

如對本學與教資源有任何意見及建議，請致函：

香港九龍塘沙福道 19 號西座 W101 室

教育局課程發展處科技教育組

總課程發展主任（科技教育）

本學習資源版權，除在鳴謝頁所列舉的圖片外，全屬於香港特別行政區政府教育局擁有。

教育局歡迎學校等教育團體使用本學習資源的內容作非牟利的教學用途。任何情況下使用本學習資源，需作出鳴謝，教育局保留本學習資源版權。

未經本局事先允許，不能以任何形式使用其中教材作出版或其他用途，否則教育局將保留一切追究的權利。

© 版權所有 2019

本學習資源由香港機械人學院製作。

目錄

章節一：機械人基礎	P.3
章節二：什麼是 Arduino?	P.7
章節三：Arduino IDE (軟件)	P.13
章節四：控制(一) – 操控 LED	P.17
章節五：控制(二) – 蜂鳴器	P.26
章節六：控制(三) – 馬達	P.33
章節七：單元專題研習 – 測謊裝置	P.39

基礎單元一：電子元件的控制

章節一：機械人基礎

I. 甚麼是機械人？

當提到機械人時，許多人會想到有手、有腳的人形機械。不過，這類機械人往往只會出現在科幻電影、娛樂場所、展覽會和玩具店中，它們與工業用的機械人大不相同。

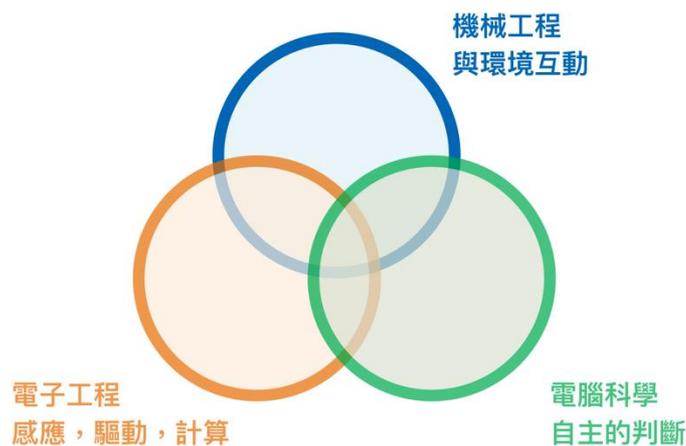
工業用機械人（Industrial Robots）簡稱為 IR，有時會被稱為機械臂，可進行簡單的動作例如提起→放下，或在機器內放入→取出工件等。同時，它們亦可進行較複雜的工作，例如運輸、握取、校準、裝配、檢驗等。



II. 機械人的定義

美國機械人協會在 1979 年將機械人定義為「一個可用程式控制，多功能的操作器。它透過程式控制和多變化的動作設計來移動材料、工件、工具或特別設備，以完成一連串的工作」。只要符合定義，沒有人的形態也可以稱為機械人。

「工業機械人」一般相等於機械臂，由多個環節與線性、旋轉式或稜柱式的連桿接合點組成。機械人一面被固定於支撐底部，一面則配上工具，用以被操控，從而執行任務。



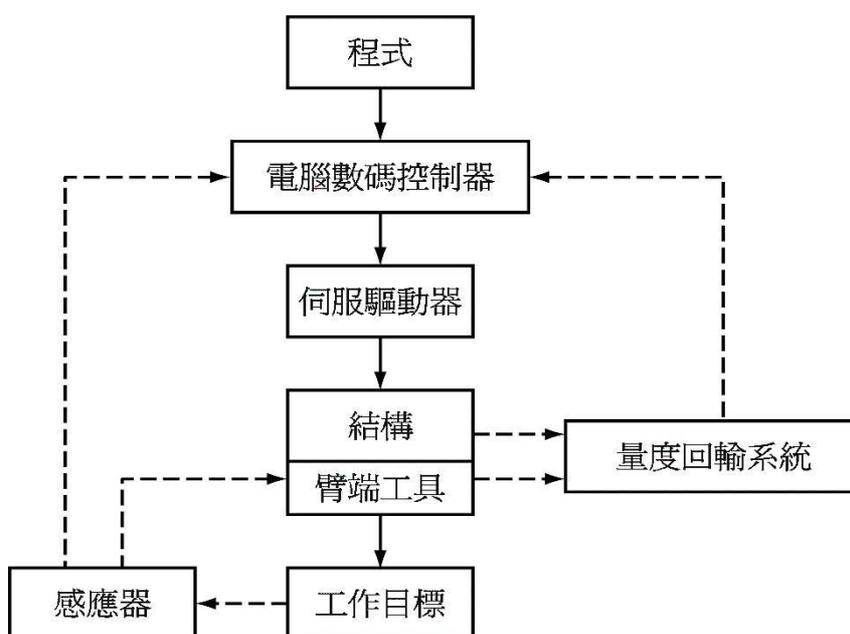
然而，時至今日，機械人已愈來愈流行。它不僅用作娛樂用途，甚至用於創新技術上，例如人類生活、動物及軍事用途。

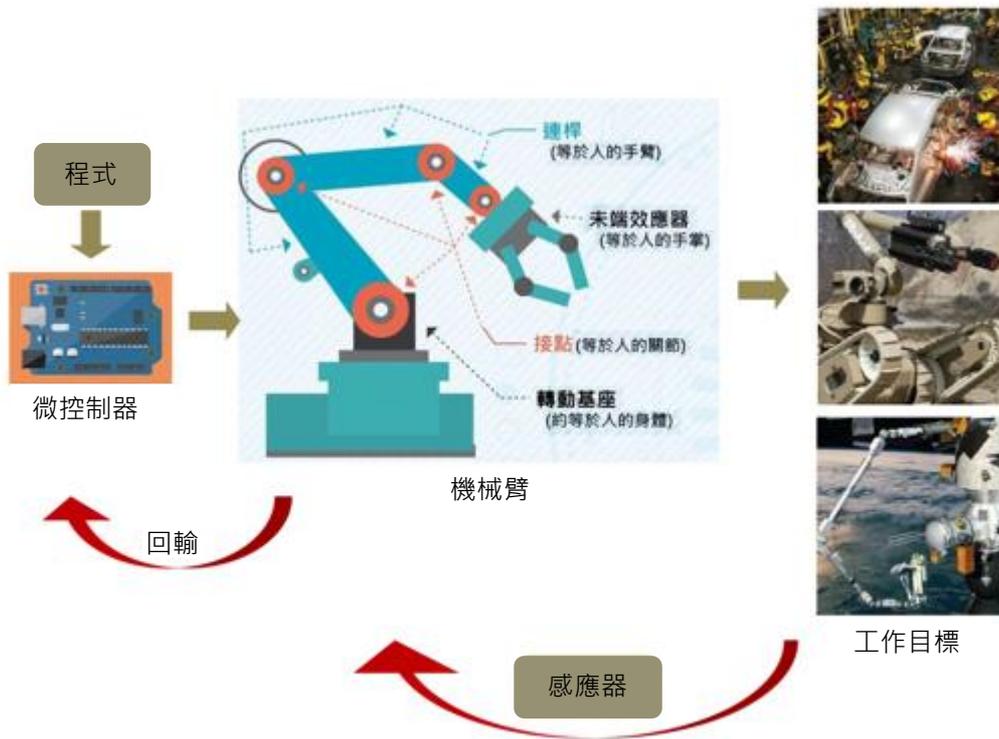
因此，機械人亦可以被解作「具備智慧及可編程的人造半獨立或全獨立（能自我控制的）物體或協作物體（具相同用途）。」

III. 機械人的設計

機械人的設計在工業上通常由六項基本元素所組成，包括：結構、臂端工具、電腦數碼控制器、驅動器、量度回輸系統和感應器。

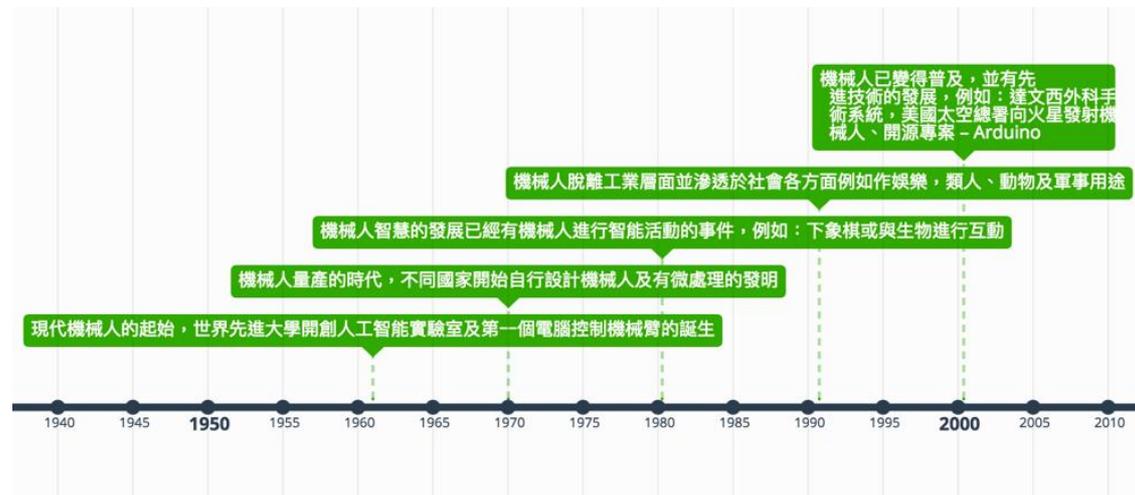
下圖顯示該六項基本元素的相互關係：





一開始我們將程式上載到微控制器，然後微控制器會對相應的輸出，例如機械臂、末端效應器做出反應。微控制器會靈活地操控不同的接點和轉動基座，以執行高準確度動作及特定工作目標。然後，機械人上的感應器會回輸一些讀數給微控制器。最後，微控制器作出相應行動以達到工作目標。

IV. 機械人的發展史以及機械人技術的提升



在 1950 年之前，機械人只是科幻小說內的人物。1960 年開始是現代機械人的起始，世界的先進大學如麻省理工學院開設人工智能實驗室，第一個有電腦控制的機械臂亦在這段時期誕生。直至 1970 年，世界正式踏入機械人量產的時代，不同國家尤其是歐洲的國家開始自行設計機械人，也開創了微處理器的發明。1980 年，人類突破機械人發展的樽頸位，機械人已發展到能夠擁有自己的智慧，它們可以下象棋或者與動物進行互動。直到 1990 年，機械人已經脫離工業層面並滲透於社會各個方面，例如軍事用途、娛樂、模仿人類及動物。踏入 2000 年，機械人已經變得普及，並且有先進技術的發展，例如達文西外科手術系統、美國太空總署向火星發射偵測機械人、以及開放源碼專案 Arduino 的誕生。

V. 溫習問題

1. 什麼是機械人？
2. 機械人學包括那幾個元素？
3. 機械人的設計由哪些基本元素組成？
4. 你能夠敘述整個機械人操作過程嗎？
5. 你能夠大概指出機械人發展史嗎？

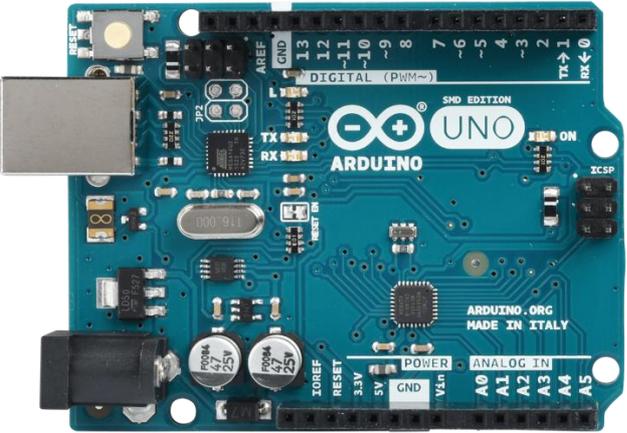
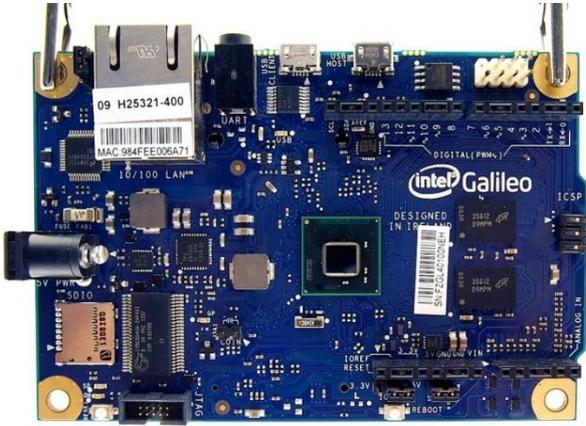
章節二：什麼是 Arduino?

