ĐIỀU CHẾ VECTOR KHÔNG GIAN CHO BIẾN TẦN BA PHA HAI MỨC TRÊN HỆ THỐNG NHÚNG ARM CORTEX

IMPLEMENTATION SPACE VECTOR PULSE WIDTH MODULATION FOR THREE-PHASE TWO-LEVEL INVERTER BASED ON ARM CORTEX EMBEDDED SYSTEM

Nguyễn Văn Đoài, Quách Đức Cường*

TÓM TẮT

Biến tần ba pha hai mức sử dụng công nghệ điều chế vector không gian (SVPWM) là một trong những cấu trúc biến tần phổ biến nhất hiện nay. Quá trình thiết kế chế tạo hệ thống thiết bị biến tần thường qua các công đoạn: mô phỏng, thực hiện thiết kế phần cứng, lập trình phần mềm trên MCU, hiệu chỉnh... Việc phân tích, mô phỏng và đề xuất giải pháp cải tiến biến tần đã có khá nhiều tài liệu đề cập. Trong bài báo này, chúng tôi sẽ tập trung vào vấn đề thực hiện giải thuật SVPWM trên hệ thống nhúng một cách chi tiết. Giải thuật SVPWM được thử nghiệm trên hệ nhúng ARM 32-bit Cortex M4 và mô hình biến tần công suất nhỏ đã minh họa cho quá trình triển khai và thực thi hệ thống.

Từ khoá: *SVPWM, kit STM32F4-discovery, biến tần ba pha nguồn áp.*

ABSTRACT

Two- level three-phase inverters using space vector modulation technology (SVPWM) is one of the most popular inverter structures. The process of designing and manufacturing inverter systems is often in some steps: simulation, hardware design, software programming on MCU, test correction, etc. Analysis, simulation and improvement of inverter has a lot of documents mentioned. In this paper, we will focus on how to develop SVPWM modulation programming for two-level three-phase inverters on an embedded system. The system is tested on ARM 32-bit Cortex M4 embedded systems and small power inverter model in order to explain the SVPWM implementation process.

Keywords: *SVPWM, STM32F4-discovery kit, Three-phase Voltage Inverter.*

Khoa Điện, Trường Đại học Công nghiệp Hà Nội
*Email: quachcuong304@gmail.com
Ngày nhận bài: 10/01/2010
Ngày nhận bài sửa sau phản biện: 25/4/2019
Ngày chấp nhận đăng: 15/8/2019

KÝ HIỆU

Ký hiệu	Đơn vị	Ý nghĩa
f	Hz	Tần số sóng sin
f_{pwm}	Hz	Tần số điều biến độ rộng xung
V_s	V	Biên độ vector điện áp quay

CHỮ VIẾT TẮT

SVPWM	Điều chế vector không gian
2L3P	Ba pha hai mức
MCU	Vi điều khiển

1. GIỚI THIỆU

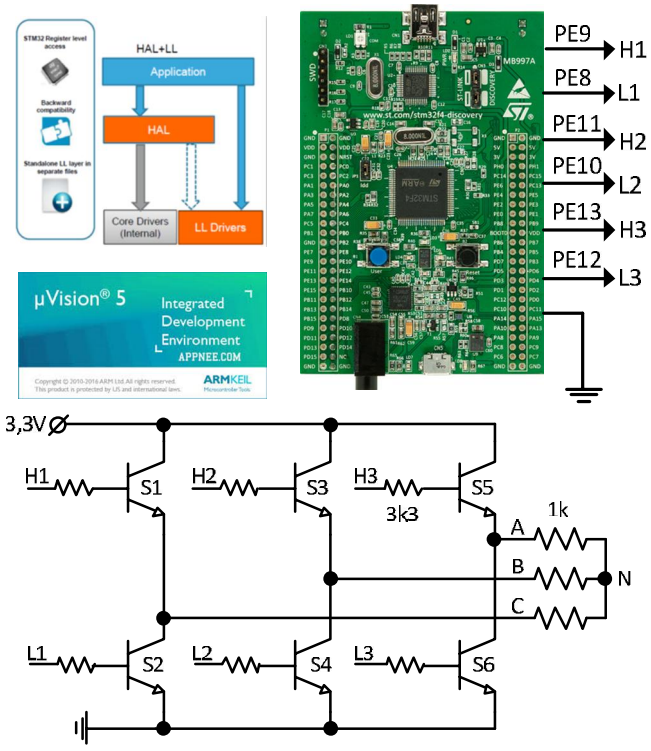
Công nghệ điều chế vector không gian (SVWPM) cho biến tần ba pha đã tạo ra cuộc cách mạng về kỹ thuật biến tần. Từ đó phát triển được một thế hệ biến tần cho phép điều khiển mềm tần số, biên độ và góc pha. Trên cơ sở nền tảng điều chế SVPWM cho biến tần ba pha hai mức (2L3P), gần đây các kỹ sư, nhà khoa học đã phát triển thêm một loạt các biến tần với cấu trúc và giải thuật điều chế SVPWM đa bậc (số bậc > 2) có sóng hài thấp và công suất lớn. Tuy vậy công nghệ SVPWM cho biến tần 2L3P vẫn là cấu trúc nền tảng và được sử dụng rất phổ biến hiện nay do: 1) cấu trúc đơn giản; 2) hoạt động tin cậy; 3) giải thuật điều chế không quá phức tạp… Đã có rất nhiều các công trình công bố về SVPWM cho biến tần 2L3P [1, 2, 3]. Song phần lớn các công trình trên chỉ công bố trên phương diện lý thuyết, tính toán mô phỏng và nếu có đề cập đến vấn đề triển khai hệ thống thực thì hầu như các tác giả chỉ đưa ra kết quả thực nghiệm. Các vấn đề chi tiết trong kỹ thuật triển khai thực hiện SVPWM trên một hệ thống nhúng đã được "che mờ" đi và ít đề cập đến. Trong bài báo này, chúng tôi trình bày cách thức triển khai thực hiện SVPWM cho biến tần 2L3P trên hệ thống nhúng ARM 32-bit Cortex M4 sử dụng lõi chíp STM32F407VGT của hãng sản xuất STMicroelectronics. Cấu trúc bài báo được trình bày theo thứ tự: 2) cấu trúc phần cứng; 3) thực hiện giải thuật điều chế; 4) thực hiện hệ thống; 5) kết quả và 6) kết luận.

