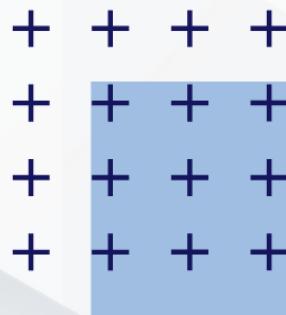
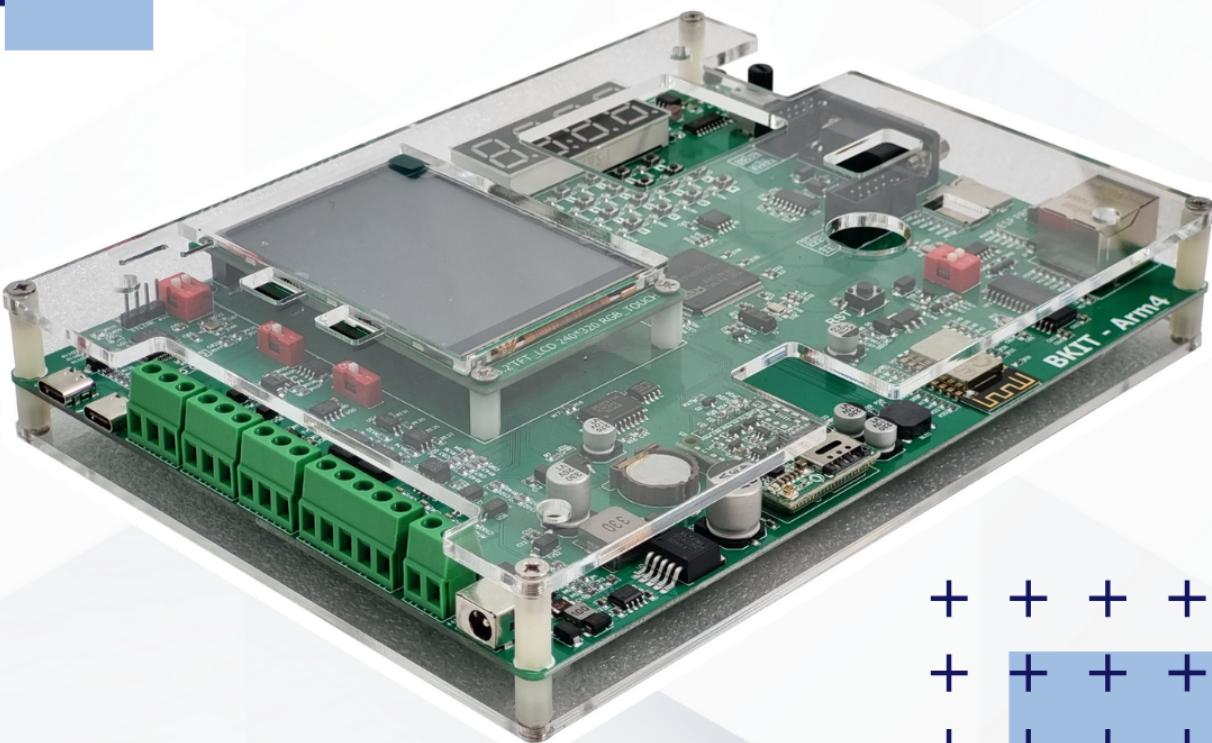


# KIT THÍ NGHIỆM BKIT

# ARM4



---

# Mục lục

---

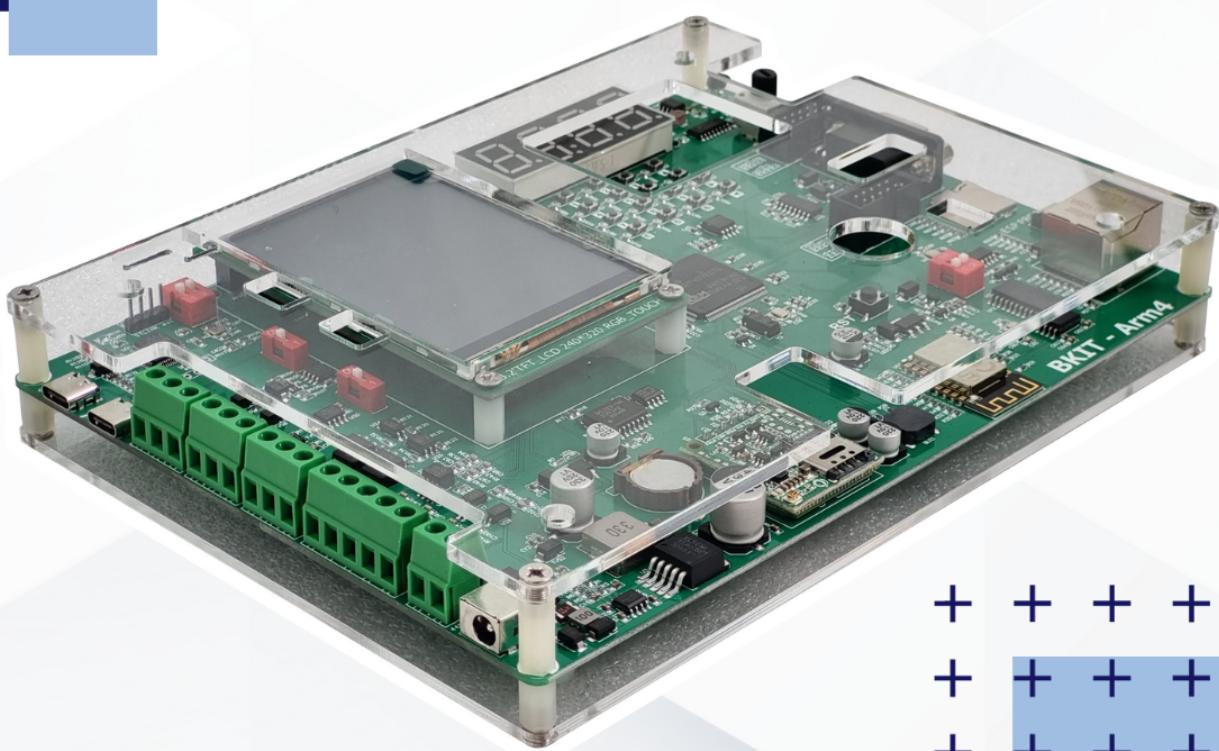
<b>Chapter 6. ADC-PWM</b>	<b>3</b>
1    Mục tiêu . . . . .	4
2    Giới thiệu . . . . .	4
3    Cơ sở lý thuyết . . . . .	5
3.1    Bộ chuyển đổi ADC . . . . .	5
3.2    Chế độ điều rộng xung - PWM . . . . .	8
4    Hướng dẫn cấu hình . . . . .	10
4.1    Cấu hình ADC: . . . . .	10
4.2    Cấu hình PWM: . . . . .	12
5    Hướng dẫn lập trình . . . . .	13
5.1    Thư viện sensor.h . . . . .	13
5.2    Thư viện buzzer.h . . . . .	14
6    Bài tập và báo cáo . . . . .	20
6.1    Bài tập 1 . . . . .	20
6.2    Bài tập 2 (Nâng cao) . . . . .	20