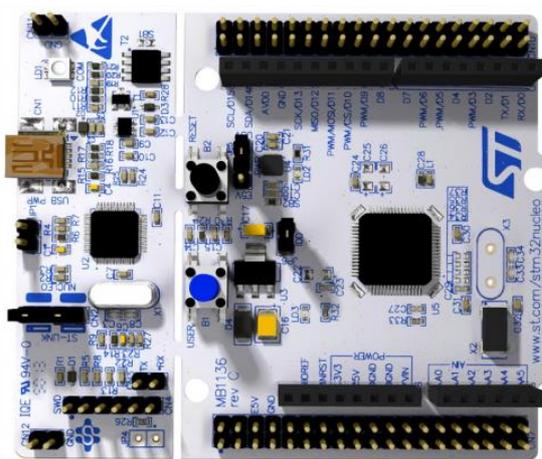
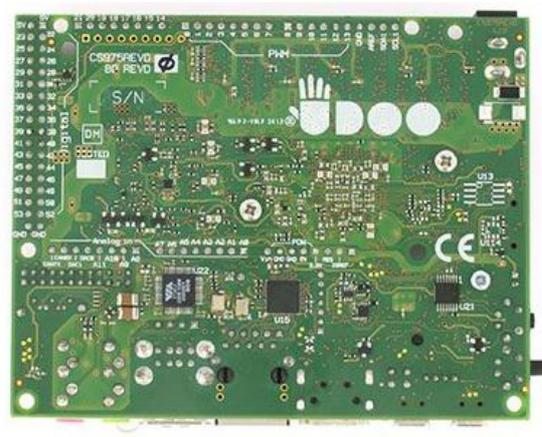
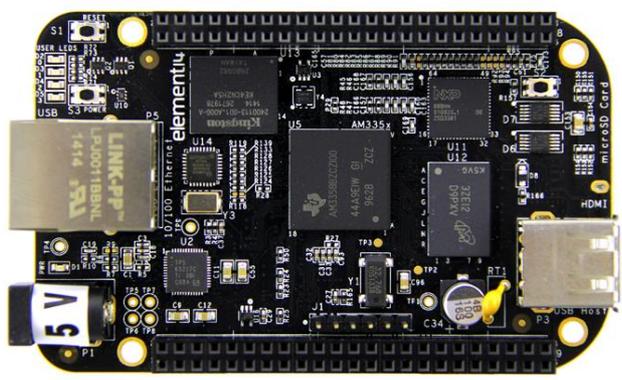
I. 了解微控制器

簡單來說，微控制器 (Microcontroller, MCU) 其實是一部微型電腦，包含了處理器、記憶體、輸入/輸出和其他週邊裝置，安裝在單一的集成電路中。但它不能如電腦般運作，沒有內置作業系統，程式開發需經由其他電腦系統編寫後再輸入。

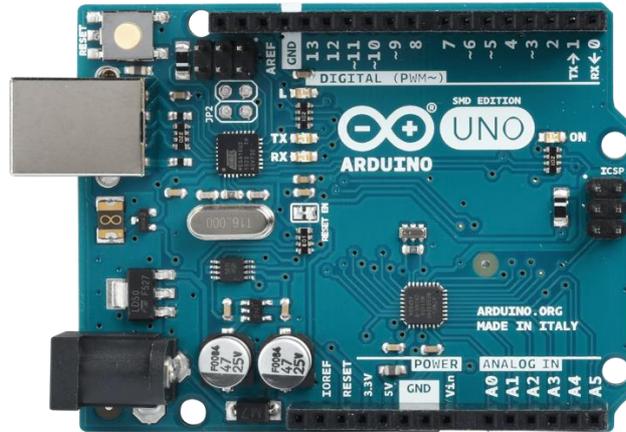
II. 不同的微控制器

早期的微控制器核心多使用由 Atmel 公司於 1996 年研發的 Atmel AVR 系列，是一種 8 位元~32 位元自動化控制精簡指令集的微控制器。它採用快閃記憶體 (Flash Memory) 作為資料存儲介質的單晶片微控制器，同時代的其他微控制器多採用一次寫入唯獨記憶體 (EPROM, EEPROM)。以下是數款較常見的微控制器：

<p>Arduino UNO</p> <p>本教材中會使用到的微控制器。這塊開發板的入門門檻低，普通人都很容易學會和使用。</p>	 A photograph of an Arduino UNO SMD Edition board. The board is blue with a white USB Type-C port on the left, a DC power jack, and a reset button. It features a USB Type-D connector, a 5V regulator, and several integrated circuits. The text 'ARDUINO UNO' and 'ARDUINO.ORG MADE IN ITALY' are visible on the board.
<p>Intel Galileo</p> <p>著名電腦處理器廠商 Intel 也有開發微控制器。這塊開發板的特點是它的腳位位置跟 Arduino Uno 相容，並且能使用 Arduino IDE 對其進行編程。</p>	 A photograph of an Intel Galileo board. The board is blue and features a central Intel processor, a USB Type-C port, a DC power jack, and a reset button. It has a similar form factor to the Arduino Uno. The text 'Intel Galileo' and 'DESIGNED IN IRELAND' are visible on the board.

<p>Nucleo 64</p> <p>Nucleo 開發板載有由 STMicroelectronics 32 位元的微處理器組成，其處理能力比 Arduino 強大，但此開發板使用較 Arduino 難。</p>	
<p>Udoo Dual</p> <p>這是一塊能載入 Window 10 IoT 的開發板，效能比 Arduino UNO 強大，卻比較難對其進行編程。</p>	
<p>BeagleBone Black</p> <p>這是一塊由德州儀器與 Digi-Key 和 Newark element14 合作生產的微控制器，於 2011 年發表的一塊開發板。</p>	

III. 什麼是 Arduino?



Arduino 是意大利 Ivrea 互動設計學院在 2005 給學生的一個作業，目的是提供一個低價而容易使用的介面，以供學生、業餘愛好者或專業人士設計包含感應器和驅動器的互動裝置。

Arduino 是一塊使用了 Atmel AVR 單晶片的電腦控制板，它可以接上各種輸入和輸出裝置，並可接上各種通訊模組。只要撰寫控制程式，便可利用它做出各式各樣的自動控制應用：例如利用溫度高低控制風扇的運轉速度、使用可變電阻控制燈光的明暗、控制馬達的轉速、利用紅外線或藍芽遙控家電、利用伺服馬達 (Servo) 控制機械臂或機械人、以及製作自動行走車和飛行器等。

Arduino 最大特色的是開放源碼，不僅軟件是開放源碼，連硬體也是開放的。傳統上，要開發微控制器的程式，開發者需要具備電子電機等相關科系的背景，不是一般人容易進入的世界。Arduino 的門檻低，普通人都很容易學會和使用。再者，Arduino 在網絡上有非常多的資源，我們只要參考網上的作品，配合自己的需求調整一下設計，就可以在短時間內完成自己的作品。

