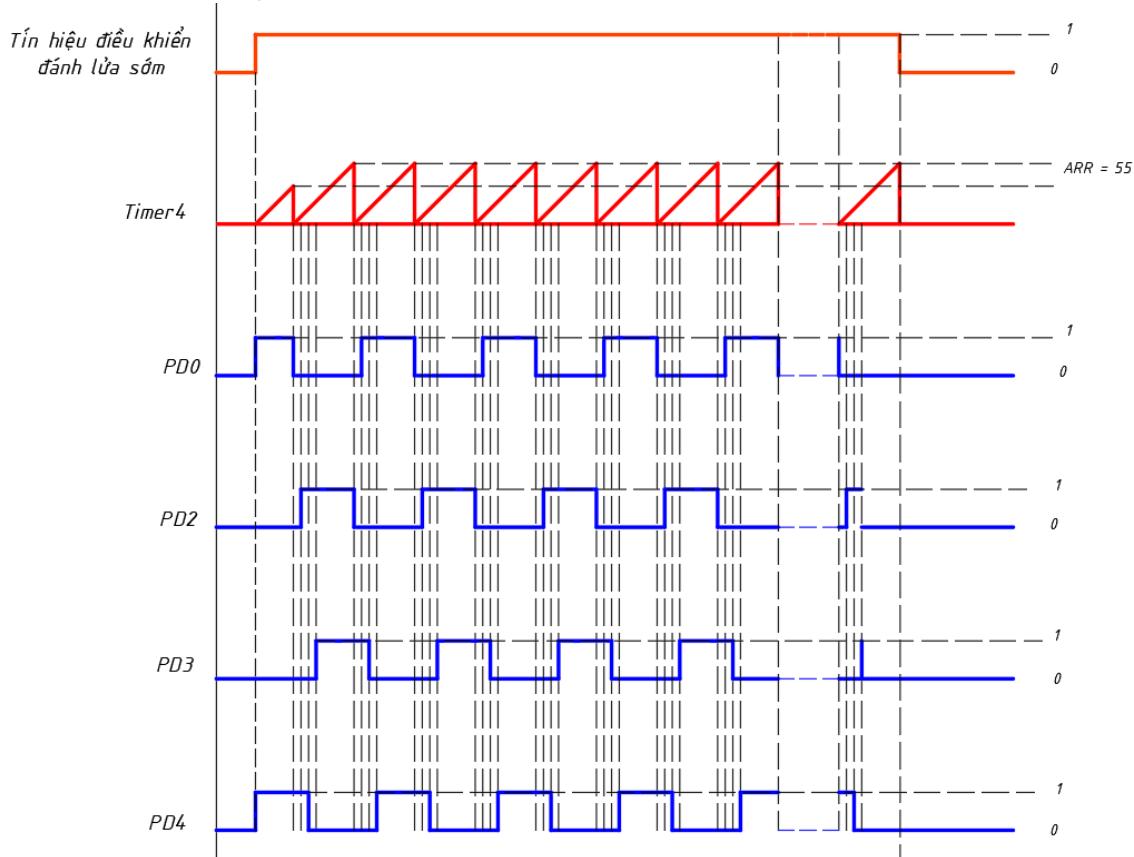
Timer/Counter 3:

- Prescaler = 1
- Tần số của STM8S= 16 MHz
- Tần số của Timer/Counter = $16\text{MHz}/1 = 16 \text{ MHz}$
- Mode: interrupt update
- TOP = ARR = 79
- Tần số làm việc: $16\text{MHz}/80 = 200000 \text{ Hz}$
- Cập nhật giá trị ARR tại BOTTOM
- Set Update Interrupt Flag lên tại TOP

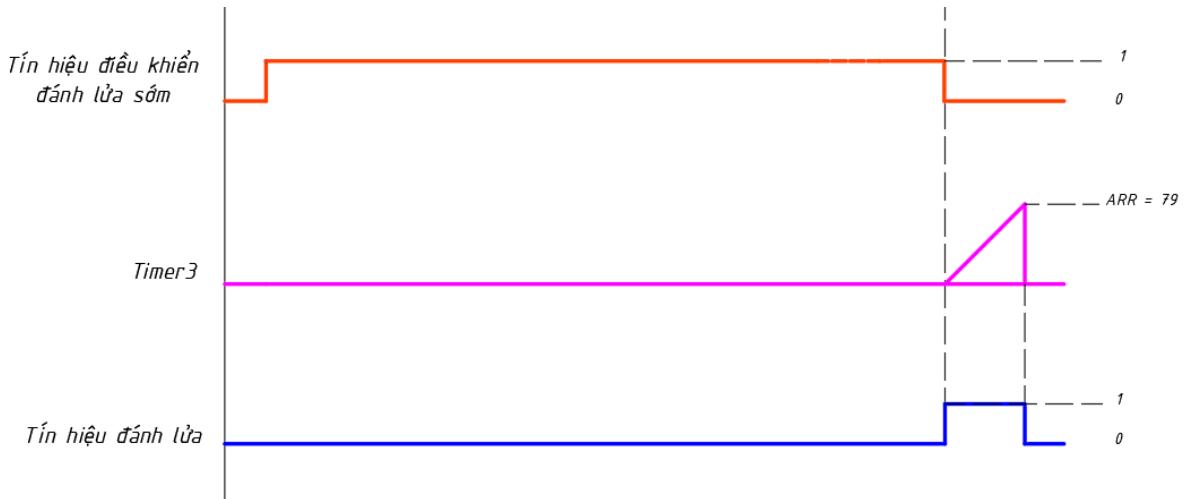
Timer/Counter 4:

- Prescaler = 1
- Tần số của STM8S= 16 MHz
- Tần số của Timer/Counter = $16\text{MHz}/1 = 16\text{ MHz}$
- Mode: interrupt update
- TOP = ARR = 55
- Tần số làm việc: $16\text{MHz}/56 = 2000000/7\text{ Hz}$
- Cập nhật giá trị ARR tại BOTTOM
- Set Update Interrupt Flag lên tại TOP

4.1.2. Giản đồ thời gian:

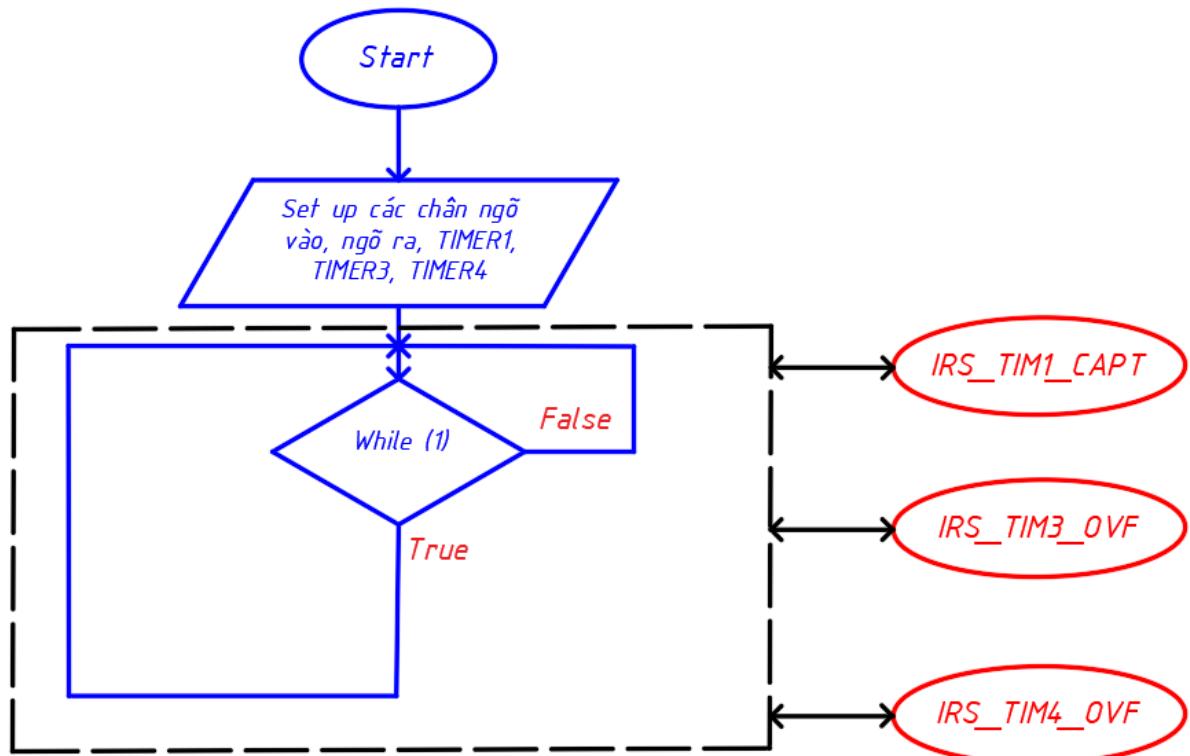


Hình 4. 1: Giản đồ thời gian Timer 4

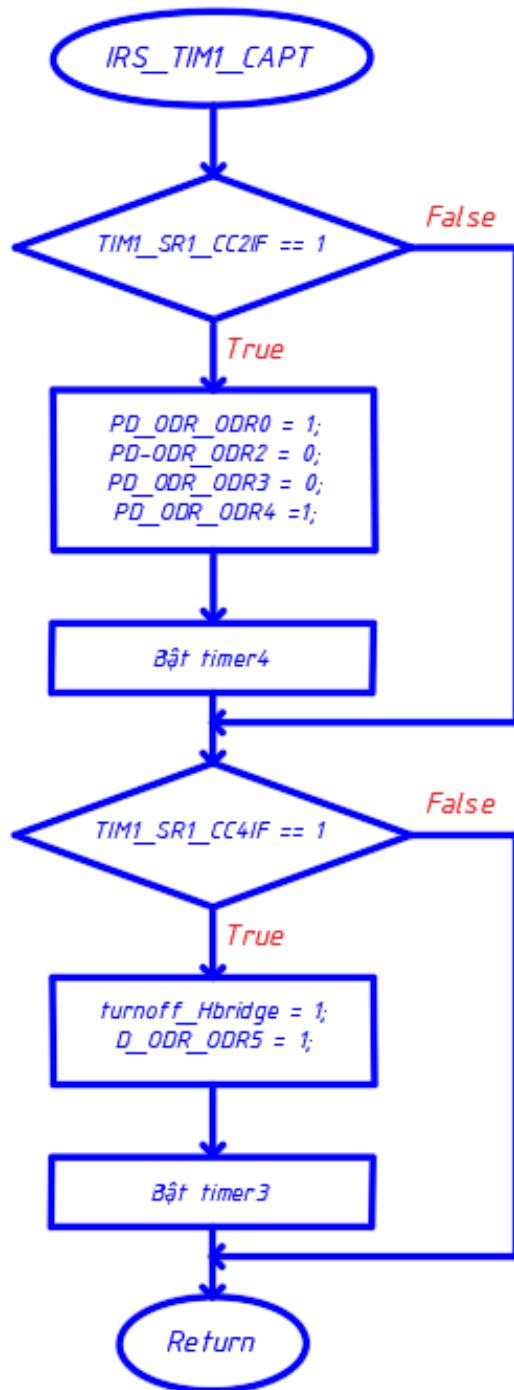


Hình 4. 2: Lưu đồ giải thuật Timer 3

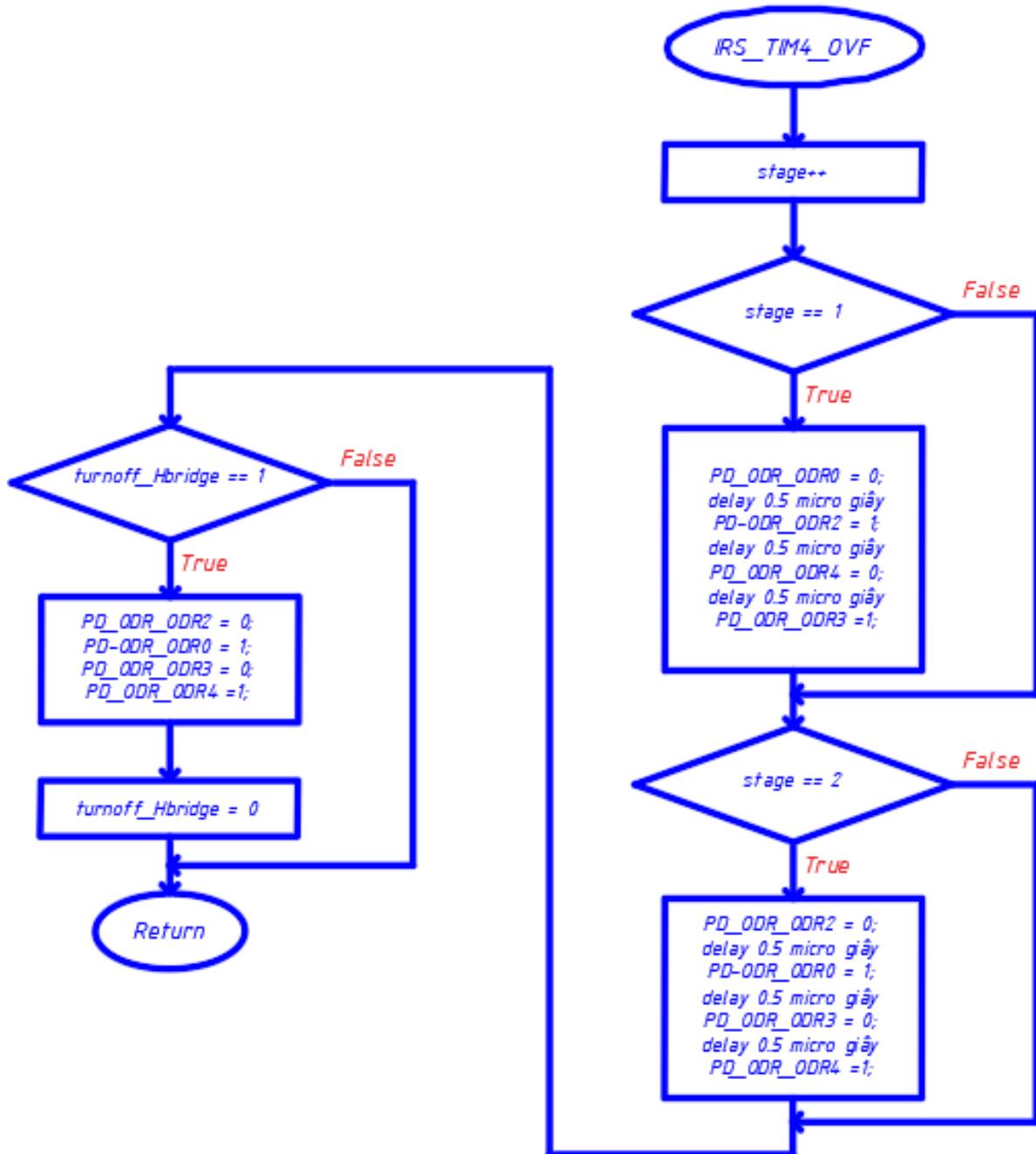
4.1.3. Lưu đồ giải thuật:



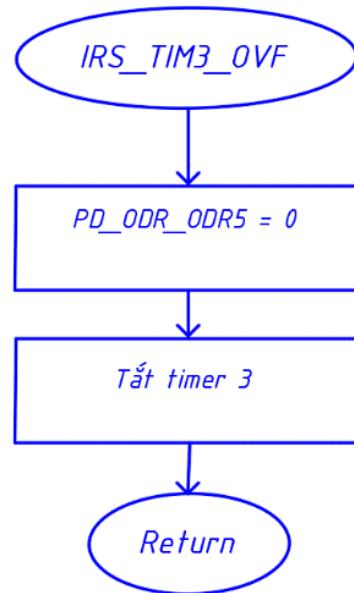
Hình 4. 3: Lưu đồ giải thuật chương trình chính



Hình 4. 4: Lưu đồ giải thuật Timer 1



Hình 4. 5: Lưu đồ giải thuật Timer 4



Hình 4. 6: Lưu đồ giải thuật Timer 3

4.1.4. Chức năng chương trình:

4.1.4.1. Timer 1:

Timer1_Input Capture:

Channel 2:

- Nhận tín hiệu cạnh lên để bật Timer 4 từ đó set trạng thái các chân PD0, PD4 ở mức cao và các chân PD2, PD3 ở mức thấp.

Channel 4:

- Nhận tín hiệu cạnh xong để bật Timer 4 từ đó set trạng thái các chân PD0, PD2, PD3, PD4 ở mức thấp.
- Set giá trị của biến turnoff_Hbridge = 1.
- Bật timer 3

4.1.4.2. Timer 4:

Timer4_Overflow:

- Thay đổi trạng thái các chân ngõ ra PD0, PD2, PD3, PD4 đúng theo cơ sở lý thuyết hoạt động của mạch cầu H.



- Nếu giá trị của biến turnoff_Hbridge = 1 thì tắt Timer 4 sau đó set giá trị của biến turnoff_Hbridge = 0.

4.2. Thiết kế giải thuật điều khiển thay đổi góc đánh lửa theo tốc độ:

4.2.1. Thiết lập vi điều khiển:

Cấu hình xung nhịp cho vi điều khiển STM8S105C6 là 16MHz.

Chân nhận tín hiệu đầu vào Digital:

- Pin PC4 (timer1 – channel 4).

Chân tạo tín hiệu đầu ra Digital:

- Pin PD0.

Timer/Counter 1:

- Prescaler = 1
- Tần số của STM8S = 16 MHz
- Tần số của Timer/Counter = $16\text{MHz}/1 = 16\text{MHz}$
- Mode: Input Capture (channel 4 – cạnh xuống) và Interrupt Update.
- TOP = ARR = 2499
- Tần số làm việc: $16\text{MHz}/2500 = 6400 \text{ Hz}$
- Kiểu đếm: up-counter
- Bắt cạnh xuống của tín hiệu đầu vào

Timer2:

- Prescaler = 1.
- Tần số của STM8S= 16 MHz.
- Tần số của Timer/Counter = $16\text{MHz}/246 = 62500 \text{ MHz}$.
- Mode: interrupt update.
- TOP = ARR
- Tần số làm việc: $16\text{MHz}/(\text{ARR} - 1)$.
- Cập nhật giá trị ARR tại BOTTOM.
- Set Update Interrupt Flag lên tại TOP.



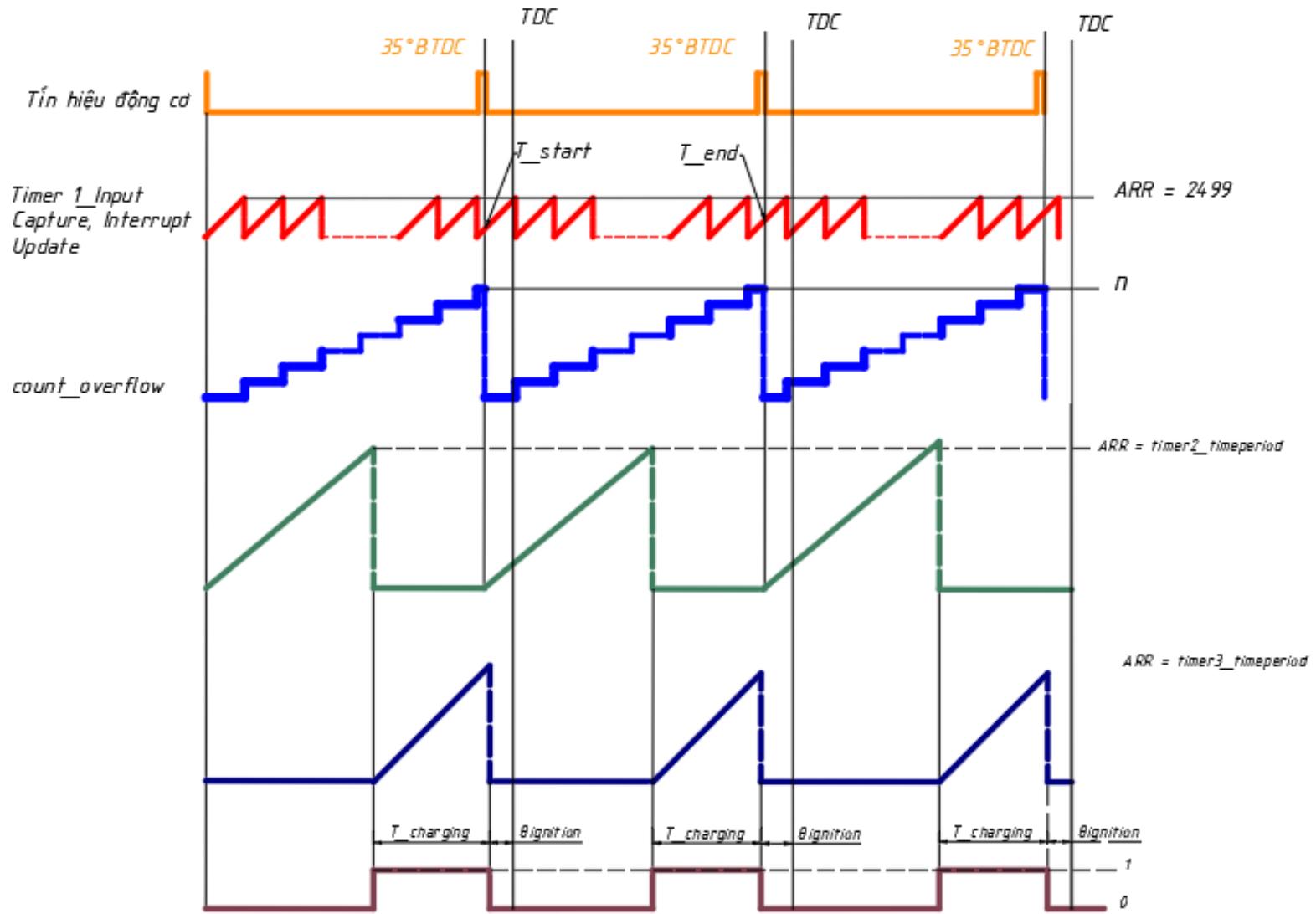
Timer3:

- Prescaler = 1.
- Tần số của STM8S= 16 MHz.
- Tần số của Timer/Counter = $16\text{Mhz}/246 = 62500 \text{ MHz}$.
- Mode: interrupt update.
- TOP = ARR
- Tần số làm việc: $16\text{MHz}/(ARR - 1)$.
- Cập nhật giá trị ARR tại BOTTOM.
- Set Update Interrupt Flag lên tại TOP.

Timer/Counter 4:

- Prescaler = 64
- Tần số của STM8S= 16 MHz
- Tần số của Timer/Counter = $16\text{Mhz}/256 = 250000 \text{ Hz}$
- Mode: interrupt update
- TOP = ARR = 249
- Tần số làm việc: $250000/250 = 1000\text{Hz}$
- Cập nhật giá trị ARR tại BOTTOM
- Set Update Interrupt Flag lên tại TOP

4.2.2. Giản đồ thời gian:



Hình 4. 7: Giản đồ thời gian thay đổi góc đánh lửa sớm theo tốc độ

4.2.3. Chức năng chương trình:

4.2.3.1. Chương trình chính:

- Cấu hình xung nhịp, các chân ngõ ra, ngõ vào.
- Thiết lập Timer/Counter 1, Timer/Counter 2, Timer/Counter 3, Timer/Counter 4.
- Bắt đầu vòng lặp vô tận.



