

Arduino 教學手冊

Tai, Wei-Hsuan

May 6, 2025

Contents

1	Arduino 介紹	3
2	電子學的基本概念	3
2.1	電壓 (Voltage)	3
2.2	電流 (Current)	3
2.3	電阻 (Resistance)	3
2.4	歐姆定律 (Ohm's Law)	3
3	如何透過 Arduino 控制元件	3
3.1	Arduino IDE 的程式結構	4
3.2	Arduino 的輸出	4
3.3	控制腳位	5
4	Appendix- C++ 語法介紹	6
4.1	變數的種類與介紹	6
4.2	if-else 語法	6
4.3	for loop	7
4.4	while loop	7

1 Arduino 介紹

平時你在寫程式時，會使用電腦的 CPU 來執程式碼，而 Arduino 的核心就是一顆微處理器，這顆微處理器可以執行你寫的程式碼，並且控制外部的電子元件，例如 LED 燈、馬達、感測器等等。

Arduino 的開發環境是 Arduino IDE，這是一個可以讓你寫程式、編譯程式、上傳程式到 Arduino 開發板的軟體，我們已經幫你們安裝好了，但如果你在家裡想要嘗試，可以去 <https://www.arduino.cc/en/software> 下載最新版本的 Arduino IDE。

2 電子學的基本概念

2.1 電壓 (Voltage)

電壓是電流的驅動力，單位是伏特 (Volt)，簡稱 V。電阻相同時，電壓越高，電流越大。

2.2 電流 (Current)

電流是電荷的流動，單位是安培 (A)，簡稱 I。電流越大，代表單位時間內通過的電荷量越多。

2.3 電阻 (Resistance)

電阻是對電流流動的阻礙，單位是歐姆 (Ω)，簡稱 R。在相同電壓下電阻越大，電流越小。

2.4 歐姆定律 (Ohm's Law)

電壓、電流和電阻之間關係，可以使用歐姆定律來表示，公式為：

$$V = I \cdot R$$

這個公式告訴我們，當電壓不變時，電流和電阻成反比；當電阻不變時，電流和電壓成正比。

如果用水管來比喻，電壓就是水壓，電流就是水流的速度，電阻就是水管的阻塞程度，水壓越大，水流的速度越快；水管阻塞程度越大，水壓就越大。

3 如何透過 Arduino 控制元件

當 Arduino 的腳位輸出電壓，並透過電路接到元件（例如 LED 或馬達）時，元件會因為電壓差而產生電流。這個電流流經元件內部，讓它發光、發聲或轉動。這就是 Arduino 控

制元件的基本原理。這一部份的內容會教你如何使用程式控制 Arduino 的腳位輸出電壓，並透過電路接到元件。

3.1 Arduino IDE 的程式結構

Arduino IDE 的程式結構主要由三個部分組成：設定區、主程式區和函式區。設定區用於初始化變數和設定腳位，主程式區則是執行的主要邏輯，而函式區則是用來定義可重複使用的函式。

以下的程式碼是你一打開 Arduino IDE 就會看到的範本程式碼：

```
1 void setup() {  
2   // put your setup code here, to run once:  
3 }  
4 void loop() {  
5   // put your main code here, to run repeatedly:  
6 }
```

其中，`setup()` 函式是用來初始化變數和設定腳位的，這個函式只有在每次重新啟動時會執行一次，而 `loop()` 函式則是用來執行主要邏輯的，這個函式會不斷重複執行，直到電源關閉或 Arduino 被重置。

一般而言，我們會在 `setup()` 函式中設定腳位的模式，例如輸入或輸出，然後在 `loop()` 函式中執行主要邏輯，例如讀取感測器的數據或控制元件的狀態。

你可以觀察到 Arduino IDE 的程式碼結構一樣是使用無數個函式來組成的，因此全域變數、自訂函式、類別等概念在 Arduino IDE 中也一樣適用，但請記得考量到 Arduino 開發板的效能，盡量避免使用過多的全域變數與類別，這樣會影響程式的執行速度。

3.2 Arduino 的輸出

平時使用電腦編譯 C++ 程式時，會使用 `cout` 來輸出結果，但在使用 Arduino 時，執行程式的核心不是電腦的 CPU，而是你手上那一塊 Arduino 開發板，所以我們不能使用 `cout` 來輸出結果，因為 Arduino 開發板並沒有螢幕可以顯示結果。而是使用 `Serial` 來輸出結果，這樣 Arduino 開發板就可以透過 USB 線將結果傳送到電腦上，然後電腦就可以使用 `Serial Monitor` 來顯示結果。

這樣的作法等同於僅借用點腦的螢幕來顯示結果，電腦本身不參與運算。

```
1 使用Serial輸出結果  
2 Serial.begin(9600); //初始化Serial，9600是傳輸速率  
3 Serial.println("Hello World!"); //輸出Hello World!並且換行  
4 Serial.print("Hello World!"); //輸出Hello World!不換行  
5 Serial.print(123); //輸出123  
6 Serial.print(a); //輸出變數a的值  
7 如果要串接變數與字串，可以這樣寫：  
8 Serial.print("a的值是" + String(a)); //輸出a的值
```

