IDE(Integrated Development Environment)

概述

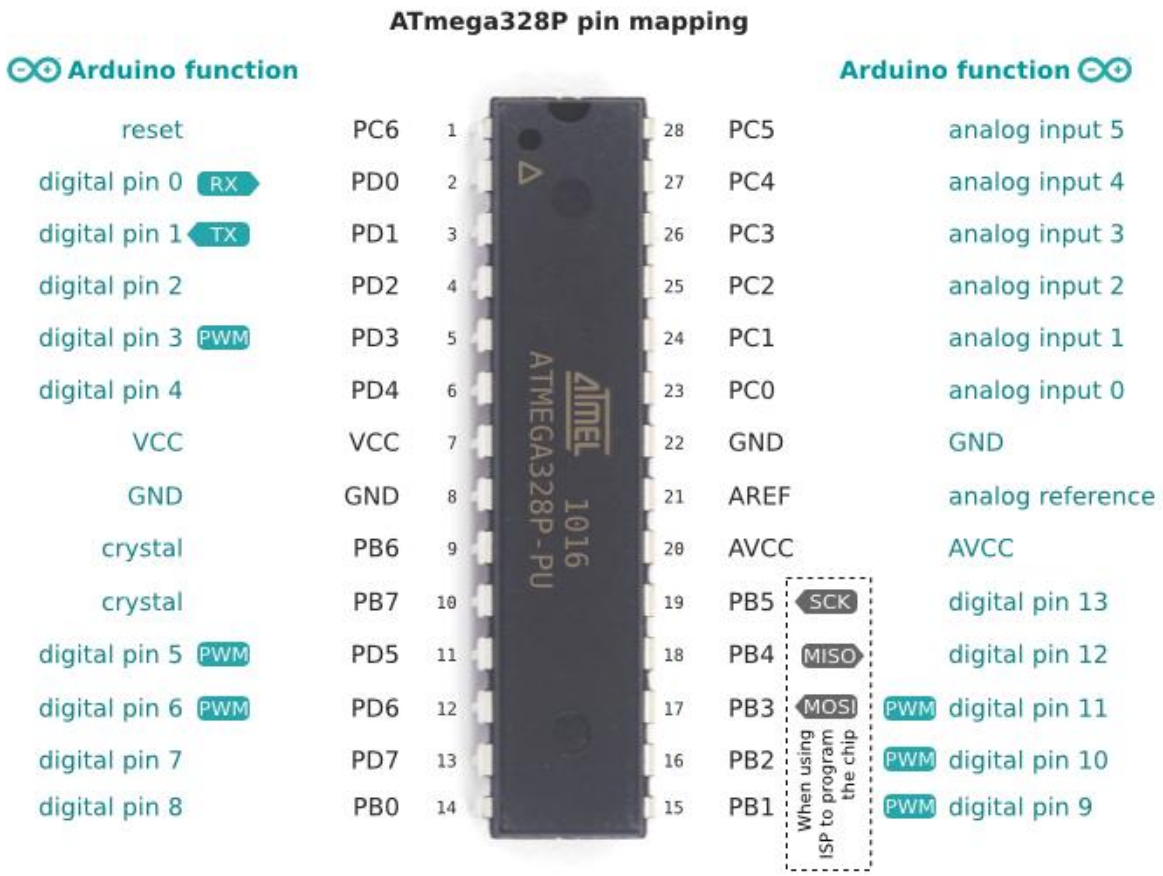
IDE 是一塊基於開放原始代碼的 Simple i/o 平台，並且具有開發語言和開發環境都很簡單、易理解的特點。讓您可以快速使用 IDE 做出有趣的東西。它是一個能夠用來感應和控制現實物理世界的一套工具。它由一個基於單片機並且開放源碼的硬件平台，和一套為 IDE 板編寫程序的開發環境組成。IDE 可以用來開發交互產品，比如它可以讀取大量的開關和傳感器信號，並且可以控制各式各樣的電燈、電機和其他物理設備。IDE 項目可以是單獨的，也可以在運行時和你電腦中運行的程序（例如：Flash，Processing，MaxMSP）進行通訊。