4.2.3.2. Timer 1:

Timer1_Overflow:

- Tăng giá trị biến count_overflow.

Timer1_Input Capture:

- Lấy giá trị từ thanh ghi CCR4 gán vào biến T_end

- Tính T_oneround.

$$T_{oneround} = T_{end} - T_{start} + count_overflow * 2499$$

- Reset giá trị biến count_overflow

- Gán giá trị của T_end cho T_start

- Tính giá trị góc đánh lửa sớm dựa vào tốc độ đo được.

- Cập nhật giá trị overflow của timer2.

- Bật timer 2.

4.2.3.3. Timer 2:

- Set chân PD0 ở mức cao.

- Cập nhật giá trị overflow của timer 3

- Bật timer 3

- Tắt timer 2

4.2.3.4. Timer 3:

- Set chân PD0 ở mức thấp.

- Tắt timer 3.

4.3. Thiết kế giải thuật điều khiển đánh lửa 2 lần/ vòng quay:

4.3.1. Thiết lập vi điều khiển:

Cấu hình xung nhịp cho vi điều khiển STM8S105C6 là 16MHz.

Chân nhận tín hiệu đầu vào Digital:

- Pin PC4 (timer1 – channel 4).

Chân tạo tín hiệu đầu ra Digital:

- Pin PD0.



Timer/Counter 1:

- Prescaler = 1
- Tần số của STM8S = 16 MHz
- Tần số của Timer/Counter = $16\text{MHz}/1 = 16\text{MHz}$
- Mode: Input Capture (channel 4 – cạnh xuống) và Interrupt Update.
- TOP = ARR = 2499
- Tần số làm việc: $16\text{MHz}/2500 = 6400 \text{ Hz}$
- Kiểu đếm: up-counter
- Bắt cạnh xuống của tín hiệu đầu vào

Timer2:

- Prescaler = 1.
- Tần số của STM8S= 16 MHz.
- Tần số của Timer/Counter = $16\text{MHz}/246 = 62500 \text{ MHz}$.
- Mode: interrupt update.
- TOP = ARR
- Tần số làm việc: $16\text{MHz}/(\text{ARR} - 1)$.
- Cập nhật giá trị ARR tại BOTTOM.
- Set Update Interrupt Flag lên tại TOP.

Timer3:

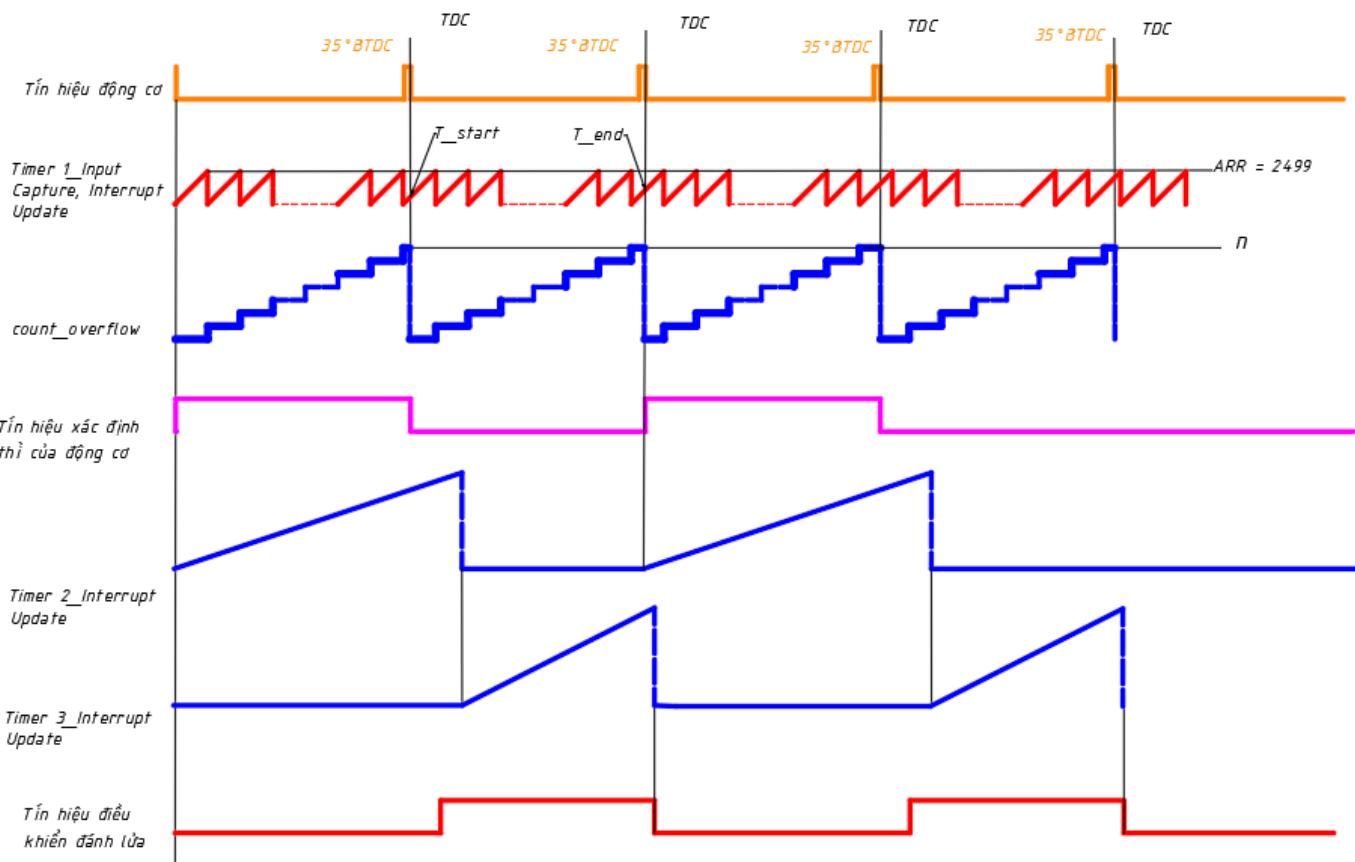
- Prescaler = 1.
- Tần số của STM8S= 16 MHz.
- Tần số của Timer/Counter = $16\text{MHz}/246 = 62500 \text{ MHz}$.
- Mode: interrupt update.
- TOP = ARR
- Tần số làm việc: $16\text{MHz}/(\text{ARR} - 1)$.
- Cập nhật giá trị ARR tại BOTTOM.

- Set Update Interrupt Flag lên tại TOP.

Timer/Counter 4:

- Prescaler = 64
- Tần số của STM8S= 16 MHz
- Tần số của Timer/Counter = $16\text{Mhz}/256 = 250000 \text{ Hz}$
- Mode: interrupt update
- TOP = ARR = 249
- Tần số làm việc: $250000/250 = 1000\text{Hz}$
- Cập nhật giá trị ARR tại BOTTOM
- Set Update Interrupt Flag lên tại TOP

4.3.2. Giản đồ thời gian:



Hình 4. 8: Giản đồ thời gian giản thuật đánh lửa 1 lần 2 vòng quay



4.3.3. Chức năng chương trình:

4.3.3.1. Chương trình chính:

- Cấu hình xung nhịp, các chân ngõ ra, ngõ vào.
- Thiết lập Timer/Counter 1, Timer/Counter 2, Timer/Counter 3, Timer/Counter 4.
- Bắt đầu vòng lặp vô tận.

4.3.3.2. Timer 1:

Timer1_Overflow:

- Tăng giá trị biến count_overflow.

Timer1_Input Capture:

- Lấy giá trị từ thanh ghi CCR4 gán vào biến T_end
- Tính T_oneround.

$$T_{oneround} = T_{end} - T_{start} + count_overflow * 2499$$

- Reset giá trị biến count_overflow
- Gán giá trị của T_end cho T_start
- Tính giá trị góc đánh lùa sớm dựa vào tốc độ được.
- Kiểm tra giá trị biến biểu thị kỳ của động cơ
- Nếu kỳ tiếp theo là kỳ nổ xả thì:
 - Cập nhật giá trị overflow của timer2.
 - Bật timer 2.
- Đảo giá trị biến biểu thị kỳ của động cơ.

4.3.3.3. Timer 2:

- Set chân PD0 ở mức cao.
- Cập nhật giá trị overflow của timer 3
- Bật timer 3
- Tắt timer 2

4.3.3.4. Timer 3:

- Set chân PD0 ở mức thấp.



- Tắt timer 3.

4.4. Thiết kế giải thuật đánh lửa nhiều lần lúc khởi động:

4.4.1. Thiết lập vi điều khiển:

Cấu hình xung nhịp cho vi điều khiển STM8S105C6 là 16MHz.

Chân nhận tín hiệu đầu vào Digital:

- Pin PC4 (timer1 – channel 4).

Chân tạo tín hiệu đầu ra Digital:

- Pin PD0.

Timer/Counter 1:

- Prescaler = 1

- Tần số của STM8S = 16 MHz

- Tần số của Timer/Counter = $16\text{MHz}/1 = 16\text{MHz}$

- Mode: Input Capture (channel 4 – cạnh xuống) và Interrupt Update.

- TOP = ARR = 2499

- Tần số làm việc: $16\text{MHz}/2500 = 6400 \text{ Hz}$

- Kiểu đếm: up-counter

- Bắt cạnh xuống của tín hiệu đầu vào

Timer2:

- Prescaler = 1.

- Tần số của STM8S= 16 MHz.

- Tần số của Timer/Counter = $16\text{MHz}/246 = 62500 \text{ MHz}$.

- Mode: interrupt update.

- TOP = ARR

- Tần số làm việc: $16\text{MHz}/(\text{ARR} - 1)$.

- Cập nhật giá trị ARR tại BOTTOM.

- Set Update Interrupt Flag lên tại TOP.

Timer3:

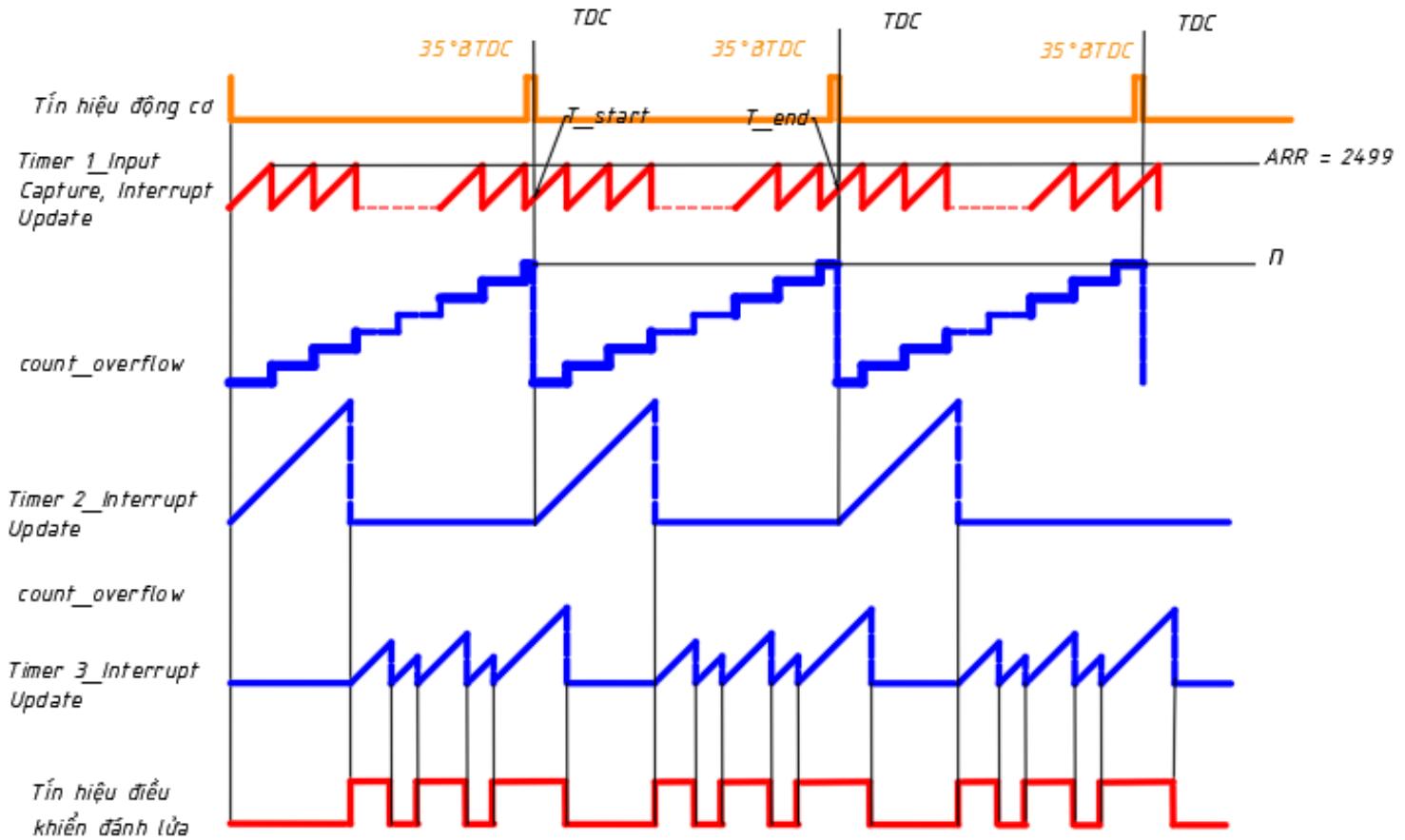


- Prescaler = 1.
- Tần số của STM8S= 16 MHz.
- Tần số của Timer/Counter = $16\text{Mhz}/246 = 62500 \text{ MHz}$.
- Mode: interrupt update.
- TOP = ARR
- Tần số làm việc: $16\text{MHz}/(\text{ARR} - 1)$.
- Cập nhật giá trị ARR tại BOTTOM.
- Set Update Interrupt Flag lên tại TOP.

Timer/Counter 4:

- Prescaler = 64
- Tần số của STM8S= 16 MHz
- Tần số của Timer/Counter = $16\text{Mhz}/256 = 250000 \text{ Hz}$
- Mode: interrupt update
- TOP = ARR = 249
- Tần số làm việc: $250000/250 = 1000\text{Hz}$
- Cập nhật giá trị ARR tại BOTTOM
- Set Update Interrupt Flag lên tại TOP

4.4.2. Giản đồ thời gian:



Hình 4. 9: Giản đồ thời gian đánh lửa nhiều lần lúc khởi động

4.4.3. Chức năng chương trình:

4.4.3.1. Chương trình chính:

- Cấu hình xung nhịp, các chân ngõ ra, ngõ vào.
- Thiết lập Timer/Counter 1, Timer/Counter 2, Timer/Counter 3, Timer/Counter 4.
- Bắt đầu vòng lặp vô tận.

4.4.3.2. Timer 1:

Timer1_Overflow:

- Tăng giá trị biến count_overflow.



Timer1_Input Capture:

- Lấy giá trị từ thanh ghi CCR4 gán vào biến T_end
- Tính T_oneround.

$$T_{oneround} = T_{end} - T_{start} + count_overflow * 2499$$

- Reset giá trị biến count_overflow
- Gán giá trị của T_end cho T_start
- Tính giá trị góc đánh lửa sớm dựa vào tốc độ được.
- Kiểm tra điều kiện đánh lửa nhiều lần
- Tính số lần đánh lửa.
- Cập nhật giá trị overflow của timer2
- Bật timer 2

4.4.3.3. Timer 2:

- Set chân PD0 ở mức cao.
- Cập nhật giá trị overflow của timer 3
- Bật timer 3
- Tắt timer 2

4.4.3.4. Timer 3:

- Kiểm tra số lần đánh lửa.
- Kiểm tra biến tính hiệu trạng thái đánh lửa.
- Kiểm tra biến kết thúc quá trình đánh lửa.
- Cập nhật giá trị overflow của timer 3.
- Bật/tắt timer 3.

4.4.3.5. Timer 4:

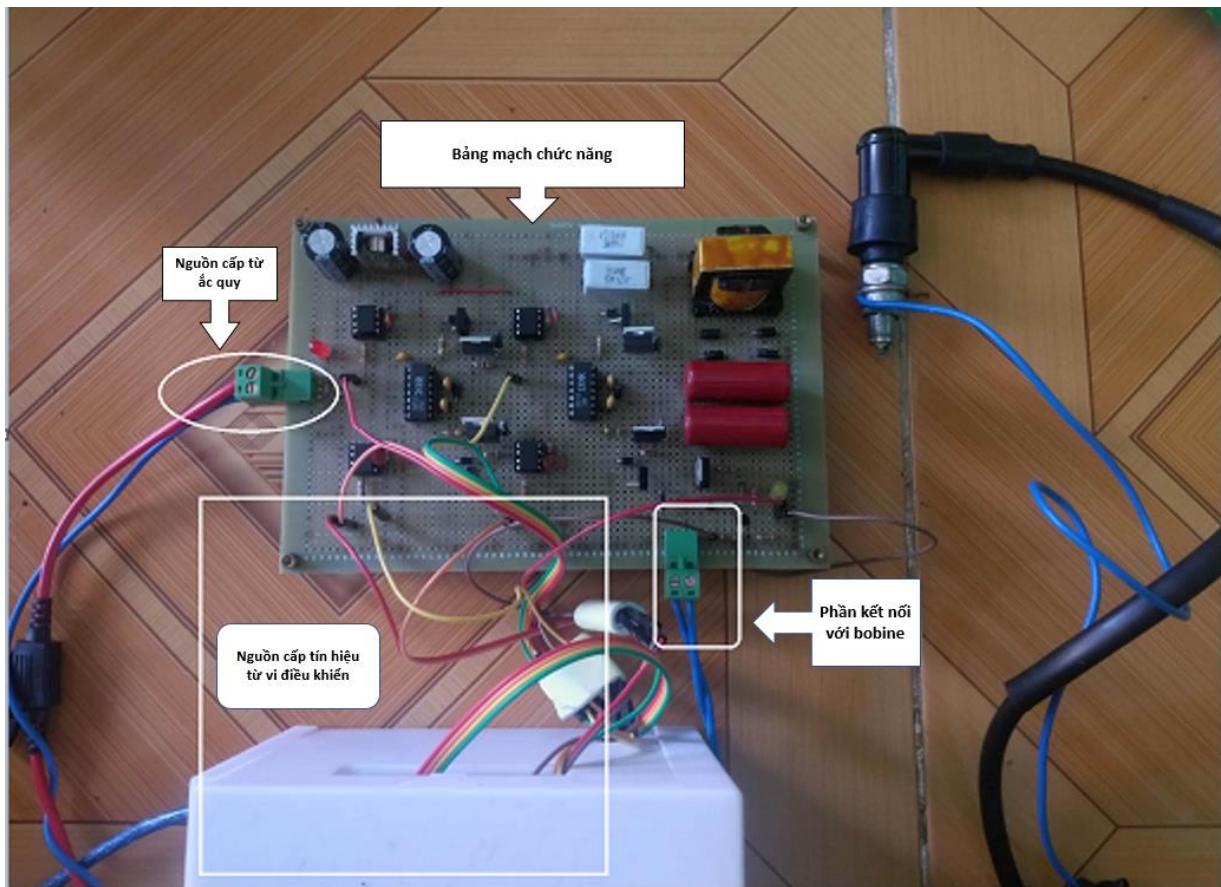
- Tăng giá trị của biến đếm lên 1 đơn vị khi biến đếm đạt giá trị 15000 tương đương với 15 giây.
- Thay đổi giá trị biến điều kiện đánh lửa nhiều lần.
- Tắt Timer 4.



4.4. Thiết kế mạch điện – điện tử điều khiển đánh lửa

Công việc thực hiện:

- Kết nối các vi điều khiển cung cấp tín hiệu tới các thành phần của mạch điện.
- Thiết kế và chế tạo các mạch chức năng điều khiển:
 - + Mạch cấp nguồn ổn áp cho các linh kiện trong mạch.
 - + Mạch DC-AC converter
 - + Mạch CDI - AC
 - + Sử dụng phần mềm Altium để đưa ra sơ đồ nguyên lý, cho ra bản vẽ mạch 2d và in mạch PCB



Hình 4. 10: Tổng quan của mạch điện đánh lửa

4.4.1. Mạch tín hiệu điều khiển

- Sử dụng board điều khiển STM8S-DISCOVERY, có trang bị vi điều khiển STM8S105C6T6 dùng để cung cấp các tín hiệu điều khiển:

- + Là vi điều khiển 8 bit
- + Lõi xử lý STM8S Core
- + Tốc độ xung nhịp: 16MHz
- + RAM: 2KB
- + Có các tính năng như LED, nút nhấn, cổng nối tiếp, cổng SPI, cổng I2C, cổng ADC, cổng UART, ...

- + Nguồn cung cấp: 5VDC

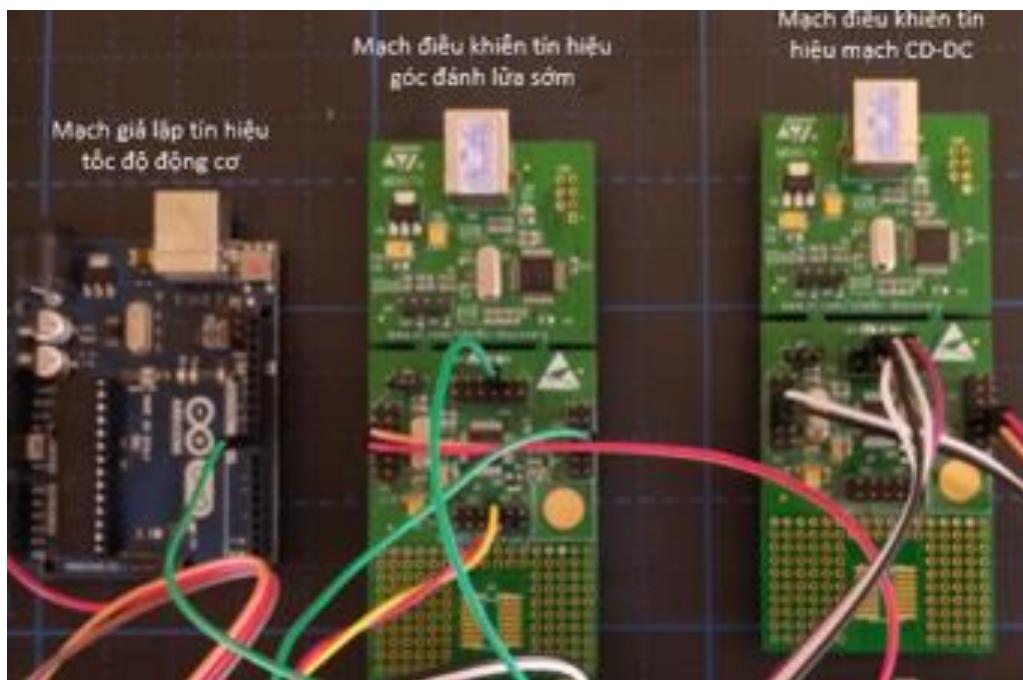


Hình 4. 11: Board vi điều khiển STM8S-DISCOVERY

- Sử dụng thêm board điều khiển Arduino Uno R3, chứa vi điều khiển ATmega328P
 - + Là vi điều khiển 8 bit
 - + Bộ nhớ Flash: 32KB
 - + Tốc độ xung nhịp: 16MHz
 - + RAM: 2KB
 - + Điện áp hoạt động: 5VDC
 - + Các tính năng: 14 chân I/O số (trong đó 6 chân có thể sử dụng để tạo tín hiệu PWM), 6 chân đầu vào ADC 10-bit, cổng nối tiếp UART, cổng SPI, cổng I2C, v.v.