# CHƯƠNG 6

---

## ADC-PWM

---



# **1 Mục tiêu**

- Tìm hiểu về module ADC và ứng dụng để đọc giá trị các cảm biến trên Kit thí nghiệm.
- Tìm hiểu về module PWM và ứng dụng để điều khiển buzzer.
- Biết cách sử dụng các thư viện liên quan đến đọc cảm biến và điều khiển buzzer.

# **2 Giới thiệu**

ADC (Analog-to-Digital Converter) là một thành phần điện tử có chức năng chính là chuyển đổi tín hiệu analog (dạng liên tục) thành tín hiệu số (dạng rời rạc). ADC đóng vai trò quan trọng trong nhiều ứng dụng điện tử và hệ thống đo lường, giúp chúng ta thu thập và xử lý dữ liệu từ các cảm biến, tín hiệu điện áp, tín hiệu âm thanh và nhiều nguồn tín hiệu khác.

Tín hiệu số là loại tín hiệu được biểu thị bằng 0 hoặc 1, trong khi tín hiệu tương tự có phạm vi giá trị lớn hơn chỉ 0 hoặc 1. Cả hai đều được sử dụng trong các thiết bị điện tử xung quanh chúng ta, nhưng cách xử lý chúng khác nhau. Đối với đầu vào tương tự, chúng ta có thể thu thập dữ liệu thời gian thực từ cảm biến, sau đó sử dụng bộ chuyển đổi tương tự sang số (ADC) để chuyển đổi dữ liệu thành dạng số cho vi điều khiển. Nhưng làm thế nào nếu chúng ta muốn điều khiển thiết bị analog từ bộ vi điều khiển? Một số bộ vi điều khiển có bộ chuyển đổi kỹ thuật số sang tương tự (DAC) tích hợp, phát ra tín hiệu tương tự để điều khiển các thiết bị analog. Chúng ta cũng có thể sử dụng DAC bên ngoài. Tuy nhiên, việc sản xuất DAC khá đắt đỏ và có kích thước khá lớn. Để giải quyết những khó khăn này và thực hiện chức năng DAC một cách tiết kiệm chi phí hơn, chúng ta có thể sử dụng kỹ thuật điều xung (PWM).

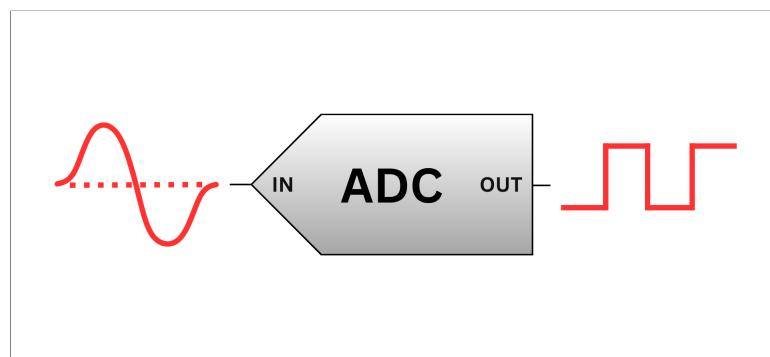
PWM (Pulse Width Modulation) là một kỹ thuật được sử dụng để điều khiển các thiết bị analog bằng cách sử dụng tín hiệu số. Kỹ thuật này có thể được sử dụng để xuất tín hiệu tương tự từ thiết bị kỹ thuật số (ví dụ như vi điều khiển).

### 3 Cơ sở lý thuyết

#### 3.1 Bộ chuyển đổi ADC

**Bộ chuyển đổi ADC hoạt động như thế nào:**

Ta có thể coi hoạt động của ADC như một bộ chia tỷ lệ toán học. Tỷ lệ cơ bản là việc chuyển đổi các giá trị từ một dải này sang một dải khác, vì vậy ADC chuyển đổi một giá trị điện áp thành một số nhị phân. Dưới đây là một số tính năng chính của ADC, và qua đó, chúng ta sẽ tìm hiểu cách nó hoạt động.