特色描述

開放原始碼的電路圖設計，開發界面免費下載，也可依需求自己修改!!下載程序簡單、方便。可簡單地與傳感器、各式各樣的電子元件連接（如：LED 燈、蜂鳴器、按鍵、光敏電阻等等），做出各種各樣有趣的東西。使用高速的微處理控制器(ATMEGA328)。開發語言和開發環境都非常的簡單、易理解，非常適合初學者學習。

Atmega328 性能描述

Digital I/O 數字輸入/輸出端口 0-13。	Analog I/O 模擬輸入/輸出端口 0-5。
支持 ISP 下載功能。	32 位元的 MCU 平台
輸入電壓：接上 USB 時無須外部供電或外部 5V~9V 直流電壓輸入。	輸出電壓：5V 直流電壓輸出和 3.3V 直流電壓輸出和外部電源輸入。



IDE 語法簡介

結構

void setup()	初始化發量，靴帶程式，調用庫函數等
void loop()	連續執行函數內的語句

常量

HIGH | LOW	表示數字 IO 口的電平，HIGH 表示高電壓（1），LOW 表示低電壓（0）。
INPUT | OUTPUT	表示數字 IO 口的方向，INPUT 表示輸入（高阻態），OUTPUT 表示

功能

串流

Serial.begin(data_rate)	串流數據傳輸設置每秒數據傳輸速率，每秒多少位數(比特率)。為了能與計算機進行通信，可選擇使用以下這些比特率：300，1200，2400，4800，9600，14400，19200，28800，38400，57600 或 115200
Serial.println(serial_data)	串流列印，同 c 語言的 printf，將資料列印到串流中
Serial.available()	檢查串流是否有資料進來
Serial.read(variable)	讀取串流資料，放到變數中供後續的運用

數字 I/O

pinMode(pin, mode)	數字 IO 口輸入輸出模式定義函數，pin 表示為 0～13，mode 表示為 INPUT 或 OUTPUT。
digitalWrite(pin, value)	數字 IO 口輸出電平定義函數，pin 表示為 0～13，value 表示為 HIGH 或 LOW。比如定義 HIGH 可以驅動 LED。
int digitalRead(pin)	數字 IO 口讀輸入電平函數，pin 表示為 0～13，value 表示為 HIGH 或 LOW。比如可以讀數字傳感器。

模擬 I/O

int analogRead(pin)	模擬IO口讀函數，pin表示為0～5（Arduino Diecimila為0～5，Arduino nano為0～7）。比如可以讀模擬傳感器（10 位AD，0～5V 表示為0～1023）。
analogWrite(pin, value)	PWM數字IO口 PWM 輸出函數，Arduino數字IO口標註了PWM的IO口可使用該函數，pin 表示3, 5, 6, 9, 10, 11，value 表示為0～255。比如可用於電機PWM調速或音樂播放。

時間函數

delay(ms)	延時函數（單位 ms）。
delayMicroseconds(us)	延時函數（單位us）。

數學函數

min(x, y)	求最小值
max(x, y)	求最大值
abs(x)	計算絕對值
log(x)	計算對數值，以10為底
ln(x)	計算對數值，以e為底
constrain(x, a, b)	約束函數，下限a，上限b，x 必須在ab 之間才能返回。
map(value, fromLow, fromHigh, toLow, toHigh)	約束函數，value 必須在fromLow 與toLow 之間和fromHigh 與toHigh 之間。
pow(base, exponent)	開方函數，base 的exponent 次方。
sq(x)	平方
sqrt(x)	開根號