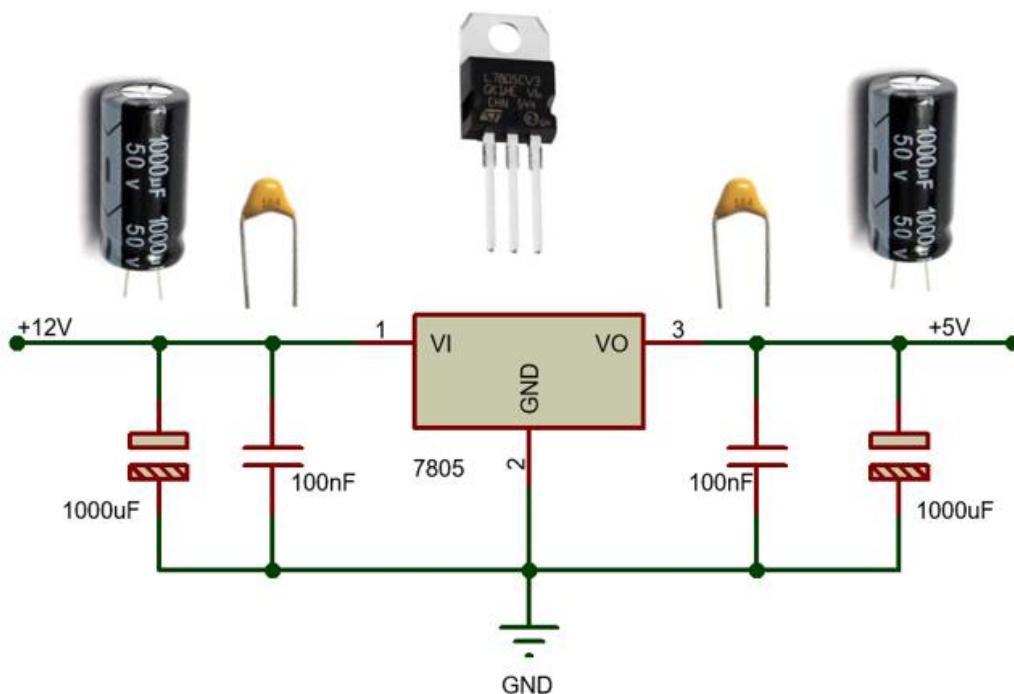
Hình 4. 12: Board vi điều khiển Arduino Uno R3



Hình 4. 13: Kết nối giữa board vi điều khiển

4.4.2. Mạch ổn áp

- Các linh kiện trong mạch cần được cung cấp một nguồn điện áp ổn định để hoạt động một cách ổn định và hiệu quả. Mạch ổn áp sẽ được sử dụng để giữ cho điện áp đầu vào và đầu ra ổn định. Trong mạch chức năng bao gồm các linh kiện hoạt động với điện áp đầu vào là 5V.



Hình 4. 14: Sơ đồ mạch ổn áp

- Mạch ổn áp sử dụng linh kiện điện tử IC ổn áp L7805, được sử dụng để chuyển đổi điện áp đầu vào không ổn định thành một điện áp DC ổn định và cố định là 5V.
- Điện áp đầu vào của L7805 có thể đạt tới 35V và cung cấp dòng lên tới 1,5A.
- L7805 là một loại vi mạch ổn áp tuyến tính, nghĩa là nó sẽ ổn định điện áp bằng cách chuyển đổi năng lượng dư thừa thành nhiệt độ. Do đó khi sử dụng L7805 cần phải đảm bảo có biện pháp tỏa nhiệt để tránh việc quá nhiệt và gây hư hỏng linh kiện

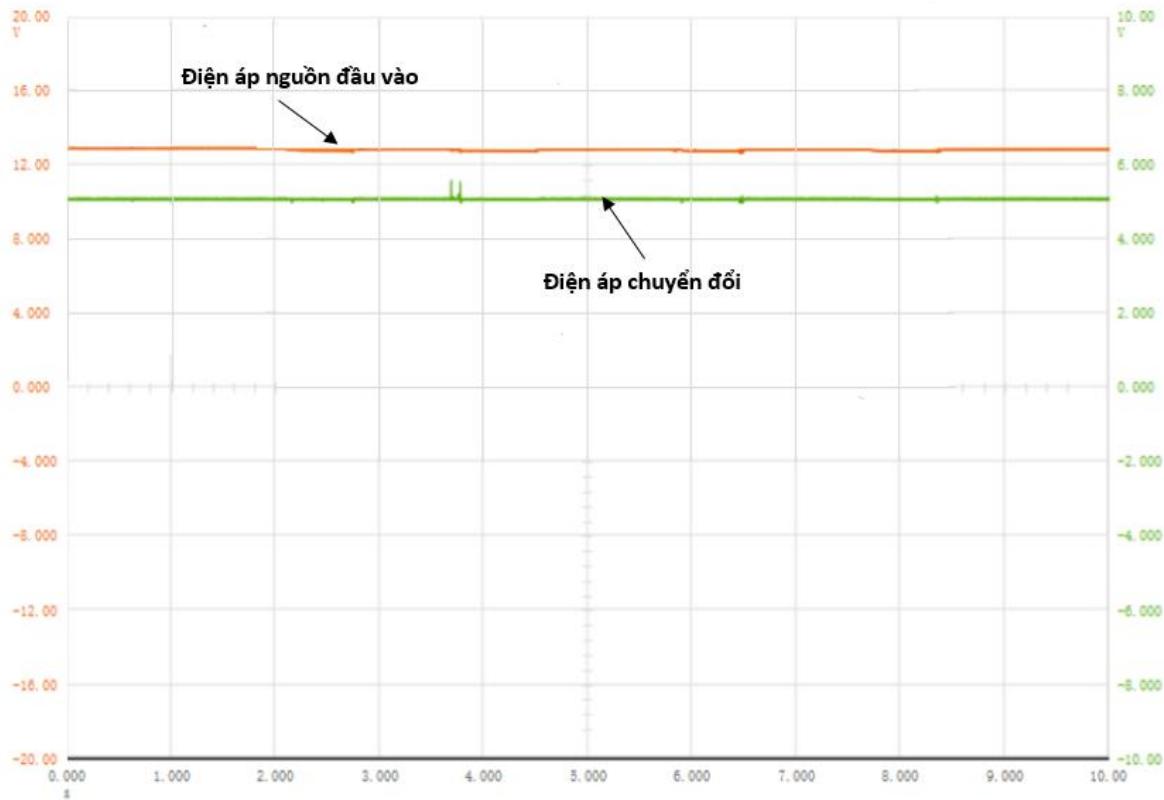


Hình 4. 15: Tản nhiệt cho IC ổn áp

- Có mắc thêm 2 loại tụ mắc giữa 2 bên điện áp đầu vào và đầu ra của IC ổn áp L7805, điều này sẽ giúp cho mạch ổn áp L7805 giảm nhiễu và ổn định đầu ra, đảm bảo hoạt động chính xác.

+ Tụ phân cực: tụ này sẽ có điện dung lớn, giúp cung cấp năng lượng cho mạch khi mạch đang hoạt động. Trong quá trình hoạt động của mạch ổn áp, đầu ra sẽ có sự dao động do tác động của nhiễu, và tụ phân cực sẽ giữ cho điện áp đầu ra ổn định bằng cách cung cấp năng lượng còn thiếu khi đầu vào không đủ.

+ Tụ không phân cực: Tụ sẽ có điện dung nhỏ, được sử dụng để lọc và giảm nhiễu, giúp đầu ra ổn định.



Hình 4. 16: Tín hiệu điện áp thực tế của mạch ốn áp

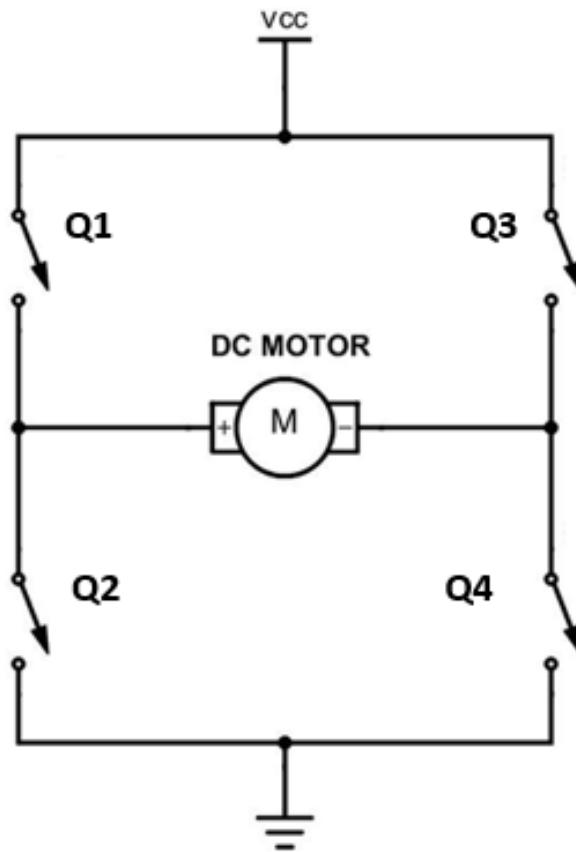
4.4.3. Mạch DC-AC converter

4.4.3.1. Mạch cầu H

* Lý thuyết

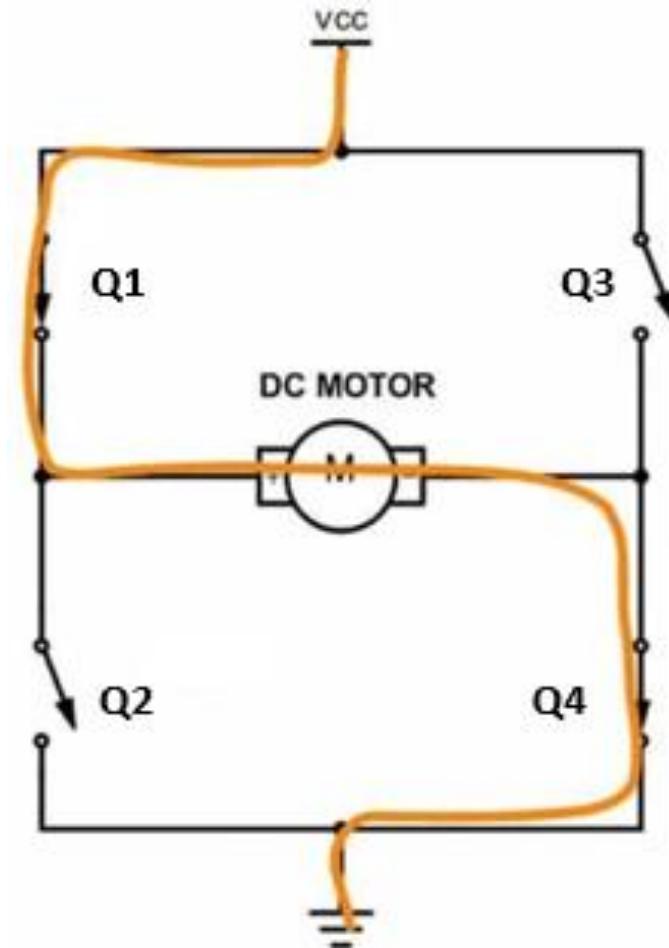
- Mạch cầu H (H – bridge) là một mạch điện tử được sử dụng để điều khiển động cơ hoặc tải điện khác theo hai hướng, cho phép điều khiển chuyển động của tải điện sang trái hoặc sang phải.

- Mạch cầu H bao gồm 4 công tắc (switch), thường là transistor hoặc MOSFET, được kết nối thành 2 cặp, mỗi cặp sẽ gồm 2 công tắc. Và các công tắc được kết nối với tải điện để tạo ra một mạch kết nối có khả năng đảo chiều dòng điện và thay đổi phương hướng dòng điện



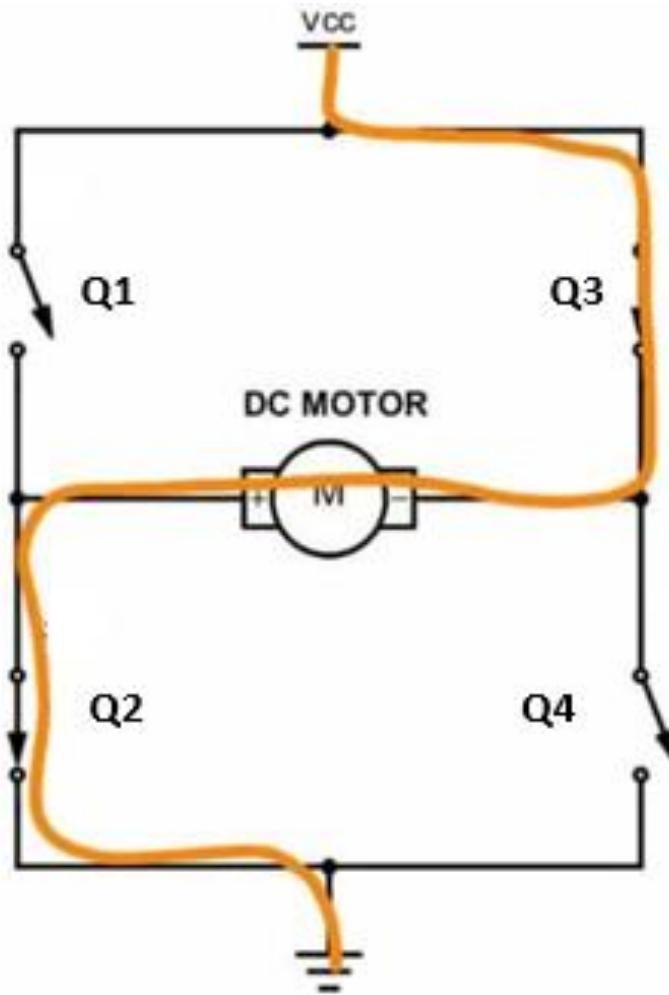
Hình 4. 17: Sơ đồ minh họa mạch cầu H

- Khi các cặp công tắc tương ứng được kích hoạt, tải điện được kết nối với nguồn điện và dòng điện sẽ chảy qua tải điện theo hướng tương ứng. Khi cặp công tắc khác được kích hoạt, tải điện sẽ được kết nối với đất, và dòng điện sẽ chảy qua tải điện theo hướng ngược lại.
- Để minh họa cụ thể, cho cặp công tắc Q1, Q4 cùng kích hoạt và cặp công tắc Q2, Q3 không được kích hoạt. Khi này dòng điện sẽ đi từ nguồn đầu vào VCC, qua công tắc Q1, qua tải, qua công tắc Q4 và về điểm nối đất. Và dòng điện sẽ chảy theo chiều tương ứng.



Hình 4. 18: Minh họa hoạt động khi cắp công tắc $Q1, Q4$ đóng

- Và khi cắp công tắc $Q2, Q3$ được kích hoạt và cắp công tắc $Q1, Q4$ không được kích hoạt. Dòng điện sẽ đi từ nguồn đầu vào VCC, qua công tắc $Q3$, qua tải, qua công tắc $Q2$ và về điểm nối đất. Và khi đó, dòng điện sẽ chảy ngược chiều lại so với chiều ban đầu.



Hình 4. 19: Minh họa hoạt động khi cặp công tắc Q2, Q3 đóng

- Còn nếu cùng kích hoạt cặp công tắc Q1, Q3 hoặc Q2, Q4 thì sẽ không có dòng điện chạy qua, mạch cầu H sẽ không hoạt động.

Q1	Q2	Q3	Q4	Kết quả
ON	OFF	OFF	ON	Dẫn
ON	OFF	ON	OFF	Không dẫn
OFF	ON	ON	OFF	Dẫn
OFF	ON	OFF	ON	Không dẫn

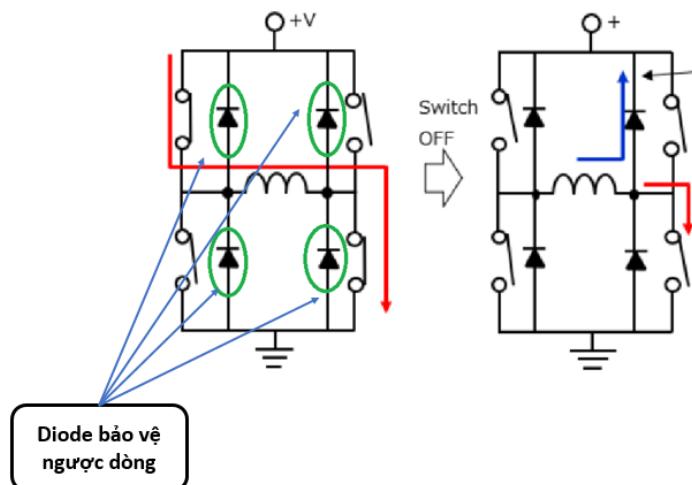
Hình 4. 20: Bảng trạng thái hoạt động lý thuyết của mạch cầu H

- Lưu ý, không được phép đóng cùng lúc công tắc Q1 và Q2 hay Q3 và Q4 hoặc cả 4 công tắc, điều này sẽ tạo ra một đường dẫn trực tiếp từ nguồn VCC xuống điểm nối đất và gây ra hiện tượng ngắn mạch, gây hư hỏng cho mạch.

* Điều khiển thực tế

- Trong thực tế khi các công tắc được kích hoạt, năng lượng được lưu trữ trong cuộn cảm và tải sẽ được truyền đến nguồn cáp. Tuy nhiên, khi công tắc đóng, với bản chất cảm ứng điện từ dòng điện khi đi qua cuộn dây hoặc động cơ DC sẽ chống lại mọi thay đổi của dòng điện chạy qua chúng. Do đó, khi dòng điện bị ngắt, điện áp tăng đột biến có làm làm hỏng các linh kiện. Hiện tượng này còn gọi là hiện tượng điện áp hồi.

- Và để khắc phục hiện tượng này, cần sử dụng các biện pháp bảo vệ, và phổ biến là sử dụng các linh kiện diode bảo vệ ngược dòng. Các diode này được gắn song song với các công tắc trong mạch cầu H, để tạo một đường dẫn thấp trở kháng cho dòng điện hồi. Khi một công tắc được tắt, năng lượng của dòng điện của cuộn dây được giải phóng và tạo ra một điện áp ngược hướng. Diode bảo vệ này cho phép dòng điện lưu qua chúng, giúp giảm điện áp hồi và bảo vệ các công tắc khỏi bị hư hỏng.

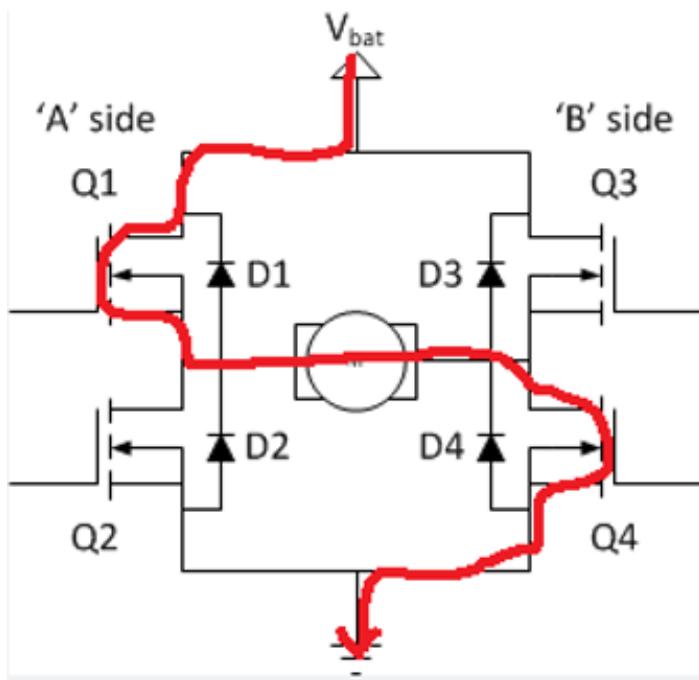


Hình 4. 21: Minh họa hoạt động của diode bảo vệ ngược dòng

- Ngoài ra, các công tắc được sử dụng trong mạch cầu H thực tế là các linh kiện bán dẫn như transistor hay mosfet, các linh kiện này không thể bật tắt ngay lập tức được mà cần phải có một khoảng thời gian, ta gọi đó là khoảng “dead time”, nếu không có khoảng thời gian này thì có thể gây ra việc trùng dãy giữa các công tắc, dẫn đến hiện tượng ngắn mạch và làm hư hỏng linh kiện.

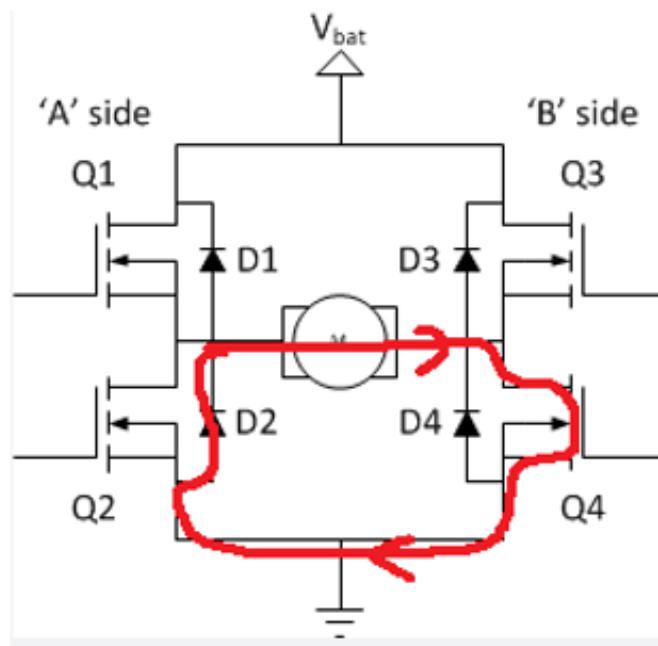
- Để minh họa một cách cụ thể hơn, hoạt động của mạch cầu H được chia ra thành nhiều trạng thái để đảm bảo an toàn:

+ Trạng thái 1: Cặp công tắc Q1, Q4 cùng đóng, cặp công tắc Q2, Q3 cùng mở. Ở trạng thái này, dòng điện sẽ đi từ nguồn cấp, qua công tắc Q1, qua tải điện, qua công tắc Q4 và về điểm nối đất, tạo thành 1 chiều dòng điện.