2. CẤU TRÚC PHẦN CỨNG

Phần cứng thử nghiệm có cấu trúc mô tả trên hình 1. Mạch động lực gồm 6 tranzitor C828 (S_1, S_2,… S_6). Tải ba pha sử dụng các điện trở có giá trị 1kΩ. Bộ điều khiển sử dụng kit STM32F4-discovery với MCU STM32F407VGT. Đây là một trong các MCU tiên tiến, hiện đại và đa dụng bậc nhất hiện nay. STM32F407VGT có các thông số cơ bản như sau: lõi chíp xử lý ARM-32bit Cortex M4 + DSP; tốc độ xung nhịp 168MHz; bộ nhớ Flash 1024kB; RAM 192kB; 24 ADC channels 12-bit tốc độ lấy mẫu lên tới 7,2 MSPS; 2 DAC channels 12-bit; 17 Timer (15 timer 16-bit, 2 timer 32-bit) với 2 Timer 16-bit dùng cho mục đích điều chế SVPWM; 3 I2C; 4 USART/2UART; 3 SPI; 2 CAN; SDIO interfaces [6].

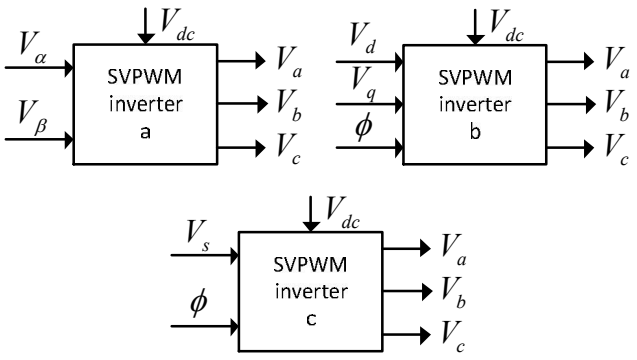
Trên kit tích hợp sẵn bộ nạp ST-Link V2 giao tiếp qua chuẩn USB thuận lợi cho quá trình lập trình thử nghiệm hệ

thống. Sử dụng trình biên dịch Keil-C V5, phần mềm cấu hình vi điều khiển STM32CubeMX và tập thư viện HAL để phát triển thử nghiệm hệ thống.



Hình 1. Mô hình thử nghiệm giải thuật điều chế biến tần ba pha hai mức SVPWM

3. GIẢI THUẬT ĐIỀU CHẾ



Hình 2. Các cấu trúc khối In-Out của biến tần

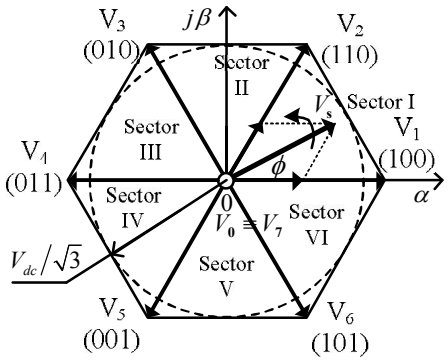
Trong các hệ thống điều khiển sử dụng biến tần là cơ cấu chấp hành đều nhìn nhận biến tần là một khâu khuếch đại có hệ số bằng 1 có chức năng sao chép nguyên trạng về biên độ, tần số và góc pha [4]. Ta có thể thấy trên các sơ đồ cấu trúc hệ thống điều khiển thì biến tần là một khâu với cặp tín hiệu vào ra tùy thuộc vào phương thức điều chế. Có 3 hình thức đầu vào như sau tương ứng với các hình 2.a, 2.b và 2.c.

1. Đầu vào là cặp điện áp trên hệ trục tọa độ cố định V_α và V_β.
2. Đầu vào là cặp điện áp trên hệ trục tọa độ quay V_d, V_q và góc pha ϕ.
3. Đầu vào là biên độ vector quay của điện áp V_s và góc pha ϕ.

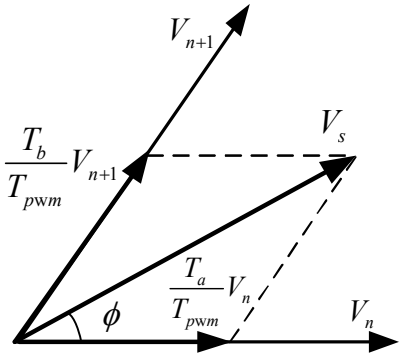
Trong bài báo này xem xét biến tần SVPWM như một khâu với tín hiệu vào là biên độ vector quay V_s và góc pha ϕ (hình 2.c). Nếu coi biến tần là lý tưởng (tần số sóng mang đủ lớn, tổn thất điện áp và công suất bằng 0, van công suất đóng cắt lý tưởng, không gian điều chế là đường tròn nội tiếp đa giác (hình 3), thời gian ngưng dẫn trên một nhánh van bằng 0,…) thì điện áp pha của biến tần tính bởi (1).

$$\begin{cases} V_a(t) = V_s cos(\omega t) = \gamma V_{s\_max} cos(\omega t) \\ V_b(t) = V_s cos(\omega t + 2\pi/3) = \gamma V_{s\_max} cos(\omega t + 2\pi/3) \\ V_c(t) = V_s cos(\omega t - 2\pi/3) = \gamma V_{s\_max} cos(\omega t - 2\pi/3) \end{cases} \quad (1)$$

Trong đó, hệ số điều chế biên độ $\gamma = [0,1]$, biên độ điện áp lớn nhất có thể điều chế được là $V_{s\_max} = V_{dc}/\sqrt{3}$ và tốc độ góc $\omega = d\phi/dt$ [1, 4].



Hình 3. Không gian điều chế V_s



Hình 4. Điều chế V_s tại Sector thứ n

Theo [1, 2, 4] để điều chế được vector điện áp quay có biên độ và góc pha biến đổi trong toàn bộ không gian góc 2π thì ta chia không gian thành 6 sector và thực hiện việc duy trì các vector gốc (V_1, V_2,… V_6) trong những khoảng thời gian quy định khi vector điện áp dịch chuyển vào các vùng sector tương ứng. Thời gian điều chế T_a, T_b và T_0 cho các vector V_n và V_{n+1} như sau [1, 4]:

$$\begin{cases} T_a = T_{pwm} \left| \frac{\sqrt{3}V_s}{V_{dc}} \right| sin\left(\frac{n}{3}\pi - \phi\right) \\ T_b = T_{pwm} \left| \frac{\sqrt{3}V_s}{V_{dc}} \right| sin\left(-\frac{n-1}{3}\pi + \phi\right) \\ T_0 = T_{pwm} - T_a - T_b \end{cases} \quad (2)$$