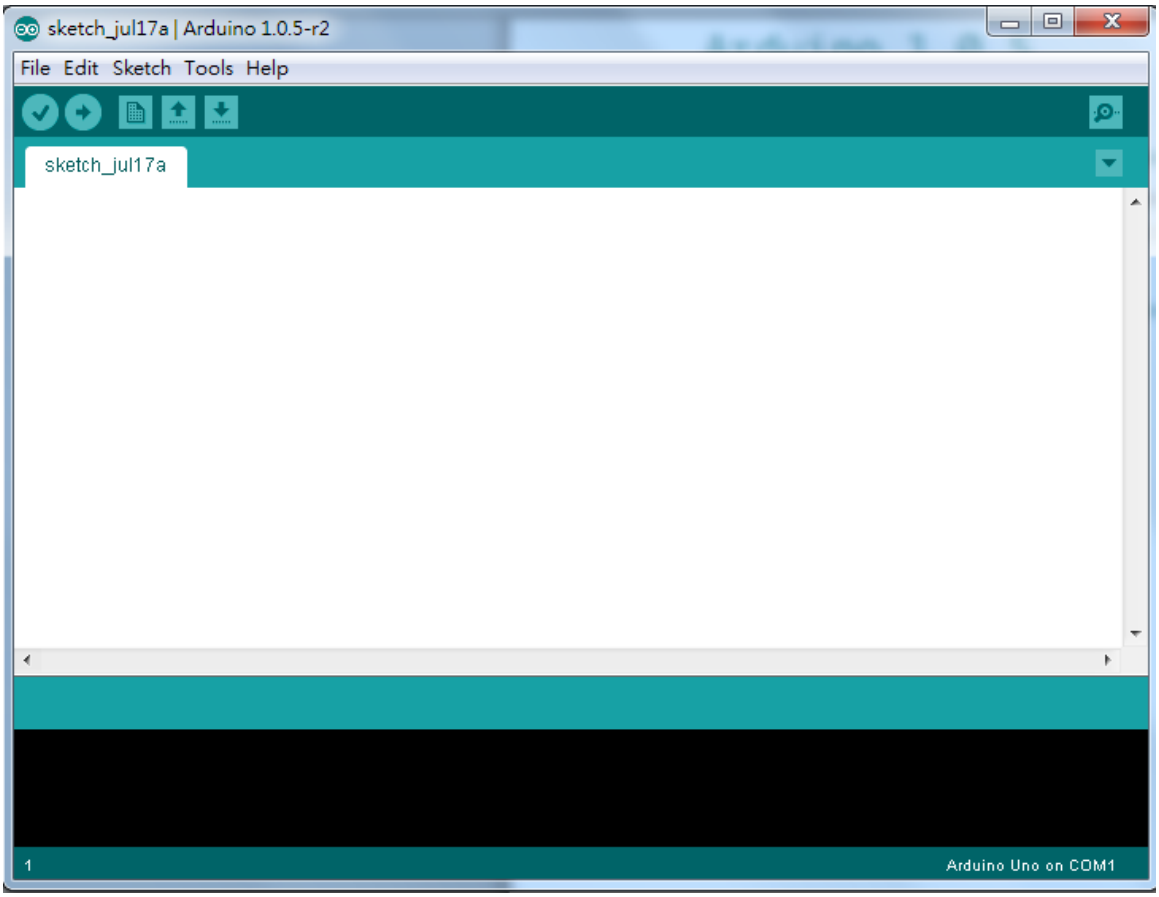
入門

第一步，Arduino 控制板拿到手後，首先需要在電腦上把驅動裝上，這樣才可以進行各種實驗。
第二步，下載 arduino 開發環境，點擊下面鏈接進行下載：

http://arduino.cc/en/main/software#toc1

選擇適合自己的作業系統下載。

第三步，安裝，下載後，NEXT 到底。



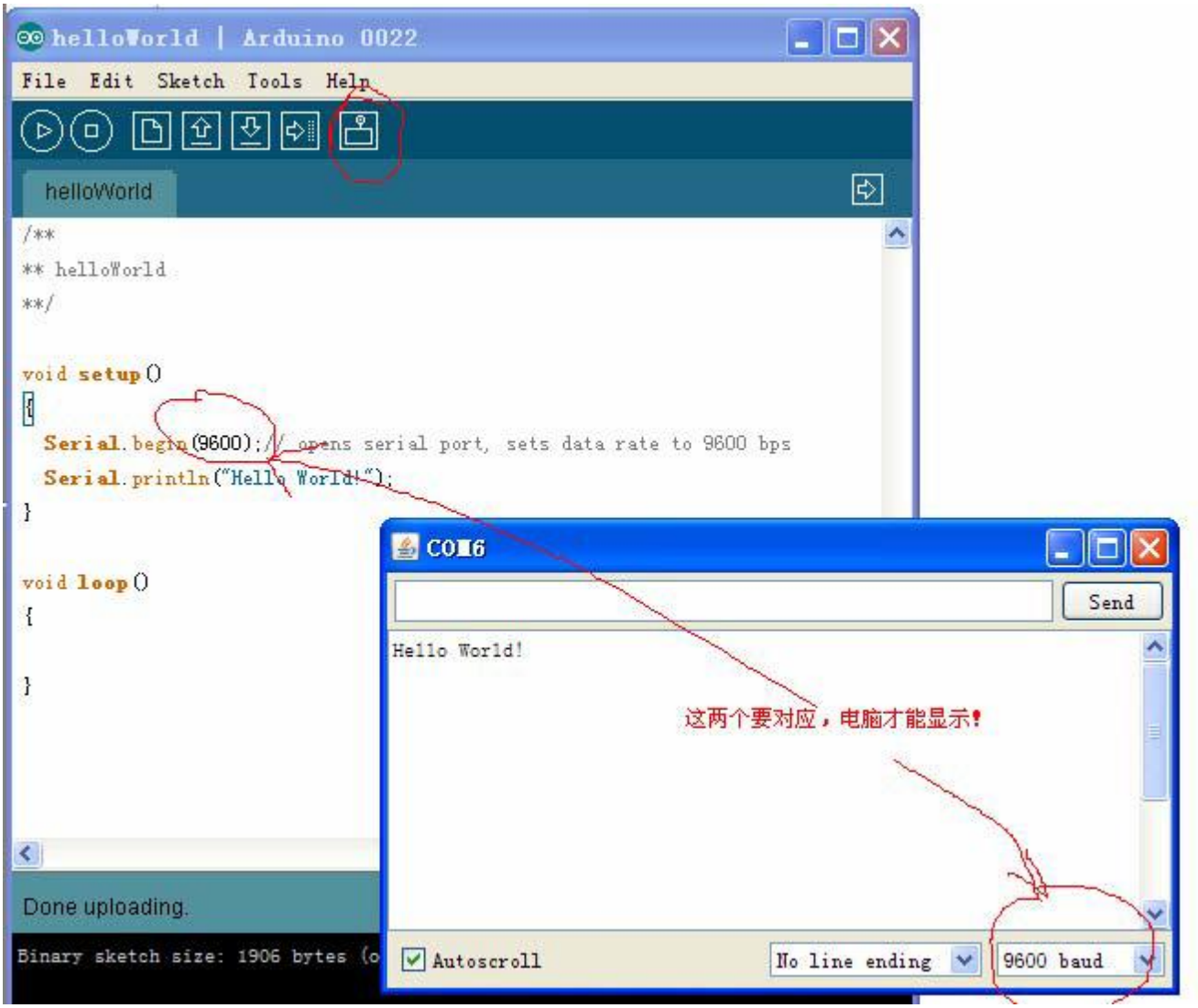