Arduino 的電路設計圖可以從網絡上下載，亦可在網上購買到價格不高的 Arduino 控制板。

IV. Arduino 控制板（硬件）

用 USB 連接電腦編程

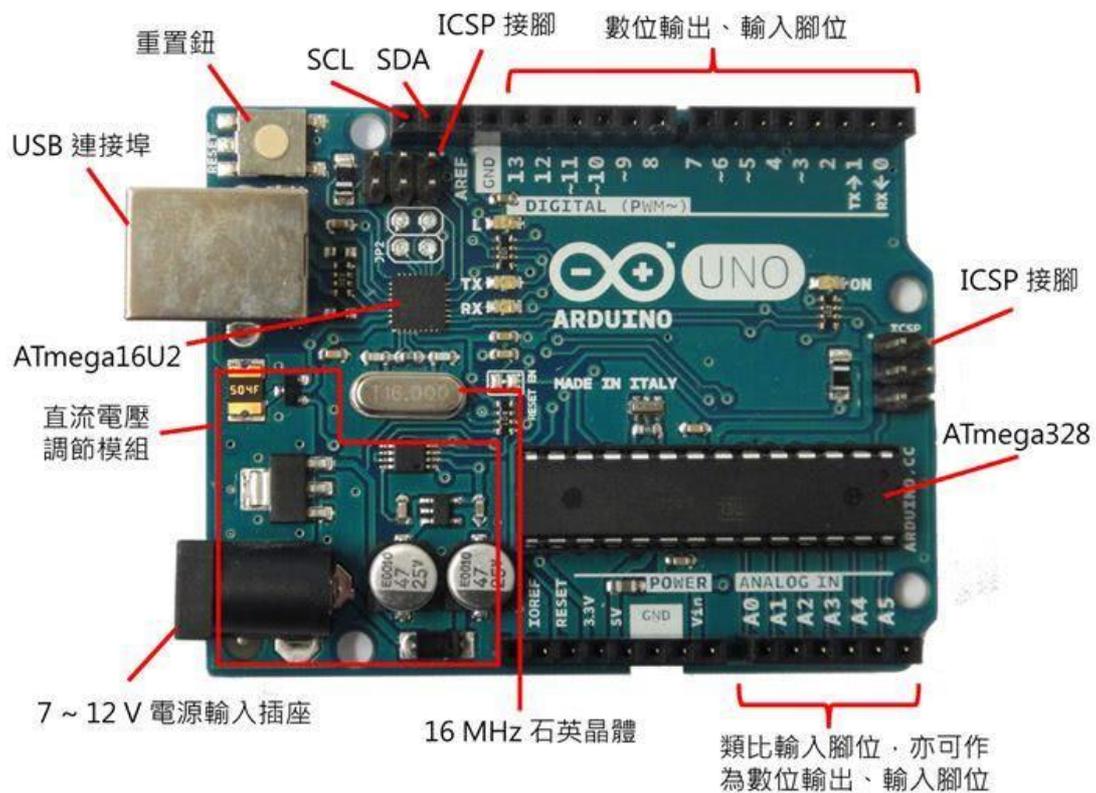
Arduino 可以利用通用的 USB 來連接電腦，編寫程式，方便使用者。

在線編程

Arduino 使用了在線編程器，把 boot loader（引導程序）上載至控制板。它引導操作系統啟動的程序，帶來的優點有：

1. 簡化的編程過程
2. 運行用戶程序時啟動
3. 從開發平台檢測方案

以下介紹 Arduino 控制板上的各元件及它提供的各式輸入和輸出的接點：



VI. 溫習問題

1. 什麼是 Arduino UNO？

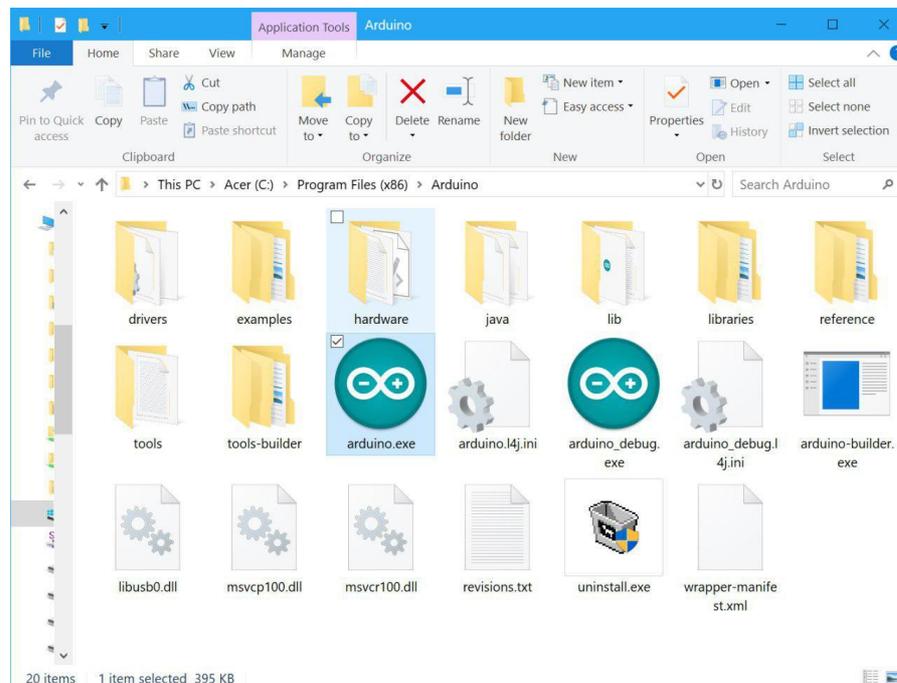
章節三： Arduino IDE（軟件）

I. 程式安裝及軟硬件連接

Arduino IDE 是 Arduino 的編程平台及環境。Arduino 接受使用很多不同的編程軟件，但這裡只集中使用從 arduino.cc 下載的 Arduino IDE。



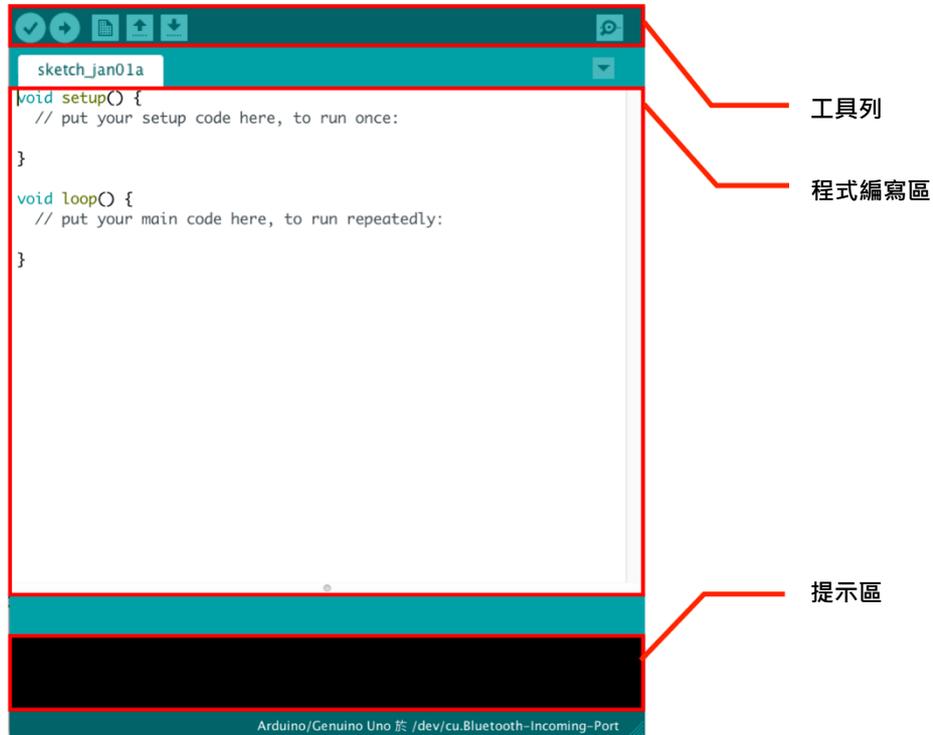
可從 www.arduino.cc 下載 Arduino IDE



Arduino 安裝檔

II. Arduino IDE 版面

1. 版面簡介

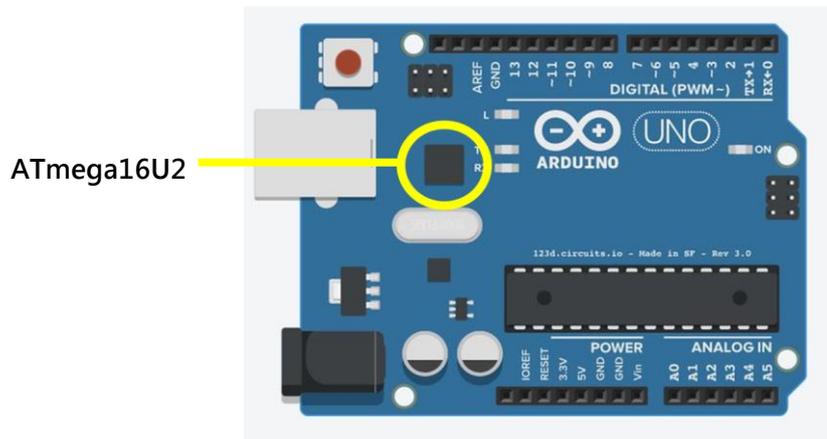


2. 工具列圖標說明：

-  Verify: 核實程式無誤
-  Upload: 上傳編譯後的程式
-  New: 開新程式
-  Open: 打開已有的程式
-  Save: 存儲程式
-  Serial Monitor: 透過串行埠與 Arduino 溝通

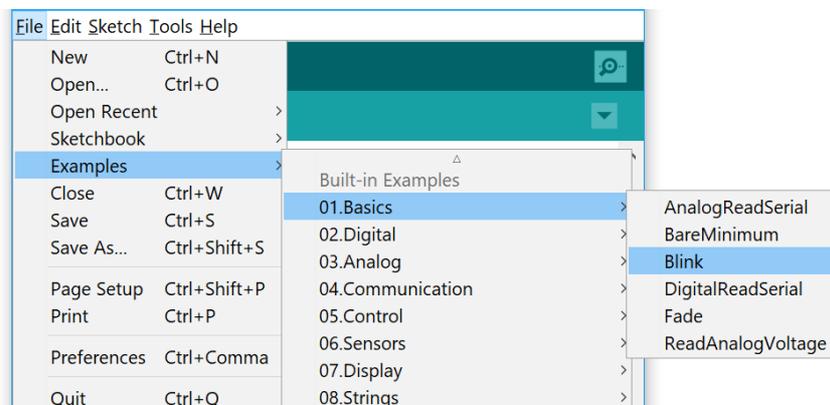
III. 動手做 – 了解 Arduino & Arduino IDE

本課節是讓同學能夠熟悉 Arduino 控制板及 Arduino IDE 的操作。要使電腦和 Arduino 控制板進行溝通，我們必須使用序列埠的方法。可是，現代電腦一般使用 USB 接口，並不能跟 Arduino 控制板直接溝通。這個時候，我們便需要一塊 USB 轉序列埠的晶片。常見的晶片型號有 ATmega8U2、CH340、CP2102、FTDI232R 及 PL2303。Arduino 控制板官方版本載有 ATmega16U2 USB 轉序列埠的晶片，這塊晶片可以幫助我們使用電腦跟 Arduino 控制板進行溝通，並燒錄程式。



這練習使用 Arduino 控制板上的 LED，無需外接其他單元。

1. 開啟 Arduino 的範例-LED Blink（閃爍的 LED）：點開 File → Examples → 01.Basics → Blink。

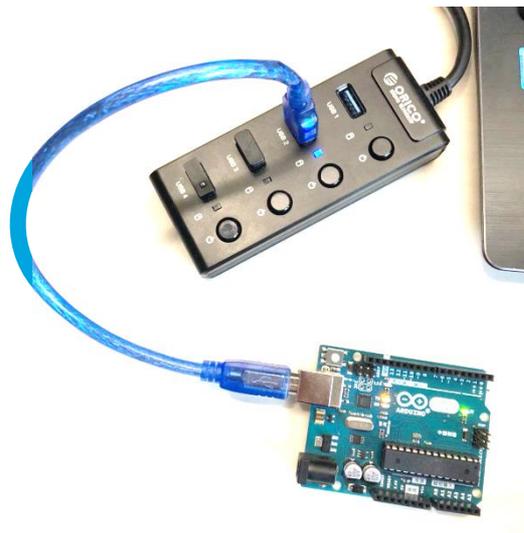


2. 點開 **Blink** 後會出現以下的程式。

```
Blink $
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
38
```

本程式主要分為兩部份，第一部份 `setup()` 內的程式碼設定 **Arduino** 的內置 **LED** 為輸出(**OUTPUT**)，第二部份 `loop()` 內的程式碼則設定內置 **LED** 不停地進行開燈關燈的動作。

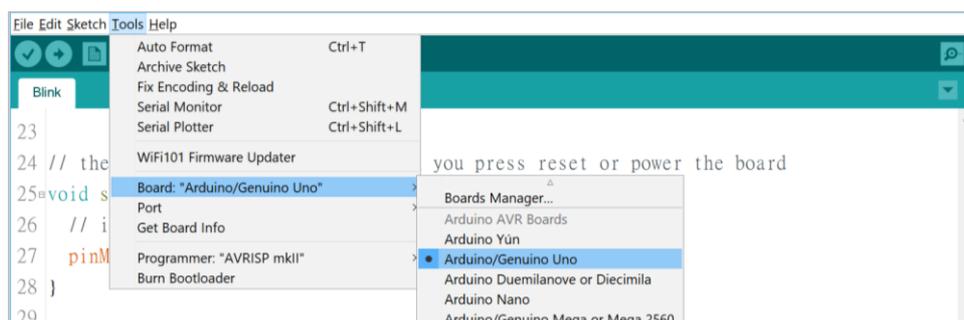
3. 接著插上 **USB** 線，把 **Arduino** 控制板與電腦連接。



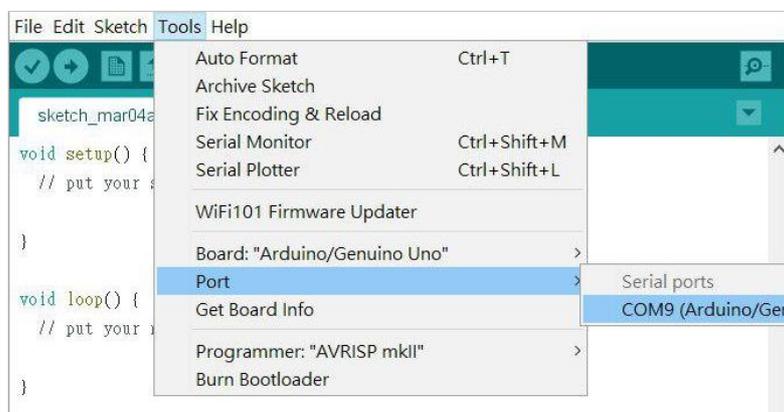
4. 連接電腦後，我們要先打開電腦的控制台 > 裝置管理員 (Device Manager)，找出 Arduino 板的通訊埠 (又稱序列埠)。下圖顯示在裝置管理員中可找到 Arduino UNO 的通訊埠(COM9)。