Trong đó n = 1, 2, 3,…, 5 chỉ vị trí sector mà vector V_s đang thuộc phạm vi không gian đó. T_a, T_b và T_0 lần lượt khoảng thời gian tồn tại của vector thứ V_n, V_{n+1}và vector V_0 hoặc vector V_7 trong một chu kỳ T_{pwm}. Thay $V_s = \gamma V_{s\_max}$ và $V_{s\_max} = V_{dc}/\sqrt{3}$ vào (2) có được:

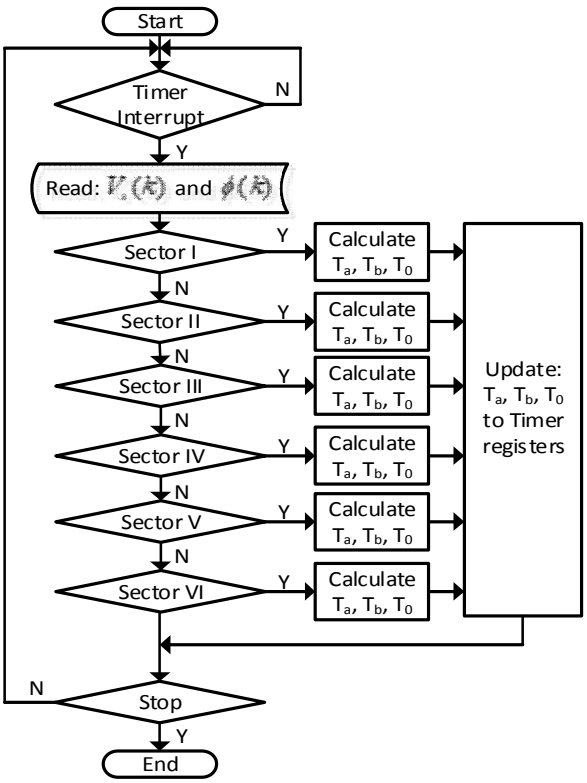
$$\begin{cases} T_a = \gamma T_{pwm} sin\left(\frac{n}{3}\pi - \phi\right) \\ T_b = \gamma T_{pwm} sin\left(-\frac{n-1}{3}\pi + \phi\right) \\ T_0 = T_{pwm} - T_a - T_b \end{cases} \quad (3)$$

Phương trình rời rạc dùng để tính toán, cập nhật dữ liệu trên MCU:

$$\begin{cases} T_a(k) = \gamma(k) T_{pwm} sin\left(\frac{n}{3}\pi - \phi(k)\right) \\ T_b(k) = \gamma(k) T_{pwm} sin\left(-\frac{n-1}{3}\pi + \phi(k)\right) \\ T_0(k) = T_{pwm}(k) - T_a(k) - T_b(k) \end{cases} \quad (4)$$

Trong (4), $\gamma(k)$ đại diện cho biên độ điện áp được điều chế, $\phi(k)$ đại diện cho góc pha của vector quay và tần số lưới xoay chiều.

Sơ đồ và giải thuật điều chế được thể hiện trên hình 5 và bảng 1 [1]. Lưu ý, tại Sector 6 thì T_a, T_b lần lượt là thời gian của V_6, V_1.



Hình 5. Giải thuật điều chế SVPWM

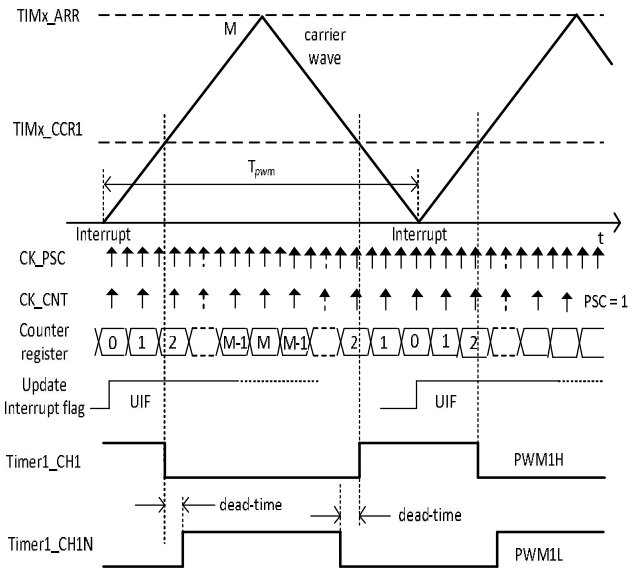
Bảng 1. Bảng đóng ngắt của các van [1]

Sector	Van $S_1S_3S_5$	Van $S_2S_4S_6$
I	S_1=T_a+T_b+T_0/2	S_4=T_0/2
	S_3=T_b+T_0/2	S_6=T_a+T_0/2
	S_5=T_0/2	S_2=T_a+T_b+T_0/2
II	S_1=T_a+T_0/2	S_4=T_b+T_0/2
	S_3=T_a+T_b+T_0/2	S_6=T_0/2
	S_5=T_0/2	S_2=T_a+T_b+T_0/2
III	S_1=T_0/2	S_4=T_a+T_b+T_0/2
	S_3=T_a+T_b+T_0/2	S_6=T_0/2
	S_5=T_b+T_0/2	S_2=T_a+T_0/2
IV	S_1=T_0/2	S_4=T_a+T_b+T_0/2
	S_3=T_a+T_0/2	S_6=T_b+T_0/2
	S_5=T_a+T_b+T_0/2	S_2=T_0/2
V	S_1=T_b+T_0/2	S_4=T_a+T_0/2
	S_3=T_0/2	S_6=T_a+T_b+T_0/2
	S_5=T_a+T_b+T_0/2	S_2=T_0/2
VI	S_1=T_a+T_b+T_0/2	S_4=T_0/2
	S_3=T_0/2	S_6=T_a+T_b+T_0/2
	S_5=T_a+T_0/2	S_2=T_b+T_0/2

4. THỰC HIỆN GIẢI THUẬT SVPWM TRÊN HỆ THỐNG NHÚNG ARM CORTEX

4.1. Mô tả cài đặt Timer

Trên MCU STM32F407VGT có tổng cộng 17 Timer ứng dụng cho nhiều bài toán khác nhau. Trong đó có 2 Timer (Timer1 và Timer8) với cấu hình 16-bit dùng cho phát triển SVPWM ba pha [6]. Trong bài báo này, chúng tôi sử dụng Timer1 để cài đặt thuật toán SVPWM. Timer hoạt động ở chế độ SVPWM thực chất là một bộ đếm phối hợp với khâu so sánh. Khi giá trị bộ đếm đạt tới ngưỡng so sánh thì khâu so sánh thực hiện chức năng lật trạng thái tại các bit-flag và các chân đầu ra trên MCU tạo ra tín hiệu PWM.