Hình 6.1: Bộ chuyển đổi ADC

**Phân loại bộ chuyển đổi ADC:**

- **Bộ chuyển đổi Flash ADC:**

- Flash ADC thường rất nhanh vì nó thực hiện nhiều so sánh đồng thời. Vì vậy, nó thích hợp cho các ứng dụng yêu cầu tốc độ cao như trong các hệ thống điều khiển hoặc xử lý tín hiệu thời gian thực.
- Độ phân giải cố định: số bit đầu ra của Flash ADC được quyết định bởi số lượng so sánh. Ví dụ, nếu có 8 so sánh, ADC sẽ có độ phân giải cố định là 8-bit.
- Tính phức tạp cao: Flash ADC yêu cầu một số lượng so sánh rất lớn, và điều này làm tăng tính phức tạp của mạch. Điều này thường dẫn đến sự gia tăng về diện tích chip và tiêu thụ điện năng so với các loại ADC khác.
- Ứng dụng chính: Flash ADC thường được sử dụng trong các ứng dụng yêu cầu tốc độ cao như radar, truyền thông số hóa, và xử lý tín hiệu thời gian thực. Dù tiêu thụ điện năng cao và chi phí tổng cộng cao, tốc độ chuyển đổi cao và độ tin cậy làm Flash ADC được sử dụng nhiều.

- **Digital Ramp ADC:**

- Digital Ramp ADC là một dạng của ADC được sử dụng để chuyển đổi tín hiệu analog thành dạng số. Nó sử dụng một nguyên tắc hoạt động đặc biệt gọi là "kỹ thuật đường dốc số".
- Độ phân giải thấp: Digital Ramp ADC thường có độ phân giải thấp hơn so với nhiều loại ADC khác. Điều này có nghĩa là chúng có khả năng chuyển đổi tín hiệu với độ chính xác tương đối thấp.
- Tốc độ chuyển đổi chậm: Do cách hoạt động, tốc độ chuyển đổi của Digital Ramp ADC thường chậm hơn so với nhiều loại ADC khác. Điều này khiến chúng không thích hợp cho các ứng dụng yêu cầu tốc độ cao.
- Ứng dụng: Digital Ramp ADC thường được sử dụng trong các ứng dụng đòi hỏi độ chính xác thấp và tốc độ chuyển đổi không quan trọng, như hệ thống đo lường tự động hoặc ứng dụng kiểm tra. Tuy nhiên, loại ADC này vẫn có hạn chế về độ phân giải và tốc độ chuyển đổi.

- **Successive Approximation Register (SAR):**

- SAR là một loại ADC chuyển đổi tín hiệu analog thành số bằng phương pháp tiếp cận tương tự, thường được sử dụng trong các ứng dụng đòi hỏi độ chính xác cao và tốc độ chuyển đổi nhanh.
- Độ phân giải: SAR hỗ trợ độ phân giải từ 8-bit đến 18-bit hoặc cao hơn, cho phép chia thành nhiều mức giá trị khác nhau. Độ phân giải càng cao, độ chính xác càng tốt, điều này quan trọng trong các ứng dụng đo lường và điều khiển yêu cầu độ chính xác cao.
- Tốc độ: SAR thường có tốc độ chuyển đổi nhanh, được đo bằng số lần chuyển đổi mỗi giây. Tuy nhiên, tốc độ này có thể giảm khi chọn độ phân giải cao hoặc xử lý nhiều kênh đầu vào cùng một lúc.
- Ứng dụng: SAR thường được sử dụng trong các ứng dụng yêu cầu độ chính xác và hiệu suất cao, như hệ thống điều khiển, thiết bị y tế, các thiết bị công nghiệp, thiết bị đo lường và các ứng dụng audio, video hoặc các ứng dụng yêu cầu tốc độ chuyển đổi cao. Tuy nhiên, SAR có hai nhược điểm chính: tiêu thụ năng lượng cao và chi phí cho độ phân giải cao, thường cần nhiều bit trong bộ đếm, điều này có thể làm tăng chi phí và diện tích vật lý của vi mạch, làm phức tạp hóa thiết kế PCB và hao phí không gian.

- **Tracking ADC:**

- Tracking ADC là một loại ADC được sử dụng để chuyển đổi tín hiệu ADC theo thời gian thực, nghĩa là nó theo dõi và ghi lại giá trị tín hiệu đầu vào

liên tục, thay vì thực hiện chuyển đổi theo cách rời rạc. Một điểm quan trọng của Tracking ADC là nó cố gắng duy trì một giá trị kỹ thuật số ứng với tín hiệu analog đầu vào trong suốt thời gian hoạt động.

