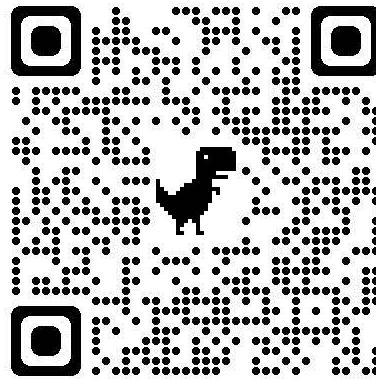


**完成以上所有步驟 = 電子零件正常運作**



## Starter Project 1 - Graphic Sensors

<https://www.youtube.com/watch?v=29ZIiJJQTFI>



# Arduino IDE

Arduino IDE 1.8.19 是一個的開發環境，提供代碼編輯器、實時錯誤檢查。



# 使用介面簡介

File Edit Sketch Tools Help



sketch\_sep06a



```
void setup() {  
    // put your setup code here, to run once:  
  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```

Arduino/Genuine Uno on COM1

啟動 Arduino IDE：您將看到主界面，包括編輯區域、工具欄和菜單欄。

sketch\_may30a | Arduino 1.8.19

檔案 編輯 儲存碼 工具 說明

sketch\_may30a

```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }  
4  
5 void loop() {  
6     // put your main code here, to run repeatedly:  
7 }  
8  
9 }
```

儲存專案

開啟範例或舊檔

開啟新專案

寫入到 arduino 上

編譯程式碼

查看序列埠，  
亦即與 arduino 互動

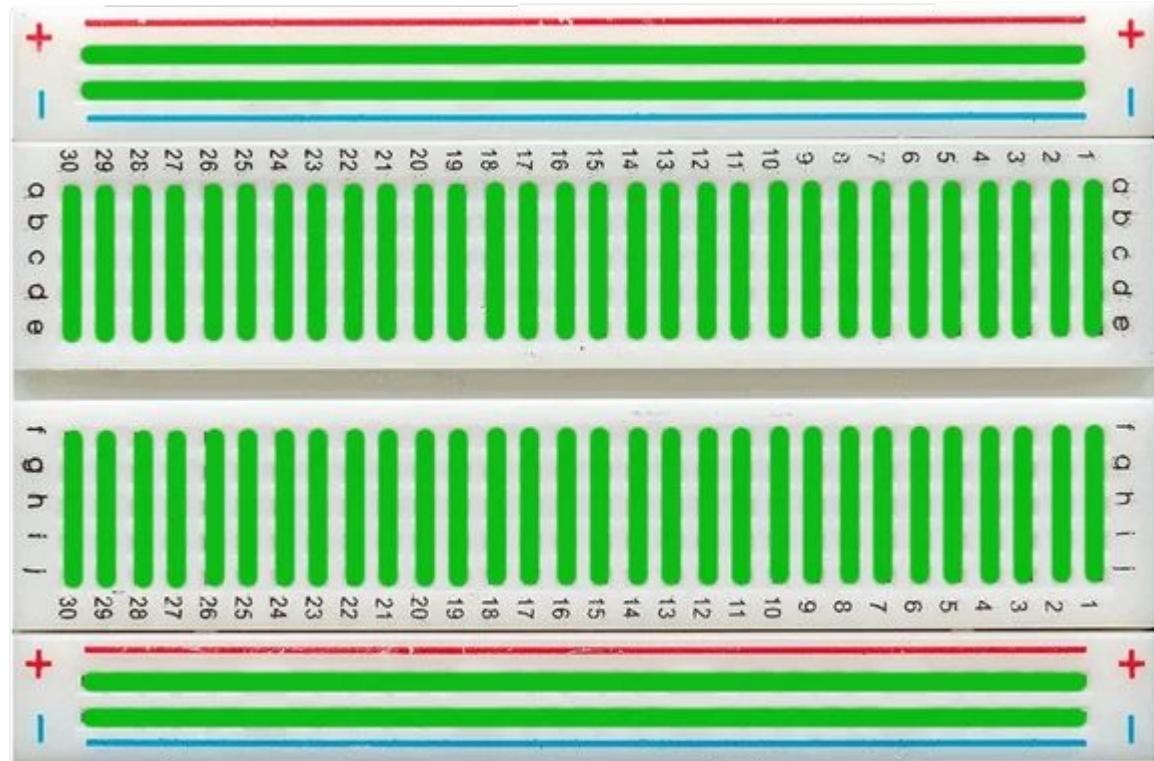
目前選擇的 arduino 硬體

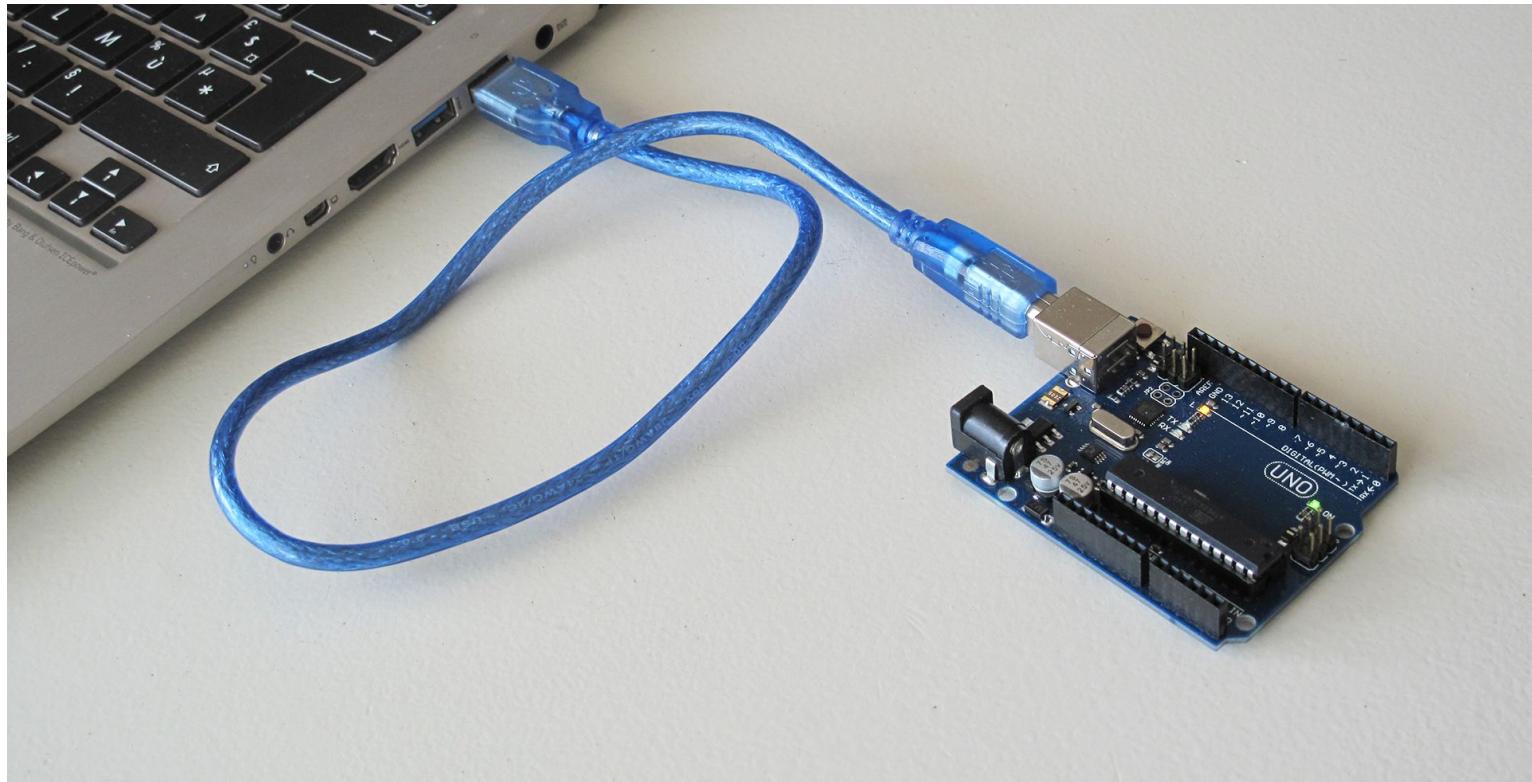
Arduino Nano, ATmega328P (Old Bootloader) 於 COM4

在使用 開發板 做到影片中的效果前

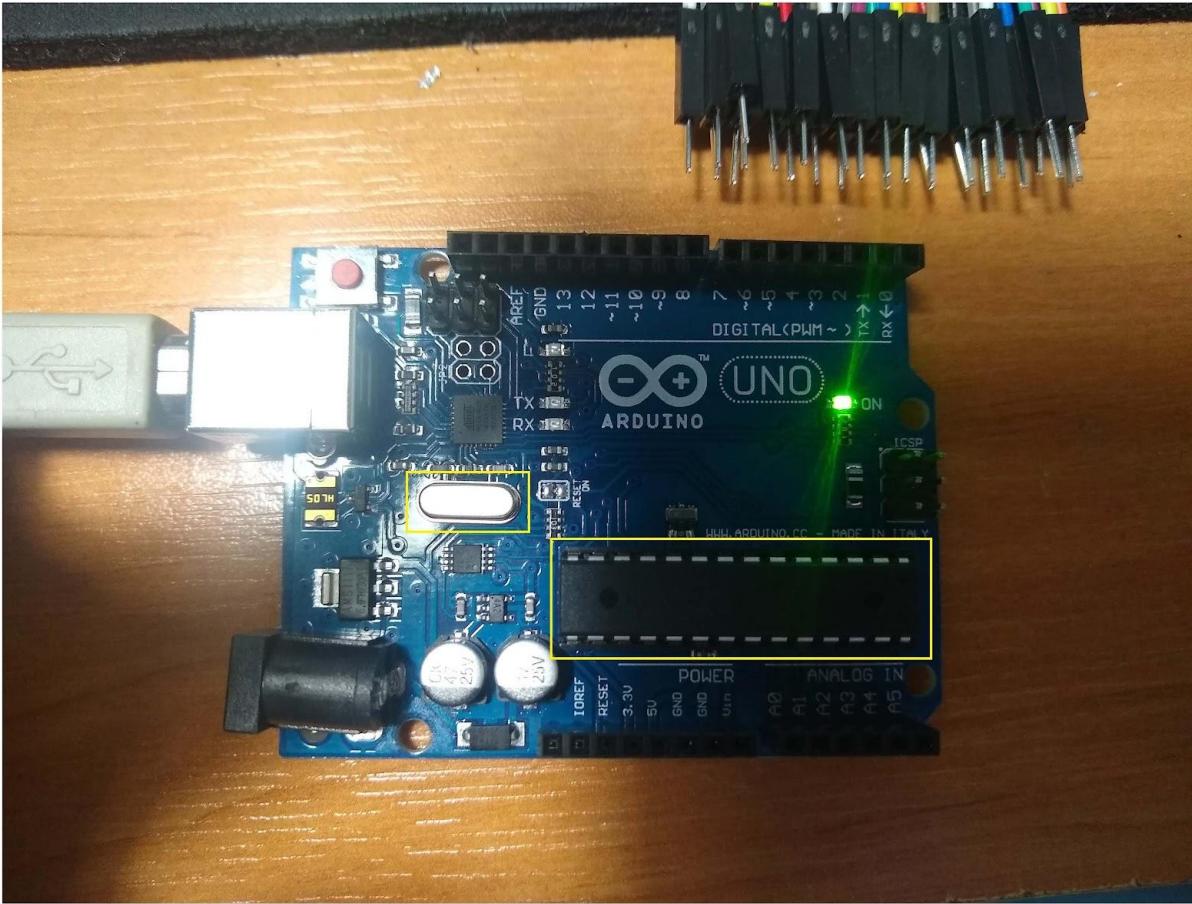
## 要做的準備工作

麵包板是電子原型設計中常見的工具，用於在無需焊接的情況下建立和測試電路。麵包板上的孔洞按照特定的方式連接，讓使用者能夠插入電子元件並使用導線將它們連接起來。每一列的五個孔洞是相互連接的，可以用來連接元件，而兩側的長條則是用來提供電源。在使用麵包板進行電路連接時，請確保將所有元件和導線正確無誤地插入相應的孔洞，並確保電源和接地線的正確連接，以避免任何可能的短路或故障。





USB連接電腦及開發板



先確定 LED 燈亮起

File Edit Sketch Tools Help



Auto Format Ctrl+T  
Archive Sketch  
Fix Encoding & Reload  
Manage Libraries... Ctrl+Shift+I  
Serial Monitor Ctrl+Shift+M  
Serial Plotter Ctrl+Shift+L  
WiFi101 / WiFiNINA Firmware Updater

Board: "Arduino Uno"

Port

Get Board Info

Programmer: "AVRISP mkII"

Burn Bootloader

Boards Manager...

Arduino AVR Boards

Bare Conductive Boards (in sketchbook) &gt;

run repeatedly:

Arduino Yún

• Arduino Uno

Arduino Duemilanove or Diecimila

Arduino Nano

Arduino Mega or Mega 2560

Arduino Mega ADK

Arduino Leonardo

Arduino Leonardo ETH

選擇開發板：在菜單欄，選擇「工具」>「開發板」，然後選擇您所使用的 Arduino 開發板。

檔案 編輯 草稿碼 工具 說明



sketch\_may30a