3.3 控制腳位

假設你今天要讓一個 LED 燈亮起來，該怎麼做呢？你需要一個正極和一個負極，負極可以直接連接到開發板上的 GND 腳位，而正極的操作就多了，開發板上有現成的 5V 腳位可以使用，但這個腳位是持續輸出電壓的，如果你想要控制他閃爍，就必須控制腳位的電壓輸出，這時候就需要用到 `digitalWrite()` 這個函式了。

```
1 使用digitalWrite控制腳位
2
3 請記得先使用pinMode設定腳位的模式
4 void setup() {
5     pinMode(13, OUTPUT); //設定腳位13為輸出模式
6 }
7
8 在loop()中做你想做的事情，例如以下程式碼會讓腳位13的電壓不斷地閃爍
9 void loop() {
10     digitalWrite(13, HIGH); //將腳位13的電壓設為5V
11     delay(1000); //延遲1秒（1000毫秒）
12     digitalWrite(13, LOW); //將腳位13的電壓設為0V
13     delay(1000); //延遲1秒
14     其中，HIGH也可以用1來表示，LOW也可以用0來表示，
15     所以這兩行程式碼也可以寫成：
16     digitalWrite(13, 1); //將腳位13的電壓設為5V
17     delay(1000); //延遲1秒
18     digitalWrite(13, 0); //將腳位13的電壓設為0V
19     delay(1000); //延遲1秒
20 }
```

4 Appendix- C++ 語法介紹

4.1 變數的種類與介紹

類型	大小 (位元組)	範圍	用途
int	4	-2,147,483,648 to 2,147,483,647	整數運算
double	8	$\pm 2.3\text{E}-308$ to $\pm 1.7\text{E}+308$	浮點數運算
char	1	-128 to 127 或 0 to 255	儲存單一字元

Table 1: C++ 常用的三種變數

雖然 C++ 有以上三種常用變數，但在 Arduino 中，我們比較常使用的是 int 與 double。要注意的是，由於 Arduino 的運算速度較慢，使用 double 會影響運算速度。

以下是操縱變數的範例程式碼：

```

1 宣告變數
2  int a=1, b=2;
3  double pi=3.14;
4
5  變數運算
6  int sum=a+b; // 計算a+b
7  int mod=a%b; // 計算a除以b的餘數
8
9  更改變數的值
10 a=3; // 將變數a的值改為3
11 b+=a; // 將變數b的值加上變數a的值
12
13 以下的寫法都是把變數a的值加1
14 a=a+1;
15 a+=1;
16 a++;
17 ++a;

```

4.2 if-else 語法

我們難免會遇到一些情況，必須根據不同的條件來執行不同的程式碼。這時候就需要用到 if-else 語法。

```

1  if-else 範例程式碼
2  if(a>b){
3      // 當a大於b時執行的程式碼
4  }else{
5      // 當a不大於b時執行的程式碼
6  }

```

在判別時，邏輯運算子也會派上用場，以下是常見的邏輯運算子：

運算子	描述	範例
&&(AND)	當兩個條件都為真時，結果為真	(a > b) && (b > c)
(OR)	當至少一個條件為真時，結果為真	(a > b) (b > c)

Table 2: C++ 邏輯運算子

```

1 邏輯運算子範例程式碼
2  if(a>b && b>c){
3     //當a大於b且b大於c時執行的程式碼
4 }else if(a>b || b>c){
5     //當a大於b或b大於c時執行的程式碼
6 }else{
7     //當a不大於b且b不大於c時執行的程式碼
8 }
```

4.3 for loop

如果要重複很多次執行同樣的程式碼，使用 for loop 會比重複寫一樣的程式碼來得簡單許多。for loop 的語法如下：

```

1 for loop 語法
2 for(初始值; 條件; 更新){
3     //要執行的程式碼
4 }
5 舉例而言，假設我們要輸出1到10的數字，我們可以這樣寫：
6
7 for(int i=1;i<=10;i++){
8     Serial.println(i);
9 }
10 其中，i是初始值，i<=10是條件，i++是更新的方式。
```

4.4 while loop

在明確知曉執行次數的情況下，使用 for loop 會比較簡單，但在不明確知曉執行次數的情況下，使用 while loop 會比較直觀。while loop 的語法如下：

```

1 while loop 語法
2 while(條件){
3     //要執行的程式碼
4 }
5 舉例而言，假設我們要輸出1到10的數字，我們可以這樣寫：
6 int i=1;
7 while(i<=10){
```

```
8     Serial.println(i);
9     i++;
10 }
```

11 在這個例子中，我們先宣告一個變數*i*，然後使用 `while` loop 來判斷*i*是否小於等於10，
12 如果是，就輸出*i*的值，然後將*i*的值加1，直到*i*大於10為止。

以上這是明確知曉執行次數的情況下，使用 `while` loop 的範例程式碼，明顯會比 `for` loop 來得繁瑣，但在不明確知曉執行次數的情況下，使用 `while` loop 會比較直觀。

```
1 String password="arduino123";
2 String input="";
3
4 while(input!=password) {
5     Serial.println("請輸入密碼：");
6     while(Serial.available()==0){
7         //等待使用者輸入
8     }
9     input=Serial.readStringUntil('\n');
10    input.trim();//去除換行符
11 }
```

```
12 Serial.println("密碼正確，歡迎進入系統！");
```

13 在這個例子中，我們使用 `while` loop 來判斷使用者輸入的密碼是否正確，
14 如果不正確，就一直要求使用者輸入，直到輸入正確為止。