- Độ phân giải: Độ phân giải của Tracking ADC xác định số bit trong dữ liệu kỹ thuật số được ghi lại bởi nó. Tracking có khả năng chuyển đổi tín hiệu analog thành dạng số với độ chính xác tốt hơn. Độ phân giải thường được đo bằng số bit, ví dụ: 12-bit, 16-bit, 18-bit, vv. Độ phân giải cao đòi hỏi nhiều tài nguyên tính toán và băng thông dữ liệu lớn hơn.
- Tốc độ: Tốc độ của Tracking ADC là tốc độ lấy mẫu và chuyển đổi tín hiệu từ dạng analog thành dạng digital. Tốc độ cao cho phép Tracking ADC theo kịp và ghi lại biến động nhanh của tín hiệu. Tuy nhiên, tốc độ cao có thể đòi hỏi nhiều công suất và băng thông dữ liệu lớn.
- Ứng dụng: Tracking ADC thường được sử dụng trong các ứng dụng yêu cầu xử lý tín hiệu thời gian thực và theo dõi biến động nhanh của tín hiệu. Các ứng dụng tiêu biểu bao gồm: Hệ thống điều khiển, Truyền thông, Xử lý tín hiệu thời gian thực, Ghi âm và video... Tuy nhiên, Tracking ADC tiêu thụ năng lượng khá lớn, dễ bị nhiễu, có chi phí và kích thước khá lớn.

- **Slope (integrating) ADC:**

- Giống như Digital Ramp ADC, Slope cũng tạo ra sóng răng cưa. Tuy nhiên, Slope sử dụng một mạch opamp (gọi là bộ tích phân).
- Ưu điểm của Slope ADC bao gồm độ phân giải cao, độ chính xác tốt và không yêu cầu mạch tỷ lệ phân đoạn để chuyển đổi tín hiệu thành dạng số. Tuy nhiên, Slope ADC cần thời gian lấy mẫu đủ dài để chuyển đổi, và tốc độ chuyển đổi thấp hơn so với một số loại ADC khác. Slope ADC thường được sử dụng trong các ứng dụng yêu cầu độ chính xác cao, chẳng hạn như trong các hệ thống đo lường và kiểm tra. Tuy nhiên độ chính xác này sẽ giảm dần theo thời gian do hiện tượng nạp xả tụ.

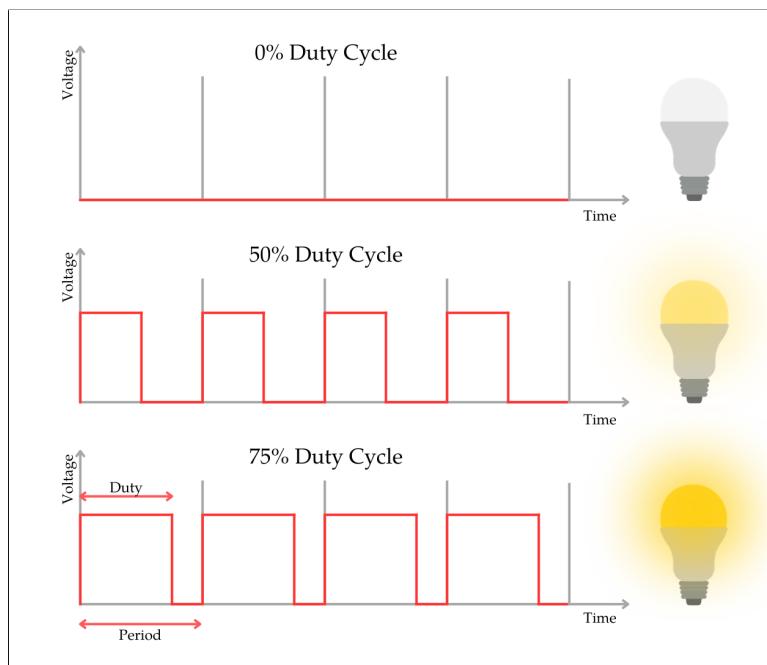
- **Một số loại ADC phức tạp khác:**

- Delta ADC
- Sigma-delta ADC
- Pipeline
- Time-interleaved
- ...

## 3.2 Chế độ điều rộnぐ xung - PWM

Điều chỉnh độ rộnぐ xung (PWM) (Pulse Width Modulation) áp dụng tín hiệu số để kiểm soát các ứng dụng nguồn, và có khả năng chuyển đổi trở lại tín hiệu tương tự mà không cần nhiều phần cứng. Các hệ thống analog thường tạo ra nhiệt do dòng điện chảy qua các điện trở trong hệ thống lớn, trong khi hệ thống số không tạo ra nhiệt độ như vậy.

- Duty Cycle: là một tham số quan trọng trong PWM và đo lường tỷ lệ thời gian mà tín hiệu PWM ở mức cao so với tổng thời gian chu kỳ. Nó thường được biểu thị dưới dạng phần trăm. Ví dụ, nếu chu kỳ là 1ms và duty cycle là 50%, thì tín hiệu PWM sẽ ở mức cao trong 0,5ms và mức thấp trong 0,5ms.