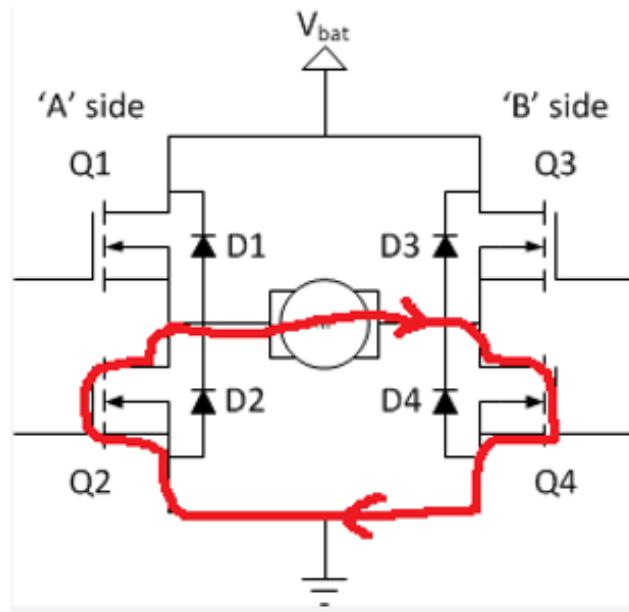
Hình 4. 22: Chiều dòng điện khi cặp công tắc Q1, Q4 cùng đóng, cặp công tắc Q2, Q3 cùng mở

+ Trạng thái 2: Công tắc Q4 đóng, các công tắc Q1, Q2, Q3 cùng mở. Dòng điện do tải điện sẽ tiếp tục chảy qua Q4, qua diode D2 và quay về tải điện.



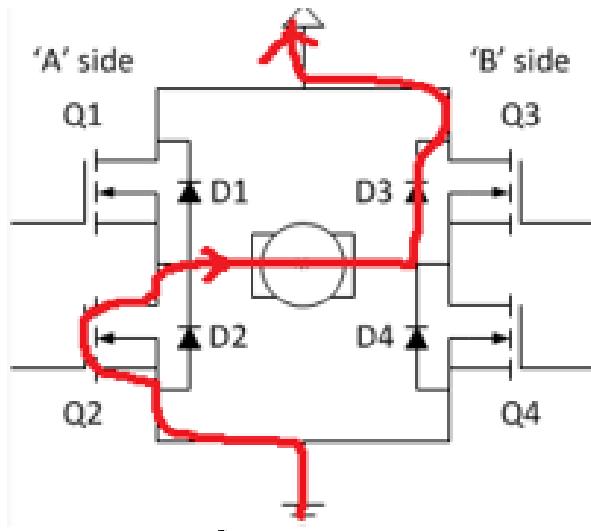
Hình 4. 23: Chiều dòng điện sau khi công tắc Q1 mở, công tắc Q4 vẫn đóng

+ Trạng thái 3: Công tắc Q2 và Q4 cùng mở, công tắc Q1 và Q3 cùng đóng. Lúc này, dòng điện từ tải điện sẽ đi qua công tắc Q4, qua công tắc Q2 và quay về tải điện.



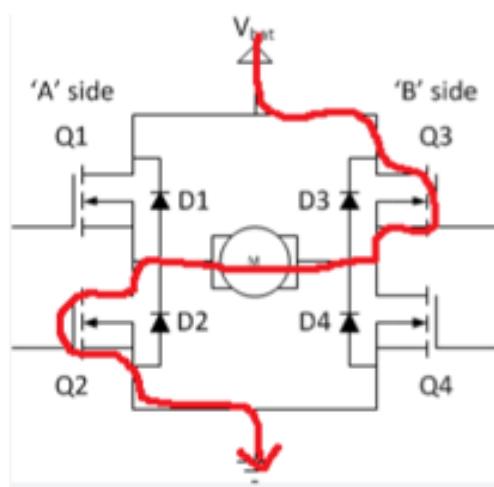
Hình 4. 24: Chiều dòng điện khi công tắc Q2, Q4 cùng đóng

- + Trạng thái 4: Công tắc Q2 đóng, các công tắc Q1, Q3, Q4 cùng mở. Dòng điện sẽ đi qua công tắc Q2, qua tải điện và qua diode D3.



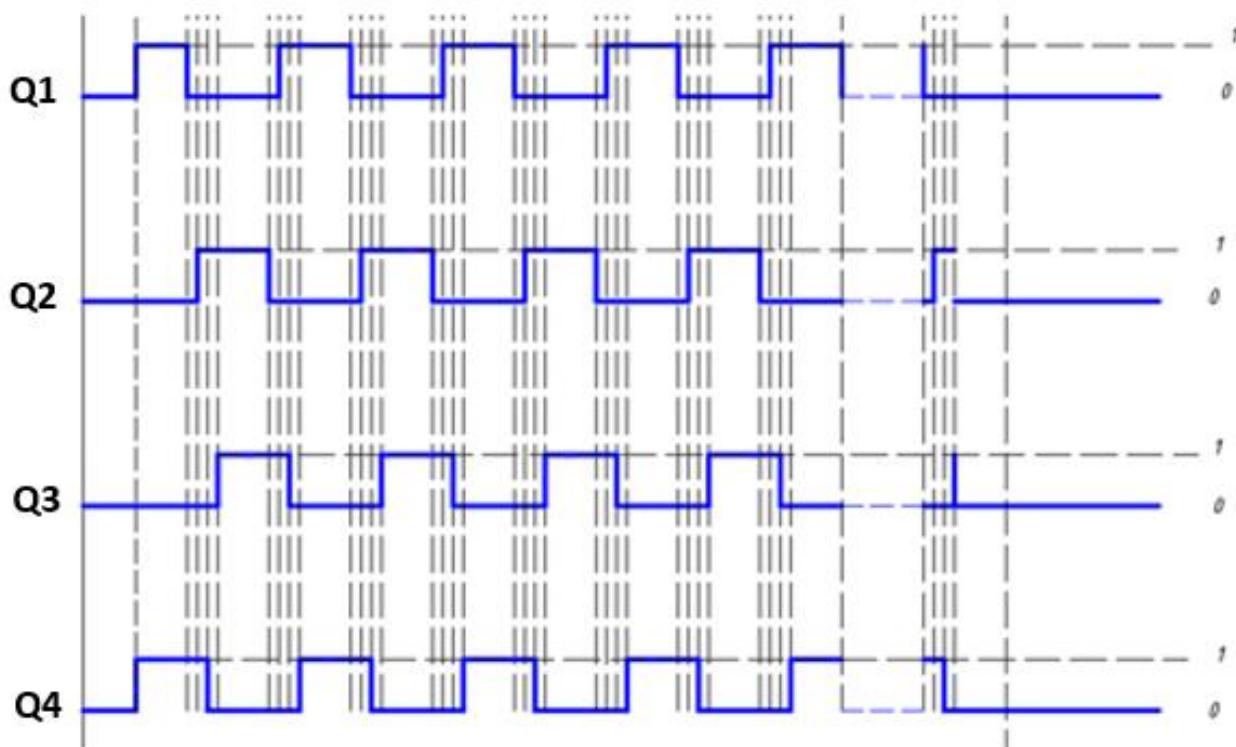
Hình 4. 25: Chiều dòng điện sau khi công tắc Q4 mở, công tắc Q2 vẫn đóng

- + Trạng thái 5: Công tắc Q2, Q3 cùng đóng, công tắc Q1, Q4 cùng mở. Khi ở trạng thái này, dòng điện từ nguồn cấp qua công tắc Q3, qua tải điện, qua công tắc Q2 và về điểm nối đất.



Hình 4. 26: Chiều dòng điện khi công tắc Q2, Q3 đóng, công tắc Q1, Q4 mở

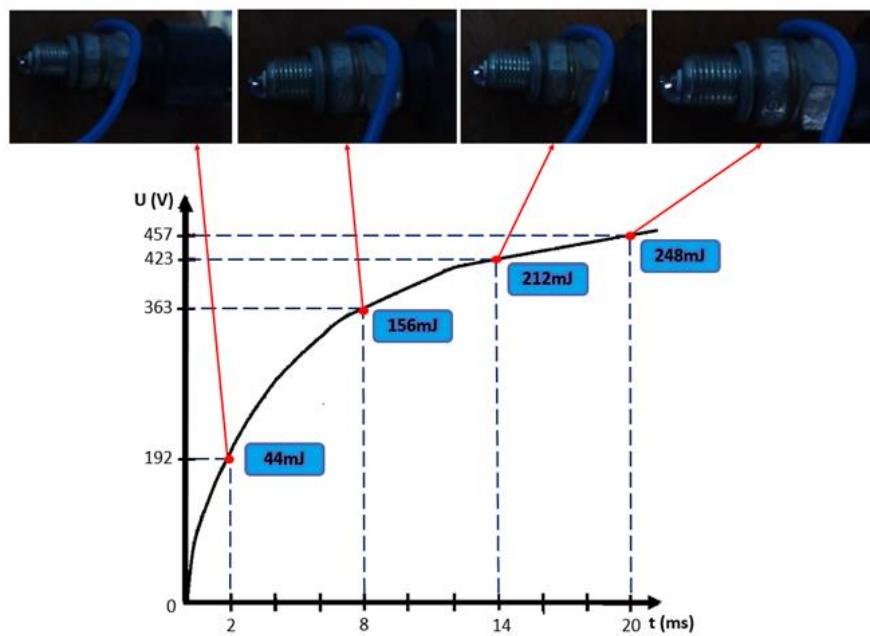
- + Các trạng thái còn lại lắp lại tương tự như trên.
- Bằng cách điều khiển mạch cầu H qua các trạng thái này làm không chuyển đổi cực tính của dòng điện ngay lực tức và thay vào đó là điều khiển qua nhiều giai đoạn, giúp làm phân tán năng lượng của dòng cảm ứng điện, giúp bảo vệ an toàn cho mạch.



Hình 4. 27: Tín hiệu điều khiển mạch cầu H

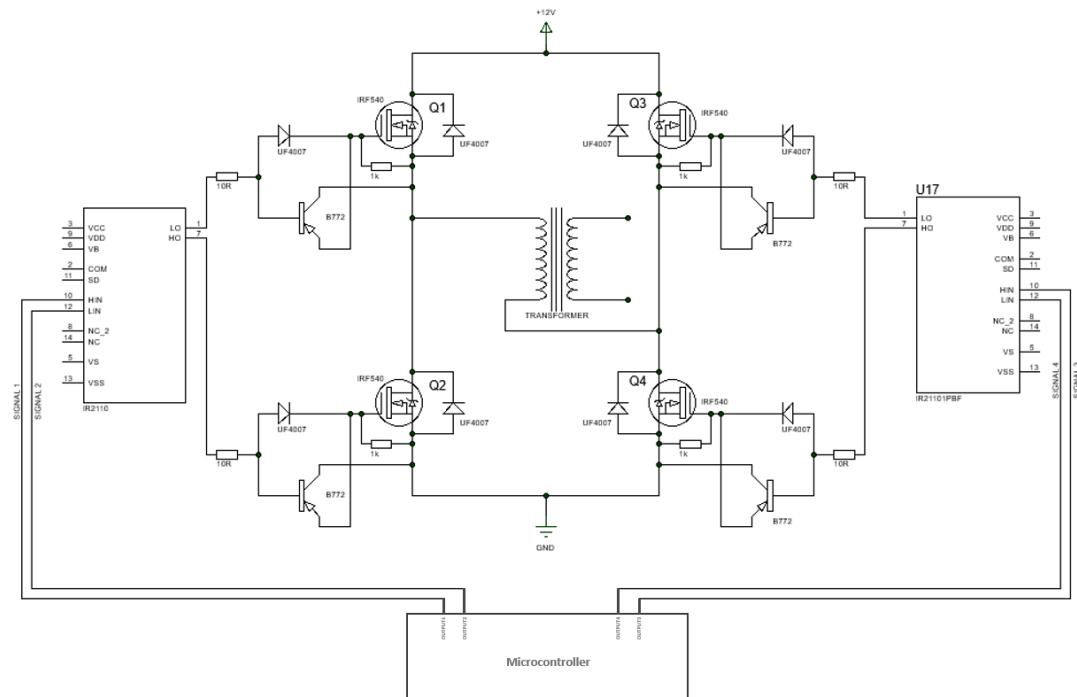


Hình 4. 28: Tín hiệu điều khiển mạch cầu H đo bằng dao động kỹ



Hình 4. 29: Đường đặc tính U - t

* Thiết kế mạch

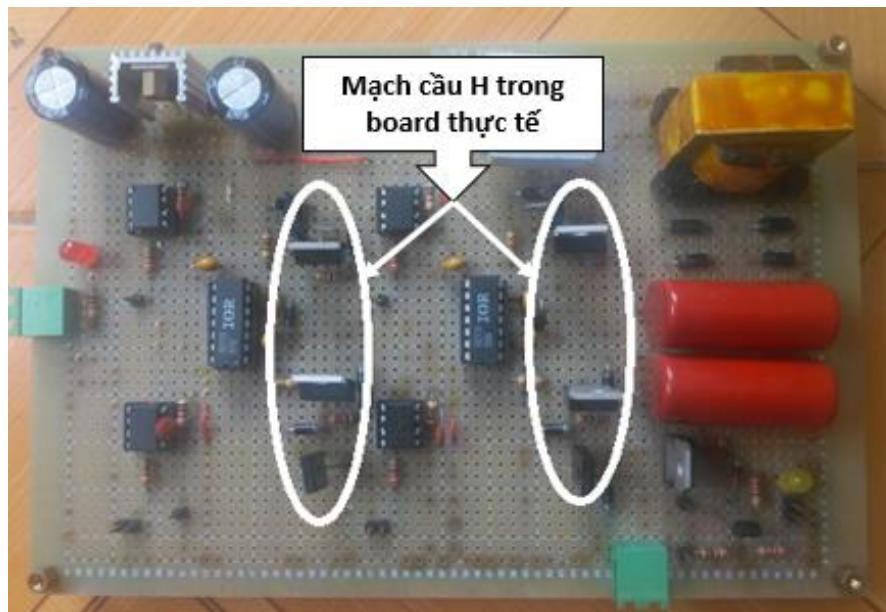


Hình 4. 30: Sơ đồ mạch cầu H của mạch thực tế

- Trong mạch cầu H thực tế, sử dụng 4 linh kiện bán dẫn mosfet IRF540 làm 4 công tắc điều khiển. Mosfet IRF540 là một loại linh kiện hiệu suất cao, được sử dụng trong các ứng dụng điều khiển công suất cao, với các thông số như:

- + Điện áp drain-source tối đa (VDS): 100V
- + Dòng drain tối đa (ID): 33A
- + Công suất tiêu thụ tối đa (PD): 150W
- + Điện trở chuyển động tuyệt đối (RDS(on)): 0.077 ohm
- + Điện trở cách điện (VGS(TH)): 2V - 4V
- + Nhiệt độ hoạt động tối đa (TJ): 175 độ C
- + Dung lượng đầu vào (Ciss): 1100pF

- + Dung lượng đầu ra (Coss): 180pF
- + Dung lượng chuyển đổi (Crss): 90pF.
- Và các thông số này đáp ứng được điều kiện hoạt động của mạch.
- Để đảm bảo mạch cầu H hoạt động ổn định, còn có thêm điện trở được mắc giữa cổng Gate và cổng Source của mosfet. Điện trở này được gọi là điện trở pull-down, việc sử dụng điện trở pull-down sẽ giúp đảm bảo cổng gate sẽ có điện áp ổn định khi không có tín hiệu điều khiển được áp lên đó. Và nếu khi không có điện trở này, nếu ở trường hợp không có tín hiệu điều khiển, điện áp trên cổng gate có thể bị chênh lệch và làm cho mosfet bị kích hoạt ngẫu nhiên, gây ra sự cố không mong muốn.
- Ngoài ra, còn trang bị thêm linh kiện transistor PNP tại mỗi mosfet như một đường xả trên MOSFET để giúp MOSFET chuyển trạng thái nhanh hơn khi không có tín hiệu cấp vào chân Gate.



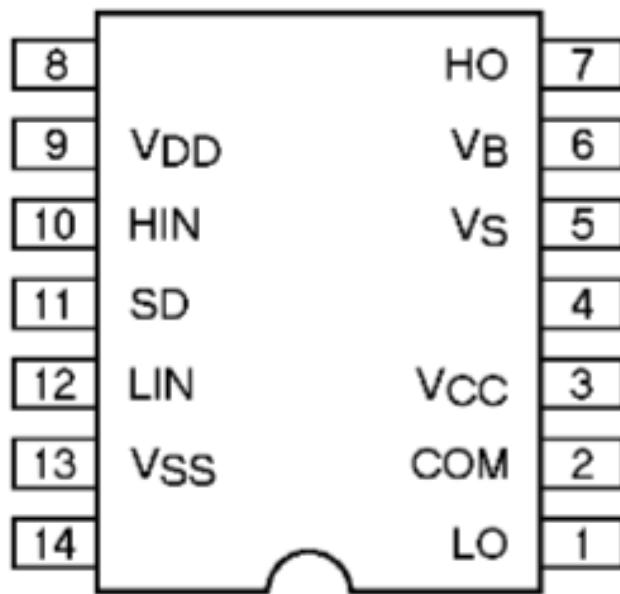
Hình 4. 31: Mạch cầu H trong board mạch thực tế

4.4.3.2. IC điều khiển mosfet



- Do mạch cầu H sử dụng các mosfet làm chức năng như công tắc, mà trong thực tế, không cung cấp tín hiệu điều khiển trực tiếp vào mosfet do tín hiệu đầu ra của vi điều khiển thường có điện áp và dòng điện rất nhỏ, không đủ để kích hoạt mosfet, và điều này có thể dẫn đến hiện tượng mosfet bị mở không đầy đủ, sẽ gây ra hư hỏng.
- Ngoài ra, tín hiệu từ vi điều khiển có thể bị nhiễu, gây ra các lỗi và có thể làm giảm độ ổn định của mạch. Vì vậy, để đảm bảo độ ổn định và hiệu suất, nên sử dụng IC driver để khuếch đại tín hiệu từ vi điều khiển và cấp cho mosfet.
- Những ưu điểm khi sử dụng IC driver để điều khiển mosfet:
 - + IC driver có khả năng tăng cường tín hiệu điều khiển, giúp đảm bảo rằng mosfet được kích hoạt và xả nhanh hơn.
 - + Có khả năng xử lý được các dòng điện và điện áp lớn, giúp đảm bảo rằng mosfet hoạt động ổn định.
 - + Ngoài ra, IC driver cũng có chức năng bảo vệ cho mosfet như bảo vệ quá dòng, quá nhiệt độ và quá áp, điều này giúp cho mosfet hoạt động ở mức độ an toàn và bảo vệ nó khỏi các hư hỏng do quá tải.
 - + Tương thích với nhiều loại mosfet khác nhau, giúp tăng tính linh hoạt khi sử dụng.
- Và trong mạch thực tế, dùng IR2110 để điều khiển mosfet, đây là một IC điều khiển MOSFET/IGBT đôi (high-low side driver), được thiết kế để điều khiển hai MOSFET/IGBT hoặc một cặp MOSFET/IGBT nối tiếp nhau.
- Các thông số của IR2110 gồm:
 - + Điện áp cung cấp: từ 10V đến 20V
 - + Dòng điện cung cấp: 2A

- + Điện áp tín hiệu đầu vào tối thiểu: 2V
 - + Điện áp tín hiệu đầu vào tối đa: 20V
 - + Tần số chuyển đổi tối đa: 500kHz
 - + Khả năng chịu điện áp đầu ra cao: đến 600V
 - + Nhiệt độ hoạt động tối đa: -55 độ C đến 150 độ C
- IR2110 bao gồm các tính năng bảo vệ quan trọng, bao gồm bảo vệ ngắn mạch, bảo vệ quá dòng, bảo vệ quá nhiệt và bảo vệ quá áp.
- IR2110 được tích hợp nhiều tính năng như bộ phân cách tín hiệu đầu vào, đảo pha, tạo xung PWM, và các bảo vệ quan trọng, cho phép nó dễ dàng tích hợp vào hệ thống điều khiển MOSFET/IGBT.



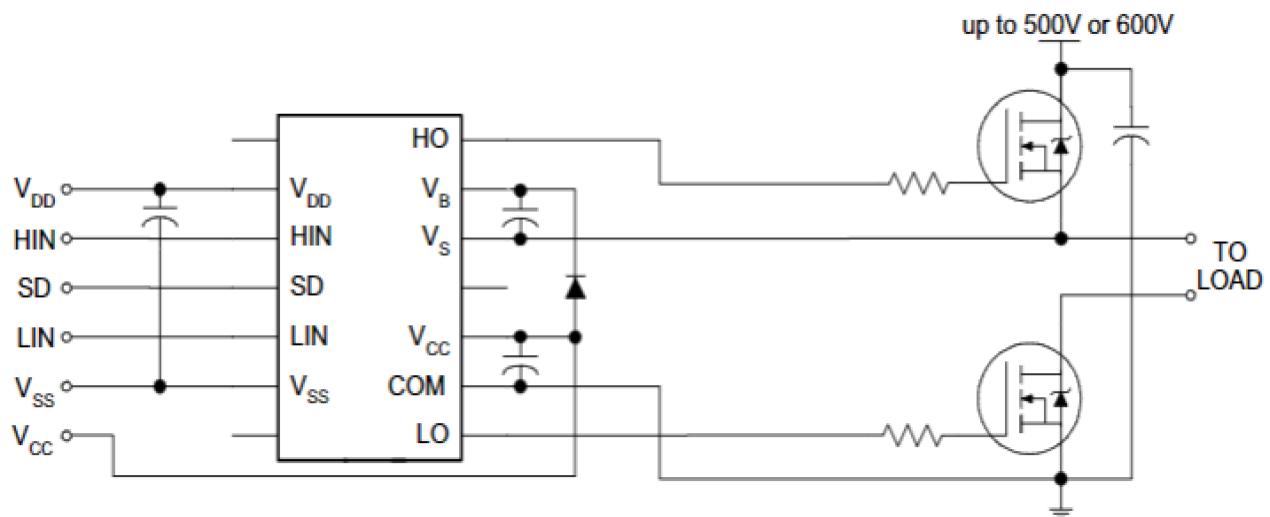
Hình 4. 32: Sơ đồ chân của IR2110

-- Chức năng của từng chân trong IR2110:

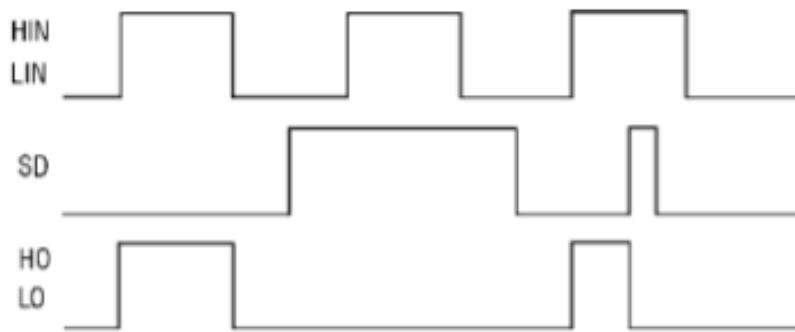


- + Chân 1 (LO): Công đầu ra bên thấp. Kết nối với cổng gate của mosfet.
 - + Chân 2 (COM): Thường được kết nối với nguồn điện âm
 - + Chân 3 (VCC): Chân cấp nguồn cho bên thấp
 - + Chân 5 (VS): Đường dẫn nguồn cấp bên cao ở trạng thái nối
 - + Chân 6 (VB): Cung cấp điện áp nối phía cao.
 - + Chân 7 (HO): Công đầu ra bên cao. Kết nối với cổng gate của mosfet.
 - + Chân 9 (VDD): Nguồn điện logic
 - + Chân 10 (HIN): Đầu vào logic cho công đầu ra bên cao (HO), cùng pha
 - + Chân 11 (SD): Đầu vào logic để shutdown
 - + Chân 12 (LIN): Đầu vào logic cho công đầu ra bên thấp (LO), cùng pha
 - + Chân 13 (VSS): Chân nối đất.
- Cách hoạt động của IR2110:
- + Chân LO là chân đầu ra kết nối với mosfet phía thấp
 - + Chân COM sẽ là một đường dẫn lại cho phía thấp. Nó được nối chung với chân VSS nối đất. Vì khi chân LIN ở mức cao, đầu ra chân LO sẽ bằng giá trị của điện áp tại chân VCC đối với chân VSS và chân COM. Còn khi đầu vào chân LIN ở mức thấp, đầu ra LO sẽ bằng giá trị của VSS là bằng không.
 - + Chân VDD là chân cung cấp điện áp logic. Giá trị đầu vào phải nằm trong khoảng 5V.
 - + Chân HIN là chân tín hiệu đầu vào để điều khiển mosfet bên cao, có thể từ bộ vi điều khiển và mức logic tín hiệu đầu vào nằm trong khoảng 4-5V.