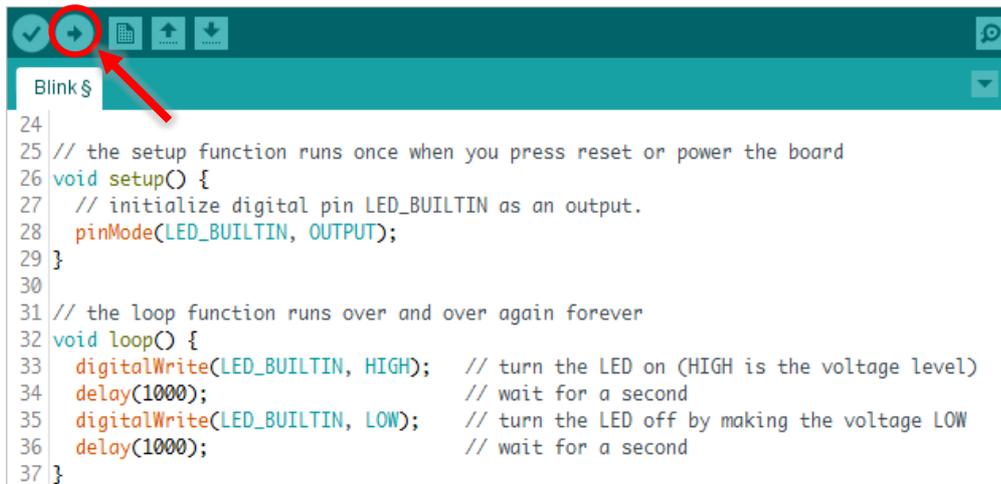
5. 在 Arduino IDE → Tools → Board: → 選取 Arduino UNO 開發板。



6. 在 Arduino IDE 上選取剛才於裝置管理員中顯示的 Arduino UNO 序列埠。

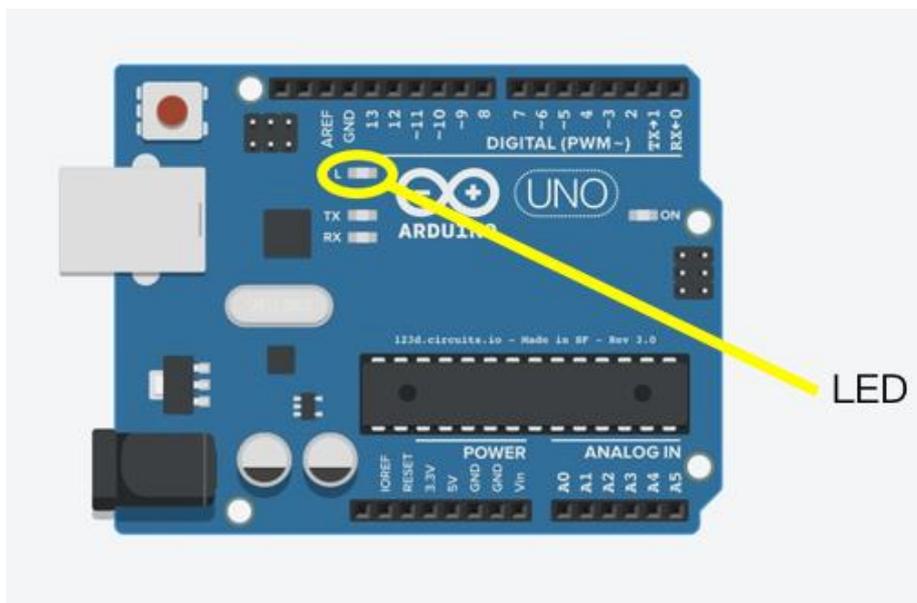


7. 按下上載（Upload），程式會被編譯並上載到 Arduino Uno 開發板上。



```
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
```

8. 若上載成功，會看到 Arduino 控制板上的 LED 閃爍。



IV. 溫習問題

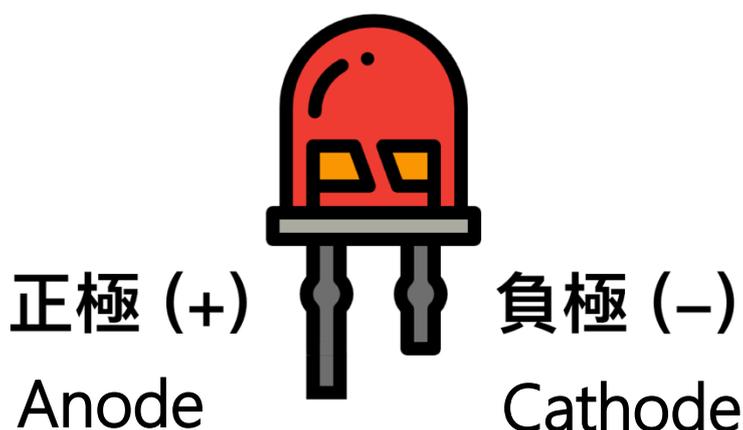
1. 什麼是 Arduino IDE？
2. 如何把 Arduino 控制板與電腦連接並進行溝通？

章節四：控制(一) – 操控 LED

I. 發光二極管 (Light-emitting Diode, LED)

發光二極管是最基本及最常用的輸出裝置，其可以發光的特性，可用於表示狀態，如合格和不合格，便可用綠色和紅色的發光二極管閃亮去表示。此外，它亦可用於程式測試，因為程式是由上而下執行，假如我們想測試其中一段指令是否已經執行，我們可以在程式的下面加上一個閃亮發光二極管的程式。

假如發光二極管閃爍，證明程式已經被執行。另外，我們可以測試訊號的強度，例如我們可把接收到的訊號用作表示 LED 燈的閃亮程度，到時我們便能從肉眼去判斷訊號強度等等的指標。以上提到的，我們都可以用發光二極管加上編程去進行。



發光二極管是一種單體導向硬件，即是在一個電路中，我們需要根據電流去接駁；若果產生反接現象，它便不會閃亮。發光二極管一般有長短腳之分，長的接腳我們稱之為**正極**，短的接腳稱之為**負極**，電流方向需要由**正極**去**負極**。

由於它有體積小、不會發熱、比較耐用的特性，發光二極管被廣泛運用於表示電子產品的信號燈，有別於傳統燈泡只用於照明的功能。在這一課，我們採用的發光二極管是一般常見的小型發光二極管，有不同的顏色可供選擇，如紅、橙、黃、綠、藍等等。

II. Arduino 程式的各部分

```
File Edit Sketch Tools Help
Blink$
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the Uno and
Leonardo, it is attached to digital pin 13. If you're unsure what
pin the on-board LED is connected to on your Arduino model, check
the documentation at http://arduino.cc

This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}

Done uploading
Binary sketch size: 1,082 bytes (of a 32,256 byte maximum)
19 Arduino Uno on COM8
```

/* 與*/之間是編程人員的備註，不會被執行

Setup(): 一個會被執行一次的函數，這裏設定13為輸出接腳

// 後面的部分，也不會被執行

loop(): 一個會不停重複執行的函數，直至截斷電源為止

III. 函數的各部分

C/C++ 編程語言的函數部分

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // turn the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off
  delay(1000);            // wait for a second
}
```

所有函數的指示需放在{及}之間

函數輸出 (void 即無輸出)

函數名稱

函數輸入() 即無輸入

函數需完成的指示需以「;」完結；例如. a = a + b;

pinMode(13, OUTPUT);

pinMode - 叫出函數，**定義針腳的模式**。函數先指出針腳的編號(1~13 或 A0~A5)，後指出動作(INPUT, OUTPUT 或 INPUT_PULLUP)。INPUT:輸入; OUTPUT:輸出; INPUT_PULLUP:啟動此針腳的 pull-up 電阻，並提供一個穩定的 5V 電壓

digitalWrite(13, HIGH);

digitalWrite - 叫出函數，**定義針腳為以數碼方式寫入 5V 或 0V**。函數先指出針腳的編號(1~13 或 A0~A5)，後指出寫入動作(LOW 或 HIGH)。HIGH:高電位(5V); LOW:低電位(0V)

delay(1000);

delay - 叫出函數，定義時間**延遲** 1000ms (必需以微秒(ms)為單位)。

在程式設計的過程中，我們會需要記錄某些資料，可能是文字，也可能是數字，我們把這些資料記錄在記憶體某個位址中，並給它一個名稱，這就是變數。Arduino 程式中有下列其中一些常用的變數：

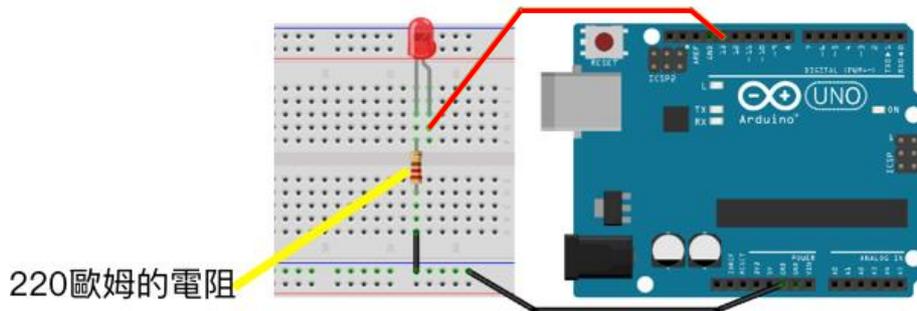
種類	記憶體大小	數值
boolean (布爾值)	8 bit (1 Byte)	1 or 0
byte (位元)	8 bit (1 Byte)	0 ~ 255
char (字元)	8 bit (1 Byte)	-128 ~ 127
int (整數)	16 bit (2 Bytes)	-32768 ~ 32767
long (長整數)	32 bit (4 Bytes)	-2147483648 ~ 2147483647
float (浮點數)	32 bit (4 Bytes)	$\pm 3.4 e \pm 38$

IV. 動手做 – LED 閃光燈

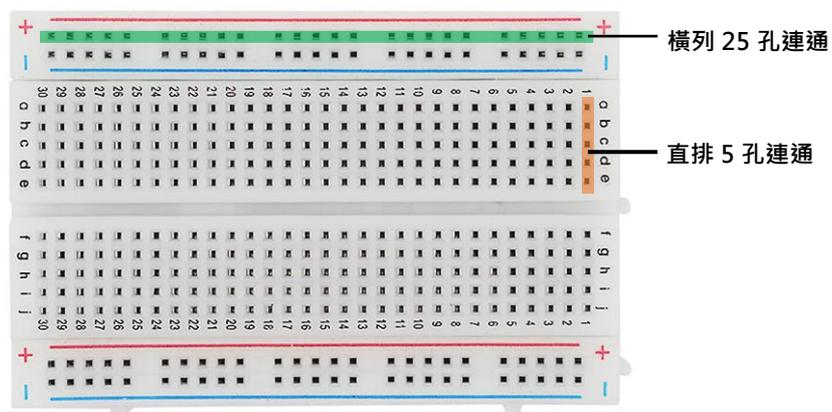
本章的第一個任務是嘗試透過 Arduino 控制 LED 閃光燈。這次任務共分為兩個部分，分別是連接 Arduino 和 LED，以及使用 Arduino 編程控制 LED。

一 連接 Arduino 和 LED

我們根據以下方法連接 LED、電阻和 Arduino。



這次任務需要的電子零件不多，而且接駁方法非常簡單。首先，我們把一盞紅色 LED 安插在麵包板上。請留意，LED 的腳位有分長短，較長的腳位是正極(Anode)；較短的腳位是負極(Cathode)。當我們要亮起 LED，電流必須要由正極流向負極。因此，當 LED 亮起時，正極必須處於高電壓，而負極必須處於低電壓。這次任務中，LED 燈的正極會連向 Arduino 的 13 號腳位，負極會連接 Arduino 的 GND (接地)。連接 LED 和 Arduino 中間有一個電阻，電阻的功用是限制流經 LED 的電流，避免電流過大影響 LED 運作。



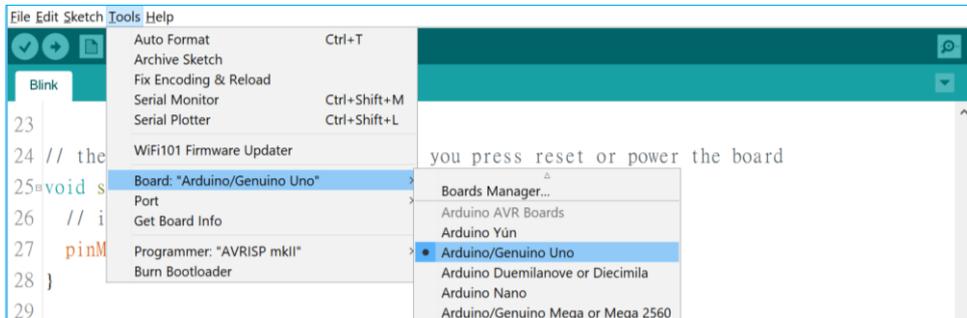
認識「麵包板」：

麵包板是不需要經由焊接過程，就可以將電路所使用的電子元件加以連接。

1. 電源軌：上圖麵包板的上下分別有兩列插孔，一般是作為電源引入的通路。上方第一行標有「+」的一列有 25 組內部連通的插孔，均為正極。上方第二行標有「-」的一列則為接地。麵包板下方第一行與第二行結構同上。
2. 接線軌：上圖連接孔分為上下兩部分，是主要的工作區域，用來插接原件和跳線。在同一直排的 5 個插孔（即 a-b-c-d-e 或 f-g-h-i-j）是互相接通的；直排與直排（即 1-30）之間以及凹槽上下部分（即 e-f）是不相通的。

二 使用 Arduino 編程控制 LED

我們打開 Arduino IDE，並把 Arduino 連接電腦。在工具裡選擇 Arduino UNO 為開發板，並檢查序列埠是否已連接 Arduino。



連接 Arduino 後，我們需要輸入以下程式，並把程式上傳到 Arduino。

程式碼全圖，可參考檔案 [1_4_1_led_flash](#)

