STM32 ???? ? 5 ??ADC ???????????? Channel?

? ????

1. 理解 **ADC** 工作原理 - 掌握模數轉換的概念
2. 學會配置不同轉換模式 - 單次、掃描、連續轉換
3. 實現多通道採樣 - 透過 UART 輸出感測器數值

? ADC ????

ADC ????

ADC (Analog-to-Digital Converter) 將類比電壓信號轉換為數位值。 STM32F446 內建 **3** 個 **ADC**, 共 **16** 個通道 。

STM32F446 ADC ??

特性	規格
分辨率	12 位（ 0-4095）
轉換時間	~3.5 微秒
採樣率	最高 2.4 MSPS (百萬次 /秒)
通道數	19 個（ 16 外部 + 3 內部）
參考電壓	3.3V（ VREF+）

ADC ????

通道	**GPIO** 接腳	用途
ADC1_IN0	PA0	外部感測器輸入 1
ADC1_IN1	PA1	外部感測器輸入 2

通道	**GPIO** 接腳	用途
ADC1_IN2	PA2	外部感測器輸入 3
ADC1_IN3	PA3	外部感測器輸入 4
ADC1_IN16	-	溫度感測器（內部）
ADC1_IN17	-	VREF+ 參考電壓
ADC1_IN18	-	VBAT（電池電壓）

??????????

```
類比電壓 (V)            數位值 (12-bit)
      ↑                       ↑
3.3V ├──────────────────  4095
2.5V ├──────────────────  3077 (約 2.5V/3.3V * 4095)
1.65V├──────────────────  2048
0.0V ├──────────────────  0
      └──────────────────   └────────
```

計算公式:

```
ADC_Value = (V_input / V_ref) × 2^Resolution - 1
          = (V_input / 3.3) × 4095


反推:
V_input = (ADC_Value / 4095) × 3.3
```

ADC ????

模式	說明	應用
Single	單次轉換一個通道	偶發採樣
Continuous	連續自動轉換	實時監測
Scan	依序轉換多個通道	多感測器
DMA	直接記憶體存取	高速大量採樣

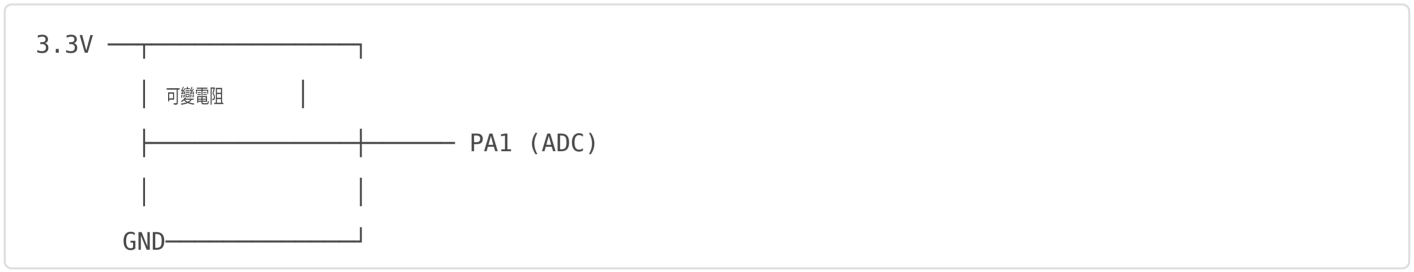
?? ????

????

元件	數量	說明
可變電阻（ 10kΩ）	1	提供類比電壓輸入
鱷魚夾	3	接線用
跳線	適量	麵包板接線

???

組件	接腳	說明
可變電阻中心腳	PA1 (ADC1_IN1)	類比輸入
可變電阻一端	3.3V	電源
可變電阻另一端	GND	地線



?? CubeMX ????

?? 1??? ADC1

1. 開啟 CubeMX
2. 點擊 **Analog → ADC1**
3. 在 Pinout 圖中選擇 **PA1** 配置為 **ADC1_IN1**

?? 2??? ADC ??