```
void setup()
// put your
}
```

```
void loop()
// put your
}
```

自動格式化

Ctrl+T

封存草稿碼

修正編碼並重新載入

管理程式庫...

Ctrl+Shift+I

序列埠監控視窗

Ctrl+Shift+M

序列繪圖家

Ctrl+Shift+L

WiFi101 / WiFiNINA Firmware Updater

開發板: "Arduino Nano"

&gt;

處理器: "ATmega328P (Old Bootloader)"

&gt;

序列埠: "COM4"

&gt;

取得開發板資訊

&gt;

燒錄器: "BusPirate as ISP"

&gt;

燒錄Bootloader

&gt;

序列埠

COM1

COM2

COM3

COM4

選擇連接埠：再次點擊「工具」，選擇「連接埠」，然後選擇 Arduino 開發板所連接的串口。

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch\_may30a | Arduino 1.8.19
- Menu Bar:** 檔案 編輯 草稿碼 工具 說明
- Toolbar:** Standard icons for file operations (New, Open, Save, Print, Upload, Download).
- Sketch Name:** sketch\_may30a
- Code Editor:** The code is divided into two sections:
  - setup() Section:** Contains the code: `void setup() { // put your setup code here, to run once; }` with the explanatory text "初始化 arduino 硬體的程式碼" (Initialize Arduino hardware program code) overlaid.
  - loop() Section:** Contains the code: `void loop() { // put your main code here, to run repeatedly; }` with the explanatory text "實際運作的程式碼" (Actual running program code) overlaid.
- Status Bar:** Arduino Nano, ATmega328P (Old Bootloader) 於 COM4

編寫程式碼：在編輯區域編寫您的 Arduino 程式碼，使用 `setup()` 和 `loop()` 函數組織程式結構。

# 編碼例子：Blink

```
void setup() {  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(1000);  
    digitalWrite(LED_BUILTIN, LOW);  
    delay(1000);  
}
```

Blink | Arduino 1.8.19

檔案 編輯 草稿碼 工具 說明

先按這裡做編譯

```
22 https://www.arduino.cc/en/Tutorial/BuiltinExamples/Blink
23 */
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 }
```

這裡會顯示編譯的結果！  
若有錯誤，就得要修訂才行

編譯完畢。

草稿碼使用了 924 bytes (3%) 的程式儲存空間。上限為 30720 bytes。  
全域變數使用了 9 bytes (0%) 的動態記憶體，剩餘 2039 bytes 給區域變數。上限為 2048 bytes。

Arduino Nano, ATmega328P (Old Bootloader) 於 COM4

驗證程式碼：點擊工具欄上的「✓」按鈕，Arduino IDE 將對您的程式碼進行編譯和錯誤檢查。

先按下上傳按鈕之後

會顯示上傳的結果在這裡！

有問題時，參考這裡的方案