```

1 int ledPin = 13;                                定義變數
2 -----
3 void setup() {
4   pinMode(ledPin, OUTPUT);                      定義 LED 腳位
5 }
6 -----
7 void loop() {
8   digitalWrite(ledPin, HIGH);
9   delay(1000);                                  設定 LED 閃光頻率
10  digitalWrite(ledPin, LOW);
11  delay(1000);
12 }
  
```

三 程式內容說明

1. 定義變數。

```
1 | int ledPin = 13;
```

程式共分為三個部分。第一部分我們定義變數 ledPin 為整數 13。整數(integer)是程式裡其中一種數據類型，定義整數變數時需要在變數的前方加上“int”。

2. 定義 LED 腳位。

```
3 | void setup() {  
4 |   pinMode(ledPin, OUTPUT);  
5 | }
```

第二部分我們定義 LED 連接 Arduino 的腳位。在上一部分變數 ledPin 定義為整數 13，這部分我們便使用“pinMode()”把 13 號腳位(Pin)定義為輸出腳位。

3. 設定 LED 閃光頻率。

```
7 | void loop() {  
8 |   digitalWrite(ledPin, HIGH);  
9 |   delay(1000);  
10 |  digitalWrite(ledPin, LOW);  
11 |  delay(1000);  
12 | }
```

我們假設 LED 每兩秒閃亮一次，所以 LED 亮起和熄滅的時間各設定為一秒。我們在程式中會使用“digitalWrite()”來控制 LED，當“digitalWrite”裡面是 HIGH 時，LED 燈就會亮起；當“digitalWrite”裡面是 LOW 時，LED 燈就會熄滅。“delay()”是控制程式停頓的時間，當中使用的單位是微秒。緊記，1 秒等如 1000 微秒。

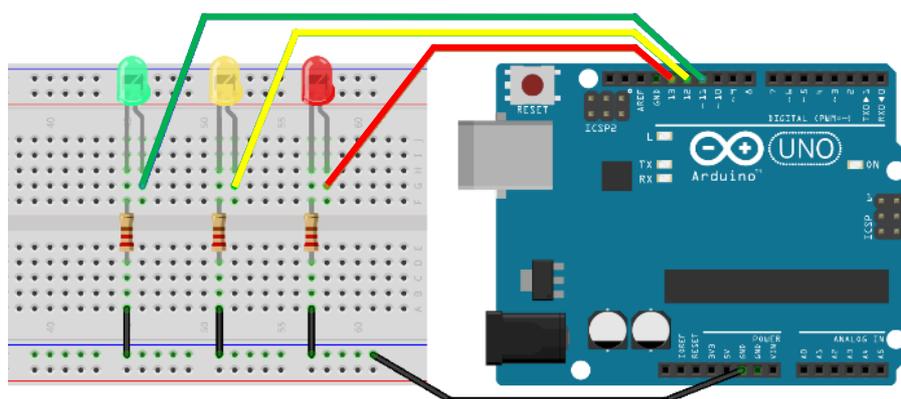
4. 當把程式上載到 Arduino UNO 後，麵包板上的紅色 LED 每一秒就會亮起一次，不停重複。

V. 動手做 – 紅綠燈

本章的第二個任務與第一個任務非常相似，這次任務我們會嘗試控制三盞不同顏色的 LED 輪流閃亮，製作我們日常生活裡經常看見的紅綠燈。

一 連接 Arduino 和 LED

我們根據以下方法連接 LED、電阻和 Arduino。



這個任務連接 LED 的方法和第一部分的任務一樣，LED 燈的正極會連向 Arduino，而負極會連向 GND。由於這次任務需要控制三盞 LED，所以我們需要用上 Arduino 上三個腳位連接 LED，分別是 11、12 和 13 號腳位。

二 使用 Arduino 編程控制紅綠燈

我們打開 Arduino IDE，並把 Arduino 連接電腦。在工具裡選擇 Arduino UNO 為開發板，並檢查序列埠是否已連接 Arduino。連接 Arduino 後，我們需要輸入以下程式，並把程式上傳到 Arduino 當中。

紅綠燈的程式碼全圖，可參考檔案 1_4_2_traffic_light

```
1 int ledPin_R = 13;
2 int ledPin_Y = 12;
3 int ledPin_G = 11;
4
5 void setup() {
6     pinMode(ledPin_R, OUTPUT);
7     pinMode(ledPin_Y, OUTPUT);
8     pinMode(ledPin_G, OUTPUT);
9 }
10
11 void loop() {
12     digitalWrite(ledPin_R, HIGH);
13     delay(1000);
14     digitalWrite(ledPin_R, LOW);
15     digitalWrite(ledPin_Y, HIGH);
16     delay(1000);
17     digitalWrite(ledPin_Y, LOW);
18     digitalWrite(ledPin_G, HIGH);
19     delay(1000);
20     digitalWrite(ledPin_G, LOW);
21     delay(1000);
22 }
```

定義變數

定義 LED 腳位

設定 LED 閃光的
次序和頻率

三 程式內容說明

1. 定義變數

```
1 int ledPin_R = 13;
2 int ledPin_Y = 12;
3 int ledPin_G = 11;
```

我們在這部分同時定義三個變數，分別是“ledPin_R”、“ledPin_Y”和“ledPin_G”。這三個變數分別儲存整數 13、12 和 11。

2. 定義 LED 腳位

```
5 void setup() {
6     pinMode(ledPin_R, OUTPUT);
7     pinMode(ledPin_Y, OUTPUT);
8     pinMode(ledPin_G, OUTPUT);
9 }
```

第二部分我們分別定義 LED 連接 Arduino 的腳位。我們使用“pinMode()”把 11、12 和 13 號腳位定義為輸出腳位。

3. 設定 LED 閃光次序和頻率

```
11 void loop() {  
12     digitalWrite(ledPin_R, HIGH);  
13     delay(1000);  
14     digitalWrite(ledPin_R, LOW);  
15     digitalWrite(ledPin_Y, HIGH);  
16     delay(1000);  
17     digitalWrite(ledPin_Y, LOW);  
18     digitalWrite(ledPin_G, HIGH);  
19     delay(1000);  
20     digitalWrite(ledPin_G, LOW);  
21     delay(1000);  
22 }
```

LED 閃光的次序會根據紅、黃、綠這三種顏色的先後次序而定。每種顏色發光的時間是一秒。當一盞 LED 亮起，前一盞亮起的 LED 便會熄滅。我們使用 “delay(1000)” 來控制 LED 亮起和熄滅的時間，而 “digitalWrite()” 控制 LED 發光和熄滅的次序。

VI. 溫習問題

1. 什麼是發光二極體？
2. 什麼是 pinMode, int x = 3; ?

章節五：控制(二) – 蜂鳴器

I. 發音體和聲音

電子軟件除了會發光之外，另一種比較普遍的就是發聲。一般可分為**揚聲器**和**蜂鳴器**兩種。蜂鳴器體型比較小且靈活，但是因為音質比較差，故適用於警報聲或提示等用途。小型的揚聲器比蜂鳴器要大，使用紙膜震動音質比蜂鳴器高，對於要求音準的電子產品，或機械人如電子琴、機械助手等會比較適合。下圖是蜂鳴器：

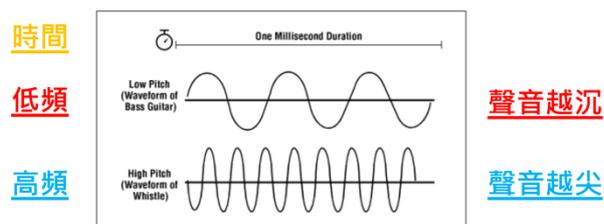


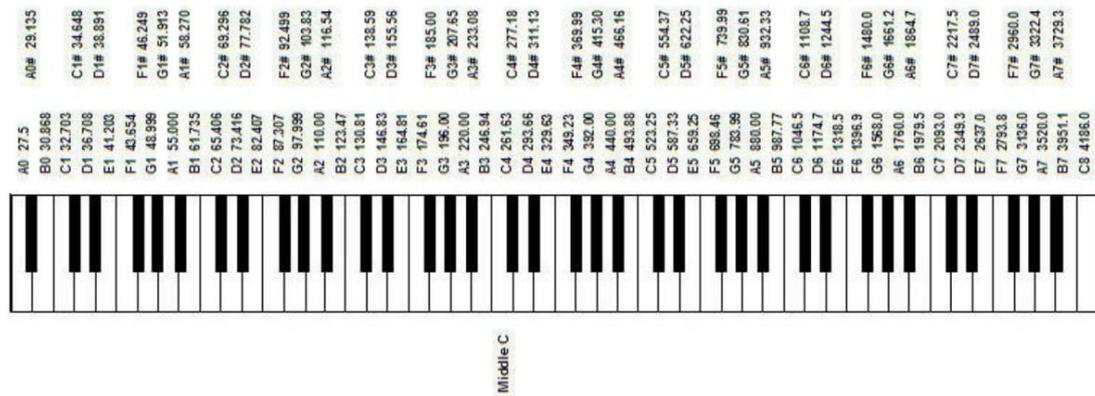
蜂鳴器內部的主要零件是蜂鳴片。蜂鳴片是一塊薄的銅片，當通電時便有電磁力推動銅片震動。由於聲音是由震動產生，震動的頻率稱為音頻。透過公式我們能夠計算出不同的音頻在傳統音階上是什麼音。撰寫音樂程式之前，我們先複習一下基本的音樂常識。

II. 音高與節拍

音頻（率）的高低稱為音高（pitch）。在音樂上，我們用 Do、Re、Mi 等音名來代表不同頻率的音高。鋼琴鍵盤就是依照聲音頻率，由 C、D、E 等階級（音階）順序來排列。每個音高和高八度的下一個音高（註：從中音「Do」向右數到第 8 個白鍵，就是比中音「Do」高八度的高音「Do」），其頻率比正好是兩倍：

頻率(Frequency)是量度震動的標準。由於聲音是一種波段，所以每種聲音都有各自獨特的頻率。聲音越尖，頻率越高；聲音越沉，頻率越低。





在本課的任務中，我們需要運用到新的語法來控制蜂鳴器，包括 for 循環及陣列。

III. for 循環

開始時，我們先要了解一下 for 循環的語法：