1. 在左側選擇 **Analog → ADC1**
2. 在 **Parameter Settings** 中設定:
   - **Resolution**: 12 Bits
   - **Data Alignment**: Right Alignment
   - **Scan Conversion Mode**: Disable (單通道)
   - **Continuous Conversion Mode**: Enable (連續轉換)

- **EOC Selection**: End of conversion flag

3. 在 **Rank** 區域添加通道:
   - 點擊 **Add** 或 **Rank 1**
   - 選擇 **Channel**: ADC_CHANNEL_1
   - **Sampling Time**: 144 Cycles (較穩定)

?? 3??? ADC ??

1. 點擊 **NVIC Settings** 標籤
2. 勾選 **ADC1 global interrupt**
3. 設定優先級:
   - **Preemption Priority**: 2
   - **Sub Priority**: 0

?? 4?????????????

若要定時採樣, 可配置 Timer 觸發 ADC:

1. 選擇 **Timers → TIM2**
2. 設定 **Trigger Output Event**: Update Event
3. 返回 ADC1, 設置:
   - **Trigger**: Timer2 Trigger Out Event
   - **External Trigger Conversion Edge**: Rising Edge

?? 5??????

點擊 **Generate Code**

? ?????

main.c - ADC ??????????

```
/* STM32 Lesson 05 - ADC Continuous Sampling
 * 功能: 連續採樣類比輸入, 透過 UART 輸出結果
 * 難度: 中級
 */

#include "main.h"
#include "adc.h"
```

```c
#include "usart.h"
#include "gpio.h"
#include <stdio.h>
#include <stdarg.h>

/* 私有定義 */
#define ADC_SAMPLE_SIZE 10  /* 每次平均取樣數 */

/* 函數宣告 */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART1_UART_Init(void);
void UART_Print(const char *format, ...);
uint16_t ADC_Get_Average(void);
float ADC_To_Voltage(uint16_t adc_value);

/* 全域變數 */
ADC_HandleTypeDef hadc1;
UART_HandleTypeDef huart1;
volatile uint16_t adc_value = 0;
volatile uint8_t adc_conversion_complete = 0;

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  MX_ADC1_Init();

  UART_Print("\r\n======= ADC Sampling Test =======\r\n");
  UART_Print("ADC Value | Voltage (V)\r\n");
  UART_Print("==================================\r\n");

  /* 啟動 ADC 連續轉換 */
  HAL_ADC_Start_IT(&hadc1);

  while (1)
  {
```

```
    if (adc_conversion_complete)
    {
      /* 讀取 ADC 值 */
      uint16_t adc_raw = HAL_ADC_GetValue(&hadc1);

      /* 轉換為電壓 */
      float voltage = ADC_To_Voltage(adc_raw);

      /* 透過 UART 輸出 */
      UART_Print("%-9d | %.2f\r\n", adc_raw, voltage);

      adc_conversion_complete = 0;

      HAL_Delay(100);  /* 每 100ms 輸出一次 */
    }
  }
}

/**
  * @brief 將 ADC 原始值轉換為電壓
  * @param adc_value: 12-bit ADC 原始值 (0-4095)
  * @return 電壓值 (V)
  */
float ADC_To_Voltage(uint16_t adc_value)
{
  /* 參考電壓 3.3V, 12-bit 分辨率 4095 */
  return (adc_value / 4095.0f) * 3.3f;
}

/**
  * @brief 獲取多次採樣的平均值
  */
uint16_t ADC_Get_Average(void)
{
  uint32_t sum = 0;

  for (uint8_t i = 0; i < ADC_SAMPLE_SIZE; i++)
  {
    sum += HAL_ADC_GetValue(&hadc1);
    HAL_Delay(1);
```

```
  }

  return sum / ADC_SAMPLE_SIZE;
}

/**
  * @brief ADC 轉換完成中斷回調
  */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if (hadc->Instance == ADC1)
  {
    adc_conversion_complete = 1;
  }
}

/**
  * @brief UART 列印函數
  */
void UART_Print(const char *format, ...)
{
  char buffer[100];
  va_list args;
  va_start(args, format);
  vsnprintf(buffer, sizeof(buffer), format, args);
  va_end(args);

  HAL_UART_Transmit(&huart1, (uint8_t *)buffer, strlen(buffer), HAL_MAX_DELAY);
}

/**
  * @brief 系統時脈配置
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
```

```
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = 2;
  RCC_OscInitStruct.PLL.PLLQ = 7;

  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
                              | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief GPIO □□□
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  __HAL_RCC_GPIOA_CLK_ENABLE();

  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}

/**
  * @brief ADC1 □□□
  */
static void MX_ADC1_Init(void)
{
  ADC_ChannelConfTypeDef sConfig = {0};

  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc1.Init.LowPowerAutoWait = DISABLE;
  hadc1.Init.LowPowerAutoPowerOff = DISABLE;

  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }

  sConfig.Channel = ADC_CHANNEL_1;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_144CYCLES;

  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
}
```

```
/**
  * @brief ADC MSP □□□
  */
void HAL_ADC_MspInit(ADC_HandleTypeDef* hadc)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  if(hadc->Instance == ADC1)
  {
    __HAL_RCC_ADC1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    GPIO_InitStruct.Pin = GPIO_PIN_1;
    GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    HAL_NVIC_SetPriority(ADC_IRQn, 2, 0);
    HAL_NVIC_EnableIRQ(ADC_IRQn);
  }
}

/**
  * @brief USART1 □□□
  */
static void MX_USART1_UART_Init(void)
{
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;

  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
```

```
}

/**
  * @brief UART MSP □□□
  */
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  if(huart->Instance == USART1)
  {
    __HAL_RCC_USART1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    GPIO_InitStruct.Pin = GPIO_PIN_9 | GPIO_PIN_10;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    HAL_NVIC_SetPriority(USART1_IRQn, 1, 0);
    HAL_NVIC_EnableIRQ(USART1_IRQn);
  }
}

/**
  * @brief ADC □□□□
  */
void ADC_IRQHandler(void)
{
  HAL_ADC_IRQHandler(&hadc1);
}

/**
  * @brief USART1 □□□□
  */
void USART1_IRQHandler(void)
{
  HAL_UART_IRQHandler(&huart1);
```

```
}

void Error_Handler(void)
{
  while(1);
}
```