Hello Brazil!

一開始，我們簡單學習一下利用 Arduino IDE 的串流工具，在電腦中顯示我們想要顯示的內容。

CODE

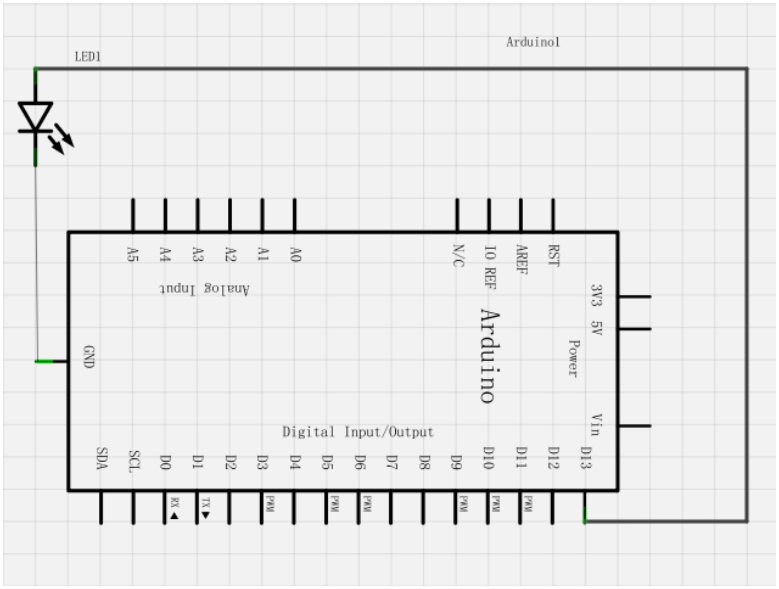
```
void setup()
{
Serial.begin(9600);// opens serial port, sets data rate to 9600 bps
Serial.println("Hello Brazil!");
}
void loop()
{


}
```