```
for (初始變數; 條件式; 計數敘述) {
    敘述 1;
    敘述 2;
}
```

在設定循環時，第一件事是決定究竟我們想循環重複多少次。假如我們想重複執行三次，例子：`for (byte i = 0 ; i < 3 ; i++) { }`。第一個循環中，變數 `i` 為 0。當執行完成第一個循環後，變數 `i` 會加 1 並執行第二個循環。最後變數 `i` 會再加 1 再執行第三個循環。由 `i` 需要小於 3，所以完成第三個循環後循環停止。

從循環語法中，我們可以知道設定循環需要寫四部份。首先是寫 `for`，然後我們需要設定起始，一般來說我們會將起始設定成 0。變數表示之後就是執行條件，在例子中我們寫入變數少於 3，這樣就能重複三次。最後就是一個重複多少次的方法，因為我們知道若變數等於 3 的話，循環就會停止，如果每次加一，由 0 至少於 3 總共需要三次，所以就會寫每次增加 1。最後編寫我們需要重複執行的程式，這樣就完成一個循環。

上段提到，起始一般而言會設定成 0，但也有例外。舉例說，我們想將第 8 至 10 插腳的發光二極管輪流定義。首先在程式中定義為輸出，那麼我們就可以利用變數 `i` 去做邏輯，然後在過程中每一次定義的插腳也遞增上去，由 8 增加至 10，第一次執行時變數

會是 8、第二次執行時變數會是 9、第三次執行時變數會是 10。這樣我們就節省了程式所佔的容量，也避免了不斷重複同一條程式碼的情況。

```
void setup() {  
    for ( byte i = 8 ; i <= 10 ; i++ ) {  
        pinMode ( i, OUTPUT );  
    }  
}
```

IV. 認識陣列 (Array)

在認識陣列之前，首先要認清究竟變數是如何儲存一個數值。在這部份，我們可以將變數想像成一個箱子，定義一個變數的話就是為箱子加上一個名稱。假設未定義數值會是空的箱子，我們定義之後，箱子就會儲存一個變數。一般而言，這個箱子只能儲存一個數值。如我們需要在程式中儲存 10 個數值的話，我們就需要寫 10 句；如我們需要儲存 100 個變數的話，便需要定義 100 句。可是，這樣的程式非常缺乏效率，因此我們可以使用陣列，亦即是定義有多少個相似的箱子；例如以下這句句法就是指箱子能夠儲存 4 個整數：

```
int Nums[4]; (資料類型 陣列名稱[陣列大小] ;)
```

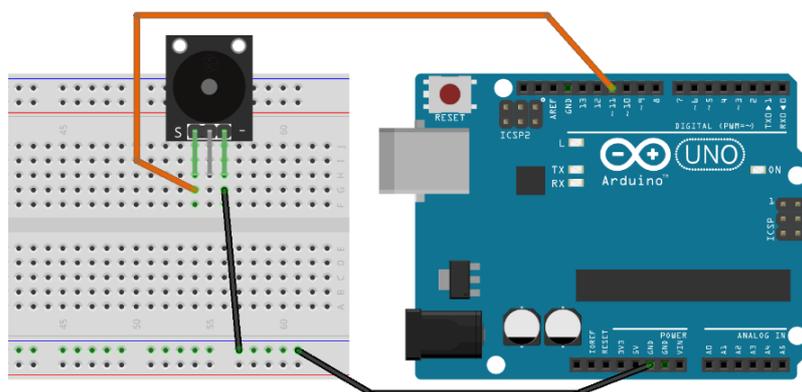
Nums	[0]	[1]	[2]	[3]
------	-----	-----	-----	-----

V. 動手做 – 蜂鳴器

本課會嘗試透過 Arduino 控制蜂鳴器播放音樂。這次任務共分為兩部分，當中包括連接 Arduino 和蜂鳴器，以及使用 Arduino 編程讓蜂鳴器播放音樂。

一 連接 Arduino 和蜂鳴器

我們根據以下方法連接蜂鳴器和 Arduino。



蜂鳴器連接 Arduino 的方法非常容易和直接，基於蜂鳴器自身帶具有高電阻率的特性，令使用蜂鳴器時不需要額外借用電子零件的協助。蜂鳴器同樣有分正極和負極，正極通常會以紅色電線標示，而負極會用黑色電線標示。我們需要把正極連接到 Arduino 11 號腳位上，而負極則連接到 Arduino 的 GND。

二 使用 Arduino 編程讓蜂鳴器播放音樂

我們打開 Arduino IDE，並把 Arduino 連接電腦。在工具裡選擇 Arduino UNO 為開發板，並檢查序列埠是否已連接 Arduino。連接 Arduino 後，我們需要輸入以下程式，並把程式上傳到 Arduino 當中。

音樂播放的程式碼全圖，可參考檔案 1_5_play_music

```
1 int buzzerPin = 11;
2 int notes_length = 25;
3 char notes[] = "ddedgfddedagddDbgfeCCbgag";
4 int beats[] = {1, 1, 2, 2, 2, 4, 1, 1, 2, 2, 2, 4,
5 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 4};
6 int tempo = 300;
7
8
9 void playTone(int frequency, int duration){
10 tone(buzzerPin, frequency);
11 delay(duration);
12 noTone(buzzerPin);
13 }
14
15
16 void playNote(char note, int duration){
17 char names[] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D'};
18 int freq[] = {261, 294, 329, 370, 392, 440, 493, 523, 587};
19 for(int i = 0; i < notes_length; i++){
20 if (names[i] == note){
21 playTone(freq[i], duration);
22 }
23 }
24 }
25
26 void setup() {
27 pinMode(buzzerPin,OUTPUT);
28 }
29
30 void loop() {
31 for(int i = 0; i < notes_length; i++){
32 if(notes[i] == ' '){
33 delay((beats[i] * tempo));
34 }
35 else{
36 playNote(notes[i], beats[i] * tempo);
37 }
38 delay(tempo/2);
39 }
40 delay(1000);
41 }
```

定義變數和陣列

定義函數

定義蜂鳴器連接腳位

使用循環播放音樂

三 程式內容說明

1. 定義變數和陣列。

```
1 int buzzerPin = 11;
2 int notes_length = 25;
3 char notes[] = "ddedgfddedagddDbgfeCCbgag";
4 int beats[] = {1, 1, 2, 2, 2, 4, 1, 1, 2, 2, 2, 4,
5 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 4};
6 int tempo = 300;
```

在這部分我們定義三個變數和兩個陣列。三個變數分別是 “buzzerPin”、 “notes_length” 和 “tempo”。“buzzerPin” 儲存著整數 11，代表蜂鳴器連接到 Arduino 的腳位。“notes_length” 儲存著整數 25，代表樂曲裡音符的總數量。“tempo” 儲存著整數 300，代表一拍等待的時間。兩個陣列分別是 “notes[]” 和 “beats[]”。“notes[]” 儲存著不同英文字母，英文字母代表樂曲中的音符。“beats[]” 儲存著不同音符的音長，當 tempo 乘以陣列 beats[] 中的每一項，得出的結果就是個音符響起的時間。

2. 定義函數。

```
9 void playTone(int frequency, int duration){
10     tone(buzzerPin, frequency);
11     delay(duration);
12     noTone(buzzerPin);
13 }
14
16 void playNote(char note, int duration){
17     char names[] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D'};
18     int freq[] = {261, 294, 329, 370, 392, 440, 493, 523, 587};
19     for(int i = 0; i < notes_length; i++){
20         if (names[i] == note){
21             playTone(freq[i], duration);
22         }
23     }
24 }
```

在這部分我們定義兩個函數，一個名叫 “playTone()”，另一個名叫 “playNote()”。函數 “playNote()” 負責根據陣列 “notes[]” 中的英文字母，設定每個音符聲音的頻率。而函數 “playTone()” 負責播放音樂。使用 “tone()” 能根據函數 “playNote()” 中計算的頻率在 “buzzerPin” 腳位播放音樂；而使用 “noTone()” 就能使蜂鳴器停止播放音樂。在 for 循環中，循環次數根據歌曲長度來定立，依照音階的音長來播放聲音。

3. 定義蜂鳴器連接腳位。

```
26 void setup() {
27     pinMode(buzzerPin, OUTPUT);
28 }
```

這部分我們使用 “pinMode()” 把 “buzzerPin” 11 號腳位定義為輸出腳位。

4. 使用循環播放音樂。