```
22 // https://www.arduino.cc/en/Tutorial/BuiltinExamples/Blink
23 */
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 }
```

上傳完畢。

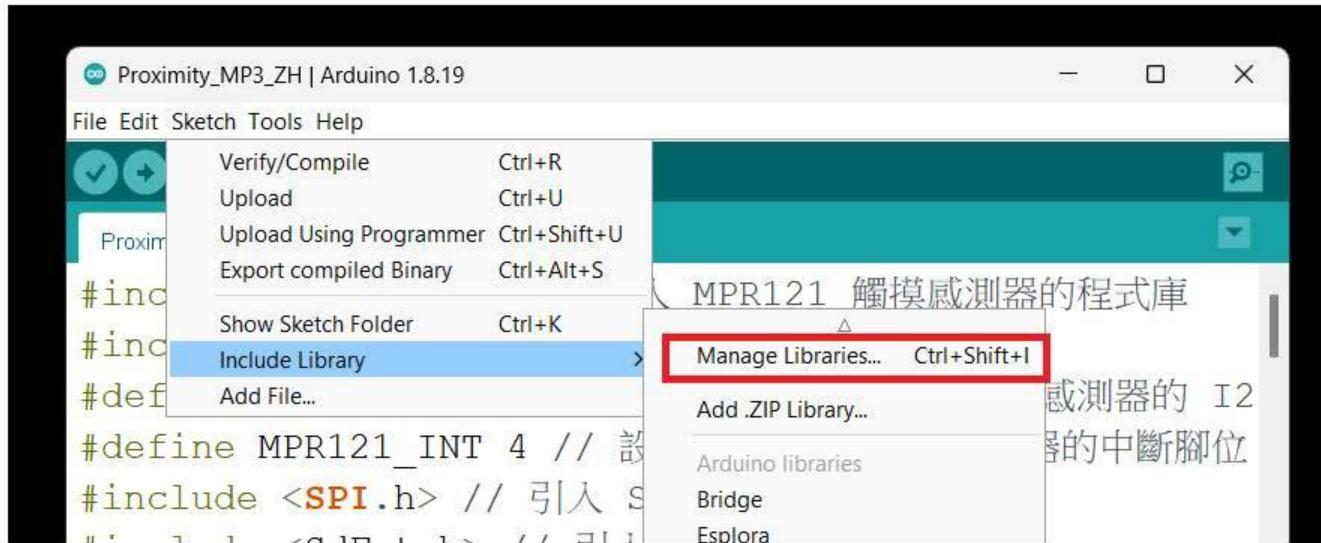
avrdude: 924 bytes of flash verified

avrdude done. Thank you.

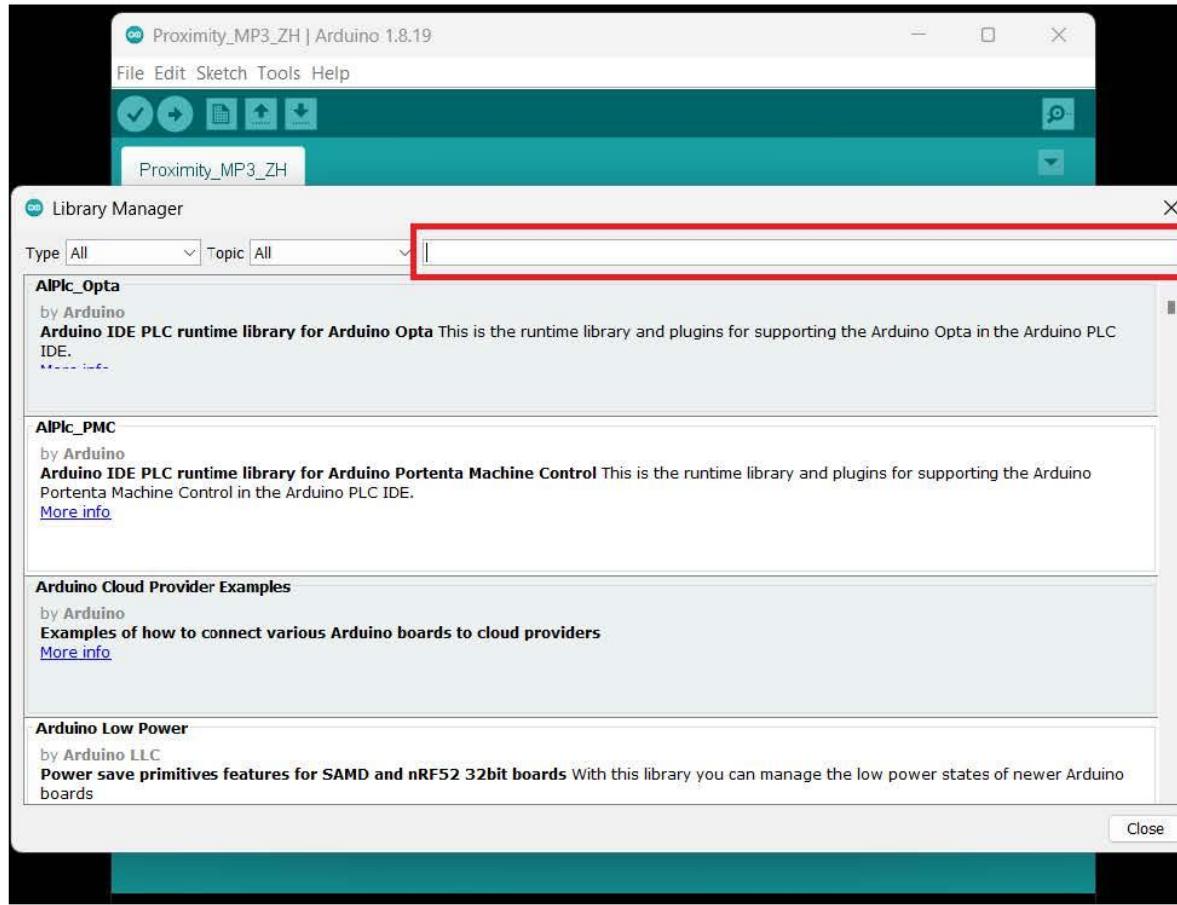
Arduino Nano, ATmega328P (Old Bootloader) 於 COM4

上傳程式碼：確保程式碼無誤後，點擊工具欄上的「→」按鈕，將程式碼上傳到 Arduino 開發板。

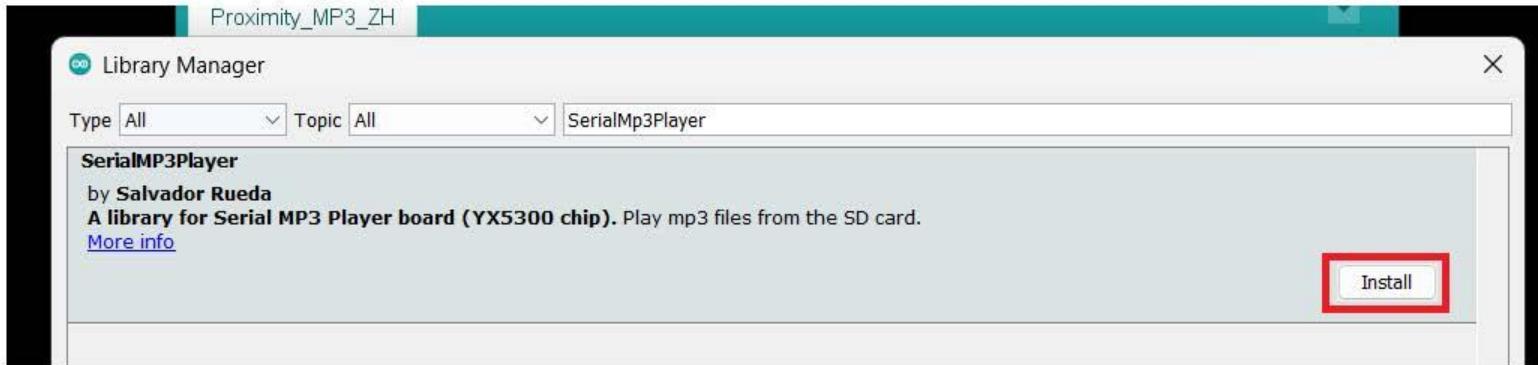
**完成以上所有步驟 = 電子零件正常運作**



在 Sketch 工具欄下, 把滑鼠移到 Inculde Library,  
選取 Manage Libraries



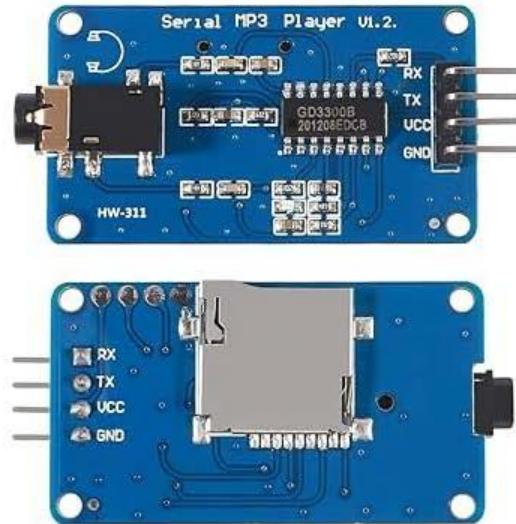
輸入 SerialMp3Player 搜索編碼庫



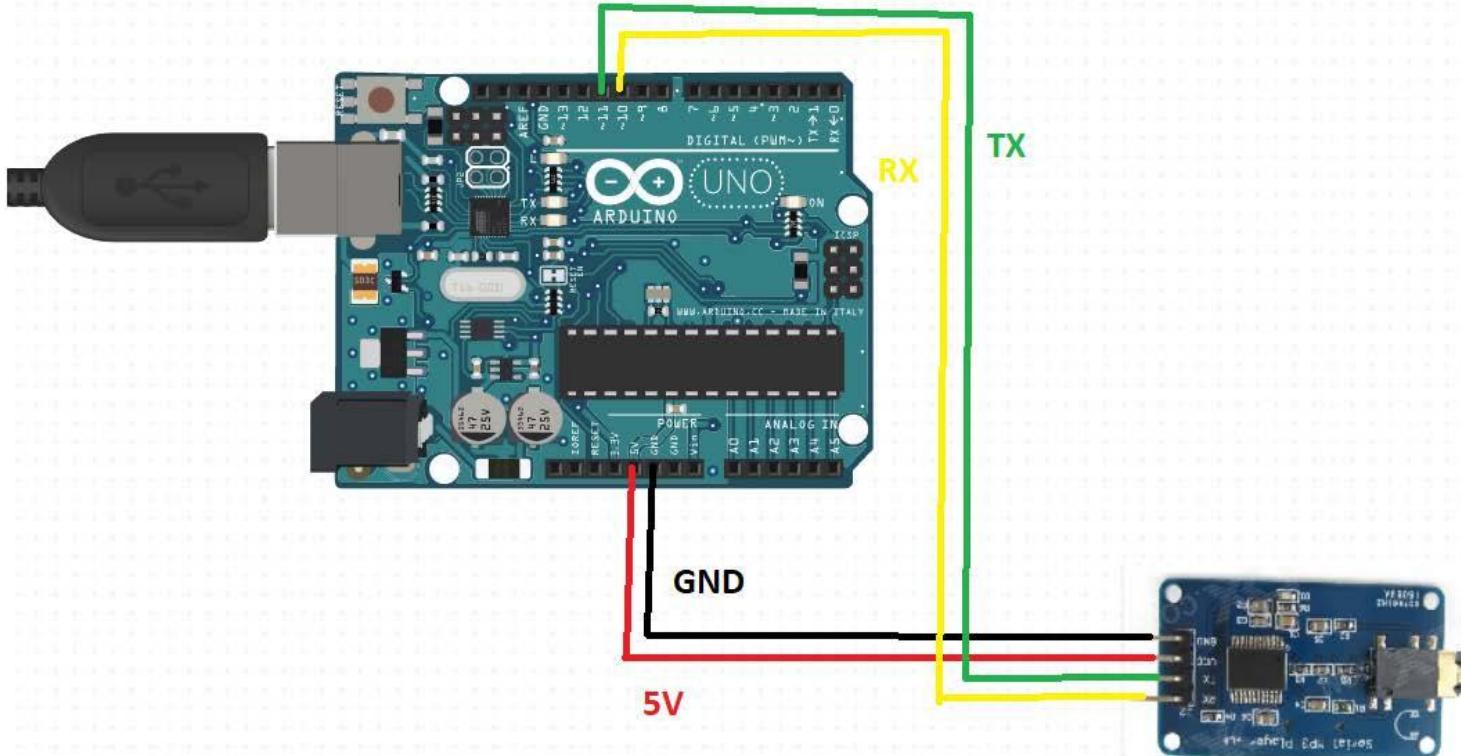
按 Install 安裝編碼庫



另外再安裝 CapacitiveSensor 編碼庫



把 Serial MP3播放模組YX5300/YX6300 接到 Arduino UNO





# 編程時間

# 將以下的編碼複製到編輯區域