1) 把代碼 upload 到 arduino 控制板。
2) 下載成功後，先從選項 tool，選擇相應的 arduino 控制板，和對應的 com。
3) 打開串口工具，在新打開的串口工具窗口的“右下角”選擇相應的比特率。



LED 閃爍

LED 小燈實驗是比較基礎的實驗之一，這次我們利用主板上 13 腳的 LED 燈來完成這個實驗，我們需要的實驗器材除了每個實驗都必須的 Arduino 控制器和 USB 下載線以外，其他的都不用，下一步我們按照下面的小燈實驗原理圖鏈接電路圖，



按照上圖鏈接好電路後，就可以開始編寫程式了，我們讓 LED 小燈閃爍，點亮 1 秒熄滅 1 秒。

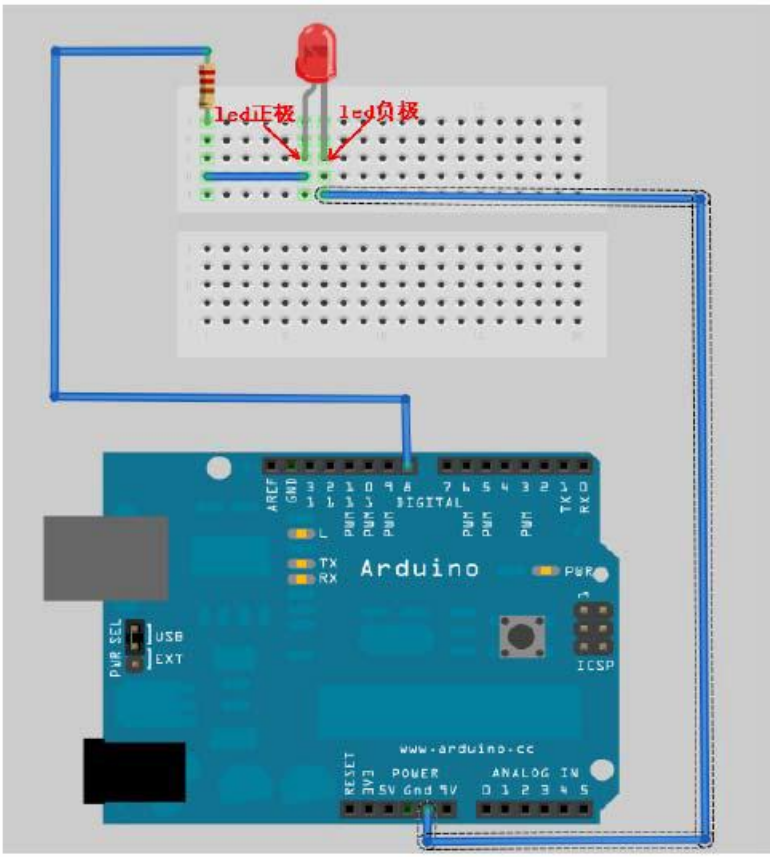
CODE

```
int ledPin = 13;
void setup()
{
pinMode(ledPin, OUTPUT);
}
void loop()
{
digitalWrite(ledPin, HIGH);
delay(1000);
digitalWrite(ledPin, LOW);
delay(1000);
}
```

upload 完程式就可以看到我們的 13 腳 LED 燈在閃爍了，這樣我們的小燈閃爍實驗就完成了。

外接 LED 閃爍實驗

本實驗選擇了接線方法 1 連接發光二極管，將 220Ω電阻的一端插在 Prototype Shield 擴展板上的第 8 個 digital I/O 口，電阻的另一端插在麵包板上，電阻和發光二極管通過導線相連，發光二極管的負端插在麵包板上與 GND 相連。具體連接如圖：