Hình 6. Giản đồ sóng của Timer1 trong chế độ PWM với sóng mang dạng tam giác cân [5]

Hình 6 mô tả giản đồ xung của Timer1 khi hoạt động ở chức năng SVPWM. Những vấn đề chúng ta cần quan tâm khi thiết lập chế độ hoạt động của Timer:

1. Hoạt động trong chế độ ngắt, khoảng thời gian giữa các ngắt của Timer chính là chu kỳ cập nhật giá trị thời gian (T_a, T_b và T_0) điều chế.

2. Thiết lập thời gian ngưng dẫn giữa hai van trên một nhánh nối tiếp (dead-time).

3. Giản đồ sóng mang điều chế phải thực hiện ở chế độ tam giác cân (một chu kỳ bộ đếm bao gồm hai bước đến lên sau đó đếm xuống).

4. Cài đặt chu kỳ xung đếm bao gồm các vấn đề liên quan đến chia xung đếm, dung hòa giữa giá trị tần số điều chế và độ phân giải điều chế…

Tần số điều chế PWM khi sóng mang ở chế độ tam giác cân xác định theo (5), [5, 6].

$$f_{pwm} = \frac{f_{CK\_PSC}}{2(PSC+1)(ARR+1)} \quad (5)$$

f_{CK_PSC} là tần số xung cấp cho timer. Khi MCU hoạt động ở tần số 168MHz, tần số f_{CK_PSC} có giá trị lớn nhất là 168MHz. PSC là giá trị thanh ghi TIM1_PSC và nó có chức năng chia xung đưa vào bộ đếm. ARR là giá trị thanh ghi TIM1_ARR, đây là thanh ghi Auto-reload cho bộ đếm. Như vậy nếu lựa chọn $PSC = 15$ và độ phân giải PWM là 10-bit ($ARR = 1023$) và $f_{CK_PSC} = 168$MHz thì sóng mang PWM sẽ có tần số $f_{pwm} = 5{,}127$kHz với độ phân giải 10-bit.

Để xác định được thời gian dead-time cần dựa vào thời gian ON/OFF của van công suất IGBT do nhà sản xuất cung cấp. Phần lớn tham số thời gian trễ Turn-ON và Turn-OFF của các IGBT vào khoảng dưới 100ns và dưới 500ns. Tuy nhiên để tăng hệ số an toàn thì thời gian dead-time cần phải lựa chọn lớn hơn tổng của Turn-ON và Turn-OFF. Và phải có một lượng dự trữ nhất định. Nếu lượng dự trữ này lớn sẽ nâng cao độ an toàn tuy vậy khi đó hệ thống sẽ phải chịu tổn thất về biên độ. Kinh nghiệm cho thấy nên lựa chọn hệ số dead-time DT vào khoảng từ 1,5µs đến 4µs là chấp nhận được. Lưu ý rằng về mặt lý thuyết giá trị dead-time DT không được vượt quá 50% chu kỳ băm xung.

Thời gian dead-time DT xác định theo giá trị DTG trong thanh ghi TIM1_BDTR [5]:

Bảng 2. Thiết lập thời gian dead-time trong thanh ghi TIM1_BDTR

TT	DTG[7:5]	Dead-time DT
1	0xx	DTG[7:0]×t_{DTS}
2	10x	(64+DTG[5:0])×2t_{DTS}
3	110	(32+DTG[4:0])×8t_{DTS}
4	111	(32+DTG[4:0])×16t_{DTS}

Với giá trị t_{DTS} = DIV×TCK_PSC. Trong đó DIV là hệ số chia xung có giá trị 1, 2, 4 được xác lập trong thanh ghi TIM1_CR1.CKD và TCK_PSC là chu kỳ xung CK_PSC [5, 6]. Ví dụ với $f_{CK_PSC} = 168$MHz, DIV = 2, DTG = 11001010 (hệ BIN) = 202 (Hệ DEC) thì dead-time DT có giá trị như sau:

$$DT = (32+10) \times 8t_{DTS} = 42 \times 8t_{DTS} = 42 \times 8 \times DIV \times t_{CK\_PSC}$$

$$DT = 42 \times 8 \times 2 \times t_{CK\_PSC} = \frac{42 \times 8 \times 2}{168000000} = 4 \times 10^{-6}\ s$$

Một số hàm trọng yếu khi cài đặt Timer1 cho điều chế SVPWM sử dụng thư viện HAL do STMicroelectronics cung cấp:

```
static void MX_TIM1_Init(void)
 { ….
 htim1.Instance = TIM1;    //khai báo sử dụng timer1
 htim1.Init.Prescaler = 15;  //cài đặt giá trị PSC, hệ số chia xung vào counter
htim1.Init.CounterMode=TIM_COUNTERMODE_CENTERALIGNED1;//chế độ xung tam giác cân
 htim1.Init.Period = 1023;  //cài đặt giá trị ARR, độ phân giải của sóng PWM
 htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV2;  //hệ số chia xung DIV = 2
 htim1.Init.RepetitionCounter = 0;
 sConfigOC.OCMode = TIM_OCMODE_PWM1;  //chế độ PWM
 sConfigOC.Pulse = 0;
 sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
 sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
 sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
 sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
 sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
 sConfigOC.OCMode = TIM_OCMODE_PWM1; //cài đặt Timer1 ở chế độ PWM
 sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
 sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
 sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
 sBreakDeadTimeConfig.DeadTime = 202;  //cài đặt thời gian dead time, giá trị DTG
 sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
 sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
 sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
 …
 HAL_TIM_MspPostInit(&htim1);
 }
```

Cài đặt Timer1 hoạt động trong chế độ ngắt khi giá trị thanh ghi counter trong Timer1 về 0x0000.

```
HAL_TIM_Base_Start_IT(&htim1);
```