```
#include "SerialMP3Player.h" // 使用MP3版的編碼庫library
#define TX 10 //to MP3 board RX //定義ARDUINO TX到MP3 RX引腳連接
#define RX 11 //to MP3 board TX //定義ARDUINO RX到MP3 TX引腳連接

SerialMP3Player mp3(RX, TX); // 定義起動MP3相關的TX, RX

//設定：有電源起動時執行一次的程序
void setup() {
    Serial.begin(9600); // 起動serial介面
    mp3.begin(9600); // 開始MP3版的連接
    delay(500); // 等待起動
    mp3.sendCommand(CMD_SEL_DEV, 0, 2); //選取 sd-card
    delay(500); // 等待起動
    mp3.setVol(15); // 設定音量
    mp3.play(1); //歌曲於SD CARD內的次序
}

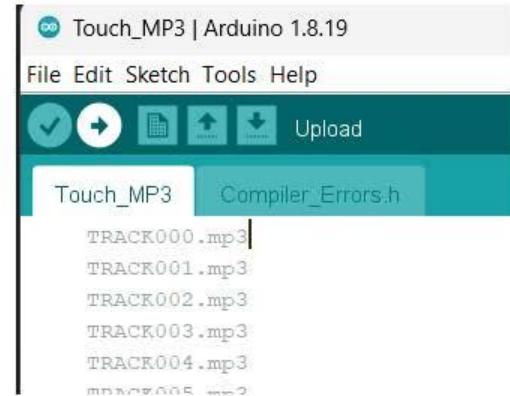
//迴圈：處理器不停執行的程序
void loop() {
}
```



## 先編譯程式碼

```
Done compiling.  
  
Sketch uses 24038 bytes (83%) of program storage space. Maximum is 28672 bytes.  
Global variables use 1459 bytes (56%) of dynamic memory, leaving 1101 bytes for local variables  
  
16 Bare Conductive Touch Board on COM3
```

如果系統控制台 (Console) 內沒有出現問題, 就可以做下一步。



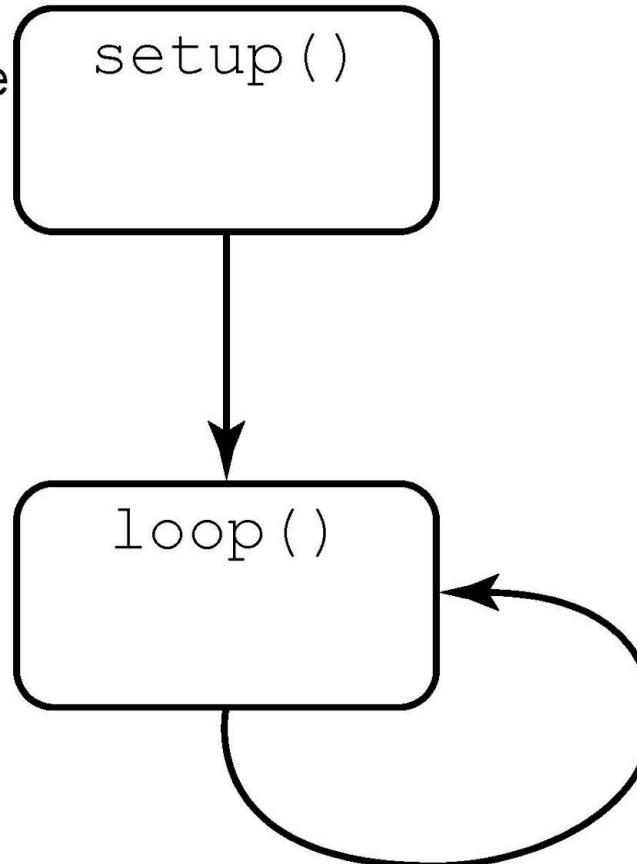
上傳程式碼

## 有關 void setup() 及 void loop()

*void setup()* 是 Arduino 程式的一個函式，用於初始化設定，例如設定序列通訊、腳位模式、變數初值等。當 Arduino 板子啟動後，會**自動執行一次** *setup()*。

*void loop()* 是 Arduino 程式的另一個函式，用於執行主要的程式邏輯，例如讀取腳位數值、判斷條件、控制輸出等。*loop()* **會一直執行**，直到板子關機或重置。

Execution starts here  
Execute once.

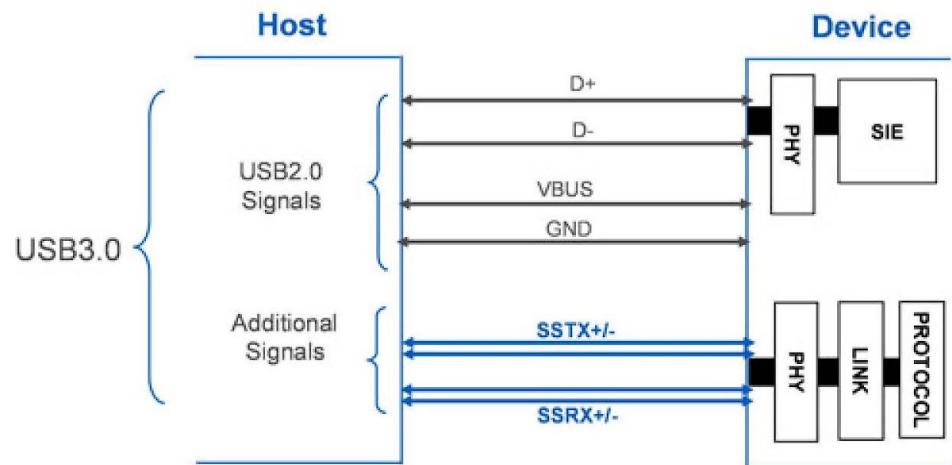


Execute repeatedly.