1) 實驗器材

Led 燈：1 個

220Ω 的電阻：1 個

麵包板實驗跳線：若干

2) 實驗連線

按照 Arduino 使用介紹將控制板、板子、麵包板連接好，下載線插好。最後，按照圖將發光二級管連接到數字的第 8 引腳。這樣我們就完成了實驗的連線部分。

3) 實驗原理

先設置數字 8 引腳為高電平點亮 led 燈，然後延時 1s，接著設置數字 8 引腳為低電平熄滅 led 燈，再延時 1s。這樣使 led 燈亮 1s、滅 1s，在規視上就形成閃爍狀態。如果想讓 led 快速閃爍，可以將延時時間設置的小一些，但不能過小，過小的話人眼就識別不出來了，看上去就像 led 燈一直在亮著；如果想讓 led 慢一點閃爍，可以將延時時間設置的大一些，但也不能過大，過大的話就沒有閃爍的效果了。

CODE

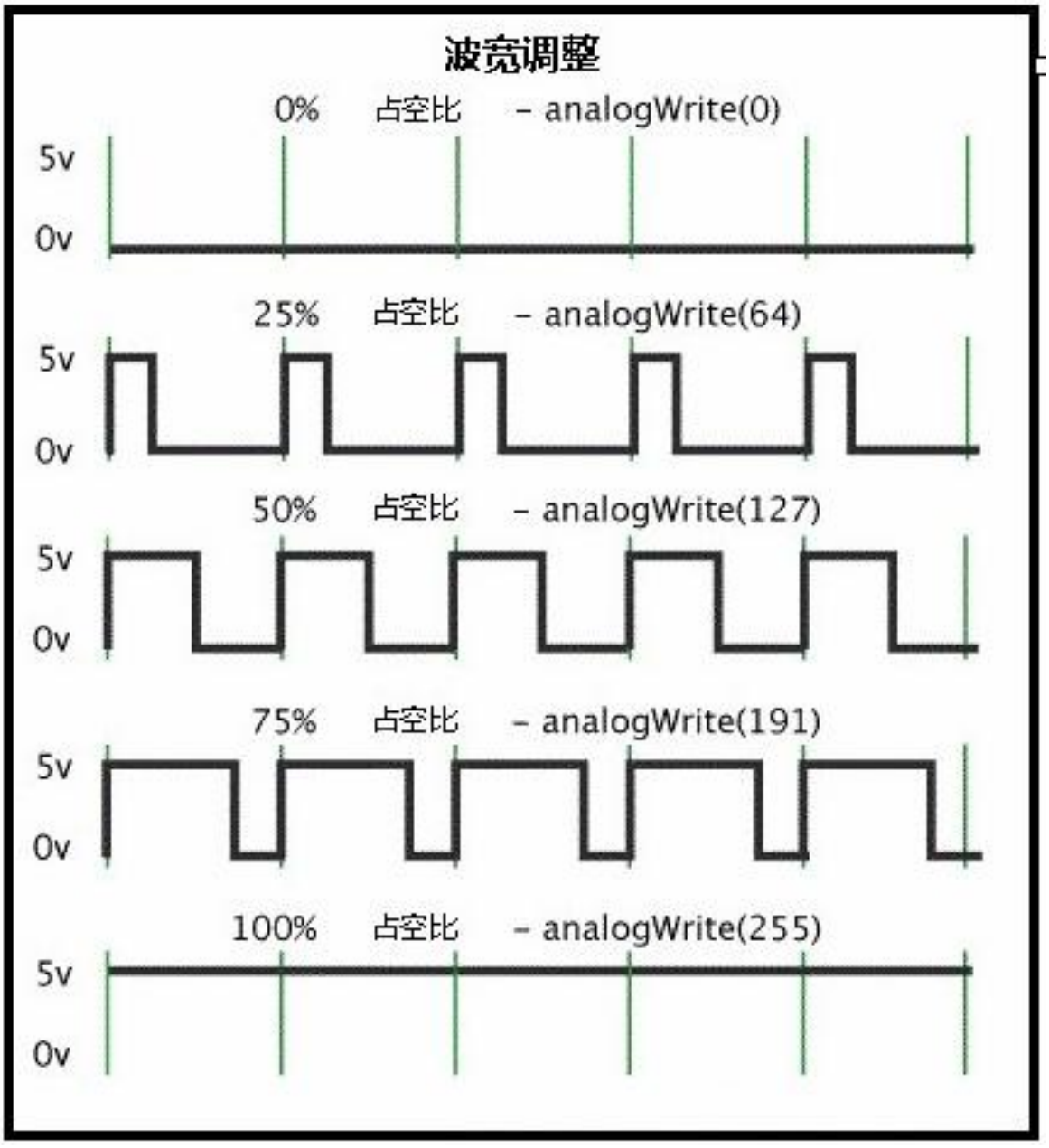
```
int ledPin=8; //設定控制 LED 的數字 IO 腳
void setup()
{
pinMode(ledPin,OUTPUT);//設定數字 IO 口的模式，OUTPUT 為輸出
}
void loop()
{
digitalWrite(ledPin,HIGH); //設定 PIN8 腳為 HIGH = 5V 左右
delay(1000); //設定延時時間，1000 = 1 秒
digitalWrite(ledPin,LOW); //設定 PIN8 腳為 LOW = 0V
delay(1000); //設定延時時間，1000 = 1 秒
}
```