```
30 void loop() {  
31   for(int i = 0; i < notes_length; i++){  
32     if(notes[i] == ' '){  
33       delay((beats[i] * tempo));  
34     }  
35   else{  
36     playNote(notes[i], beats[i] * tempo);  
37   }  
38   delay(tempo/2);  
39 }  
40 delay(1000);  
41 }
```

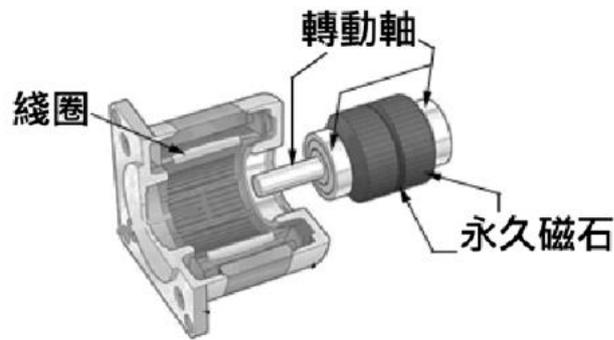
在第四部分，我們使用 for 循環由 0 到 25 把所有在 “notes[]” 定義的音符在程式裡運行一次。在這個循環當中有一個 if-else 條件，if 的條件是檢查陣列 “notes[]” 當中有沒有空白鍵。如果有空白鍵，程式便會把它當成休止符，若不是空白鍵，程式便會把它當成正常音符。程式的最後有一個 “delay(1000)” 等待，這個停頓是歌曲和歌曲之間的停頓，讓我們更容易聽到歌曲。

VII. 溫習問題

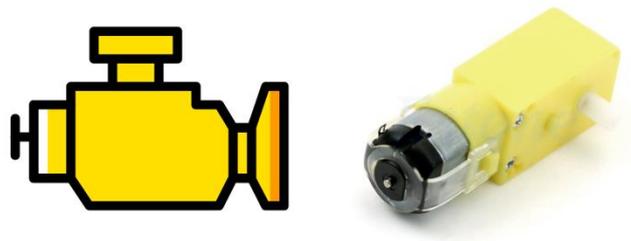
1. 什麼是 for 循環？
2. 什麼是陣列？
3. 什麼是蜂鳴器？
4. 什麼是 tone(), noTone()？

章節六：控制(三) – 馬達

I. 認識直流馬達



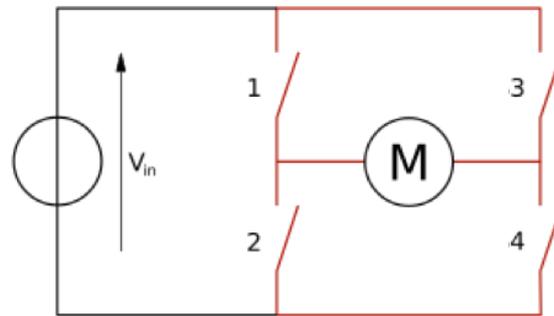
馬達有很多，這一課所採用的馬達是直流馬達，同樣也稱之為黃色馬達。要組成一個馬達，不可或缺的零件包括永久磁鐵和線圈。透過電源供電令線圈產生磁場，中間的永久磁鐵便會根據磁場產生轉向。相同的電流會將馬達轉向相同方向，令馬達能夠轉動。根據這個邏輯，如果把電源供應相反接上，便會令馬達轉向相反方向。



黃色馬達或直流馬達只有供電和接地兩條線。但是，我們之後會接觸的伺服馬達，就可能含有四至六條連接線，能讓馬達運轉指定的圈數或角度提供高穩定性。高準確度的操作一般用於工業控制，例如機械臂、影印機等等。

除了電風扇、吹風機及電鑽等電器會直接把負載（如：扇葉）和馬達相連，多數的動力裝置都會採用齒輪箱，滑輪等裝置來降低馬達的轉速，改變動力輸出方向以增加扭力。

II. 控制馬達正反轉的 H 橋式馬達控制電路



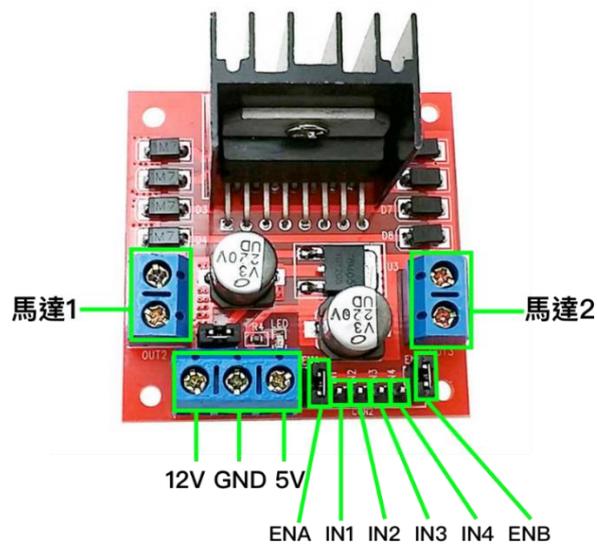
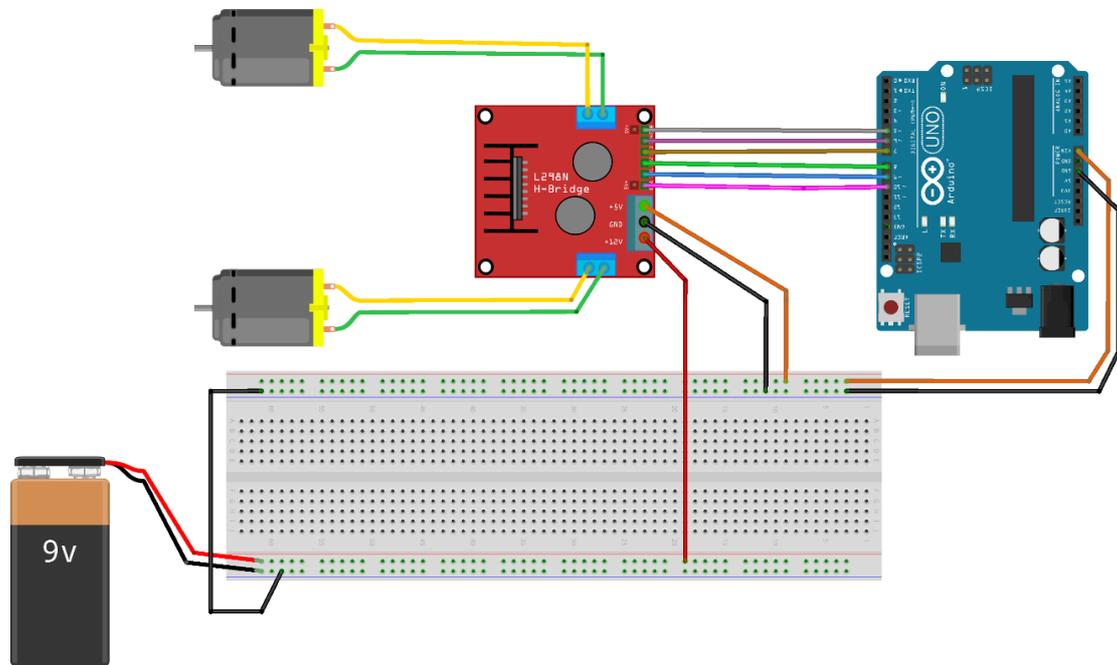
根據上面對馬達的介紹，我們清楚知道如果要令馬達轉向，便需要把電流方向調轉。但是，在現實生活中，我們不可能用這種方法來操控機械人。因此我們需要在馬達額外增加一個馬達驅動器，用作轉向之用。這個馬達驅動器叫做 H 橋式馬達控制。我們可以看到馬達被四個通道連住，操作方法：情況一是第二和第三通電令馬達向左轉；情況二是第一和第四通電令馬達向右轉。

III. 動手做 – 控制馬達

本課節希望同學能夠完成控制馬達的任務。一般而言，控制馬達需要馬達驅動器，馬達的轉動方向可分為順時針和逆時針旋轉，這分別對應不同的電流流向方向。因此，若果希望馬達能夠在程式中順時針或逆時針旋轉，我們必須利用馬達驅動器去達成這個任務。

一 連接 Arduino、直流馬達和馬達驅動器

我們根據以下方法連接 Arduino、直流馬達和馬達驅動器。



馬達驅動器放大圖

二 使用 Arduino 編程讓馬達驅動器驅動直流馬達

我們打開 Arduino IDE，並把 Arduino 連接電腦。在工具裡選擇 Arduino UNO 為開發板，並檢查序列埠是否已連接 Arduino。連接 Arduino 後，我們需要輸入以下程式，並把程式上傳到 Arduino 當中。

雙直流馬達的程式碼全圖，可參考檔案 1_6_two_dc_motor

```
1 //MotorA
2 const int motorENA = 10;
3 const int motorIN1 = 9;
4 const int motorIN2 = 8;
5
6 //MotorB
7 const int motorENB = 5;
8 const int motorIN3 = 7;
9 const int motorIN4 = 6;
10
11 void setup(){
12     pinMode(motorENA, OUTPUT);
13     pinMode(motorIN1, OUTPUT);
14     pinMode(motorIN2, OUTPUT);
15     pinMode(motorENB, OUTPUT);
16     pinMode(motorIN3, OUTPUT);
17     pinMode(motorIN4, OUTPUT);
18 }
19
20 void loop(){
21     motorForward();
22     delay(2000);
23     motorBackward();
24     delay(2000);
25     motorStop();
26     delay(1000);
27 }
28
29 void motorStop(){
30     digitalWrite(motorENA, HIGH);
31     digitalWrite(motorIN1, LOW);
32     digitalWrite(motorIN2, LOW);
33     digitalWrite(motorENB, HIGH);
34     digitalWrite(motorIN3, LOW);
35     digitalWrite(motorIN4, LOW);
36 }
37
38 void motorForward(){
39     digitalWrite(motorENA, HIGH);
40     digitalWrite(motorIN1, HIGH);
41     digitalWrite(motorIN2, LOW);
42     digitalWrite(motorENB, HIGH);
43     digitalWrite(motorIN3, HIGH);
44     digitalWrite(motorIN4, LOW);
45 }
46
47 void motorBackward(){
48     digitalWrite(motorENA, HIGH);
49     digitalWrite(motorIN1, LOW);
50     digitalWrite(motorIN2, HIGH);
51     digitalWrite(motorENB, HIGH);
52     digitalWrite(motorIN3, LOW);
53     digitalWrite(motorIN4, HIGH);
54 }
```

定義腳位

設定腳位為輸出

主程式

馬達驅動上的
控制邏輯

三 程式內容說明

1. 第一個部分為定義腳位，這樣定義能讓我們快速編寫程式。

```
1 //MotorA
2 const int motorENA = 10;
3 const int motorIN1 = 9;
4 const int motorIN2 = 8;
5
6 //MotorB
7 const int motorENB = 5;
8 const int motorIN3 = 7;
9 const int motorIN4 = 6;
```

2. 第二部分為設定腳位的工作模式，這裡是跟馬達驅動器進行溝通。Arduino Uno 是發送數據到馬達驅動器，因此腳位模式選擇輸出。

```
11 void setup(){
12     pinMode(motorENA, OUTPUT);
13     pinMode(motorIN1, OUTPUT);
14     pinMode(motorIN2, OUTPUT);
15     pinMode(motorENB, OUTPUT);
16     pinMode(motorIN3, OUTPUT);
17     pinMode(motorIN4, OUTPUT);
18 }
```

3. 重複的部分是馬達向前走兩秒後、退後兩秒後、然後停止，等待一秒後重複向前行向後行的動作。

```
20 void loop(){
21     motorForward();
22     delay(2000);
23     motorBackward();
24     delay(2000);
25     motorStop();
26     delay(1000);
27 }
```

4. 這些程式是根據馬達驅動器所接收的訊號來定義，例如當 IN1、IN2、IN3、IN4 也是低電壓時，馬達便會處於停止的狀態。