? ?????

????

 ADC 輸出示例:

```
======= ADC Sampling Test =======
ADC Value | Voltage (V)
==================================
2048      | 1.65
3072      | 2.47
1024      | 0.82
4095      | 3.30
```

轉動可變電阻， ADC 值應隨之變化，電壓範圍應為 0.0V ~ 3.3V

????

問題	原因	解決方案
ADC 值不變	未啟動連續轉換	確認 `HAL_ADC_Start_IT()` 已調用
數值抖動大	採樣時間過短或雜訊干擾	增加 `SamplingTime` 至 144 Cycles
電壓計算不准	參考電壓設定錯誤	確認公式中 V_ref 為 3.3V
無 **UART** 輸出	UART 未初始化	檢查 USART1 初始化函數

? ????

???????

```
/* CubeMX 設定 */
hadc1.Init.ScanConvMode = ENABLE;  /* 啟用掃描模式 */
hadc1.Init.NbrOfConversion = 3;    /* 3 個通道 */

/* 配置多個通道 */
sConfig.Channel = ADC_CHANNEL_0;
sConfig.Rank = 1;
HAL_ADC_ConfigChannel(&hadc1, &sConfig);

sConfig.Channel = ADC_CHANNEL_1;
sConfig.Rank = 2;
HAL_ADC_ConfigChannel(&hadc1, &sConfig);

sConfig.Channel = ADC_CHANNEL_2;
sConfig.Rank = 3;
HAL_ADC_ConfigChannel(&hadc1, &sConfig);
```

?? DMA ??????

後續課程（第 7 節 DMA）將深入講解透過 DMA 進行高效率的 ADC 採樣

? ????

???? - ? 6 ????? Timer + PWM

第 6 節將實現:

- 定時器基本配置與中斷
- **PWM** 信號產生
- 呼吸燈效果實現

□ **ADC** □□□□□□□□

Revision #2
Created 2026-04-01 02:06:17 UTC by TaipeiTechRacing
Updated 2026-04-06 06:23:30 UTC