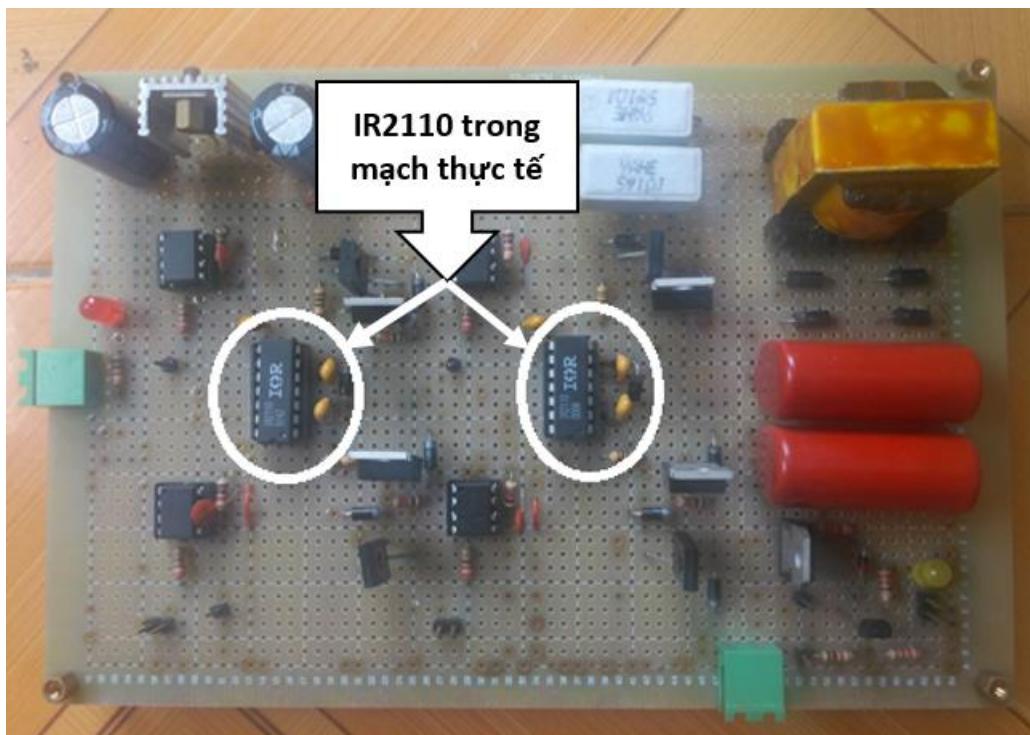
- + Chân LIN là chân tín hiệu đầu vào để điều khiển Mosfet bên thấp, có thể từ bộ vi điều khiển và mức logic tín hiệu đầu vào nằm trong khoảng 4-5V.
- + Chân SD được sử dụng làm chân shutdown, hoặc có thể sử dụng nó cho mạch bảo vệ.
- + Chân VB được sử dụng như một nguồn cung cấp nối phía cao để cung cấp điện áp nối cho MOSFET phía cao.
- + Tụ bootstrap được sử dụng giữa VB và VS để vận hành đầy đủ MOSFET phía cao. Nó đóng một vai trò rất quan trọng.



Hình 4. 33: Minh họa cách kết nối của IR2110



Hình 4. 34: Sơ đồ thời gian các tín hiệu đầu vào/đầu ra



Hình 4. 35: IR2110 trong mạch thực tế

4.4.4. Mạch CDI - AC

- Là mạch điện tử sử dụng để tạo ra điện áp cao để kích hoạt tia lửa trong động cơ đốt trong, mạch sử dụng nguyên lý xả điện từ bên trong tụ điện để tạo ra một điện áp cao trong khoảng thời gian ngắn.



- Trong mạch CDI, năng lượng cung cấp cho bobine được tích lũy trong tụ theo công thức:

$$W_C = \frac{C \times U^2}{2}$$

Với C: điện dung của tụ điện (Farad)

U: điện áp trên tụ điện

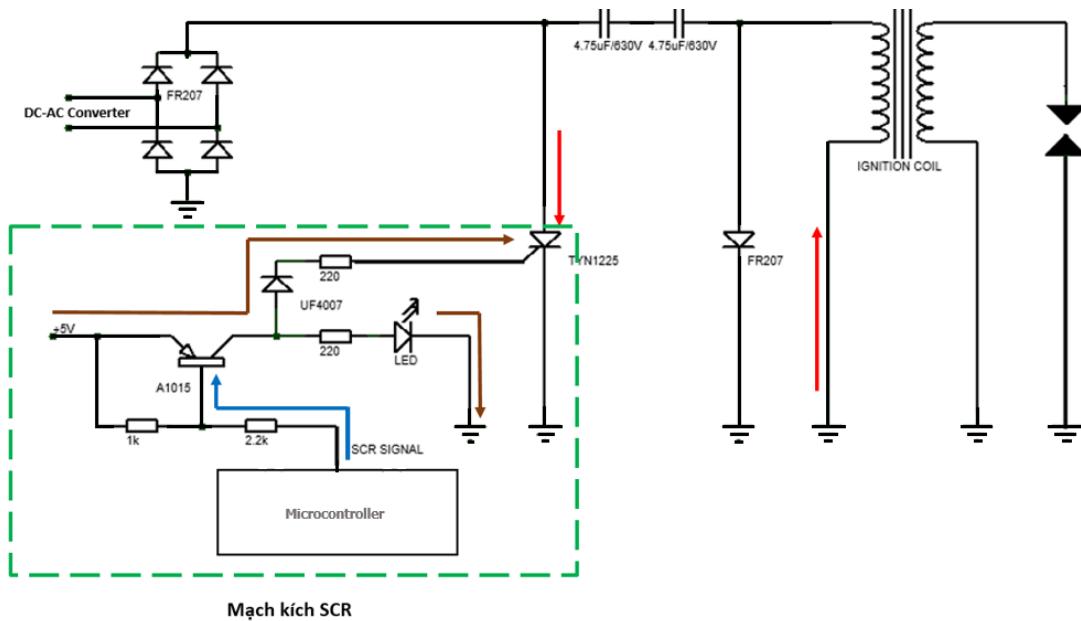
- Và trong mạch CDI, giá trị điện dung C của tụ điện rất quan trọng. Giá trị điện dung của tụ ảnh hưởng trực tiếp đến thời gian sạc và thời gian xả của tụ. Nếu giá trị điện dung quá nhỏ, tụ sẽ sạc và xả rất nhanh, không đủ thời gian tích tụ đủ năng lượng để đánh lửa. Còn nếu giá trị điện dung quá lớn, tụ sẽ sạc và xả chậm, làm giảm hiệu suất đánh lửa. Do đó, cần chọn giá trị điện dung tụ phù hợp để đảm bảo hiệu suất. Và trong mạch CDI, giá trị điện dung của tụ thường nằm trong khoảng $0,5\mu F - 3\mu F$.

- Với điện áp đầu ra nạp vào tụ từ mạch DC-AC converter có giá trị lớn 500V, và nhằm bảo vệ an toàn cho tụ cũng như hoạt động của mạch, do đó lựa chọn 2 tụ có giá trị $630V/4.75\mu F$ mắc nối tiếp với nhau. Với 2 tụ mắc nối tiếp với nhau, giá trị của tổng 2 tụ là:

$$U = U_1 + U_2 = 630 + 630 = 1260V$$

$$\frac{1}{C} = \frac{1}{C_1} + \frac{1}{C_2} = \frac{1}{4.75} + \frac{1}{4.75} \Rightarrow C = 2.375\mu F$$

- Vậy khi mắc 2 tụ nối tiếp thì sẽ có tổng điện áp chịu được tối đa là 1260V, giá trị điện dung là $2,375\mu F$.



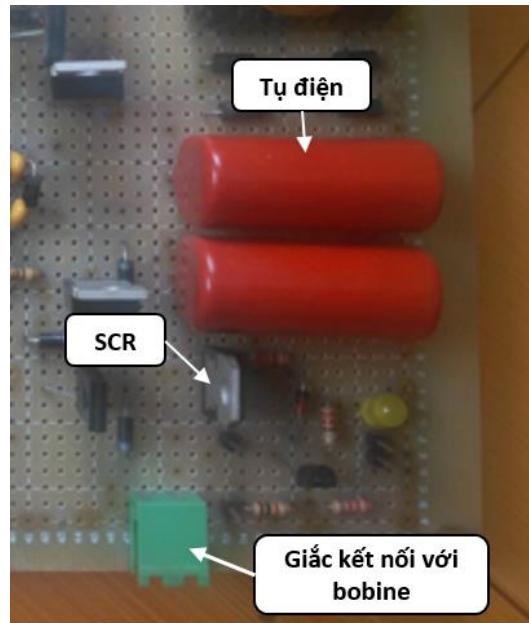
Hình 4. 36: Sơ đồ mạch đánh lửa CDI

- Nguyên lý hoạt động:

+ Ban đầu điện áp từ mạch DC-AC Converter qua diode chỉnh lưu được nạp tụ điện. Sau khi có tín hiệu đánh lửa từ vi điều khiển vào chân SIGNAL, dòng điện sẽ đi từ nguồn 5V, qua mạch kích transistor A1015 và vào chân Gate của SCR. Và khi SCR được dẫn, tụ điện sẽ xả điện qua SCR, qua điểm mass, tới cuộn sơ cấp của bobine đánh lửa và về tụ điện. Và do có sự biến thiên đột ngột về dòng điện trên cuộn sơ cấp nên có cảm ứng lên cuộn thứ cấp của bobine, sức điện động cao áp tạo ra tia lửa ở bugi.

+ Và để đảm bảo cho việc đánh lửa ổn định, có mắc một diode FR207 (fast diode) song song với cuộn dây sơ cấp của bobine đánh lửa, có tác dụng giúp dập tắt dòng xả ngược của tụ điện.

- SCR trong mạch CDI được sử dụng là linh kiện điện tử TYN1225, với khả năng điều khiển đầu vào cực thấp và được sử dụng chủ yếu trong ứng dụng đánh lửa. TYN1225 có thể chịu được dòng điện tối đa lên đến 25A và điện áp chịu đựng là 1200V. Ngoài ra, TYN1225 còn có khả năng chống nhiễu tốt và hoạt động ổn định ở nhiệt độ cao.

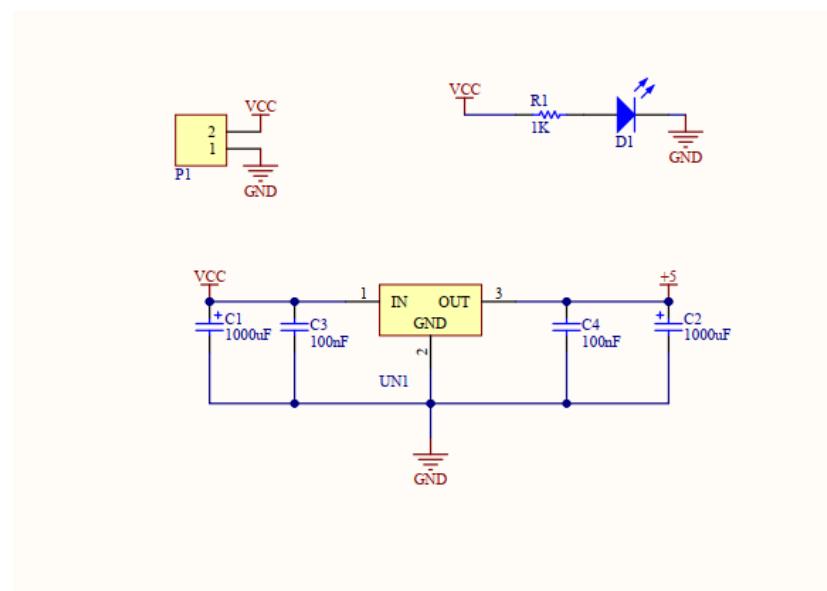


Hình 4. 37: Mạch CDI trên board mạch thực tế

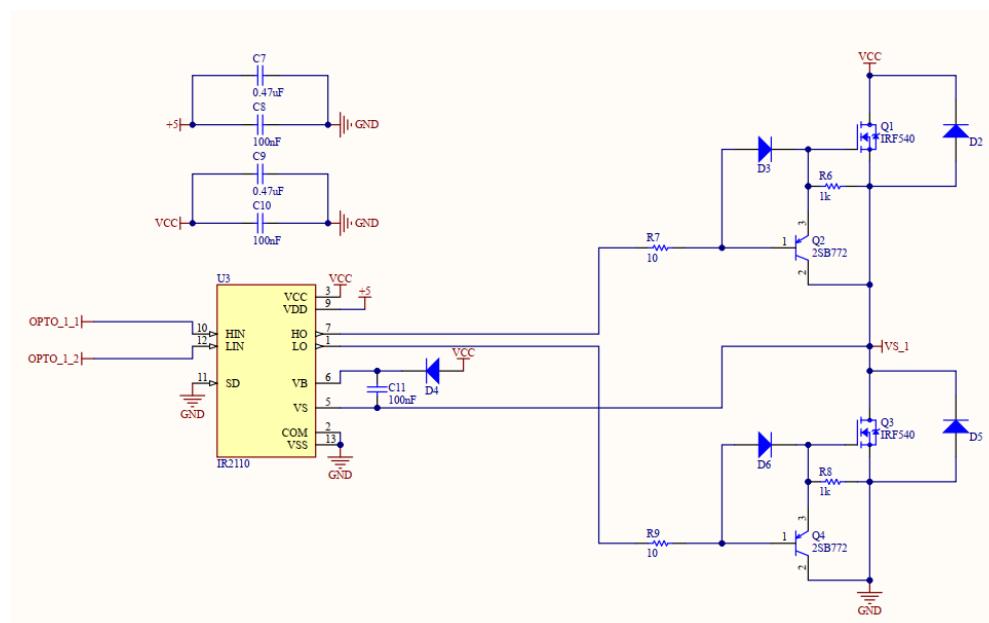
4.4.5. Mạch in PCB

- Sử dụng công cụ vẽ Altium vẽ mạch in PCB. Đây là một phần mềm thiết kế PCB chuyên nghiệp, cho phép tạo ra và thiết kế các mạch in PCB với đầy đủ các thành phần điện tử, đường dẫn, lóp, vùng, và lõi cảm. Có thể tạo và sắp xếp các thành phần theo ý muốn, tạo đường dẫn kết nối giữa chúng và xác định các vùng và lớp định nghĩa.
- Altium cung cấp công cụ vẽ sơ đồ mạch, tạo ra sơ đồ điện tử và kết nối chúng với các thành phần PCB tương ứng. Ngoài ra có thể vẽ các biểu đồ điện tử phức tạp, sử dụng ký hiệu và kỹ thuật mô phỏng linh hoạt.

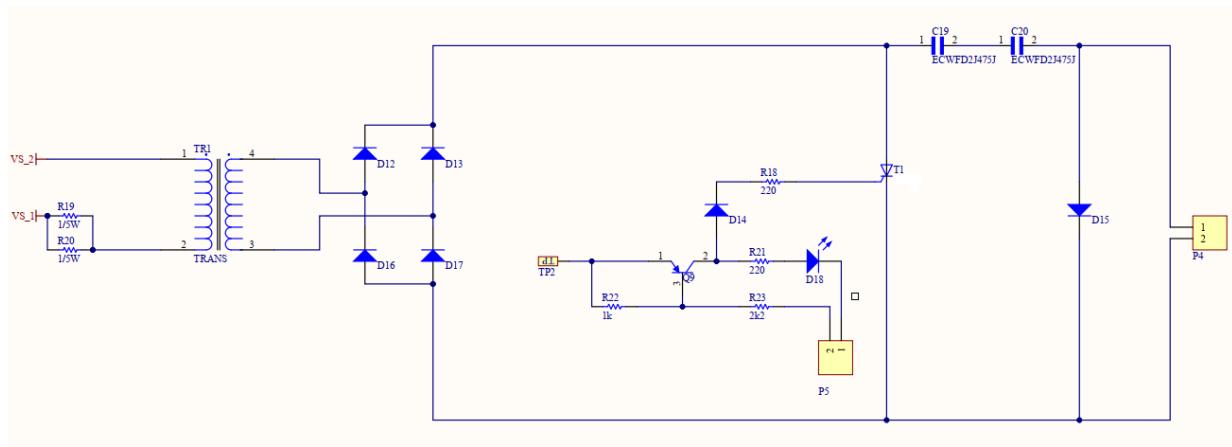
4.4.5.1. Sơ đồ nguyên lý (Schematic)



Hình 4. 38: Schematic của mạch nguồn ôn áp trên Altium

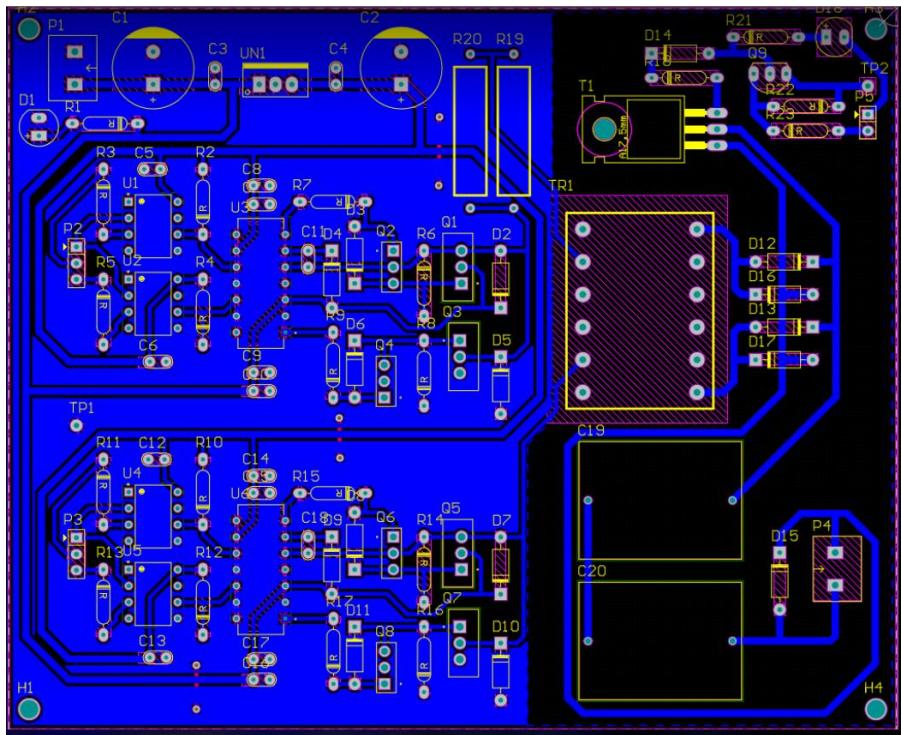


Hình 4. 39: Schematic của IC điều khiển MOSFET trên Altium

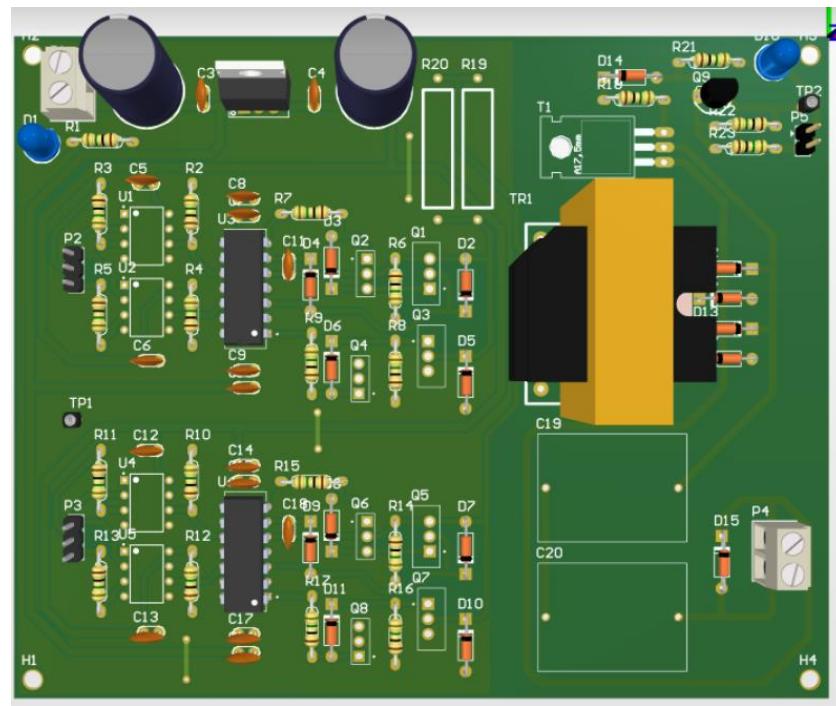


Hình 4. 40: Schematic của mạch CDI-AC trên Altium

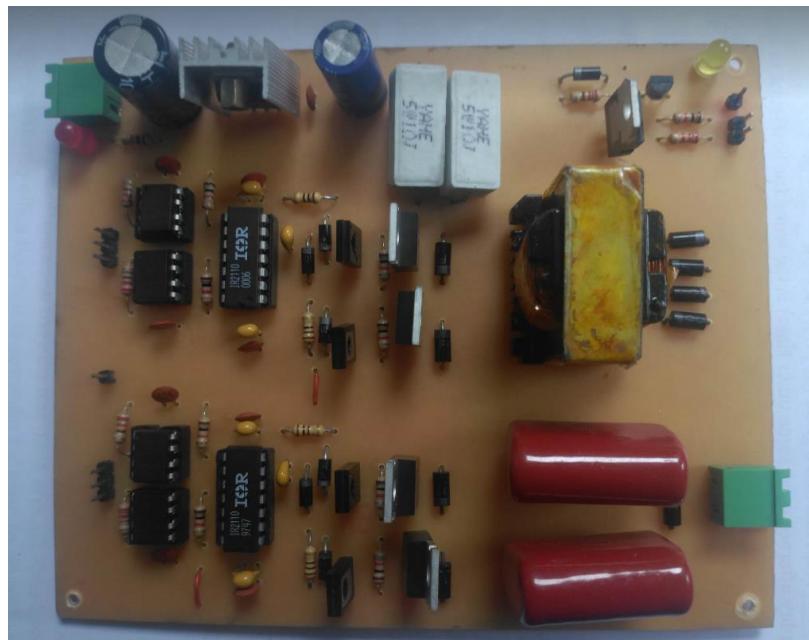
4.4.5.2. Sơ đồ mạch in



Hình 4. 41: Sơ đồ đi dây của mạch in PCB



Hình 4. 42: Hình 3d mạch in PCB



Hình 4. 43: Mạch in thực tế



CHƯƠNG V: THIẾT KẾ CÔNG NGHỆ

5.1. Phương án công nghệ

- Sử dụng testboard và các linh kiện điện tử có sẵn ngoài thị trường

- Chế tạo mạch in PCB, sử dụng các linh kiện chân cắm

=> Lựa chọn sử dụng cả 2 phương án:

+ Sử dụng testboard và các linh kiện điện tử có sẵn ngoài thị trường: giúp kiểm tra sự hoạt động của mạch

+ Chế tạo mạch in PCB, sử dụng các linh kiện chân cắm: giúp tăng tính thẩm mỹ và hiệu quả hoạt động.