4.2. Chuẩn hóa dữ liệu lập trình

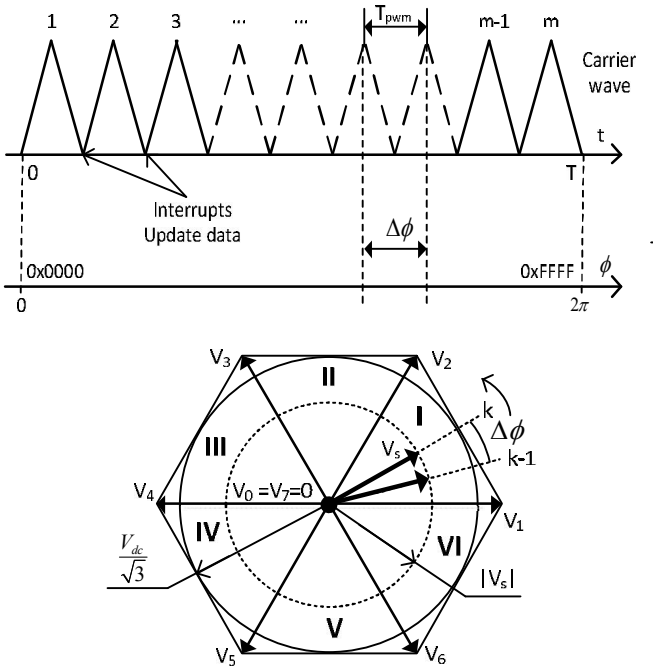
Trước khi lập trình thực hiện giải thuật điều chế SVPWM cần tiến hành chuẩn hóa dữ liệu. Công đoạn này thực chất là lượng tử hóa dữ liệu để tính toán trên MCU. Từ (4) chúng ta cần phải chuẩn hóa 4 tham số: biến thời gian T; hệ số điều chế γ; giá trị hàm sin trong khoảng từ 0 đến $\pi/3$; góc quay ϕ. Chuẩn hóa dữ liệu thể hiện như trên bảng 3.

Bảng 3. Chuẩn hóa dữ liệu lập trình

TT	Tham số	Dải dữ liệu		Dải dữ liệu chuẩn hóa		Số bit
		Giá trị	Đơn vị	Hex	Dec	
1	Thời gian	$[0, T_{pwm}]$	s	[0x0000, 0x03FF]	[0, 1023]	10
2	Hệ số điều chế γ	$[0, 1]$		[0x0000, 0x7FFF]	[0, 32767]	15
3	Hàm sin	$[0, 1]$		[0x0000, 0x7FFF]	[0, 32767]	15
4	Góc pha ϕ	`$[0, 2\pi]$	rad	[0x0000, 0xFFFF]	[0, 65535]	16

Từ (4) và bảng 2 có được công thức tính toán thời gian điều chế SVPWM trên MCU.

$$\begin{cases} T_a(k) = \gamma(k) \times 0x03FF \times sin(n \times 0x2AAA - \phi(k)) \\ T_b(k) = \gamma(k) \times 0x03FF \times sin(-(n-1) \times 0x2AAA + \phi(k)) \\ T_0(k) = T_{pwm}(k) - T_a(k) - T_b(k) \end{cases} \quad (6)$$



Hình 7. Phân chia đường tròn điều chế và lượng tử hóa thành phần góc quay ϕ

Thông thường tần số sóng mang PWM từ 2,5kHz đến 20kHz [4] lớn hơn nhiều lần tần số sóng sin ba pha. Trong 1 chu kỳ sóng sin sẽ có $m = f_{pwm}/f$ chu kỳ sóng mang PWM. Nếu sử dụng hàm sin có sẵn trong thư viện để tính thì thời gian tính toán quá lớn không đủ kịp cho việc cập nhật dữ liệu. Do vậy thường sử dụng bảng tra để tính hàm sin. Lượng góc (giá trị được chuẩn hóa) được cập nhật trong 1 chu kỳ sóng mang PWM:

$$\Delta\phi(k) = \frac{f(k)}{f_{pwm}} \times 0xFFFF = 65535\frac{f(k)}{f_{pwm}} \quad (7)$$

Giải thuật điều chế trong các sector là như nhau do đó chỉ cần tra cứu hàm sin trong phạm vi góc $[0,\ \pi/3]$. Giả sử ta chia đường tròn điều chế ra làm $N = 2047$ điểm tra (11-bit) thì một sector sẽ có 341 điểm tra. Lượng phân giải góc đã chuẩn hóa được tra nhỏ nhất:

$$d\Delta\phi = \frac{0xFFFF}{N} = \frac{0xFFFF}{2047} = 32{,}015 \approx 32 = 2^5 \quad (8)$$

Từ (8) suy ra chỉ số i trong bảng tra hàm $sin(i)$ được xác định như trong (9). Lưu ý tham chiếu tới hàm tra bảng sinetable[(**unsigned int**)(angle1 >> 5)] trong code.

$$i \approx \frac{\phi(k)}{d\Delta\phi} = \frac{\phi(k)}{2^5} \quad (9)$$

Tần số sóng sin nhỏ nhất mà hệ còn thực hiện bảng tra được để điều chế SVPWM:

$$f_{min} = \frac{f_{pwm}}{N} \quad (10)$$

Để thực hiện việc điều chế sóng sin ở dải tần thấp có thể sử dụng hai giải pháp: 1) tăng số điểm chia N; 2) giảm tần số điều biến f_{pwm}. Dù sử dụng giải pháp nào cũng cần lưu ý phải đảm bảo tỷ lệ $f_{pwm}/f \geq 100$ để giảm sóng hài bậc cao khi điều chế.

4.3. Lập trình giải thuật SVPWM trên các sector

Một chu kỳ tính toán và cập nhật dữ liệu điều chế sẽ thực hiện theo trình tự sau:

1. Xảy ra sự kiện ngắt của Timer1, kiểm tra cờ ngắt UIF.
2. Xóa cờ ngắt UIF.
3. Đọc giá trị biên độ V_s và góc pha ϕ.
4. Thực hiện tính toán điều chế cập nhật các giá trị thời gian T_a, T_b và T_0 chuyển hóa chúng thành các giá trị thanh ghi TIMx_CCR1, TIM1_CCR2 và TIM1_CCR3.

Chương trình ngắt của Timer1 điều chế SVPWM. Theo mặc định hàm ngắt của Timer sẽ nằm trong file stm32f4xx_it.c do CubeMX và HAL drive thiết lập quy định.

```
void TIM1_UP_TIM10_IRQHandler(void)
{
    if(TIM1->SR & TIM_SR_UIF)  //kiểm tra cờ ngắt UIF, nếu UIF =1
  { TIM1->SR &= ~TIM_SR_UIF;  //xóa cờ UIF
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,GPIO_PIN_SET); //đưa chân PD14 lên 1
        svpwm(angle,gamma);     //thực hiện hàm điều chế SVPWM
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,GPIO_PIN_RESET);}     //đưa chân PD14 về 0
    HAL_TIM_IRQHandler(&htim1);
}
```

Để linh hoạt hóa quá trình sử dụng module SVPWM nên xây dựng một file mã nguồn C riêng dùng chứa các chương trình con. File C này có tên svpwm.c chứa hai chương trình con: 1) hàm cài đặt timer **void** MX_TIM1_Init(**void**); 2) hàm điều chế SVPWM **void** svpwm(**int** angle, **int** gamma); 3) bảng tra hàm sin **const int** sinetable[341].