Hình 6.2: Ví dụ về Duty Cycle

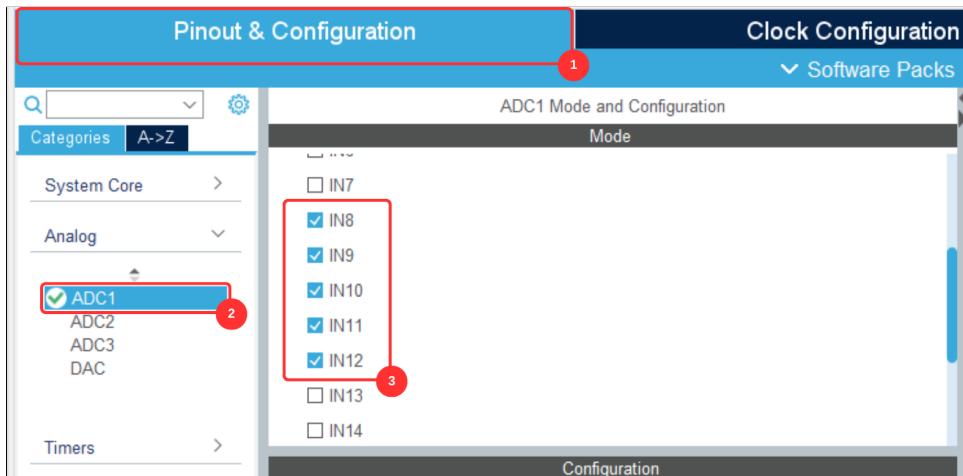
- Mạch cầu H: là một mạch điện tử thường được sử dụng để điều khiển động cơ DC hoặc tải điện áp cao. Nó bao gồm bốn công tắc điện tử (thường là transistor hoặc MOSFET) được kết hợp thành một cấu trúc dạng hình chữ "H". Bằng cách kiểm soát các công tắc, bạn có thể thay đổi hướng và tốc độ quay của động cơ hoặc điều khiển công suất đầu ra của tải.
- Switching Mode Power Supplies (SMPS): là một cách chuyển đổi hiệu suất cao sử dụng PWM để cung cấp năng lượng điện cho các thiết bị điện tử và hệ thống. SMPS thường sử dụng một cấu trúc chuyển đổi đặc biệt để biến đổi nguồn điện đầu vào thành nguồn điện đầu ra có điện áp và dòng điện

ổn định. Quá trình này tiết kiệm năng lượng và thường có hiệu suất cao hơn so với các nguồn cung cấp dựa trên cấu trúc tuyến tính. PWM được sử dụng trong SMPS để kiểm soát đầu ra và duy trì điện áp hoặc dòng điện ổn định bằng cách điều chỉnh duty cycle của tín hiệu PWM.

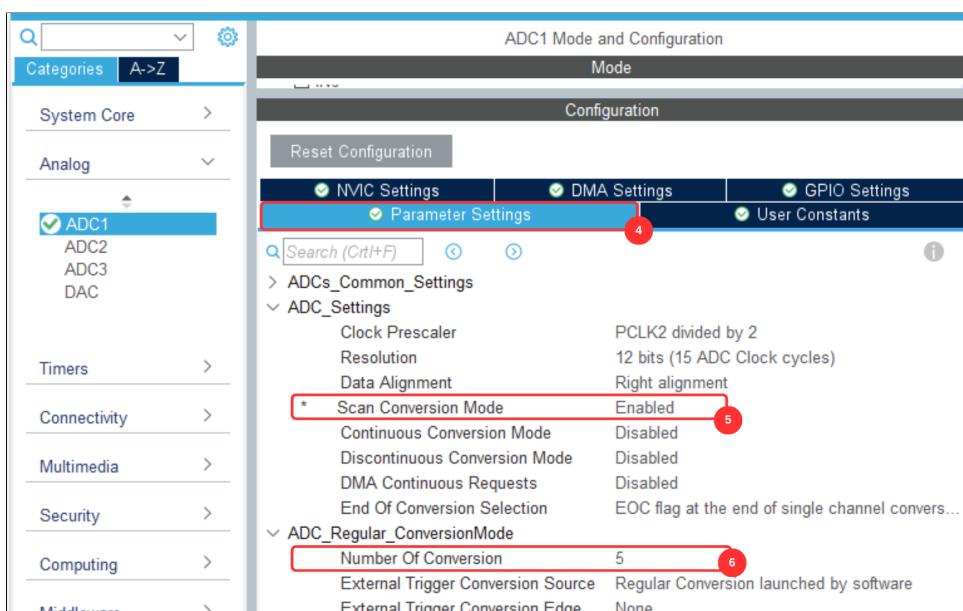
## 4 Hướng dẫn cấu hình

### 4.1 Cấu hình ADC:

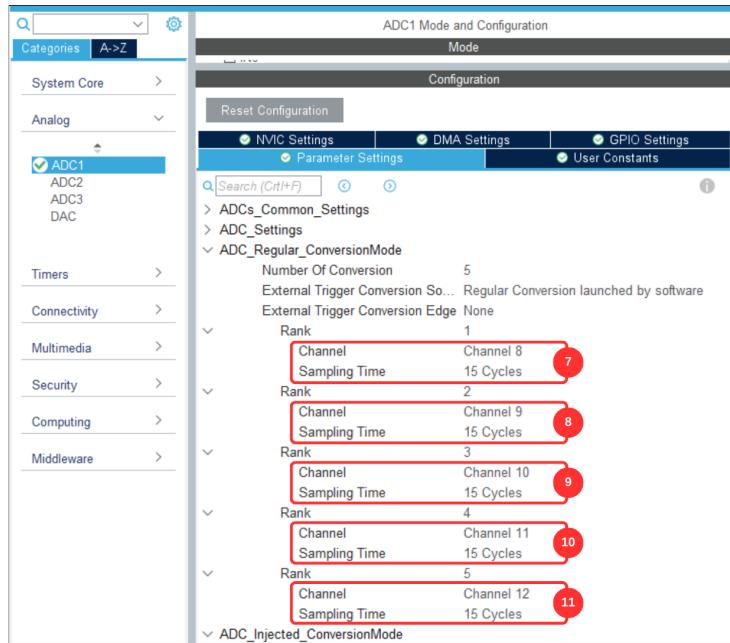
Trên Kit thí nghiệm, chúng ta sẽ giao tiếp với các ngoại vi thông qua ADC1 trên MCU. Trong file .ioc chúng ta sẽ config ADC1 như sau:



Hình 6.3: Config các kênh ADC

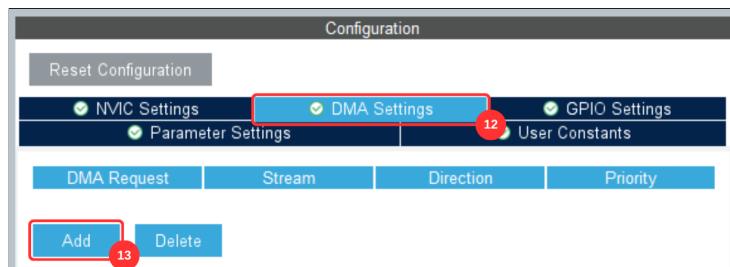


Hình 6.4: Config ADC

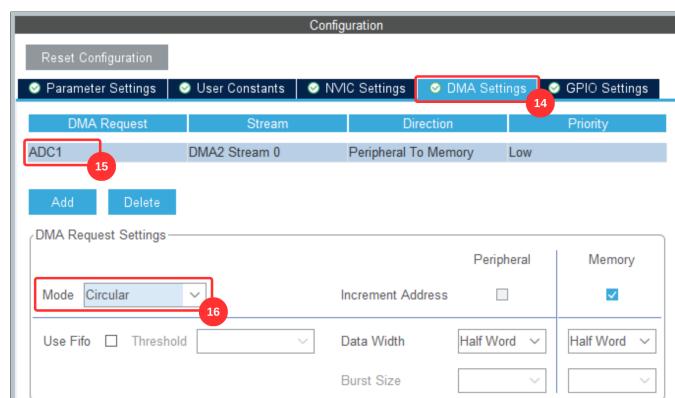


Hình 6.5: Config ADC

Trong thiết kế của Kit thí nghiệm, để có thể sử dụng đầy đủ các cảm biến trên bo mạch, ta cần phải sử dụng nhiều CHANNEL khác nhau của 1 module ADC. Để làm được việc này, ta bắt buộc phải sử dụng cơ chế DMA để hỗ trợ. Đây là cơ chế giúp trao đổi dữ liệu giữa bộ nhớ và ngoại vi mà không cần thông qua CPU.

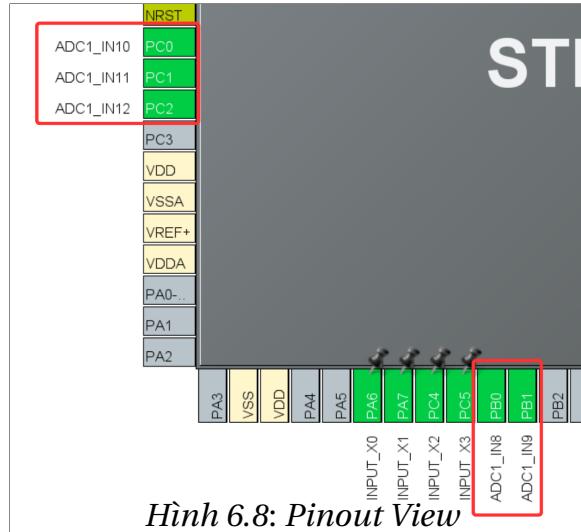


Hình 6.6: Config DMA



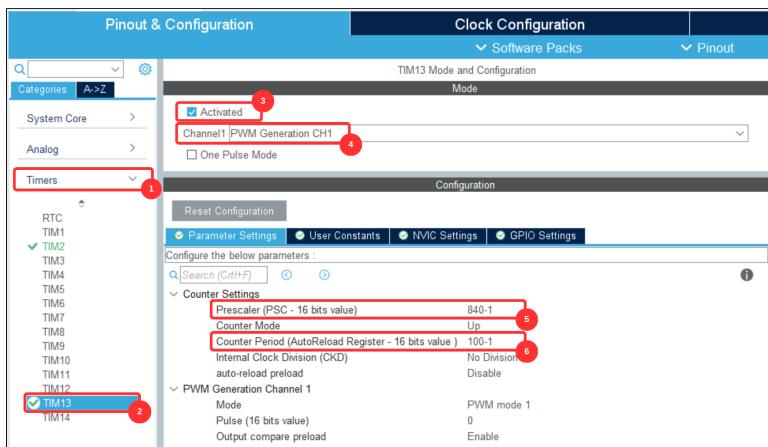
Hình 6.7: Config DMA

Sau khi cấu hình, ta có Pinout view như sau:



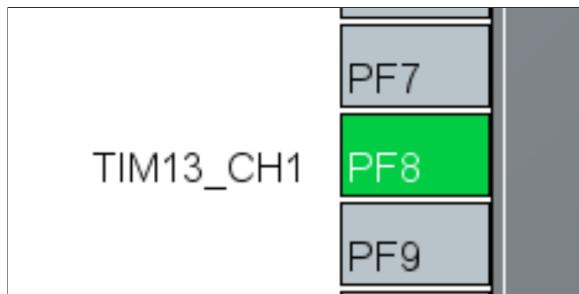
## 4.2 Cấu hình PWM:

Ta sẽ điều khiển buzzer bằng một xung có tần số 1000Hz. Dựa theo thiết kế của Kit thí nghiệm, buzzer sẽ được điều khiển bằng chân **CHANNEL1** của **TIM13**. Vậy nên ta sẽ cần config TIM13 với tần số 1000Hz. Việc config này đã được giải thích ở **LAB 2**. Với cách config dưới, duty\_cycle tối đa của xung PWM sẽ tương ứng với giá trị của **Counter Period**. Trong quá trình lập trình, để thay đổi duty\_cycle ta chỉ cần thay đổi giá trị tham số trong khoảng 0-99.