```
#include "SerialMP3Player.h">// 使用MP3版的編碼庫library  
  
#include <CapacitiveSensor.h>// 使用可感應導電墨水的CAP SENSE編碼庫library  
  
#define TX 10 //to MP3 board RX //定義ARDUINO TX到MP3 RX引腳連接  
#define RX 11 //to MP3 board TX //定義ARDUINO RX到MP3 TX引腳連接
```

Arduino程式碼包含兩個庫：SerialMP3Player和CapacitiveSensor。SerialMP3Player用於控制MP3播放板，CapacitiveSensor用於讀取導電墨水的觸碰輸入。

#define指令創建了兩個名為TX和RX的常量，分別設為10和11，用於表示Arduino與MP3播放板之間串行通訊的接口引腳。



```
//設定：有電源起動時執行一次的程序
void setup() {
    Serial.begin(9600);          // 起動serial介面
    mp3.begin(9600);            // 開始MP3版的連接
    delay(500);                 // 等待起動
    mp3.sendCommand(CMD_SEL_DEV, 0, 2); //選取 sd-card
    delay(500);                 // 等待起動
    mp3.setVol(15);             // 設定音量
    mp3.play(1);                //歌曲於SD CARD內的次序
}
```

setup()函數初始化串口和MP3播放器，並設定音量和播放歌曲。

*Serial.begin(9600)* 和 *mp3.begin(9600)* 開始串口和MP3播放器的通訊，傳輸速率為9600位元/秒。

*mp3.sendCommand(CMD\_SEL\_DEV, 0, 2)* 選擇SD卡作為存儲設備。