5.2. Quy trình chế tạo

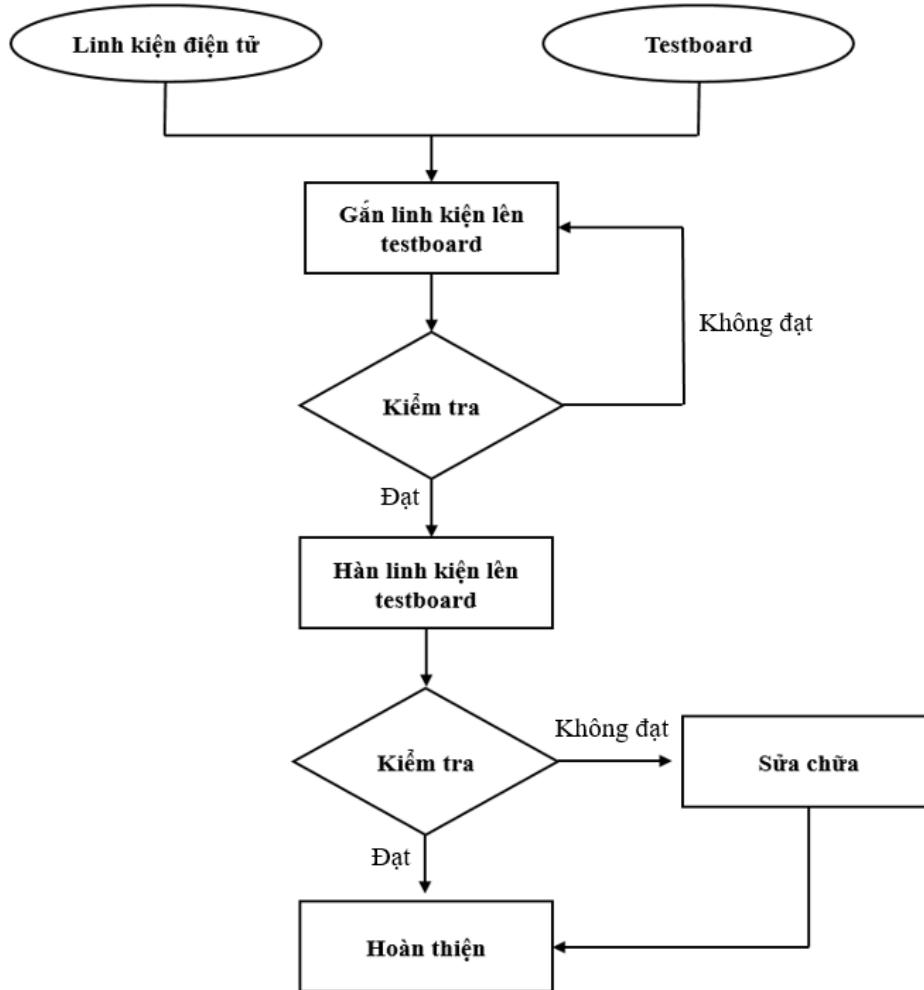
* Đối với testboard

- Dụng cụ, thiết bị:

+ Testboard, linh kiện điện tử có sẵn ngoài thị trường

+ Dụng cụ hàn như mỏ hàn, thiếc hàn, nhựa thông, ...

- Sơ đồ quy trình chế tạo



Hình 5. 1: Sơ đồ quy trình chế tạo testboard mạch thực tế

* Đối với mạch in PCB

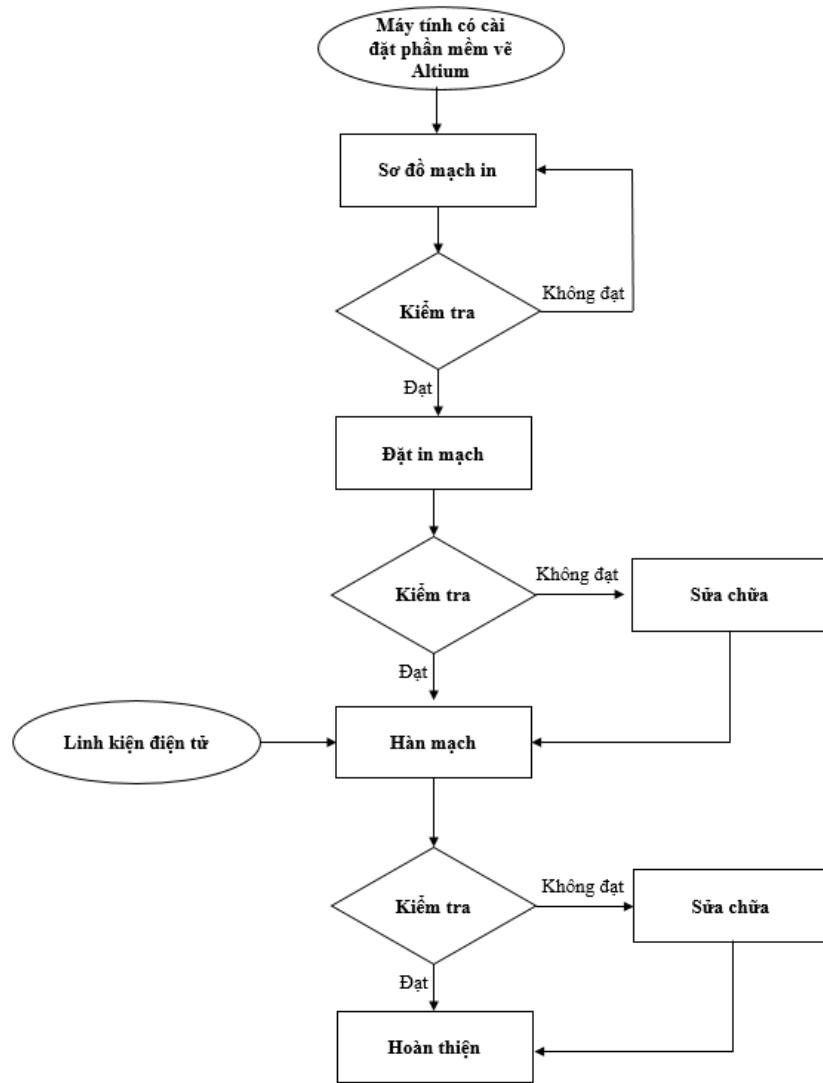
- Dụng cụ, thiết bị:

- + Máy tính có cài đặt phần mềm vẽ mạch in Altium

- + Dụng cụ hàn như mỏ hàn, thiếc hàn, nhựa thông, ...

- + Các linh kiện điện tử có sẵn ngoài thị trường

- Sơ đồ quy trình chế tạo



Hình 5. 2: Sơ đồ quy trình chế tạo mạch in PCB thực tế

5.3. Quy trình kiểm tra

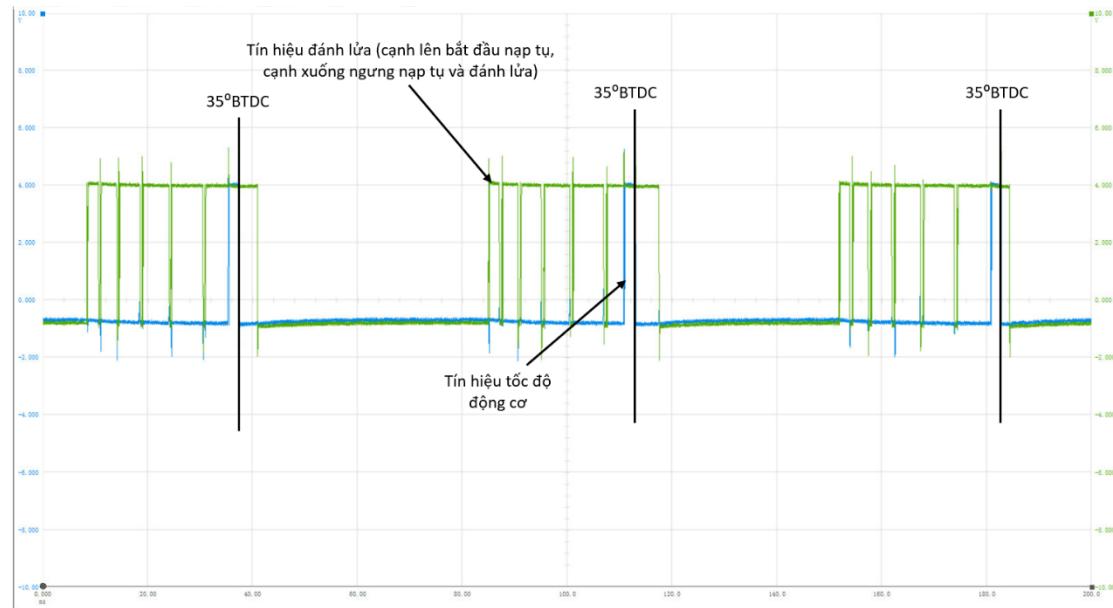
- Dụng cụ, thiết bị kiểm tra:

+ Dao động ký

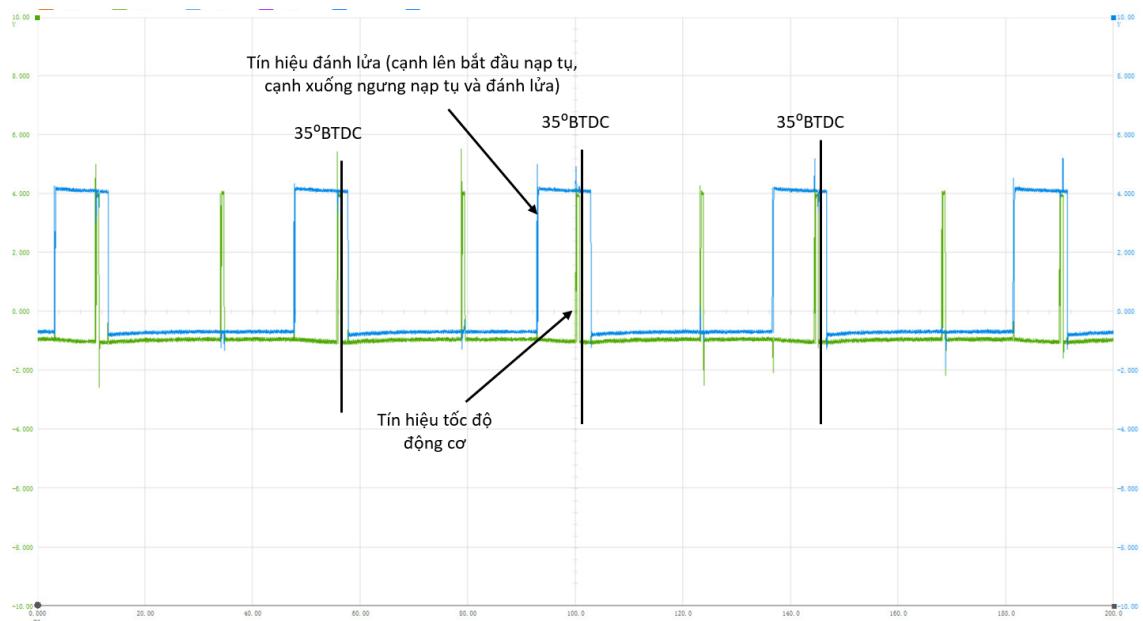
+ Đồng hồ VOM

CHƯƠNG VI: KẾT QUẢ ĐẠT ĐƯỢC

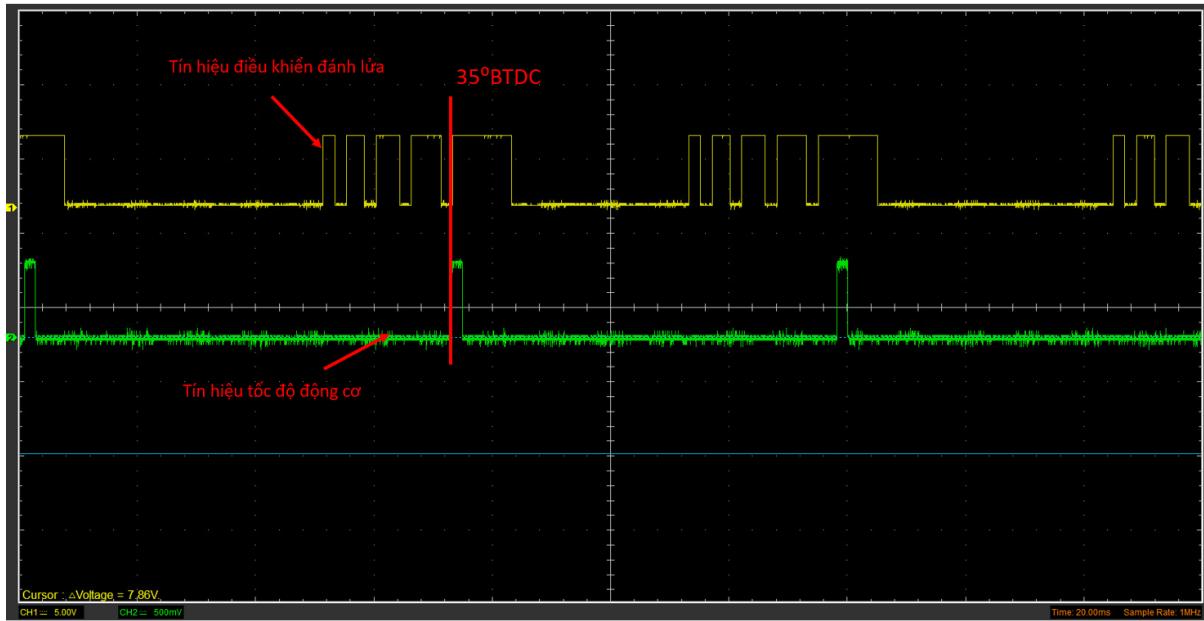
6. Kết quả sau khi thực hiện



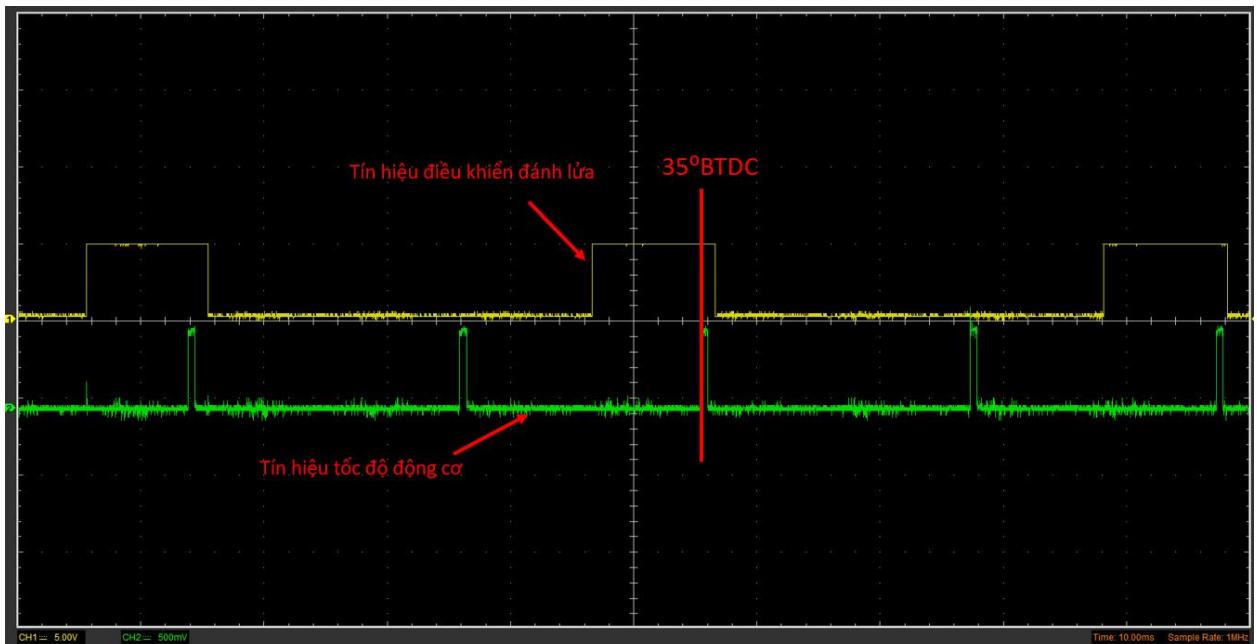
Hình 6. 1: Tín hiệu đánh lửa nhiều lần lúc khởi động



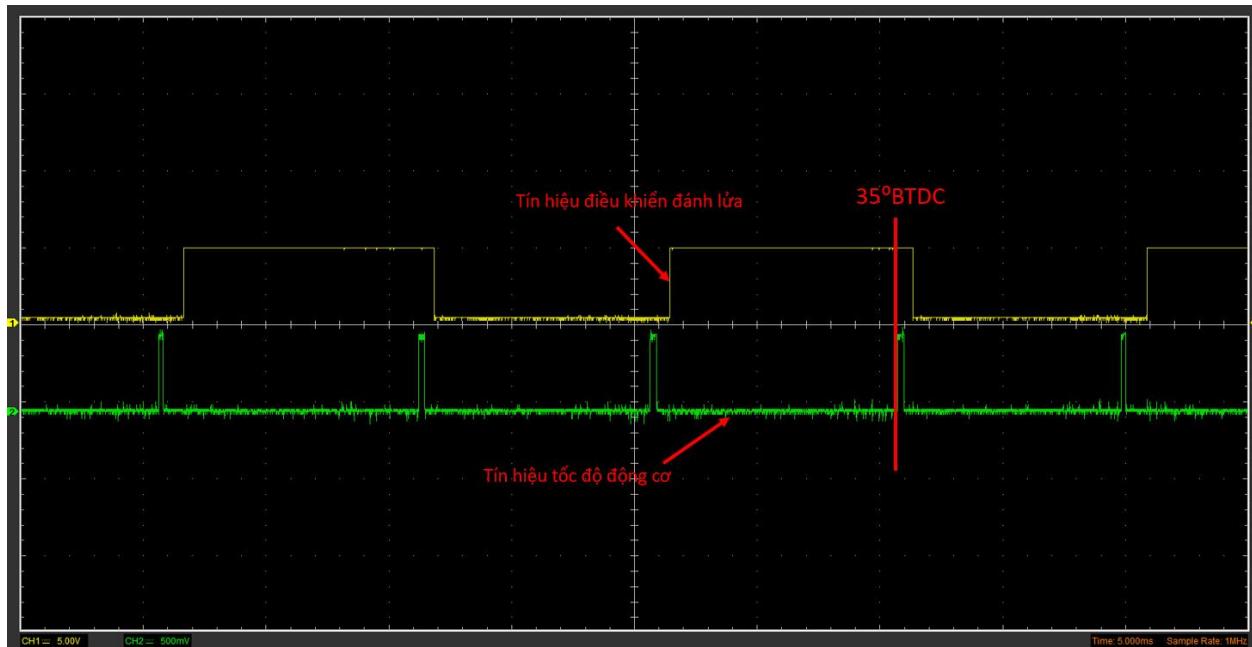
Hình 6. 2: Tín hiệu đánh lửa nhiều 1 lần 2 vòng quay trực khuỷu



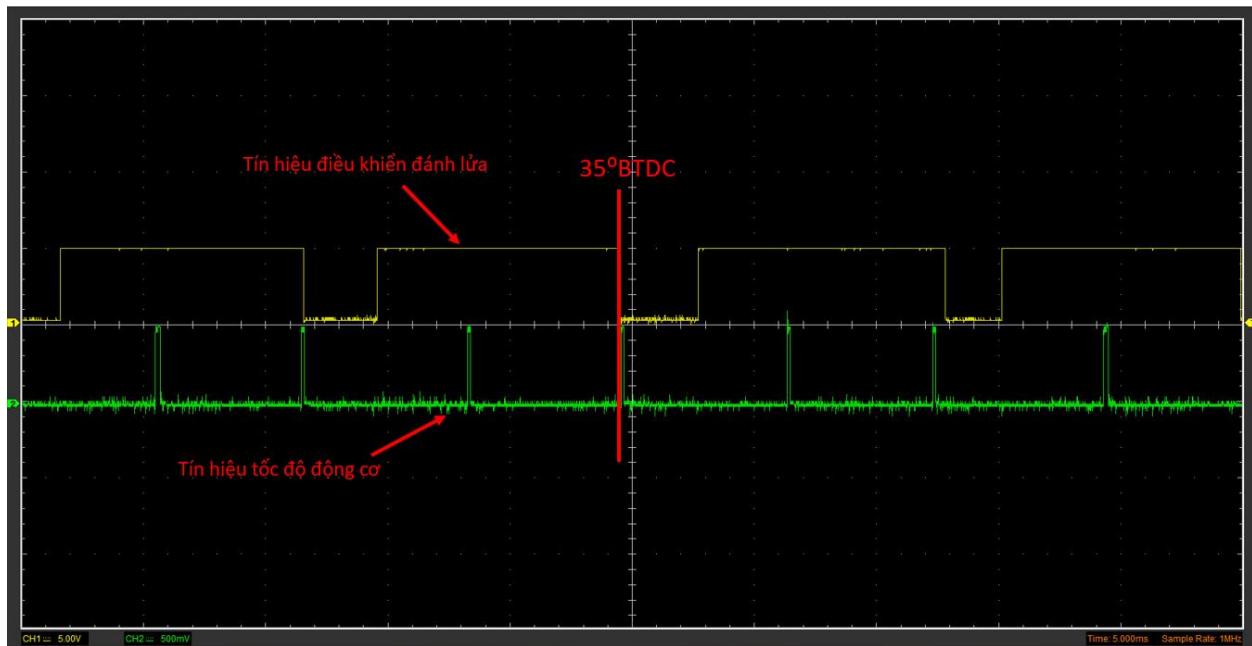
Hình 6. 3: Đánh lửa nhiều lần ở tốc độ động cơ 800 vòng/phút