Hình 6.9: Config PWM

Sau khi cấu hình, ta có Pinout view như sau:



Hình 6.10: Pinout View

## 5 Hướng dẫn lập trình

### 5.1 Thư viện sensor.h

Các file sensor.h và sensor.c có thể được sao chép từ project mẫu **Bai6\_ADC\_PWM**.

#### **void sensor\_init()**

- **Mô tả:** Khởi tạo các cảm biến.
- **Tham số:** Không có.
- **Giá trị trả về:** Không có.

#### **void sensor\_Read()**

- **Mô tả:** Đọc và lưu giá trị các cảm biến từ module ADC.
- **Tham số:** Không có.
- **Giá trị trả về:** Không có.

#### **uint16\_t sensor\_GetLight()**

- **Mô tả:** Đọc giá trị cảm biến ánh sáng.
- **Tham số:** Không có.
- **Giá trị trả về:** Giá trị trả về từ cảm biến ánh sáng (0-4095).

#### **uint16\_t sensor\_GetPotentiometer()**

- **Mô tả:** Đọc giá trị biến trổ.
- **Tham số:** Không có.

- **Giá trị trả về:** Giá trị trả về từ biến trỏ (0-4095).

#### **uint16\_t sensor\_GetVoltage()**

- **Mô tả:** Đọc giá trị điện áp được cấp vào Kit thí nghiệm.
- **Tham số:** Không có.
- **Giá trị trả về:** Điện áp được cấp vào Kit thí nghiệm (V).

#### **uint16\_t sensor\_GetCurrent()**

- **Mô tả:** Đọc giá trị dòng điện tiêu thụ của Kit thí nghiệm.
- **Tham số:** Không có.
- **Giá trị trả về:** Dòng điện tiêu thụ (mA).

#### **uint16\_t sensor\_GetTemperature()**

- **Mô tả:** Đọc giá trị cảm biến nhiệt độ.
- **Tham số:** Không có.
- **Giá trị trả về:** Nhiệt độ cảm biến đo được (°C).

## **5.2 Thư viện buzzer.h**

Các file buzzer.h và buzzer.c có thể được sao chép từ project mẫu **Bai6\_ADC\_PWM**.

#### **void buzzer\_init()**

- **Mô tả:** Khởi tạo buzzer.
- **Tham số:** Không có.
- **Giá trị trả về:** Không có.

#### **void buzzer\_SetVolume(uint8\_t dutyCycle)**

- **Mô tả:** Thay đổi âm lượng buzzer bằng độ rộng xung.
- **Tham số:**
  - **dutyCycle:** Giá trị độ rộng xung (0-99).

- **Giá trị trả về:** Không có.

Dưới đây là chương trình mẫu giúp in các giá trị đọc được từ cảm biến và dùng 3 nút nhấn để điều khiển buzzer. Thông thường với giá trị duty cycle = 50%, buzzer sẽ kêu với cường độ lớn nhất. Chương trình mẫu sẽ điều khiển duty cycle = 50% nếu nhấn nút "mũi tên lên", duty cycle = 0% nếu nhấn nút "mũi tên xuống", duty cycle = 25% nếu nhấn nút "mũi tên phải".

```

1 // ...
2 /* USER CODE BEGIN Includes */
3 #include "software_timer.h"
4 #include "led_7seg.h"
5 #include "button.h"
6 #include "lcd.h"
7 #include "picture.h"
8 #include "ds3231.h"
9 #include "sensor.h"
10 #include "buzzer.h"
11 /* USER CODE END Includes */
12
13 // ...
14
15 /* USER CODE BEGIN PFP */
16 void system_init();
17 void test_LedDebug();
18 void test_Buzzer();
19 void test_AdC();
20 /* USER CODE END PFP */
21
22 // ...
23
24 int main(void)
25 {
26
27     // ...
28
29     /* USER CODE BEGIN 2 */
30     system_init();
31     /* USER CODE END 2 */
32
33     /* Infinite loop */

```

```

34  /* USER CODE BEGIN WHILE */
35  lcd_Clear(BLACK);
36  while (1)
37  {
38      while(!flag_timer2);
39      flag_timer2 = 0;
40      button_Scan();
41      test_LedDebug();
42      test_Adc();
43      test_Buzzer();
44  /* USER CODE END WHILE */

45
46  /* USER CODE BEGIN 3 */
47 }
48 /* USER CODE END 3 */
49 }

50
51 // ...
52
53 /* USER CODE BEGIN 4 */
54 void system_init(){
55     timer_init();
56     button_init();
57     lcd_init();
58     sensor_init();
59     buzzer_init();
60     setTimer2(50);
61 }
62
63 uint8_t count_led_debug = 0;
64
65 void test_LedDebug(){
66     count_led_debug = (count_led_debug + 1)%20;
67     if(count_led_debug == 0){
68         HAL_GPIO_TogglePin(DEBUG_LED_GPIO_Port , DEBUG_LED_Pin);
69     }
70 }
71
72 uint8_t isButtonUp()