*mp3.setVol(15)* 設定音量為15, *mp3.play(1)* 播放SD卡中的第一首歌曲。

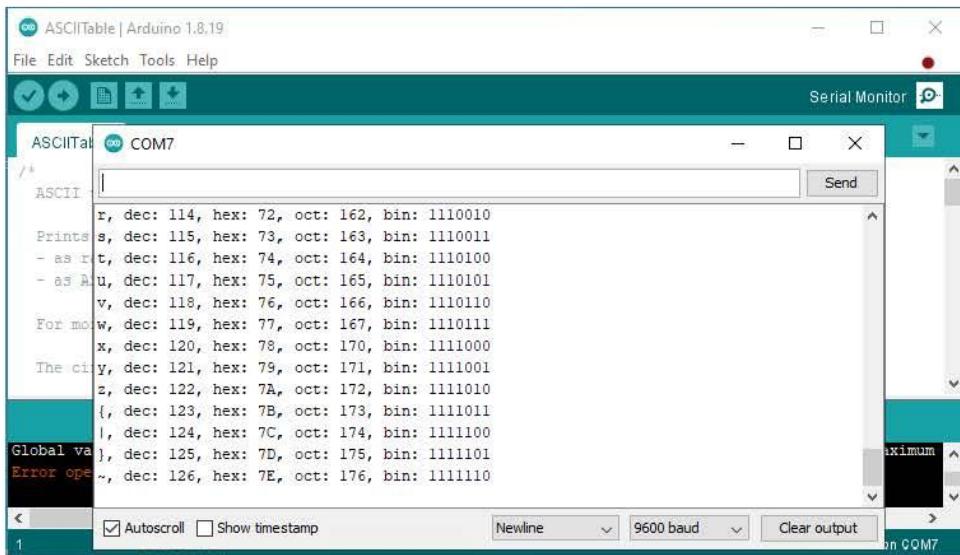
*delay(500)* 在相關操作之間提供必要的等待時間，以確保操作的成功執行。

# 編碼內容

- *Serial.begin();*

*Serial.begin()* 是 Arduino 程式語言的一個函式，用於啟動序列通訊介面，在程式執行時可以透過該介面與外部設備或電腦進行資料傳輸。

- 右圖內的是 Serial Monitor



## 有關上一段編碼的補充

- *pinMode();*

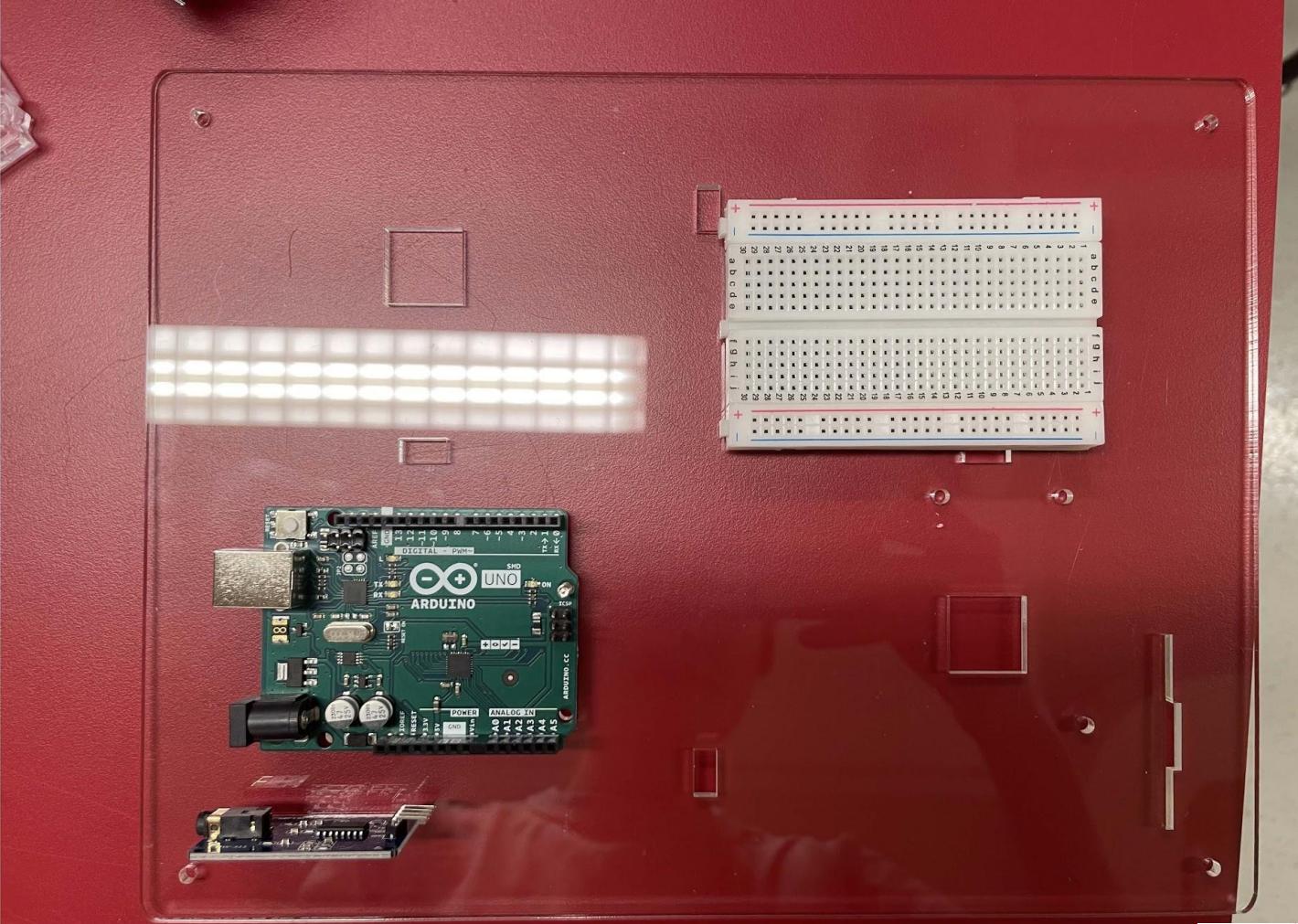
*pinMode()* 是 Arduino 程式語言的一個函式，用於設定腳位的輸入或輸出模式，以便與外部電路或設備進行數位或類比訊號的傳輸。

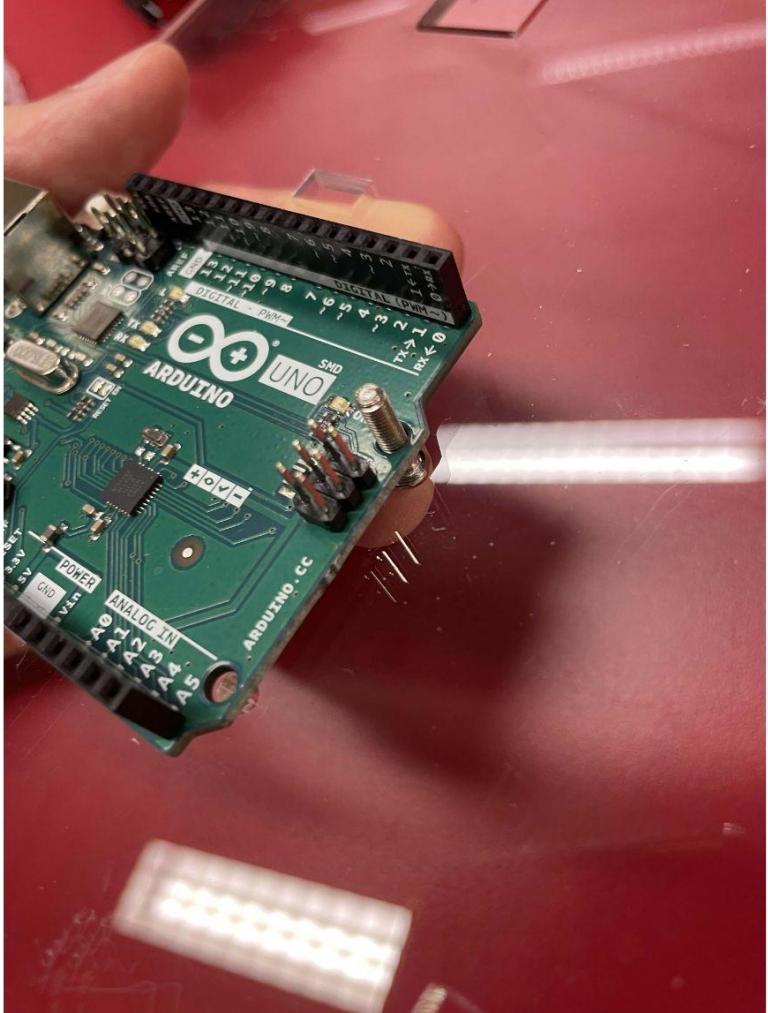
- *Serial.println();*

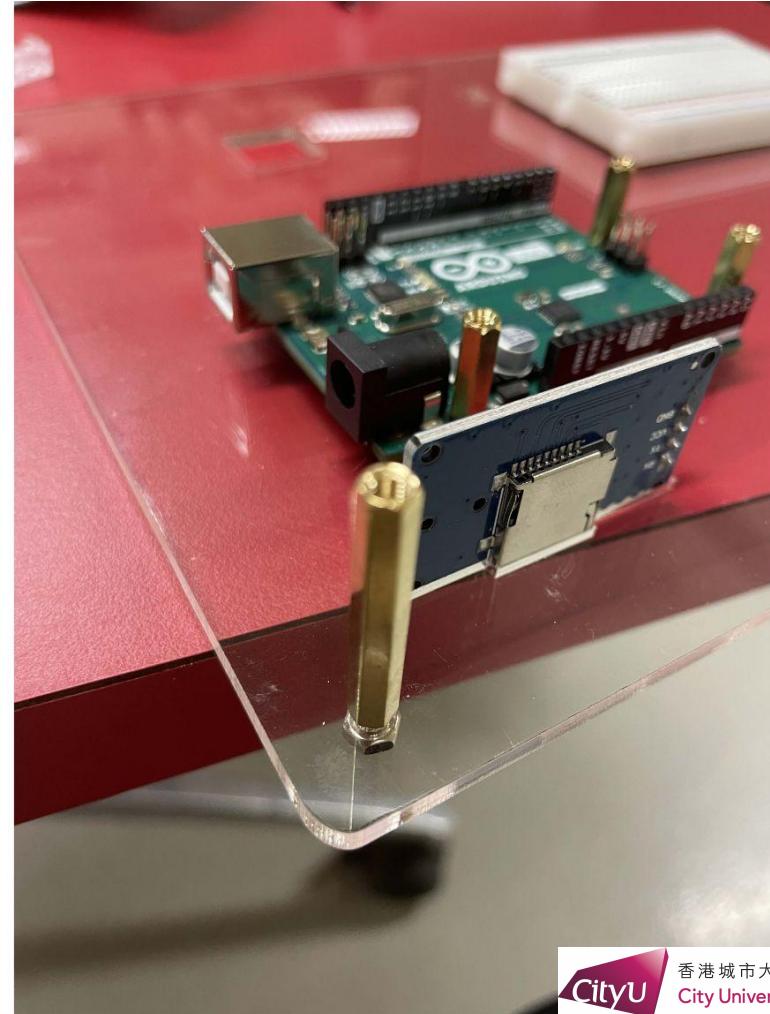
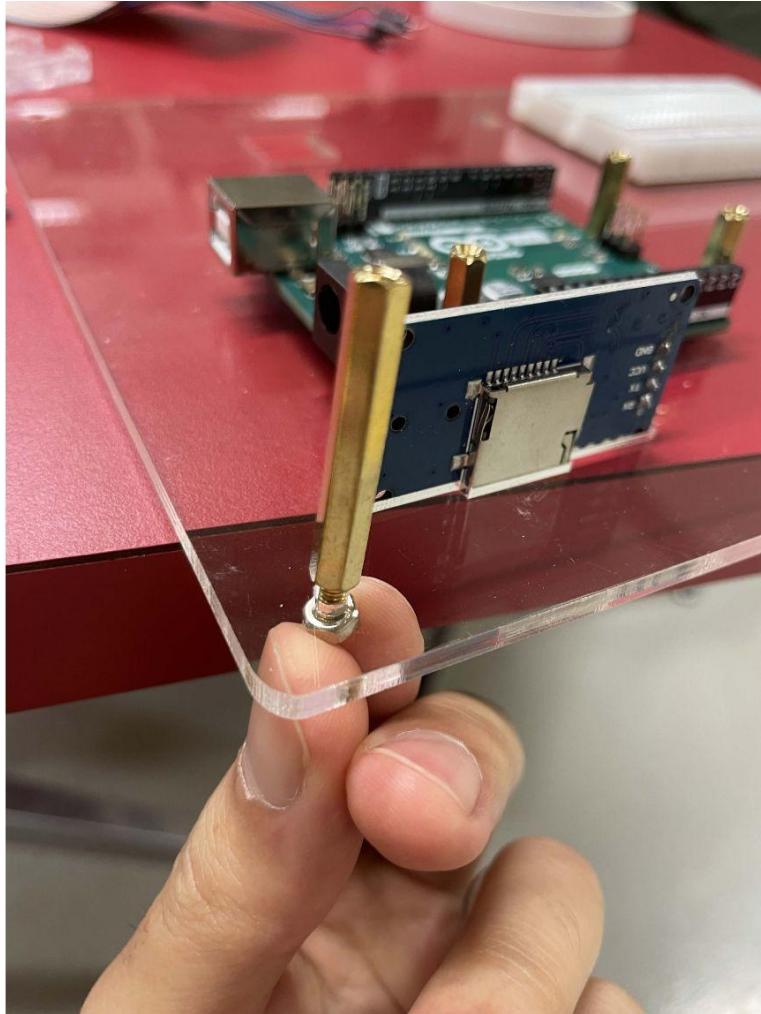
*Serial.println()* 是 Arduino 程式語言的一個函式，用於透過序列通訊介面輸出文字或數字訊息，並在最後自動換行。

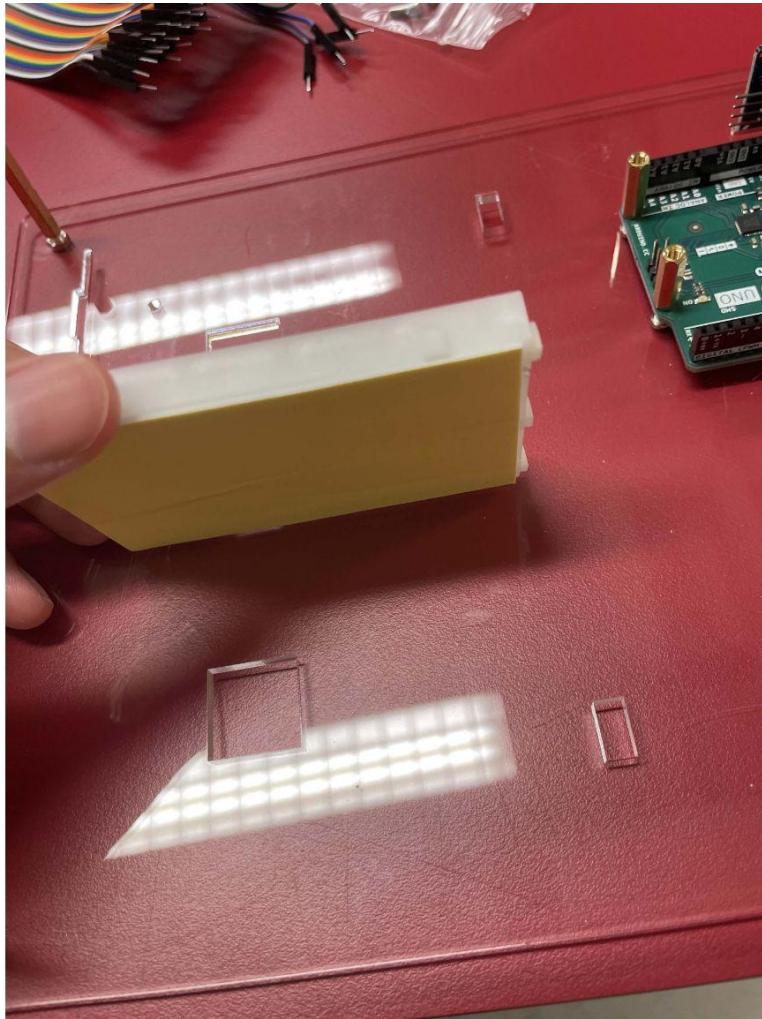
# 硬件組裝

[https://github.com/JC-Project-IDEA/PHASE-2-  
Student-Workshop/blob/main/%E7%B5%84%E8%A3%9D%E6%8C%87%E5%8D%97.pdf](https://github.com/JC-Project-IDEA/PHASE-2-Student-Workshop/blob/main/%E7%B5%84%E8%A3%9D%E6%8C%87%E5%8D%97.pdf)









# 組裝完成效果



## sketch\_apr29c §