Lưu ý: 1) tính hàm sin trong chương trình bằng cách tra bảng; 2) các giải thuật điều chế SVPWM trong các sector 2, 3,.. 6 tương tự như trong sector 1; 3) thời gian dẫn của các van công suất tương ứng với giá trị thanh ghi các TIM1_CCR1, TIM1_CCR2 và TIM1_CCR3 trong MCU được xác định theo bảng 1. Nội dung file svpwm.c bao gồm:

```
#include "stm32f4xx_hal.h"              //khai báo sử dụng thư viện HAL
/*==========ĐỊNH NGHĨA CÁC HẰNG SỐ==========*/
#define  VECTOR1     0                  //0 độ
#define  VECTOR2     0x2AAA             //60 độ, 10922
#define  VECTOR3     0x5555             //120 độ
#define  VECTOR4     0x8000             //180 độ
#define  VECTOR5     0xAAAA             //240 độ
#define  VECTOR6     0xD555             //300 độ
#define  SIXTY_DEG   0x2AAA             //lượng góc 60 độ
#define  VOLTS_LIMIT  0x7FFF            //điện áp giới hạn 0x7FFF
```

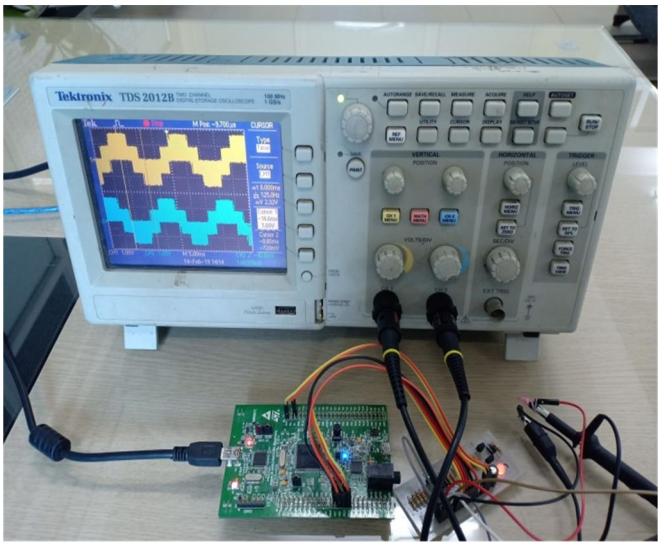
```
#define Tpwm 0x03FF //chu kỳ sóng mang
const int sinetable[341] = {0,100,201,301,..., 28174,28225,28276}; //341 phần tử
/*=============HÀM CÀI ĐẶT TIMER1=============*/
static void MX_TIM1_Init(void)
{ ... }
/*=====CHƯƠNG TRÌNH ĐIỀU CHẾ SVPWM TRONG CÁC SECTOR====*/
void svpwm(int angle, int gamma)
{ unsigned int angle1, angle2, half_T0, Ta, Tb;
/*================SECTOR1================*/
 if(angle < VECTOR2)
 {angle2 = angle - VECTOR1;
    angle1 = SIXTY_DEG - angle2;
  Ta = sinetable[(unsigned int)(angle1 >> 5)]; //chia 32, 60 d0 tuong ung 341 diem
    Ta = ((long)Ta*(long)gamma) >> 15; //chia 32768
    Ta = ((long)Ta*(long)Tpwm) >> 15; //chia 32768
    Tb = sinetable[(unsigned int)(angle2 >> 5)];//chia 32
    Tb = ((long)Tb*(long)gamma) >> 15; //chia 32768
    Tb = ((long)Tb*(long)Tpwm) >> 15; //chia 32768, sin(x)*sqrt(3)*T*Vs/Vdc
    half_T0 = (Tpwm - Ta - Tb) >> 1;
    TIM1->CCR1 = Ta + Tb + half_T0; //cập nhật dữ liệu cho kênh PWM1
    TIM1->CCR2 = Tb + half_T0; //cập nhật dữ liệu cho kếnh PWM2
    TIM1->CCR3 = half_T0; } //cập nhật dữ liệu cho kênh PWM3
/*================SECTOR2================*/
 else if(angle < VECTOR3) //điều chế trong sector 2
 { ... }
/*==============SECTOR3==============*/
 else if(angle < VECTOR4) //điều chế trong sector 3
 { ... }
/*=============SECTOR4=============*/
 else if(angle < VECTOR5) //điều chế trong sector 4
 { ... }
/*=============SECTOR5=============*/
 else if(angle < VECTOR6) //điều chế trong sector 5
 { ... }
/*=============SECTOR6=============*/
 else if (angle<0xFFFF) //điều chế trong sector 6
 { ... }
 else
 {
  angle=0x0000;
 }
}
```

5. KẾT QUẢ THỰC HIỆN

5.1. Cài đặt tham số hệ thống

Kit STM32F4-discovery được cài đặt hoạt động ở tần số xung nhịp 168MHz. Tần số xung cấp cho Timer1 là $f_{CK_PSC} = 168$MHz. Các hệ số: $PSC = 15$, $ARR = 1023$, $DIV = 2$, $DTG = 202$. Số lượng điểm chia không gian đường tròn điều chế là 2047 điểm. Như vậy đặc tính của sóng điều chế PWM trên phương diện tính toán lý thuyết như sau:

- Tần số điều chế sóng PWM: $f_{pwm} = 5{,}127$kHz.
- Độ phân giải khi điều chế: 10-bit.
- Thời gian ngưng dẫn giữa hai van trên một nhánh là dead-time: $DT = 4{,}0$µs.