```

```

73 {
74     if (button_count[3] == 1)
75         return 1;
76     else
77         return 0;
78 }
79
80 uint8_t isButtonDown()
81 {
82     if (button_count[7] == 1)
83         return 1;
84     else
85         return 0;
86 }
87
88 uint8_t isButtonRight()
89 {
90     if (button_count[11] == 1)
91         return 1;
92     else
93         return 0;
94 }
95
96 uint8_t count_adc = 0;
97
98 void test_Adc(){
99     count_adc = (count_adc + 1)%20;
100    if(count_adc == 0){
101        sensor_Read();
102        lcd_ShowStr(10, 100, "Voltage:", RED, BLACK, 16, 0);
103        lcd_ShowFloatNum(130, 100,sensor_GetVoltage(), 4, RED,
104                         BLACK, 16);
105        lcd_ShowStr(10, 120, "Current:", RED, BLACK, 16, 0);
106        lcd_ShowFloatNum(130, 120,sensor_GetCurrent(), 4, RED,
107                         BLACK, 16);
108        lcd_ShowStr(10, 140, "Light:", RED, BLACK, 16, 0);
109        lcd_ShowIntNum(130, 140, sensor_GetLight(), 4, RED,
110                         BLACK, 16);
111        lcd_ShowStr(10, 160, "Potentiometer:", RED, BLACK, 16,
112

```

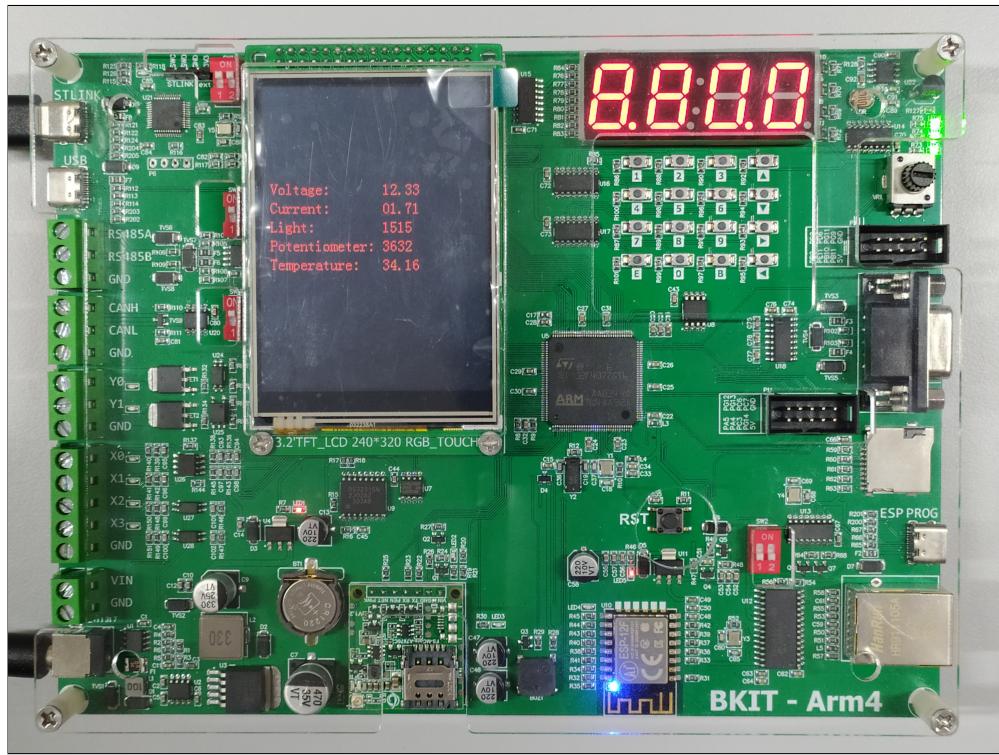
```

109     0) ;
110     lcd_ShowIntNum(130, 160, sensor_GetPotentiometer(), 4,
111     RED, BLACK, 16);
112     lcd_ShowStr(10, 180, "Temperature:", RED, BLACK, 16, 0)
113     ;
114     lcd_ShowFloatNum(130, 180, sensor_GetTemperature(), 4,
115     RED, BLACK, 16);
116 }
117 }
118 }
119
120 void test_Buzzer(){
121     if(isButtonUp()){
122         buzzer_SetVolume(50);
123     }
124     if(isButtonDown()){
125         buzzer_SetVolume(0);
126     }
127 }
128 }
129 /* USER CODE END 4 */

```

Program 6.1: Chương trình mẫu đọc cảm biến và điều khiển buzzer.

Hình ảnh kết quả trên Kit thí nghiệm:



Hình 6.11: Kết quả hiện thực trên Kit thí nghiệm

## **6 Bài tập và báo cáo**

### **6.1 Bài tập 1**

Giả lập hệ thống quan trắc môi trường. Hiển thị các thông số đo được lên màn hình gồm:

- Công suất điện tiêu thụ (mW).
- Ánh sáng (Strong hoặc Weak).
- Nhiệt độ ( $^{\circ}\text{C}$ ).
- Độ ẩm (%), được giả lập bằng biến trỏ, giá từ từ 0-100% tương ứng với khoảng giá trị ADC từ nhỏ nhất tới lớn nhất đọc được từ biến trỏ).
- Led đồng hồ sẽ hiển thị thời gian của hệ thống.

Các dữ liệu đo được sẽ được gửi lên máy tính thông qua Uart-RS232. Khi độ ẩm vượt ngưỡng ( $> 70\%$ ) cho phép hệ thống sau vào trạng thái báo động:

- Buzzer sẽ báo động mỗi 1s.
- Thông báo sẽ được gửi liên tục đến máy tính mỗi 1s.

### **6.2 Bài tập 2 (Nâng cao)**

Trên màn hình LCD, vẽ biểu đồ đường hiển thị công suất tiêu thụ theo thời gian. Giả sử chu kì lấy mẫu là 15s.