```
#include "Compiler_Errors.h"

#include <MPR121.h>
#include <MPR121_Datastream.h>
#include <Wire.h>

#include <SPI.h>
#include <SdFat.h>
#include <FreeStack.h>
#include <SFEMP3Shield.h>

const uint32_t BAUD_RATE = 115200;
const uint8_t MPR121_ADDR = 0x5C;
const uint8_t MPR121_INT = 4;

const bool WAIT_FOR_SERIAL = false;

const bool MPR121_DATASTREAM_ENABLE = false;

uint8_t result;
uint8_t lastPlayed = 0;

SFEMP3Shield MP3player;

const bool REPLAY_MODE = true;

SdFat sd;

void setup() {

}

void loop() {

}
```

# 打開新的 Arduino File (File -> New / Ctrl + N) 將以下的編碼複製到編輯區域

```
#include "SerialMP3Player.h">// 使用MP3版的編碼庫library  
  
#include <CapacitiveSensor.h>// 使用可感應導電墨水的CAP SENSE編碼庫library  
  
#define TX 10 //to MP3 board RX //定義ARDUINO TX到MP3 RX引腳連接  
#define RX 11 //to MP3 board TX //定義ARDUINO RX到MP3 TX引腳連接
```

# 將以下的編碼複製到編輯區域

```
SerialMP3Player mp3(RX, TX); // 定義起動MP3相關的TX, RX  
  
CapacitiveSensor sensor1 = CapacitiveSensor(2, 3);  
CapacitiveSensor sensor2 = CapacitiveSensor(4, 5);  
CapacitiveSensor sensor3 = CapacitiveSensor(6, 7);  
  
//定義CAP SENSE導電感應引腳連接，兩者使用ARDUINO的DIGITAL引腳，並配合電阻達到感應運作  
//前者為SEND PIN，後者為RECIEVE PIN要連接到紙上
```

這段程式碼建立了一個名為 mp3 的 SerialMP3Player 對象，並使用先前定義的 RX 和 TX 引腳。同時，它還創建了 3 個 CapacitiveSensor 對象 (sensor1 到 sensor3)，用於讀取不同引腳上的觸控輸入。每個對象都接收兩個引腳號碼，一個用於發送脈衝，一個用於接收脈衝，以測量導電墨水的觸控。

# 將以下的編碼複製到編輯區域

```
void setup() {  
    Serial.begin(9600);          // 起動serial介面  
    mp3.begin(9600);            // 開始MP3版的連接  
    delay(500);                 // 等待起動  
    mp3.sendCommand(CMD_SEL_DEV, 0, 2); // 選取 sd-card  
    delay(500);                 // 等待起動  
    mp3.setVol(50); // 設定音量  
}
```

這段Arduino **setup()** 函數與先前的版本相似，但音量設置有所不同。

**Serial.begin(9600)** 和 **mp3.begin(9600)** 開始串口和MP3播放器的通訊，傳輸速率為 9600位元/秒。

**delay(500)** 提供操作之間的必要等待時間。**mp3.sendCommand(CMD\_SEL\_DEV, 0, 2)** 選擇SD卡作為存儲設備。

最後，**mp3.setVol(50)** 將音量設定為 50，這是與以前版本的主要區別。

# 將以下的編碼複製到編輯區域

```
void loop() {  
    long measurement1 = sensor1.capacitiveSensor(10); //讀取SENSOR的數值  
    long measurement2 = sensor2.capacitiveSensor(10); //讀取SENSOR的數值  
    long measurement3 = sensor3.capacitiveSensor(10); //讀取SENSOR的數值  
  
    Serial.print(measurement1); //SERIAL PRINT SENSOR的數值以方便MAPPING  
    Serial.print("\t");  
    Serial.print(measurement2);  
    Serial.print("\t");  
    Serial.print(measurement3);  
    Serial.println("\t");  
  
    if (measurement1 >= 60) { //決定觸發起動歌曲的條件 (值)  
        mp3.play(1); //歌曲於SD CARD內的次序  
    }  
    if (measurement2 >= 60) { //決定觸發起動歌曲的條件 (值)  
        mp3.play(2); //歌曲於SD CARD內的次序  
    }  
    if (measurement3 >= 60) { //決定觸發起動歌曲的條件 (值)  
        mp3.play(3); //歌曲於SD CARD內的次序  
    }  
  
    delay(50); //迴圈再執行的中間位  
}
```

```
long measurement1 = sensor1.capacitiveSensor(10); //讀取SENSOR的數值  
long measurement2 = sensor2.capacitiveSensor(10); //讀取SENSOR的數值  
long measurement3 = sensor3.capacitiveSensor(10); //讀取SENSOR的數值  
long measurement4 = sensor4.capacitiveSensor(10); //讀取SENSOR的數值  
long measurement5 = sensor5.capacitiveSensor(10); //讀取SENSOR的數值
```

*loop()* 函數讀取五個觸控感應器的值，並根據每個感應器的讀數決定是否播放特定的歌曲。每個感應器的讀數使用 *capacitiveSensor(10)* 方法獲取，結果儲存在 *measurement1* 到 *measurement5* 變數中。然後，*measurement2* 的值透過串口列印出來供調試。

```
...
if (measurement1 >= 60) { //決定觸發起動歌曲的條件(值)
    mp3.play(4);          //歌曲於SD CARD內的次序
}
...
delay(50);
```

接著，若任一感應器的讀數大於或等於60，將播放對應的歌曲。例如，如果 *measurement1* 大於或等於60，則播放SD卡中的第四首歌曲。同樣，如果 *measurement2* 大於或等於60，則播放第二首歌曲，依此類推。

最後，*delay(50)* 保證在每次迴圈迭代之間有短暫的暫停，以避免過度使用處理器資源。

## sketch\_apr29c §

```
#include "Compiler_Errors.h"

#include <MPR121.h>
#include <MPR121_Datastream.h>
#include <Wire.h>

#include <SPI.h>
#include <SdFat.h>
#include <FreeStack.h>
#include <SFEMP3Shield.h>

const uint32_t BAUD_RATE = 115200;
const uint8_t MPR121_ADDR = 0x5C;
const uint8_t MPR121_INT = 4;

const bool WAIT_FOR_SERIAL = false;

const bool MPR121_DATASTREAM_ENABLE = false;

uint8_t result;
uint8_t lastPlayed = 0;

SFEMP3Shield MP3player;

const bool REPLAY_MODE = true;