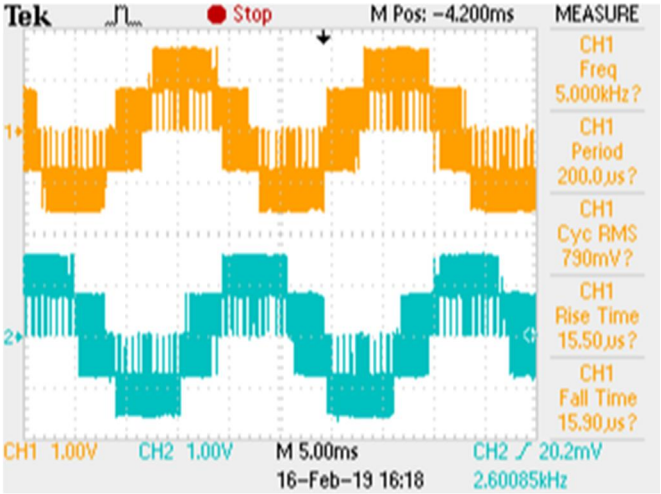


Hình 8. Mô hình thực nghiệm điều chế SVPWM cho biến tần ba pha hai mức

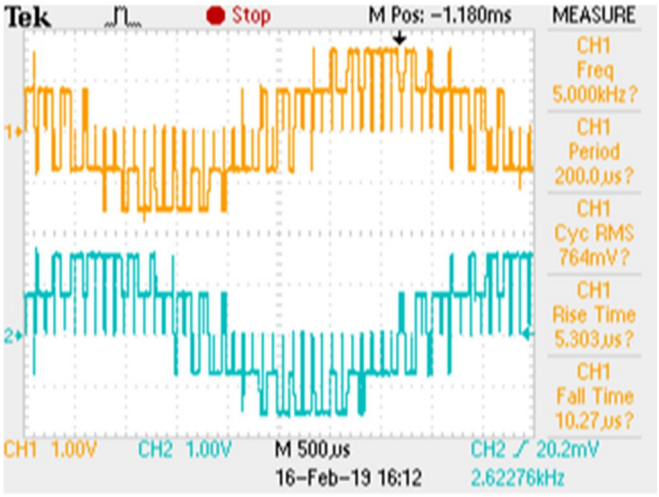
Trong quá trình thử nghiệm sử dụng vòng lặp để tạo ra biến góc pha $\phi(k)$ biến đổi theo chu kỳ phục vụ quá trình khảo sát và đánh giá.

5.2. Kết quả thực nghiệm giải thuật SVPWM khi góc pha $\phi(k)$ có chu kỳ ổn định

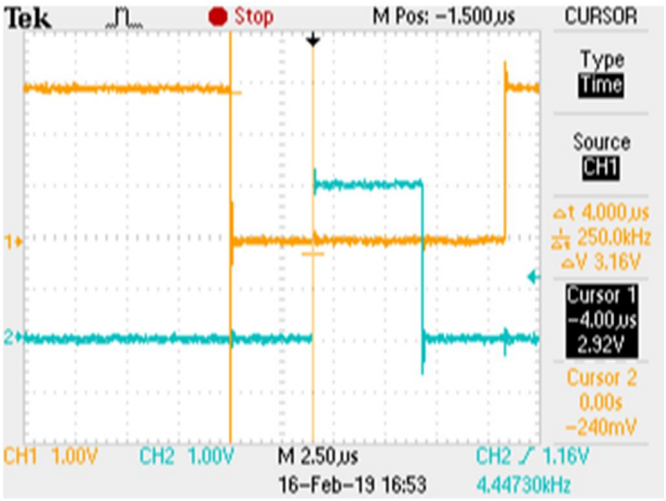
Hình 9 và 10 lần lượt là điện áp pha khi góc pha biến đổi tuyến tính đồng thời có chu kỳ lần lượt là $20ms$ (tần số $f = 50$Hz) và $4ms$ (tần số $f = 250$Hz). Hình 11 là kết quả thực nghiệm đo thời gian dead-time. Kết quả cho thấy dạng điện áp sóng sin (chưa xử lý lọc) phù hợp với lý thuyết, giá trị thời gian dead-time bằng 4,0µs và tần số PWM bằng 5,19kHz.



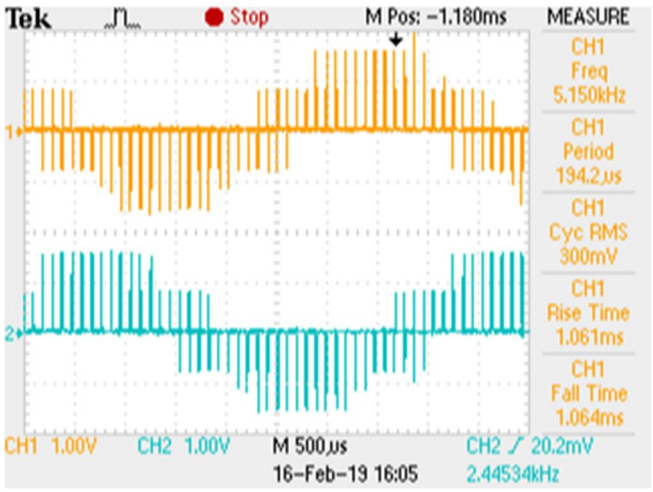
Hình 9. Giản đồ áp pha khi f = 50Hz



Hình 10. Giản đồ áp pha khi $f = 250Hz$

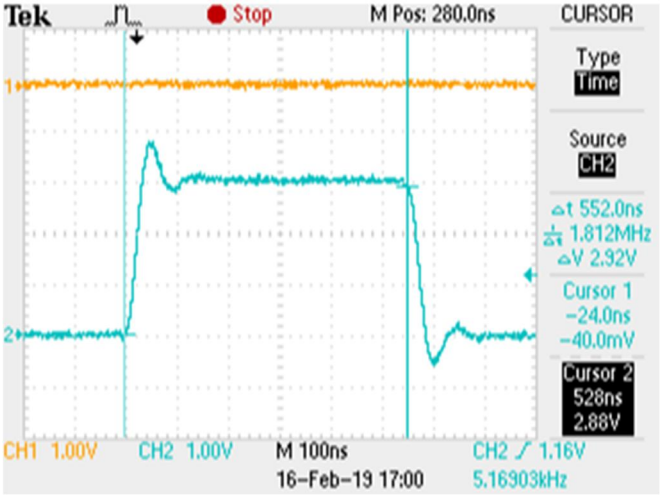


Hình 11. Khảo sát dead-time, DT = 4µs

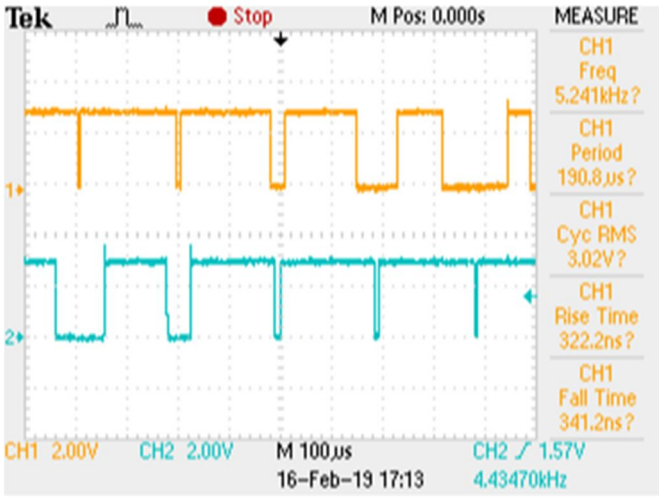


Hình 12. Giảm biên độ áp ($f = 250Hz$)