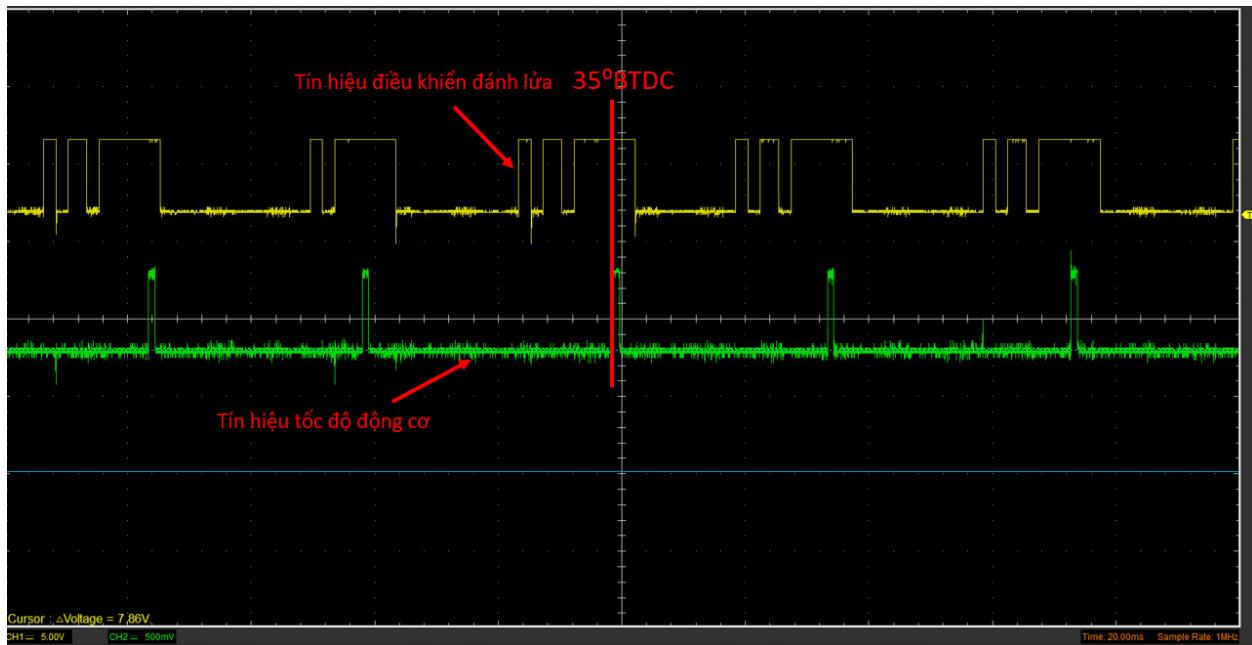
Hình 6. 4: Đánh lửa 1 lần/2 vòng quay tốc độ động cơ 3000 vòng/phút



Hình 6. 5: Đánh lửa 1 lần/2 vòng quay ở tốc độ động cơ 6500 vòng/phút



Hình 6. 6: Đánh lửa 1 lần/2 vòng quay ở tốc độ động cơ 10000 vòng/phút



Hình 6. 7: Đánh lửa nhiều lần ở tốc độ động cơ 1700 vòng/phút



Hình 6. 8. Hệ thống hoạt động thực tế



Hình 6. 9: Hình ảnh tia lửa thực tế



PHỤ LỤC

Code tham khảo điều khiển góc đánh lửa sớm (thay đổi góc đánh lửa theo tốc độ động cơ, đánh lửa 1 lần/ 2 vòng quay, đánh lửa nhiều lần lúc khởi động)

#include "stm8s.h"

#include "iostm8s105c6.h"

#include "stdbool.h"

//-----
-----//

uint32_t timer1_periodtime = 1999;

uint32_t timer2_periodtime;

uint32_t Temp_timer2_periodtime;

uint32_t timer3_periodtime;

uint32_t Temp_timer3_periodtime;

uint32_t T_ignition;

uint32_t T_charge;

uint32_t Temp_T_ignition;

uint32_t Temp_T_charge;

uint32_t multiple_ignition_compare_value;

uint32_t Temp_multiple_ignition_compare_value;

uint32_t Temp_multiple_ignition_compare_value1;

uint32_t multiple_ignition_T_charge[5] = {124, 187, 249, 312, 374};

uint32_t map_ignition[20] = {9, 11, 12, 14, 16, 17, 19, 21, 22, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35};

uint32_t count_time_15s = 0;

bool flag_UPDATE = 0;

bool flag_UPDATE_value = 0;



```
bool flag_UPDATE_timer2_timerperiod = 0;  
bool flag_UPDATE_timer3_timerperiod = 0;  
bool flag_count_overflow;  
bool ignition_exhaust_stroke_identification = 0;  
bool identification_period = 1;  
bool multiple_ignition = 0;  
bool multiple_ignition_status = 0;  
bool multiple_ignition_last_time = 0;  
bool hold_multiple_ignition = 1;  
bool redundancy_timer2 = 0;  
  
int count = 0;  
int count_identification_period = 20;  
int multiple_ignition_count_time = 0;  
int multiple_ignition_case = 0;  
int Temp_multiple_ignition_case = 0;  
  
uint32_t count_overflow = 0;  
uint32_t temp_count_overflow = 0;  
uint32_t T_end = 0;  
uint32_t T_start = 0;  
uint32_t T_oneround = 0;  
uint32_t T_oneround_previous = 0;  
uint32_t T_oneround_compared = 0;
```



```
//-----
-----//  
  
void HSI_16MHz_config(void);  
void GPIO_config(void);  
void TIMER1_setup(void);  
void TIMER2_setup(void);  
void TIMER3_setup(void);  
void TIMER4_setup(void);  
void ADC2_setup(void);  
uint16_t Read_ADC(int num_of_channel);  
float map(float number1, float number2, float number3, float number4, float number5);  
uint16_t abs(float num);  
//-----  
-----//  
  
int main(void)  
{  
    HSI_16MHz_config();  
    GPIO_config();  
    ADC2_setup();  
    TIMER1_setup();  
    TIMER2_setup();  
    TIMER3_setup();  
    TIMER4_setup();  
    while (1)  
    {  
    }  
}
```



```
}

// Timer1 - Input capture interrupt
//-----
-----//  
  
INTERRUPT_HANDLER(TIM1_CAP_COM_IRQHandler, 12)  
  
{  
    if (TIM1_SR1_CC4IF == 1)  
    {  
        T_end = (uint16_t)((TIM1_CCR4H << 8) | (uint16_t)(TIM1_CCR4L));  
        T_oneround = T_end - T_start + 2500 * count_overflow;  
        count_overflow = 0;  
        T_start = T_end;  
        if (T_oneround >= 480000)  
        {  
            if (hold_multiple_ignition == 1)  
            {  
                multiple_ignition = 1; // Enable the multiple ignition flag  
            }  
            else  
            {  
                multiple_ignition = 0;  
            }  
        }  
        else  
        {  
    }
```



```
multiple_ignition = 0;  
}  
-----//  
if (T_oneround >= 1920000) // RPM <=500  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[0]) / (float)360  
    / (float)256);  
}  
else if (T_oneround < 1920000 && T_oneround >= 1280000) // 500< RPM <= 750  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[1]) / (float)360  
    / (float)256);  
}  
else if (T_oneround < 1280000 && T_oneround >= 9600000) // 750<RPM<=1000  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[2]) / (float)360  
    / (float)256);  
}  
else if (T_oneround < 960000 && T_oneround >= 768000) // 1000<RPM<=1250  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[3]) / (float)360  
    / (float)256);  
}  
else if (T_oneround < 768000 && T_oneround >= 640000) // 1250<RPM<=1500  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[4]) / (float)360  
    / (float)256);
```



```
}

else if (T_oneround < 640000 && T_oneround >= 548571) // 1500<RPM<=1750
{
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[5]) / (float)360
    / (float)256);

}

else if (T_oneround < 548571 && T_oneround >= 480000) // 1750<RPM<=2000
{
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[6]) / (float)360
    / (float)256);

}

else if (T_oneround < 480000 && T_oneround >= 384000) // 2000<RPM<=2500
{
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[7]) / (float)360
    / (float)256);

}

else if (T_oneround < 384000 && T_oneround >= 320000) // 2500<RPM<=3000
{
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[8]) / (float)360
    / (float)256);

}

else if (T_oneround < 320000 && T_oneround >= 274285) // 3000<RPM<=3500
{
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[9]) / (float)360
    / (float)256);

}

else if (T_oneround < 274285 && T_oneround >= 240000) // 3500<RPM<=4000
```



```
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[10]) /  
    (float)360) / (float)256);  
  
}  
  
else if (T_oneround < 240000 && T_oneround >= 213333) // 4000<RPM<=4500  
  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[11]) /  
    (float)360) / (float)256);  
  
}  
  
else if (T_oneround < 213333 && T_oneround >= 192000) // 4500<RRM<=5000  
  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[12]) /  
    (float)360) / (float)256);  
  
}  
  
else if (T_oneround < 192000 && T_oneround >= 174545) // 5000<RPM<=5500  
  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[13]) /  
    (float)360) / (float)256);  
  
}  
  
else if (T_oneround < 174545 && T_oneround >= 160000) // 5500<RPM<=6000  
  
{  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[14]) /  
    (float)360) / (float)256);  
  
}  
  
else if (T_oneround < 160000 && T_oneround >= 147692) // 6000<RPM<=6500  
  
{
```



```
T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[15]) /  
(float)360) / (float)256);  
  
}  
  
else if (T_oneround < 147692 && T_oneround >= 137142) // 6500<RPM<=7000  
  
{  
  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[16]) /  
(float)360) / (float)256);  
  
}  
  
else if (T_oneround < 137142 && T_oneround >= 128000) // 7000<RPM<=7500  
  
{  
  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[17]) /  
(float)360) / (float)256);  
  
}  
  
else if (T_oneround < 128000 && T_oneround >= 106666) // 7500<RPM<=9000  
  
{  
  
    T_ignition = (uint32_t)((float)T_oneround * (float)(map_ignition[18]) /  
(float)360) / (float)256);  
  
}  
  
else // RPM > 9000  
  
{  
  
    T_ignition = (uint32_t)((float)T_oneround_previous * (float)(map_ignition[19]) /  
(float)360) / (float)256);  
  
}  
  
// T_ignition = (uint32_t)(float)T_oneround_previous * (((float)map(Read_ADC(1),  
0, 1023, 0, 35) / (float)360) / (float)256);  
  
//-----//  
  
if (multiple_ignition == 1)
```



```
{  
    multiple_ignition_status = 1;  
  
    multiple_ignition_compare_value = (uint16_t)((float)T_oneround_previous * 145  
    / 360 + (float)T_oneround * (35) / 360) / 256 - T_ignition);  
  
    if (multiple_ignition_compare_value < 1000)  
    {  
        multiple_ignition_case = 1;  
  
        Temp_timer2_periodtime = (uint16_t)((float)T_oneround_previous +  
        (float)T_oneround * 25 / 360) / 256) - 625 - 156;  
  
        TIM2_ARRH = (Temp_timer2_periodtime >> 8);  
  
        TIM2_ARRL = Temp_timer2_periodtime;  
  
        TIM2_CR1_CEN = 1; // Enable TIMER2  
  
    }  
  
    else if (multiple_ignition_compare_value >= 1000 &&  
    multiple_ignition_compare_value < 1281)  
    {  
        multiple_ignition_case = 2;  
  
        Temp_timer2_periodtime = (uint16_t)((float)T_oneround_previous +  
        (float)T_oneround * 25 / 360) / 256) - 625 - 249 - 312;  
  
        TIM2_ARRH = (Temp_timer2_periodtime >> 8);  
  
        TIM2_ARRL = Temp_timer2_periodtime;  
  
        TIM2_CR1_CEN = 1; // Enable TIMER2  
  
    }  
  
    else if (multiple_ignition_compare_value >= 1281 &&  
    multiple_ignition_compare_value < 1625)  
    {  
        multiple_ignition_case = 3;  
    }  
}
```



```
Temp_timer2_periodtime = (uint16_t)((float)T_oneround_previous +
(float)T_oneround * 25 / 360) / 256) - 625 - 249 - 312 - 374;

TIM2_ARRH = (Temp_timer2_periodtime >> 8);

TIM2_ARRL = Temp_timer2_periodtime;

TIM2_CR1_CEN = 1; // Enable TIMER2

}

else if (multiple_ignition_compare_value >= 1625 &&
multiple_ignition_compare_value < 2031)

{

multiple_ignition_case = 4;

Temp_timer2_periodtime = (uint16_t)((float)T_oneround_previous +
(float)T_oneround * 25 / 360) / 256) - 625 - 249 - 312 - 374 - 437;

TIM2_ARRH = (Temp_timer2_periodtime >> 8);

TIM2_ARRL = Temp_timer2_periodtime;

TIM2_CR1_CEN = 1; // Enable TIMER2

}

else

{

multiple_ignition_case = 5;

Temp_timer2_periodtime = (uint16_t)((float)T_oneround_previous +
(float)T_oneround * 25 / 360) / 256) - 625 - 249 - 312 - 374 - 437 - 499;

TIM2_ARRH = (Temp_timer2_periodtime >> 8);

TIM2_ARRL = Temp_timer2_periodtime;

TIM2_CR1_CEN = 1; // Enable TIMER2

}

}
```



```
else if (multiple_ignition == 0)
{
    if (count_identification_period > 0)
    {
        if (T_ignition + (uint16_t)(map(Read_ADC(2), 0, 1023, 624, 937)) <=
T_oneround_previous)
        {
            T_charge = (uint16_t)(map(Read_ADC(2), 0, 1023, 124, 937));
        }
        else
        {
            T_charge = T_oneround_previous - T_ignition;
        }
        Temp_timer2_periodtime = (uint16_t)((T_oneround_previous * 395 / 360) / 256
- T_ignition - T_charge;
        TIM2_ARRH = (Temp_timer2_periodtime >> 8);
        TIM2_ARRL = Temp_timer2_periodtime;
        // Phase of identifying the revolution involving ignition and exhaust stroke
        testvalue19 = (uint32_t)(abs((float)T_oneround - (float)T_oneround_compared) /
(float)T_oneround * 100);
        if ((uint32_t)(abs((float)T_oneround - (float)T_oneround_compared) /
(float)T_oneround * 100) <= 3)
        {
            if (T_oneround < T_oneround_previous)
            {
                ignition_exhaust_stroke_identification = 0; // Next revolution invokes intake
and compress stroke
            }
        }
    }
}
```



```
    }  
  
    else  
  
    {  
  
        ignition_exhaust_stroke_identification = 1; // Next revolution involves ignition  
        and exhaust stroke  
  
    }  
  
    count_identification_period--;  
  
}  
  
else  
  
{  
  
    count_identification_period = 20;  
  
}  
  
}  
  
else  
  
{  
  
    if (ignition_exhaust_stroke_identification == 1)  
  
    {  
  
        if (T_ignition + (uint16_t)(map(Read_ADC(2), 0, 1023, 624, 937)) <=  
        (T_oneround_previous + T_oneround))  
  
        {  
  
            T_charge = (uint16_t)(map(Read_ADC(2), 0, 1023, 124, 937));  
  
        }  
  
    else  
  
    {  
  
        T_charge = T_oneround_previous + T_oneround - T_ignition;  
  
    }  
}
```



```
Temp_timer2_periodtime = (uint16_t)((T_oneround_previous + T_oneround *  
395 / 360) / 256) - T_ignition - T_charge;  
  
TIM2_ARRH = (Temp_timer2_periodtime >> 8);  
  
TIM2_ARRL = Temp_timer2_periodtime;  
  
TIM2_CR1_CEN = 1; // Enable TIMER3  
  
}  
  
ignition_exhaust_stroke_identification = !ignition_exhaust_stroke_identification;  
}  
  
}  
  
T_oneround_compared = T_oneround_previous;  
  
T_oneround_previous = T_oneround;  
  
TIM1_SR1_CC4IF = 0; // Clear the interrupt flag  
  
}  
  
}  
  
// Timer1 - Overflow interrupt  
  
-----  
-----//  
  
INTERRUPT_HANDLER(TIM1_UPD_OVF_TRG_BRK_IRQHandler, 11)  
  
{  
  
count_overflow++;  
  
TIM1_SR1 UIF = 0; // Clear the interrupt flag  
  
}  
  
// Timer2 - Oveflow Interrupt  
  
-----  
-----//  
  
INTERRUPT_HANDLER(TIM2_UPD_OVF_BRK_IRQHandler, 13)
```



```
{  
    if (multiple_ignition == 1)  
    {  
        PD_ODR_ODR0 = 1; // Set pin D0 at high level  
        TIM2_CR1_CEN = 0; // Disable TIMER2  
        TIM3_CR1_CEN = 1; // Enable TIMER3  
        Temp_multiple_ignition_compare_value = multiple_ignition_compare_value;  
        TIM3_ARRH = (multiple_ignition_T_charge[multiple_ignition_count_time] >> 8);  
        TIM3_ARRL = (multiple_ignition_T_charge[multiple_ignition_count_time]);  
        multiple_ignition_status = 0;  
        Temp_multiple_ignition_case = multiple_ignition_case;  
    }  
    else  
    {  
        TIM2_CR1_CEN = 0; // Disable TIMER2  
        PD_ODR_ODR0 = 1; // Set pin D0 at high level  
        Temp_T_charge = T_charge;  
        TIM3_ARRH = (Temp_T_charge >> 8);  
        TIM3_ARRL = Temp_T_charge;  
        TIM3_CR1_CEN = 1; // Enable TIMER3  
    }  
    TIM2_SR1_UIF = 0; // Clear the update interrupt flag of TIMER 2  
}  
// Timer 3 - Overflow Interrupt
```



```
//-----
-----//  
INTERRUPT_HANDLER(TIM3_UPD_OVF_TRG_IRQHandler, 15)  
{  
    if (multiple_ignition == 1)  
    {  
        switch (Temp_multiple_ignition_case)  
        {  
            case 1:  
                if (multiple_ignition_last_time == 1)  
                {  
                    PD_ODR_ODR0 = 0;  
                    TIM3_CR1_CEN = 0; // Disable TIMER3  
                    multiple_ignition_last_time = 0;  
                    multiple_ignition_count_time = 0;  
                }  
                else  
                {  
                    if (multiple_ignition_count_time < 1)  
                    {  
                        if (multiple_ignition_status == 1)  
                        {  
                            PD_ODR_ODR0 = 1; // Set pin PD0 at high level  
                            TIM3_ARRH = (multiple_ignition_T_charge[multiple_ignition_count_time]  
                            >> 8);  
                            TIM3_ARRL = multiple_ignition_T_charge[multiple_ignition_count_time];  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```



```
TIM3_CR1_CEN = 1; // Enable TIMER3
multiple_ignition_status = 0;
}
else
{
    multiple_ignition_count_time++;
    PD_ODR_ODR0 = 0; // Set pin PD0 at low level
    TIM3_ARRH = (124 >> 8);
    TIM3_ARRL = 124;
    TIM3_CR1_CEN = 1; // Enable TIMER3
    multiple_ignition_status = 1;
}
}
else
{
    PD_ODR_ODR0 = 1; // Set pin PD0 at high level
    multiple_ignition_last_time = 1;
    TIM3_ARRH = 624 >> 8;
    TIM3_ARRL = 624;
    TIM3_CR1_CEN = 1; // Enable TIMER3
}
}
break;
case 2:
if (multiple_ignition_last_time == 1)
```



```
{  
    PD_ODR_ODR0 = 0;  
  
    TIM3_CR1_CEN = 0; // Disable TIMER3  
  
    multiple_ignition_last_time = 0;  
  
    multiple_ignition_count_time = 0;  
}  
  
else  
  
{  
    if (multiple_ignition_count_time < 2)  
    {  
        if (multiple_ignition_status == 1)  
        {  
            PD_ODR_ODR0 = 1; // Set pin PD0 at high level  
  
            TIM3_ARRH = (multiple_ignition_T_charge[multiple_ignition_count_time]  
            >> 8);  
  
            TIM3_ARRL = multiple_ignition_T_charge[multiple_ignition_count_time];  
  
            TIM3_CR1_CEN = 1; // Enable TIMER3  
  
            multiple_ignition_status = 0;  
        }  
  
    else  
  
    {  
        multiple_ignition_count_time++;  
  
        PD_ODR_ODR0 = 0; // Set pin PD0 at low level  
  
        TIM3_ARRH = (124 >> 8);  
  
        TIM3_ARRL = 124;  
    }  
}
```



```
TIM3_CR1_CEN = 1; // Enable TIMER3
multiple_ignition_status = 1;
}
}
else
{
    PD_ODR_ODR0 = 1; // Set pin PD0 at high level
    multiple_ignition_last_time = 1;
    TIM3_ARRH = 624 >> 8;
    TIM3_ARRL = 624;
    TIM3_CR1_CEN = 1; // Enable TIMER3
}
}
break;
case 3:
if (multiple_ignition_last_time == 1)
{
    PD_ODR_ODR0 = 0;
    TIM3_CR1_CEN = 0; // Disable TIMER3
    multiple_ignition_last_time = 0;
    multiple_ignition_count_time = 0;
}
else
{
    if (multiple_ignition_count_time < 3)
```



```
{  
    if (multiple_ignition_status == 1)  
    {  
        PD_ODR_ODR0 = 1; // Set pin PD0 at high level  
        TIM3_ARRH = (multiple_ignition_T_charge[multiple_ignition_count_time]  
>> 8);  
        TIM3_ARRL = multiple_ignition_T_charge[multiple_ignition_count_time];  
        TIM3_CR1_CEN = 1; // Enable TIMER3  
        multiple_ignition_status = 0;  
    }  
    else  
    {  
        multiple_ignition_count_time++;  
        PD_ODR_ODR0 = 0; // Set pin PD0 at low level  
        TIM3_ARRH = (124 >> 8);  
        TIM3_ARRL = 124;  
        TIM3_CR1_CEN = 1; // Enable TIMER3  
        multiple_ignition_status = 1;  
    }  
}  
else  
{  
    PD_ODR_ODR0 = 1; // Set pin PD0 at high level  
    multiple_ignition_last_time = 1;  
    TIM3_ARRH = (624 >> 8);  
}
```