SdFat sd;

void setup() {

}

void loop() {

}
```

# 除錯階段

# 有機會遇上的問題

1. 在編碼中遺漏了特定符號 如 : ( ), ,
2. 編碼不在 {} 之內

留意!!! Syntax 句法

- 指一門語言裡支配句子結構, 直到組成句子的規則或過程。

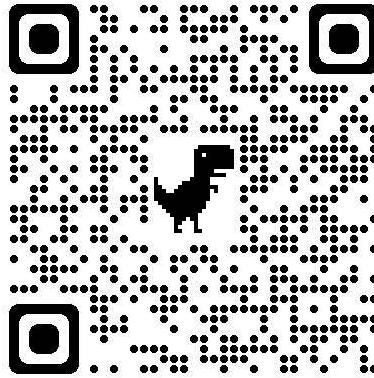
<https://en.wikipedia.org/wiki/Syntax>

如發現編碼排位混亂 可以使用 CTRL + T 自動整理編碼功能



Physics of AUM/OM mantra made  
visible - CYMATICS - Sound of  
Creation

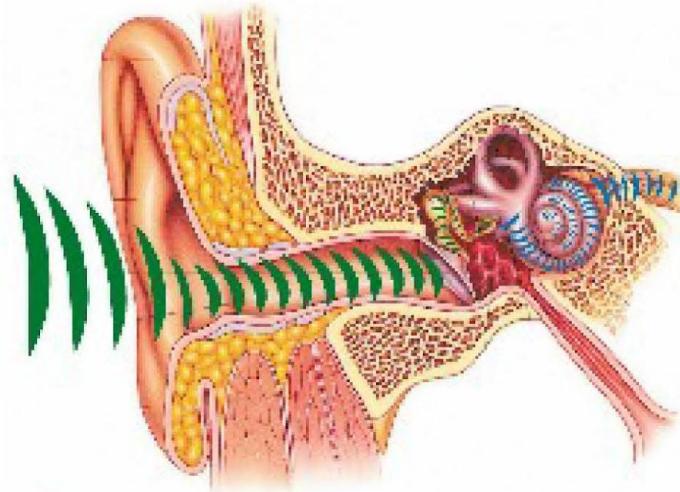
<https://www.youtube.com/watch?v=fozlNrtdzrQ>



# 聲音是？

聲音是振動產生的聲波，通過介質（氣體、固體、液體）傳播並能被人或動物聽覺器官所感知的波動現象。

聲音的頻率一般會以赫茲表示，記為 Hz，指每秒鐘週期性震動的次數。而分貝是用來表示聲音強度的單位，記為 dB。



綠色代表聲波以震動能傳遞到耳膜及聽小骨

藍色代表由內耳轉換為電波能

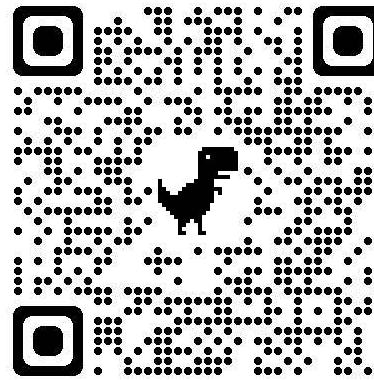
再經由聽神經傳達至大腦聽覺區

## Oscillator 振盪器

Oscillator 是指能夠產生定期波形的電子電路或元件，可用於產生時鐘訊號、調整頻率和相位等。在電子學和通訊系統中，oscillator 常用於產生高頻信號，如無線電訊號，以及在數位電路中作為時序控制器。

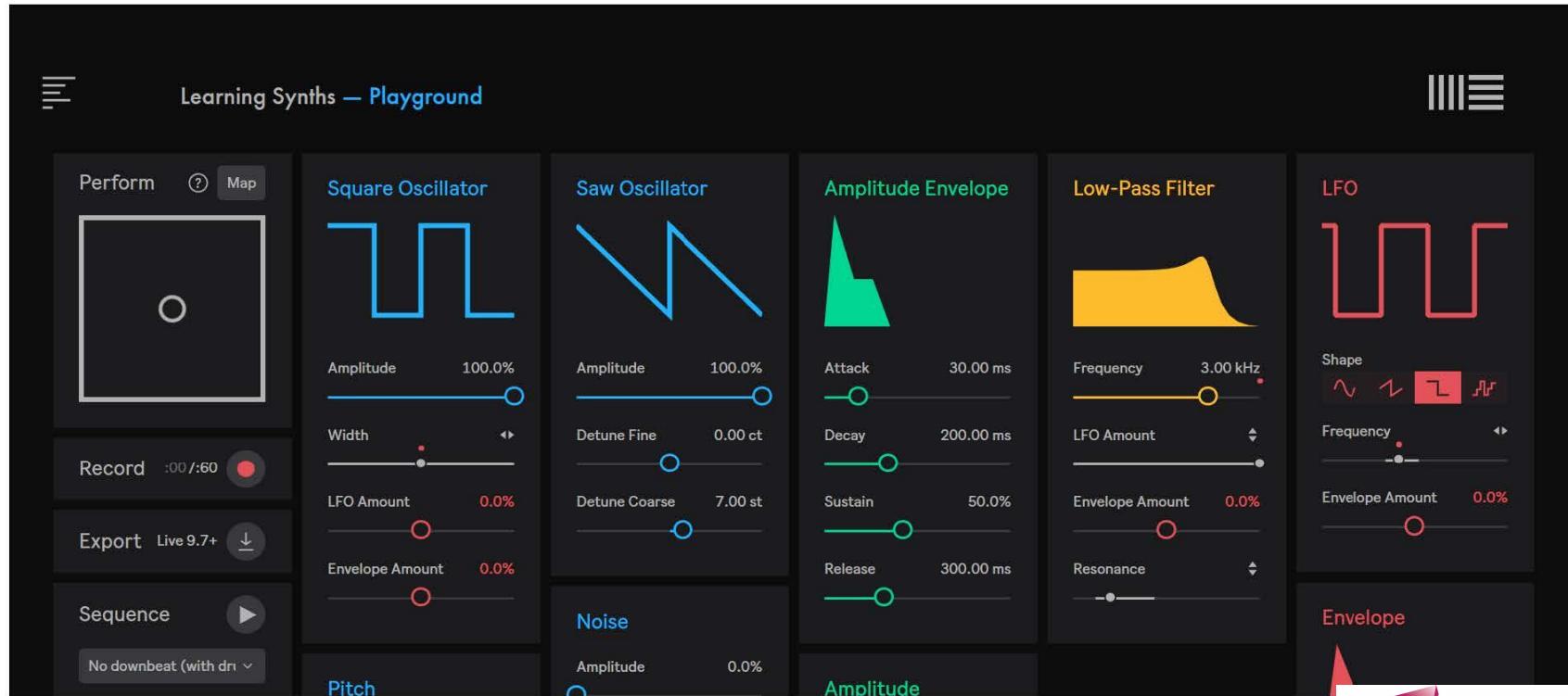
## A Brief History of Synthesizers

[https://www.youtube.com/watch?v=5sjr  
eF6H\\_rY](https://www.youtube.com/watch?v=5sjreF6H_rY)

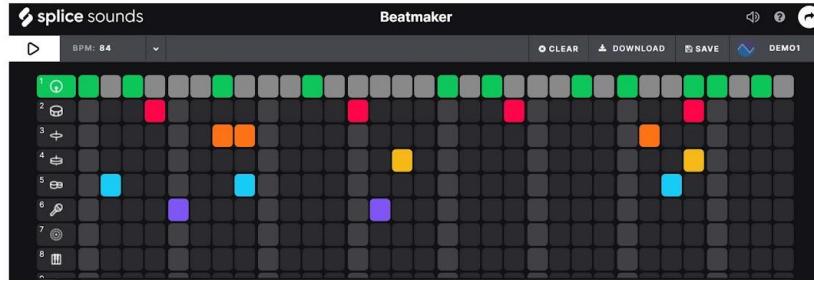


# 學一下 Synthesizer (合成器)

<https://learningsynths.ableton.com/en/playground>



# 有趣的網上音樂創作工具



<https://splice.com/sounds/beatmaker/026f92d3afda>



<https://dotpiano.com/mEvKj97M0O0>

## 除錯時段

作品在硬件上或編碼上有任何問題的話，請告知身邊的工作人員及老師。

# 簡報時間

現在邀請大家分享自己的作品

(\*今まで)  
こうじやなくて



# 賽馬會科藝共融計劃

## Jockey Club Project IDEA

Inclusive Digital and Experimental Art





香港城市大學  
City University of Hong Kong

# 賽馬會科藝共融計劃

## Jockey Club Project IDEA

Inclusive Digital and Experimental Art

專業 創新 胸懷全球  
Professional · Creative  
For The World





香港城市大學  
City University of Hong Kong

# 賽馬會科藝共融計劃

## Jockey Club Project IDEA

Inclusive Digital and Experimental Art

專業 創新 胸懷全球  
Professional · Creative  
For The World