PWM 輸出實驗

原理

Width Modulation 就是通常所說的 PWM，譯為脈衝寬度調製，簡稱脈寬調製。度調製（PWM）是一種對模擬信號電平進行數字編碼的方法，由於計算機不能擬電壓，只能輸出 0 或 5V 的的數字電壓值，我們就通過使用高分辨率計數器方波的佔空比被調製的方法來對一個具體模擬信號的電平進行編碼。PWM 信號數字的，因為在給定的任何時刻，滿幅值的直流供電要麼是 5V(ON)，要麼是 0VOFF)。電壓或電流源是以一種通(ON)或斷(OFF)的重複脈衝序列被加到模擬負載上去的時候即是直流供電被加到負載上的時候，斷的時候即是供電被斷開的時候。只要帶寬足夠，任何模擬值都可以使用 PWM 進行編碼。輸出的電壓值是通過通和斷的行計算的。

輸出電壓=（接通時間/脈衝時間）*最大電壓值



Arduino 端口的輸入電壓只有兩個 0V 與 5V。如我想要 3V 的輸出電壓怎麼辦… ，也許你會說串聯電阻? OK，這個方法是正確的。但是如果我想 1V,3V,3.5V 等等之間來回變動怎麼辦呢？不可能不停地切換電阻吧。這種情況下… ， 就需要使用 PWM 了。他是怎麼控制的呢，對於 arduino 的數字端口電壓輸出只有 LOW 與 HIGH 兩個開關，對應的就是 0V 與 5V 的電壓輸出，我們本把 LOW 定義為 0，HIGH 定義為 1 。一秒內讓 arduino 輸出 500 個 0 或者 1 的信號。如果這 500 個全部為 1，那就是完整的 5V，如果全部為 0，那就是 0V。如果 010101010101 這樣輸出，剛好一半一半，這樣輸出端口實際的輸出電壓是 2.5V。這個類似我們放映電影，我們所看的電影並不是完全連續的，它其實是每秒輸出 25 張圖片，在這種情況下人的肉眼是分辨不出來的，看上去就是連續的了。 PWM 也是同樣的道理，如果想要不同的電壓，就控制 0 與 1 的輸出比例就 ok 。當然… 這和真實的連續輸出還是有差別的，單位時間內輸出的 0,1 信號越多，控制的就越精確。在上圖中，綠線之間代表一個週期，其值也是 PWM 頻率的倒數。換句話說，如果 arduino PWM 的頻率是 500Hz，那麼兩綠線之間的周期就是 2 毫秒。 analogWrite() 命令中可以操控的範圍為 0-255， analogWrite(255)表示 100%佔空比（常開）， analogWrite(127)佔空比大約為 50%（一半的時間）。

在 Arduino 語法中， 我們使用函數： analogWrite()

analogWrite() ：作用是給端口寫入一個模擬值(PWM 波)。可以用來控制 LED 燈的亮度變化，或者以不同的速度驅動馬達。當執行 analogWrite()命令後，端口會輸出一個穩定的佔空比的方波。除非有下一個命令來改變它。 PWM 信號的頻率大約為 490Hz.在使用 ATmega168、 ATmega328 與 UNO 的 arduino 控制板上，其工作在 3,5,6,9,10,11 端口。 Arduino Mega ，2560 控制板，可以工作於 2-13 號端口。在更古老的基於 ATmega8 的 arduino 控制板上，analogWrite()命令只能工作於 9,10,11 號端口。在使用 analogWrite()命令前，可以不使用 pinMode()命令把端口定義為輸出端口，當然如果定義了更好，這樣利於程序語言規範。