```
TIM3_ARRL = 624;  
TIM3_CR1_CEN = 1; // Enable TIMER3  
}  
}  
break;  
case 4:  
if (multiple_ignition_last_time == 1)  
{  
    PD_ODR_ODR0 = 0;  
    TIM3_CR1_CEN = 0; // Disable TIMER3  
    multiple_ignition_last_time = 0;  
    multiple_ignition_count_time = 0;  
}  
else  
{  
    if (multiple_ignition_count_time < 4)  
    {  
        if (multiple_ignition_status == 1)  
        {  
            PD_ODR_ODR0 = 1; // Set pin PD0 at high level  
            TIM3_ARRH = (multiple_ignition_T_charge[multiple_ignition_count_time]  
            >> 8);  
            TIM3_ARRL = multiple_ignition_T_charge[multiple_ignition_count_time];  
            TIM3_CR1_CEN = 1; // Enable TIMER3  
            multiple_ignition_status = 0;  
        }  
    }  
}
```



```
    }  
  
    else  
  
    {  
  
        multiple_ignition_count_time++;  
  
        PD_ODR_ODR0 = 0; // Set pin PD0 at low level  
  
        TIM3_ARRH = (124 >> 8);  
  
        TIM3_ARRL = 124;  
  
        TIM3_CR1_CEN = 1; // Enable TIMER3  
  
        multiple_ignition_status = 1;  
  
    }  
  
}  
  
else  
  
{  
  
    PD_ODR_ODR0 = 1; // Set pin PD0 at high level  
  
    multiple_ignition_last_time = 1;  
  
    TIM3_ARRH = (624 >> 8);  
  
    TIM3_ARRL = (624);  
  
    TIM3_CR1_CEN = 1; // Enable TIMER3  
  
}  
  
}  
  
break;  
  
case 5:  
  
if (multiple_ignition_last_time == 1)  
  
{  
  
    PD_ODR_ODR0 = 0;
```



```
TIM3_CR1_CEN = 0; // Disable TIMER3
multiple_ignition_last_time = 0;
multiple_ignition_count_time = 0;
}
else
{
if (multiple_ignition_count_time < 5)
{
if (multiple_ignition_status == 1)
{
PD_ODR_ODR0 = 1; // Set pin PD0 at high level
TIM3_ARRH = (multiple_ignition_T_charge[multiple_ignition_count_time]
>> 8);
TIM3_ARRL = multiple_ignition_T_charge[multiple_ignition_count_time];
TIM3_CR1_CEN = 1; // Enable TIMER3
multiple_ignition_status = 0;
}
else
{
multiple_ignition_count_time++;
PD_ODR_ODR0 = 0; // Set pin PD0 at low level
TIM3_ARRH = (124 >> 8);
TIM3_ARRL = 124;
TIM3_CR1_CEN = 1; // Enable TIMER3
multiple_ignition_status = 1;
}
```



```
    }  
}  
else  
{  
    PD_ODR_ODR0 = 1; // Set pin PD0 at high level  
    multiple_ignition_last_time = 1;  
    TIM3_ARRH = 624 >> 8;  
    TIM3_ARRL = 624;  
    TIM3_CR1_CEN = 1; // Enable TIMER3  
}  
}  
break;  
}  
}  
else  
{  
    PD_ODR_ODR0 = 0; // Set pin D0 at low level  
    TIM3_CR1_CEN = 0; // Disable TIMER3  
}  
}  
TIM3_SR1_UIF = 0; // Clear the update interrupt flag of TIMER 3  
}  
//-----  
-----//  
INTERRUPT_HANDLER(TIM4_UPD_OVF_IRQHandler, 23)  
{
```



```
// Enable flag_UPDATE after 10 ms
count++;
count_time_15s++;
if (count == 10)
{
    flag_UPDATE = 1;
    count = 0;
}
if (count_time_15s == 150000)
{
    hold_multiple_ignition = 0;
}
TIM4_SR UIF = 0; // Clear the update interrupt flag of TIMER4
}

//-----
-----//  

INTERRUPT_HANDLER(EXTI_PORTB_IRQHandler, 4)
{
}  

// Clock config
//-----
-----//  

void HSI_16MHz_config(void)
{
    CLK_DeInit();
    CLK_HSICmd(ENABLE);
```



```
CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);  
CLK_SYSCLKConfig(CLK_PRESCALER_CPUDIV1);  
CLK_ClockSwitchConfig(CLK_SWITCHMODE_AUTO, CLK_SOURCE_HSI,  
DISABLE, CLK_CURRENTCLOCKSTATE_DISABLE);  
  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_SPI, DISABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_I2C, DISABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_ADC, ENABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_AWU, DISABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_UART1, DISABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_UART3, DISABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER1, ENABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER2, ENABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER3, ENABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER4, ENABLE);  
CLK_PeripheralClockConfig(CLK_PERIPHERAL_CAN, DISABLE);  
}  
// GPIO config  
//-----  
-----//  
void GPIO_config(void)  
{  
    GPIO_DeInit(GPIOC);  
    GPIO_Init(GPIOC, GPIO_PIN_4, GPIO_MODE_IN_PU_NO_IT);  
    GPIO_DeInit(GPIOB);  
    GPIO_Init(GPIOB, GPIO_PIN_1, GPIO_MODE_IN_FL_NO_IT);
```



```
GPIO_Init(GPIOB, GPIO_PIN_2, GPIO_MODE_IN_FL_NO_IT);

GPIO_DeInit(GPIOD);
GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_HIGH_FAST);
GPIO_Init(GPIOD, GPIO_PIN_2, GPIO_MODE_OUT_PP_HIGH_FAST);
}

// Timer 1 set up
//-----
-----//  

void TIMER1_setup(void)
{
    TIM1_DeInit();
    TIM1_TimeBaseInit(0, TIM1_COUNTERMODE_UP, 2499, 0);
    TIM1_ICInit(TIM1_CHANNEL_4, TIM1_ICPOLARITY_FALLING,
    TIM1_ICSELECTION_DIRECTTI, TIM1_ICPSC_DIV1, 1);
    TIM1_ARRPreloadConfig(ENABLE);
    TIM1_EGR_UG = 1;
    TIM1_ITConfig(TIM1_IT_UPDATE, ENABLE);
    TIM1_ITConfig(TIM1_IT_CC4, ENABLE);
    TIM1_Cmd(ENABLE);
    enableInterrupts();
}
// Timer2 set up
//-----
-----//  

void TIMER2_setup(void)
```



```
{  
    TIM2_DeInit();  
  
    TIM2_TimeBaseInit(TIM2_PRESCALER_256, timer2_periodtime);  
  
    TIM2_ARRPreloadConfig(DISABLE);  
  
    TIM2_ITConfig(TIM2_IT_UPDATE, ENABLE);  
  
    enableInterrupts();  
}  
  
// Timer 3 set up  
//-----  
-----//  
  
void TIMER3_setup(void)  
{  
    TIM3_DeInit();  
  
    TIM3_TimeBaseInit(TIM3_PRESCALER_256, timer3_periodtime);  
  
    TIM3_ARRPreloadConfig(DISABLE);  
  
    TIM3_ITConfig(TIM3_IT_UPDATE, ENABLE);  
  
    enableInterrupts();  
}  
  
// Timer 4 set up  
//-----  
-----//  
  
void TIMER4_setup(void)  
{  
    TIM4_DeInit();  
  
    TIM4_TimeBaseInit(TIM4_PRESCALER_64, 249);  
  
    TIM4_ARRPreloadConfig(DISABLE);
```



```
TIM4_ITConfig(TIM4_IT_UPDATE, ENABLE);
TIM4_Cmd(ENABLE);
enableInterrupts();
}

// ADC2 set up
//-----
-----//  

void ADC2_setup(void)
{
    // Right alignment
    ADC_CR2_ALIGN = 1;
    // Enable ADC2 and start conversion
    ADC_CR1_ADON = 1;
}
// Read ADC
//-----
-----//  

uint16_t Read_ADC(int num_of_channel)
{
    uint16_t value_of_ADC_conversion;
    // Choose channel for conversion
    ADC_CSR_CH = num_of_channel;
    // Enable ADC1 and start conversion
    ADC_CR1_ADON = 1;
    // Wait for the conversion complete
```



```
while (ADC_CSR_EOC == 0)
{
    ;
    // Clear the conversion complete bit
    ADC_CSR_EOC = 0;
    // get the value of ADC conversion
    value_of_ADC_conversion = ADC_DRL;
    value_of_ADC_conversion |= (ADC_DRH << 8);

    return value_of_ADC_conversion;
}

// Map function
//-----
-----//
float map(float number1, float number2, float number3, float number4, float number5)
{
    return (number5 - ((number3 - number1) / (number3 - number2)) * (number5 -
    number4));
}

// Abs function
//-----
-----//
uint16_t Abs(float num)
{
    uint16_t result_1;
    if (num < 0)
    {
```



```
result_1 = (uint16_t)(0 - num);
}
else
{
    result_1 = (uint16_t)(num);
}
return result_1;
}
```



Code tham khảo điều khiển mạch CDI-DC

```
#include "stm8s.h"  
#include "iostm8s105c6.h"  
#include "stdbool.h"  
  
int stage;  
bool turnoff_Hbridge = 0;  
  
void HSI_16MHz_config(void);  
void GPIO_config(void);  
void TIMER1_setup(void);  
void TIMER4_setup(void);  
void TIMER3_setup(void);  
void EXIT_setup(void);  
  
int main(void)  
{  
    HSI_16MHz_config();  
    GPIO_config();  
    TIMER1_setup();  
    TIMER4_setup();  
    TIMER3_setup();  
    PD_ODR_ODR0 = 0;  
    PD_ODR_ODR2 = 0;
```



```
PD_ODR_ODR3 = 0;  
PD_ODR_ODR4 = 0;  
PD_ODR_ODR5 = 1;  
while (1)  
{  
    ;  
}  
INTERRUPT_HANDLER(TIM4_UPD_OVF_IRQHandler, 23)  
{  
    TIM4_ARR = 79;  
    stage++;  
    if (stage == 1)  
    {  
        PD_ODR_ODR0 = 0; // Turn off the T1  
        // Delay 0.5 micro second  
        asm("nop");  
        asm("nop");  
        asm("nop");  
        asm("nop");  
        asm("nop");  
        asm("nop");  
        asm("nop");  
        PD_ODR_ODR2 = 1; // Turn on the T2  
        // Delay 0.5 micro second  
        asm("nop");  
        asm("nop");  
        asm("nop");
```



```
asm("nop");
asm("nop");
asm("nop");
asm("nop");
PD_ODR_ODR4 = 0; // Turn off the T4
// Delay 0.5 micro second
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
PD_ODR_ODR3 = 1; // Turn on the T3
}
else if (stage == 2)
{
    PD_ODR_ODR2 = 0; // Turn off the T2
    // Delay 0.5 micro second
    asm("nop");
    asm("nop");
    asm("nop");
    asm("nop");
    asm("nop");
    asm("nop");
    asm("nop");
    PD_ODR_ODR0 = 1; // Turn on the T1
    // Delay 0.5 micro second
```



```
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");

PD_ODR_ODR3 = 0; // Turn off the T3
                    // Delay 0.5 micro second
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");

PD_ODR_ODR4 = 1; // Turn on the T4
stage = 0;
}

if(turnoff_Hbridge == 1)
{
    PD_ODR_ODR0 = 0; // Turn off the T1
    PD_ODR_ODR2 = 0; // Turn off the T2
    PD_ODR_ODR3 = 0; // Turn off the T3
    PD_ODR_ODR4 = 0; // Turn off the T4
```



```
TIM4_CR1_CEN = 0;  
turnoff_Hbridge = 0;  
}  
TIM4_SR UIF = 0; // Clear the update interrupt flag  
}  
//-----  
-----//  
INTERRUPT_HANDLER(TIM1_CAP_COM_IRQHandler, 12)  
{  
if (TIM1_SR1_CC2IF == 1)  
{  
turnoff_Hbridge = 0;  
PD_ODR_ODR0 = 1; // Turn on the T1  
PD_ODR_ODR2 = 0; // Turn off the T2  
PD_ODR_ODR3 = 0; // Turn off the T3  
PD_ODR_ODR4 = 1; // Turn on the T4  
  
TIM4_ARR = 55;  
TIM4_CR1_CEN = 1; // Enable Timer 4  
TIM1_SR1_CC2IF = 0; // Clear the input capture interrupt flag of channel 2  
}  
if (TIM1_SR1_CC4IF == 1)  
{  
turnoff_Hbridge = 1;  
PD_ODR_ODR5 = 0; // Trigger SCR
```



```
TIM3_CR1_CEN = 1; // Enable the timer 2
TIM1_SR1_CC4IF = 0; // Clear the input capture interrupt flag of channel 4
}
}

//-----
-----//  

INTERRUPT_HANDLER(TIM3_UPD_OVF_BRK_IRQHandler, 15)
{
    PD_ODR_ODR5 = 1; // Disable SCR
    TIM3_CR1_CEN = 0; // Enable the timer 3
    TIM3_SR1 UIF = 0; // Clear the update interrupt flag
}
//-----
-----//  

void HSI_16MHz_config(void)
{
    CLK_DeInit();
    CLK_HSICmd(ENABLE);
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
    CLK_SYSCLKConfig(CLK_PRESCALER_CPUDIV1);
    CLK_ClockSwitchConfig(CLK_SWITCHMODE_AUTO, CLK_SOURCE_HSI,
DISABLE, CLK_CURRENTCLOCKSTATE_DISABLE);

    CLK_PeripheralClockConfig(CLK_PERIPHERAL_SPI, DISABLE);
    CLK_PeripheralClockConfig(CLK_PERIPHERAL_I2C, DISABLE);
    CLK_PeripheralClockConfig(CLK_PERIPHERAL_ADC, DISABLE);
```



```
CLK_PeripheralClockConfig(CLK_PERIPHERAL_AWU, DISABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_UART1, DISABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_UART3, DISABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER1, ENABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER2, DISABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER3, ENABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_TIMER4, ENABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_CAN, DISABLE);
}

//-----
-----//  
  
void GPIO_config(void)
{
    GPIO_DeInit(GPIOD);
    GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_HIGH_FAST);
    GPIO_Init(GPIOD, GPIO_PIN_2, GPIO_MODE_OUT_PP_HIGH_FAST);
    GPIO_Init(GPIOD, GPIO_PIN_3, GPIO_MODE_OUT_PP_HIGH_FAST);
    GPIO_Init(GPIOD, GPIO_PIN_4, GPIO_MODE_OUT_PP_HIGH_FAST);
    GPIO_Init(GPIOD, GPIO_PIN_5, GPIO_MODE_OUT_PP_HIGH_FAST);

    GPIO_DeInit(GPIOC);
    GPIO_Init(GPIOC, GPIO_PIN_2, GPIO_MODE_IN_PU_NO_IT);
    GPIO_Init(GPIOC, GPIO_PIN_4, GPIO_MODE_IN_PU_NO_IT);
}
```



```
//-----
-----//  
  
void TIMER1_setup(void)  
{  
    TIM1_DeInit();  
  
    TIM1_TimeBaseInit(0, TIM1_COUNTERMODE_UP, 2499, 0);  
  
    TIM1_ICInit(TIM1_CHANNEL_2, TIM1_ICPOLARITY_RISING,  
    TIM1_ICSELECTION_DIRECTTI, TIM1_ICPSC_DIV1, 1);  
  
    TIM1_ICInit(TIM1_CHANNEL_4, TIM1_ICPOLARITY_FALLING,  
    TIM1_ICSELECTION_DIRECTTI, TIM1_ICPSC_DIV1, 1);  
  
    TIM1_ARRPreloadConfig(DISABLE);  
  
    TIM1_ITConfig(TIM1_IT_CC2, ENABLE);  
  
    TIM1_ITConfig(TIM1_IT_CC4, ENABLE);  
  
    TIM1_Cmd(ENABLE);  
  
  
    enableInterrupts();  
}  
//-----
-----//  
  
void TIMER3_setup(void)  
{  
    TIM3_DeInit();  
  
    TIM3_TimeBaseInit(TIM3_PRESCALER_1, 15999);  
  
    TIM3_ARRPreloadConfig(DISABLE);  
  
    TIM3_ITConfig(TIM3_IT_UPDATE, ENABLE);  
  
    TIM3_Cmd(DISABLE);
```



```
enableInterrupts();  
}  
//-----  
----//  
void TIMER4_setup(void)  
{  
    TIM4_DeInit();  
    TIM4_TimeBaseInit(TIM4_PRESCALER_1, 55);  
    TIM4_ARRPreloadConfig(DISABLE);  
    TIM4_ITConfig(TIM4_IT_UPDATE, ENABLE);  
    TIM4_Cmd(DISABLE);  
  
    enableInterrupts();  
}  
//-----  
----//
```



Code tham khảo giả lập tốc độ động cơ

```
int count = 0;  
  
int ADC_value;  
  
int tolerance;  
  
float T_cycle_intake_compression_stroke;  
  
float T_on_intake_compression_stroke;  
  
float T_cycle_ignition_exhaust_stroke;  
  
float T_on_ignition_exhaust_stroke;  
  
uint16_t Ne;  
  
bool flag_UPDATE;  
  
bool flag_value_UPDATE;  
  
bool flag_TIMER1_status_disable;  
  
bool ignition_exhaust_stoke = 0;  
  
//-----//  
  
void setup()  
{  
    //Set up pin mode  
    //Pin 9 as ouput  
    DDRB |= B00000010;  
    //Pin A0 as input  
    pinMode(A1, INPUT);  
    //TIMER 0  
    //Reset TIMER0
```



```
TCCR0A = 0;  
TCCR0B = 0;  
TIMSK0 = 0;  
//Set up TIMER0  
TCCR0B |= B00000001;  
TCCR0A |= B00000011;  
TIMSK0 |= B00000010;  
OCR0A = 5;  
//TIMER1  
//Reset TIMER1  
TCCR1A = 0;  
TCCR1B = 0;  
TIMSK1 = 0;  
//Set up TIMER1  
//Fast PWM with TOP as ICR1 (Non-inverted mode)  
//Prescaler as 256  
TCCR1A |= B10000010;  
TCCR1B |= B00011101;  
TIMSK1 |= B00000001; //Interrupt when overflow  
//Calculate T_cycle and T_on  
ADC_value = analogRead(A0);  
if(flag_UPDATE == 1)  
{  
    if(ADC_value >= 102)
```



```
if(flag_TIMER1_status_disable == 1)
{
    TCCR1B |= B00011101;
    flag_TIMER1_status_disable = 0;
    Ne = map(ADC_value, 102, 1023, 60, 12000);
    T_cycle_intake_compression_stroke = 60/(float)(Ne)/1024*16000000;
    T_on_intake_compression_stroke = T_cycle_intake_compression_stroke*10/360;
    T_cycle_ignition_exhaust_stroke = T_cycle_intake_compression_stroke * 9 /10;
    T_on_ignition_exhaust_stroke = T_cycle_ignition_exhaust_stroke * 10 /360;
}
else
{
    Ne = map(ADC_value, 102, 1023, 60, 12000);
    T_cycle_intake_compression_stroke = 60/(float)(Ne)/1024*16000000;
    T_on_intake_compression_stroke = T_cycle_intake_compression_stroke*10/360;
    T_cycle_ignition_exhaust_stroke = T_cycle_intake_compression_stroke * 9 /10;
    T_on_ignition_exhaust_stroke = T_cycle_ignition_exhaust_stroke * 10 /360;
}
else
{
    TCCR1B = 0;
    flag_TIMER1_status_disable = 1;
}
}
```



```
// Update value for T_cycle and T_on
ICR1 = (uint16_t)T_cycle_intake_compression_stroke - 1;
OCR1A = (uint16_t)T_on_intake_compression_stroke - 1;
}

//-----
ISR(TIMER0_COMPA_vect)
{
    count++;
    if(count == 1)
    {
        flag_UPDATE = 1;
    }
}

//-----
ISR(TIMER1_OVF_vect)
{
    ignition_exhaust_stoke = !ignition_exhaust_stoke;
    tolerance = random(-3, 3);
    Serial.println(tolerance);
    if (ignition_exhaust_stoke == 1)
    {
        ICR1 = (uint16_t)(T_cycle_ignition_exhaust_stoke*(100 + tolerance)/100) - 1;
        OCR1A = (uint16_t)(T_on_ignition_exhaust_stoke*(100 + tolerance)/100) - 1;
    }
    else

```



```
{  
    ICR1 = (uint16_t)(T_cycle_intake_compression_stroke*(100 + tolerance)/100) - 1;  
    OCR1A = (uint16_t)(T_on_intake_compression_stroke*(100 + tolerance)/100) - 1;  
}  
  
if(flag_value_UPDATE == 1)  
{  
    T_cycle_intake_compression_stroke = 60/(float)(Ne)/1024*16000000;  
    T_on_intake_compression_stroke = T_cycle_intake_compression_stroke*10/360;  
    T_cycle_ignition_exhaust_stroke = T_cycle_intake_compression_stroke * 9 /10;  
    T_on_ignition_exhaust_stroke = T_cycle_ignition_exhaust_stroke * 10 /360;  
    flag_value_UPDATE = 0;  
}  
}  
//-----//  
  
void loop()  
{  
    if(flag_UPDATE == 1)  
    {  
        ADC_value = analogRead(A1);  
        if(ADC_value >= 102)  
        {  
            if(flag_TIMER1_status_disable == 1)  
            {  
                TCCR1B |= B00011101;  
                flag_TIMER1_status_disable = 0;  
            }  
        }  
    }  
}
```



```
Ne = map(ADC_value, 102, 1023, 60, 12000);
T_cycle_intake_compression_stroke = 60/(float)(Ne)/1024*16000000;
T_on_intake_compression_stroke = T_cycle_intake_compression_stroke*10/360;
T_cycle_ignition_exhaust_stroke = T_cycle_intake_compression_stroke * 9 /10;
T_on_ignition_exhaust_stroke = T_cycle_ignition_exhaust_stroke * 10 /360;
}
else
{
Ne = map(ADC_value, 102, 1023, 60, 12000);
T_cycle_intake_compression_stroke = 60/(float)(Ne)/1024*16000000;
T_on_intake_compression_stroke = T_cycle_intake_compression_stroke*10/360;
T_cycle_ignition_exhaust_stroke = T_cycle_intake_compression_stroke * 9 /10;
T_on_ignition_exhaust_stroke = T_cycle_ignition_exhaust_stroke * 10 /360;
}
else
{
TCCR1B = 0;
flag_TIMER1_status_disable = 1;
}
flag_value_UPDATE = 1;
flag_UPDATE = 0;
}
//-----//
```



TÀI LIỆU THAM KHẢO

- [1]. Tài liệu đào tạo TOYOTA
- [2]. Bài giảng hệ thống đánh lửa, môn học Hệ thống điện – điện tử ô tô, bộ môn ô tô – máy động lực, khoa kỹ thuật giao thông, Đại học Bách Khoa TPHCM.
- [3].RM0016 Reference manual, STM8S Series and STM8AF Series 8-bit microcontrollers
- [4]. UM0817 User Manual, STM8S-DISCOVERY
- [5]. L7805 Datasheet (PDF) - STMicroelectronics
- [6]. IR2110 Datasheet (PDF) - International Rectifier
- [7]. IRF540N Datasheet (PDF) - Intersil Corporation
- [8]. TYN1225 Datasheet (PDF) - STMicroelectronics.
- [9]. FR207 Datasheet (PDF) - Won-Top Electronics
- [10]. STM8S105C6T6 Datasheet (PDF) - STMicroelectronics