Hình 12 biểu diễn việc điều chỉnh biên độ điện áp khi lựa chọn hệ số điều chế $\gamma = 0,2$ (giảm 5 lần so với biên độ lớn nhất). Để biết được lượng thời gian MCU dùng cho việc điều chế SVPWM, ta thực hiện việc đưa chân PD14 lên mức logic 1 trước khi gọi hàm svpwm(angle,gamma) và khi thực hiện xong hàm thì kéo PD14 về mức logic 0. Từ độ rộng xung tại PD14 suy ra được thời gian điều chế SVPWM là 552ns (hình 13). Lượng thời gian này nhỏ hơn khoảng thời gian ngưng dẫn do đó đảm bảo về khe thời gian cập nhật dữ liệu. Hình 14 biểu diễn sóng điều chế PWM.

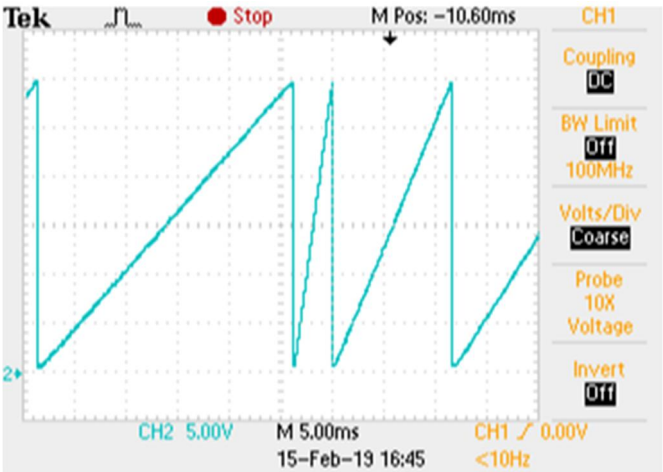


Hình 13. Thời gian điều chế SVPWM

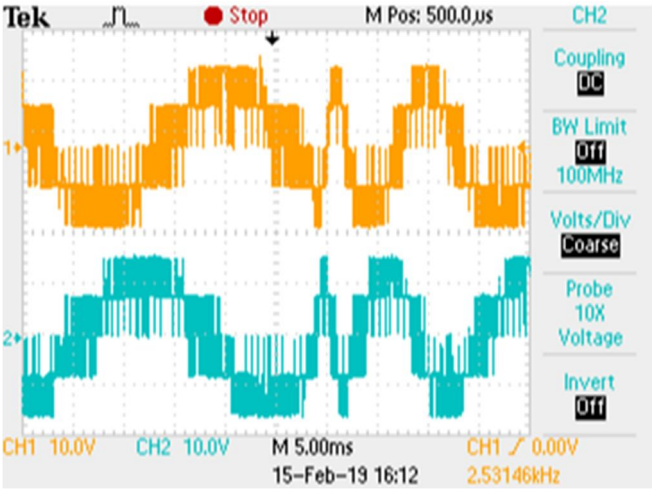


Hình 14. Sóng điều chế PWM

5.3. Kết quả thực nghiệm giải thuật SVPWM khi góc pha $\phi(k)$ có chu kỳ không ổn định



Hình 15. Giản đồ góc pha khi chu kỳ của $\phi(k)$ biến đổi



Hình 16. Giản đồ điện áp pha khi chu kỳ của $\phi(k)$ biến đổi

Hình 15 là giản đồ góc pha $\phi(k)$ biến đổi tuyến tính và có chu kỳ không phải hằng số. Hình 16 là điện áp pha tương ứng với sự biến đổi của chu kỳ góc pha $\phi(k)$.

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo đã trình bày quá trình từng bước thực hiện giải thuật SVPWM cho biến tần 2L3P trên hệ thống nhúng ARM 32-bit Cortex M4 sử dụng lõi chíp xử lý STM32F407VGT của hãng STMicroelectronics. Trong đó nhấn mạnh đến vấn đề kỹ thuật lượng tử hóa và lập trình thực thi giải thuật điều chế SVPWM. Kết quả thử nghiệm trên mô hình thực cho thấy hệ thống hoạt động đúng với nguyên lý, chức năng SVPWM và đảm bảo được đặc tính mở rộng, hiệu chỉnh khi cần thiết. Xu hướng trong thời gian tới nhóm nghiên cứu sẽ chuẩn hóa driver các module chức liên quan đến điều chế biến tần, đặc biệt là vấn đề điều chế biến tần đa bậc và biến tần ma trận.

LỜI CẢM ƠN

Nhóm tác giả xin chân thành gửi lời cảm ơn tới Bộ môn Tự động hóa, Khoa Điện và Khoa Điện tử Trường Đại học Công nghiệp Hà Nội đã hỗ trợ trong quá trình nghiên cứu.

TÀI LIỆU THAM KHẢO

[1]. Duc-Cuong Quach, Quan Yin, Yu-Feng Shi and Chun-Jie Zhou, 2012. *Design and Implementation of Three-phase SVPWM Inverter with 16-bit dsPIC*. IEEE 12th International Conference on Control-Automation-Robotics and Vision, Guangzhou, China, Dec.2012, pp. 1181-1186.

[2]. Tang Lifang, 2013. *Study of the SVPWM Converter Bassed on TMS320F24X*. Third International Conference on Intelligent System Design and Engineering Applications, pp.1316-1319.

[3]. Pengcheng Du, Liyi Li , Jiaxi Liu, Rui Yang, 2018. *A Novel Simplified 3-Level SVPWM Modulation Method Based on the Conventional 2-LSVPWM Modulation Method*. 2018 21st International Conference on Electrical Machines and Systems (ICEMS), pp.1799-1803.

[4]. Nguyễn Phùng Quang, 2005. *Truyền động điện thông minh*. NXB Khoa học và Kỹ thuật, Hà Nội, 250-255.

[5]. STMicroelectronics inc, 2016. *The datasheet of STM32F4xx Microcontroller*.

[6]. STMicroelectronics inc, 2017. *User Manual Description of STM32F4 HAL drivers.*

AUTHORS INFORMATION

Nguyen Van Doai, Quach Duc Cuong

Faculty of Electrical Engineering Technology, Hanoi University of Industry