```
29 void motorStop(){
30     digitalWrite(motorENA, HIGH);
31     digitalWrite(motorIN1, LOW);
32     digitalWrite(motorIN2, LOW);
33     digitalWrite(motorENB, HIGH);
34     digitalWrite(motorIN3, LOW);
35     digitalWrite(motorIN4, LOW);
36 }
37
38 void motorForward(){
39     digitalWrite(motorENA, HIGH);
40     digitalWrite(motorIN1, HIGH);
41     digitalWrite(motorIN2, LOW);
42     digitalWrite(motorENB, HIGH);
43     digitalWrite(motorIN3, HIGH);
44     digitalWrite(motorIN4, LOW);
45 }
46
47 void motorBackward(){
48     digitalWrite(motorENA, HIGH);
49     digitalWrite(motorIN1, LOW);
50     digitalWrite(motorIN2, HIGH);
51     digitalWrite(motorENB, HIGH);
52     digitalWrite(motorIN3, LOW);
53     digitalWrite(motorIN4, HIGH);
54 }
```

VIII. 溫習問題

1. 什麼是直流馬達？
2. 什麼是 H 橋式馬達控制？

章節七：單元專題研習 – 測謊裝置

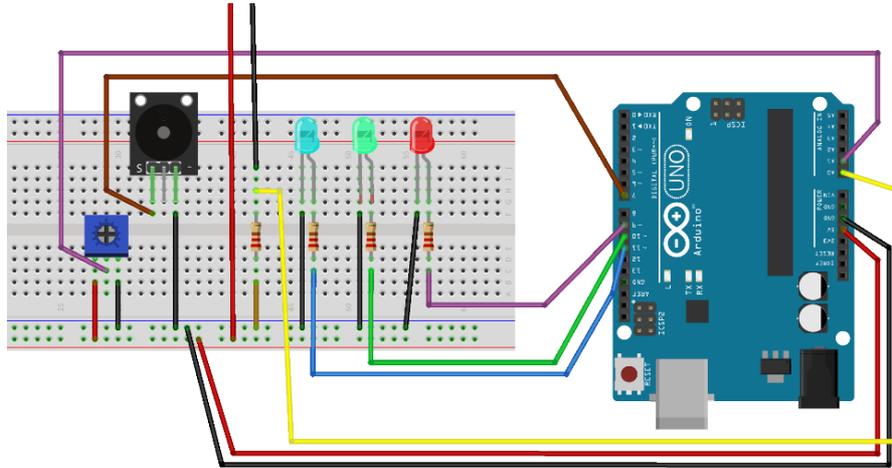
機器人學中包含了不同的元素，如科學、科技、工程等。同學進行單元專題研習時，需要利用不同的學科知識，例如一開始你們需要由科學研究做起，探討電流的原理和人體皮膚的特性，再找出如何利用科學去判斷一個人有否說謊。其次你們需要構思設計，如何利用單元一所學到的電子元件去設計一個機械人。製造完原形之後，需要進行測試。最後，你們會根據測試結果去改良你們的設計。在進行專題研習中，期望同學可以得到更多科學知識和工程知識，亦希望透過與人合作，增強溝通技巧、解難和領導能力。

工程設計 階段／步驟	相關知識	探究和設計考慮	應用相關知識 和探究結果
界定問題 (確認要求和限制)	瞭解測謊機的運作原理，以及參考現行或以前測謊機所用的標準	釐定設計要求和限制	/
研究	<ul style="list-style-type: none"> 電流的原理和不同物件有不同的導電性 人體皮膚的導電性 不同情緒、環境和天氣對皮膚導電性的影響(例如緊張會導致皮膚導電性下降) 量度皮膚導電性的方法 	選擇處理方法(如在課堂上進行，可選一項或多項，視乎教學目標、課時及材料限制而定)	/
構想設計	<ul style="list-style-type: none"> 選擇 Arduino 為這次實驗的處理器 運用類比訊號來量度皮膚的導電性 使用 LED 燈和蜂鳴器來顯示測謊者被問話時的反應 	預測： 說謊時因緊張而令皮膚導電性下降，類比訊號能有效量度皮膚導電性的變化	先使用 Arduino 進行測試
測試模型	<ul style="list-style-type: none"> 影響皮膚導電性的因素，包括人體排汗、環境濕度和情緒 	假設情緒是影響人體皮膚導電性的主要因素，然後針對其餘變因逐一進行公平	/

	<ul style="list-style-type: none"> 根據電壓分配定則 (Potential Divider) 來設計電路圖量度皮膚的導電性 使用紅色和綠色 LED 燈以及蜂鳴器來顯示測謊者被問話時的反應 使用藍色 LED 燈來顯示測謊機的接觸狀態 使用定位器減低測謊機使用在在不同人身上的誤差 	<p>測試,以驗證每種變因對效能的影響</p> <p>公平測試方法:</p> <p>找出人體排汗怎樣影響皮膚導電性</p> <ul style="list-style-type: none"> 獨立變數:人體排汗多寡 因變數:皮膚導電性 控制變數:運動量、人體體溫、新陳代謝等 <p>找出不同環境濕度怎樣影響皮膚導電性</p> <ul style="list-style-type: none"> 獨立變數:環境濕度 因變數:皮膚導電性 控制變數:氣溫、天氣、通風設施等 	/
解決設計／製作／測試過程中遇到的問題	<p>測試過程顯示要得到較佳效果及令量度更為準確,需要:</p> <ul style="list-style-type: none"> 測謊者被問話時不斷接觸測謊機 每位使用者使用前調教測謊機內的定位器 在設有良好通風設備的地方進行測試 	將這些發現納入為是項工程設計的必要程序	/
分析及評估測試結果及出現的問題	<ul style="list-style-type: none"> 電壓和電阻之間的線性關係 [歐姆定律 – Ohms Law] (工程知識) 測謊機的準確度 	<ul style="list-style-type: none"> 利用歐姆定律分析數據,評估設計是否已達到既定目標 根據測試結果,評定歐姆定律是測謊機能成功測謊的關鍵 單靠量度皮膚導電性作為測謊條件並不準確 	參考分析及評估所得,研究改良方法

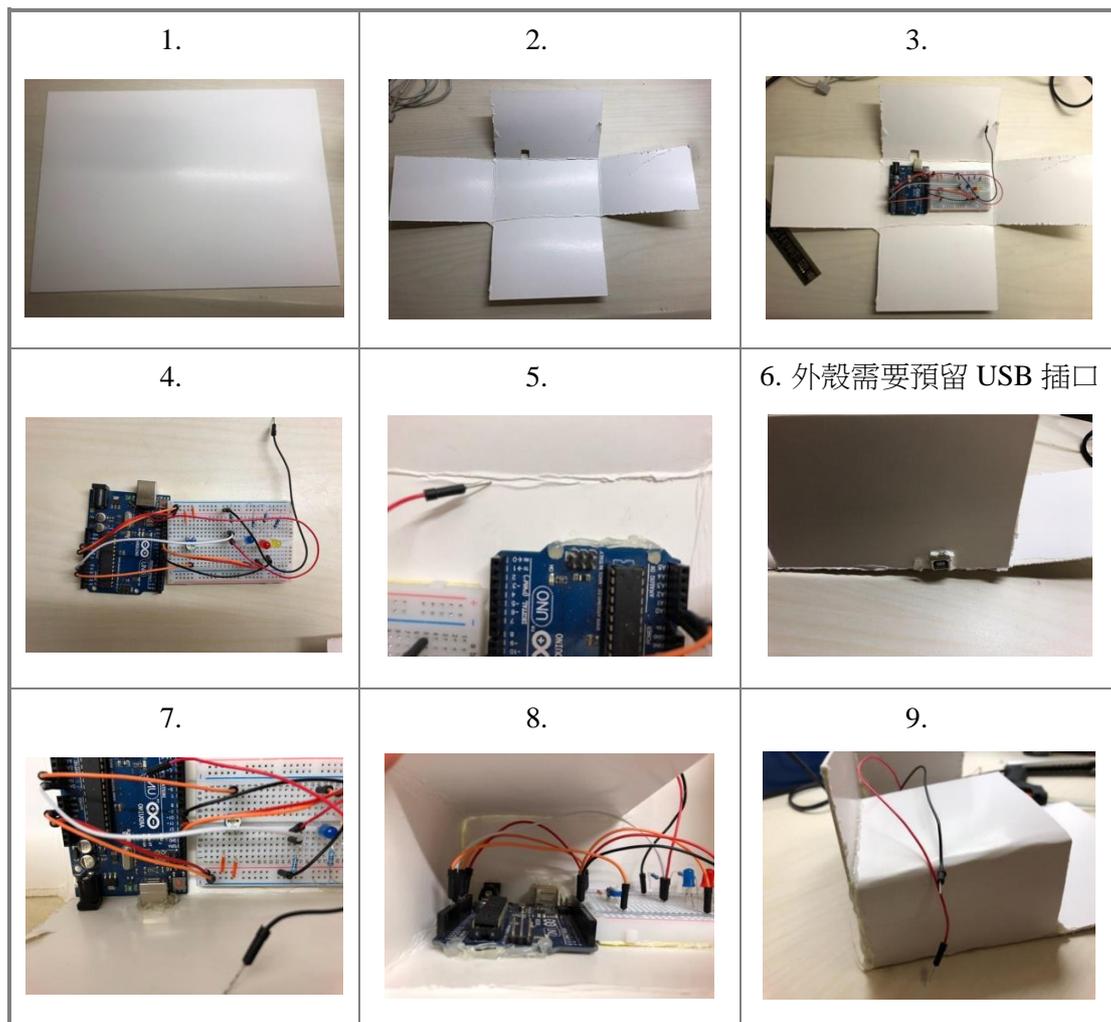
一 測謊裝置的電路製作

我們根據以下方法設置測謊裝置的電路：



二 測謊裝置的外觀設計

同學可根據自己的構思，為你的測謊裝置設計及製作其外殼，用以保護 Arduino 板及其他電子原件。以下是以卡紙製測謊裝置外殼的參



三 使用 Arduino 編程測謊裝置

我們打開 Arduino IDE，並把 Arduino 連接電腦。在工具裡選擇 Arduino UNO 為開發板，並檢查序列埠是否已連接 Arduino。連接 Arduino 後，我們需要輸入以下程式，並把程式上傳到 Arduino 當中。

測謊裝置的程式碼全圖，可參考檔案 1_7_lie_detector

```
1 const int red_led_pin = 9;
2 const int green_led_pin = 10;
3 const int blue_led_pin = 11;
4 const int buzzer_pin = 7;
5 const int offset_pin = 15;
6 const int sensor_pin = 14;
7 int band = 25;
8
9 -----
9 void setup() {
10   pinMode(red_led_pin, OUTPUT);
11   pinMode(green_led_pin, OUTPUT);
12   pinMode(blue_led_pin, OUTPUT);
13   pinMode(buzzer_pin, OUTPUT);
14   pinMode(offset_pin, INPUT);
15   pinMode(sensor_pin, INPUT);
16   Serial.begin(9600);
17 }
18
19 -----
19 void loop() {
20   int offset = analogRead(offset_pin);
21   int value = analogRead(sensor_pin);
22   if (value > offset - band){
23     digitalWrite(red_led_pin, HIGH);
24     buzzer();
25     Serial.println("Lie test fail.");
26   }
27   else if (value < offset - band){
28     digitalWrite(blue_led_pin, HIGH);
29     Serial.println("Lie detector ready");
30   }
31   else{
32     digitalWrite(green_led_pin, HIGH);
33     Serial.println("Lie test pass.");
34   }
35 }
36
37 -----
37 void buzzer(){
38   for(int i = 0; i < 1000; i++){
39     digitalWrite(buzzer_pin, HIGH);
40     delayMicroseconds(100);
41     digitalWrite(buzzer_pin, LOW);
42     delayMicroseconds(100);
43   }
44 }
```

定義變數

把變數設定為輸出，
打開串行通訊頻道

用類比輸入改變誤差值，透
過偵測水份判定是否說謊

如測試不合格蜂鳴器會響起

四 操作原理

這個測謊機，假定了使用者說謊時手會冒汗，由於冒汗時會輕微改變手指的導電特性，這種特性可以通過接上電路讓 Arduino 能感應其改變，通過編寫程式作判斷，便能製作出這台測謊機。

五 程式內容說明

1. 定義變數。

```
1 const int red_led_pin = 9;
2 const int green_led_pin = 10;
3 const int blue_led_pin = 11;
4 const int buzzer_pin = 7;
5 const int offset_pin = 15;
6 const int sensor_pin = 14;
7 int band = 25;
```

在第一部份中，我們定義變數。首先是紅、綠、藍三粒發光二極管的插腳設定分別為 9、10、11。我們定義蜂鳴器的接腳 7，至於兩個類比接腳分別定義為 14、15。最後加上我們測試出來的標準定義 25。

2. 把變數設定為輸出，打開串行通信頻道。

```
9 void setup() {
10   pinMode(red_led_pin, OUTPUT);
11   pinMode(green_led_pin, OUTPUT);
12   pinMode(blue_led_pin, OUTPUT);
13   pinMode(buzzer_pin, OUTPUT);
14   pinMode(offset_pin, INPUT);
15   pinMode(sensor_pin, INPUT);
16   Serial.begin(9600);
17 }
```

在第二部份中，我們需要把第一部份所定義的變數設為輸出和輸入，例如是紅、綠、藍三粒發光二極管和蜂鳴器及兩個類比接腳，並打開串行通信頻道傳輸率為 9600。

3. 用類比輸入改變誤差值，透過偵測水分去判定是否說謊。

```
19 void loop() {
20   int offset = analogRead(offset_pin);
21   int value = analogRead(sensor_pin);
22   if (value > offset - band){
23     digitalWrite(red_led_pin, HIGH);
24     buzzer();
25     Serial.println("Lie test fail.");
26   }
27   else if (value < offset - band){
28     digitalWrite(blue_led_pin, HIGH);
29     Serial.println("Lie detector ready");
30   }
31   else{
32     digitalWrite(green_led_pin, HIGH);
33     Serial.println("Lie test pass.");
34   }
35 }
```

在第三個部份中，我們首先透過函數“analogRead()”拿取在兩個類比輸入接腳的數值，如果兩條接觸了我們首次電線收取到的數值大於預先定立的標準，我們會亮起紅燈，響起蜂鳴器並打印出“Lie detector fail.”的字眼。假如數值沒有超過，便會打出“Lie detector ready”，最後打出“Lie test pass”。

4. 測試不合格蜂鳴器會鳴響。

```
37 void buzzer(){
38   for(int i = 0; i < 1000; i++){
39     digitalWrite(buzzer_pin, HIGH);
40     delayMicroseconds(100);
41     digitalWrite(buzzer_pin, LOW);
42     delayMicroseconds(100);
43   }
44 }
```

在第四部份中，我們定了一個叫蜂鳴器的方法，並寫了一個重複一千次的回轉，不斷地開關像蜂鳴器、警報器的聲音。