語法

analogWrite(pin, value)

參數

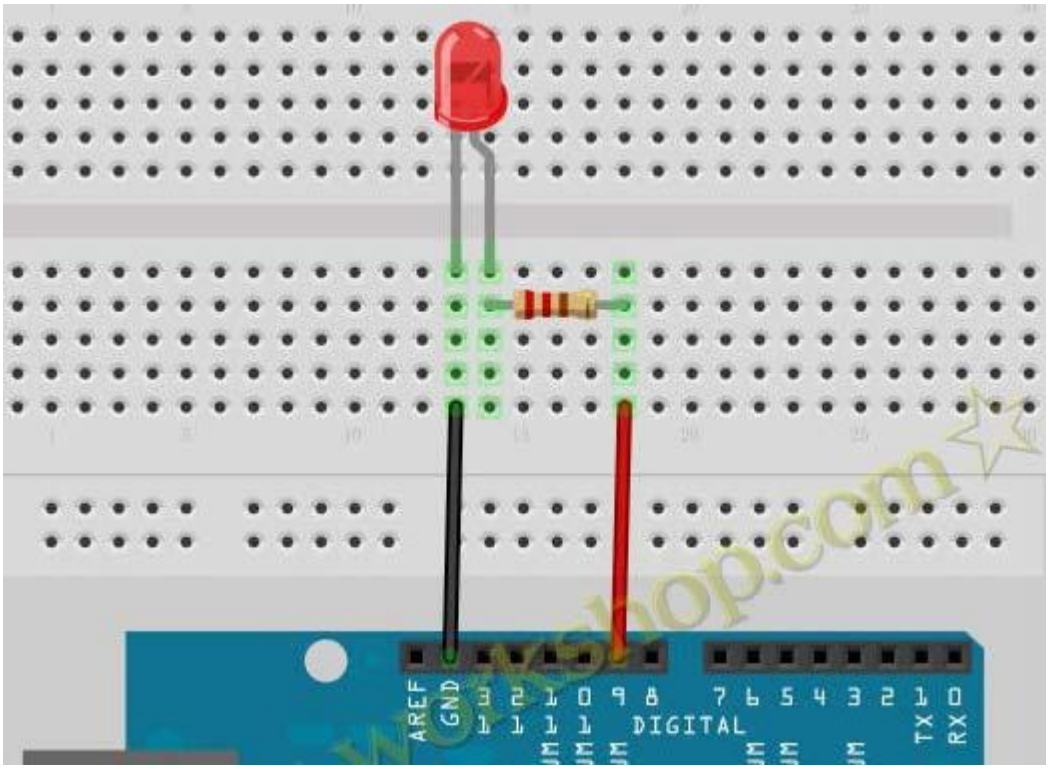
Pin： 寫入的端口
value:佔空比:在 0-255 之間。

註釋與已知問題

當 PWM 輸出與 5,6 號端口的時候，會產生比預期更高的佔空比。原因是 PWM 輸出所使用的內部時鐘，millis()與 delay()兩函數也在使用。所以要注意使用 5,6 號端口時，空佔比要設置的稍微低一些，或者會產生 5,6 號端口無法輸出完全關閉的信號。

實驗

按照 Arduino 使用介紹將控制板、麵包板連接好最後，按照圖將發光二級管通過 220 歐姆電阻連接到數字的第 9 引腳。這樣我們就完成了實驗的連線部分。



CODE

```
int brightness = 0; //定義整數型變量 brightness 與其初始值，此變量用來表示 LED 的亮度。
int fadeAmount = 5; //定義整數型變量 fadeAmount，此變量用來做亮度變化的增減量。
void setup() {
pinMode(9, OUTPUT);// 設置 9 號口為輸出端口:
}
void loop() {
analogWrite(9, brightness);//把 brightness 的值寫入 9 號端口
brightness = brightness + fadeAmount;//改變 brightness 值，使亮度在下一次循環發生改變
if (brightness == 0 || brightness == 255) {
fadeAmount = -fadeAmount ; //在亮度最高與最低時進行翻轉
}
delay(30); //延時 30 毫秒
}
```

將程序下載到實驗板後我們可以觀察到，通過 PWM 來控制一盞 LED 燈，讓它慢慢變亮再慢慢變暗，如此循環。

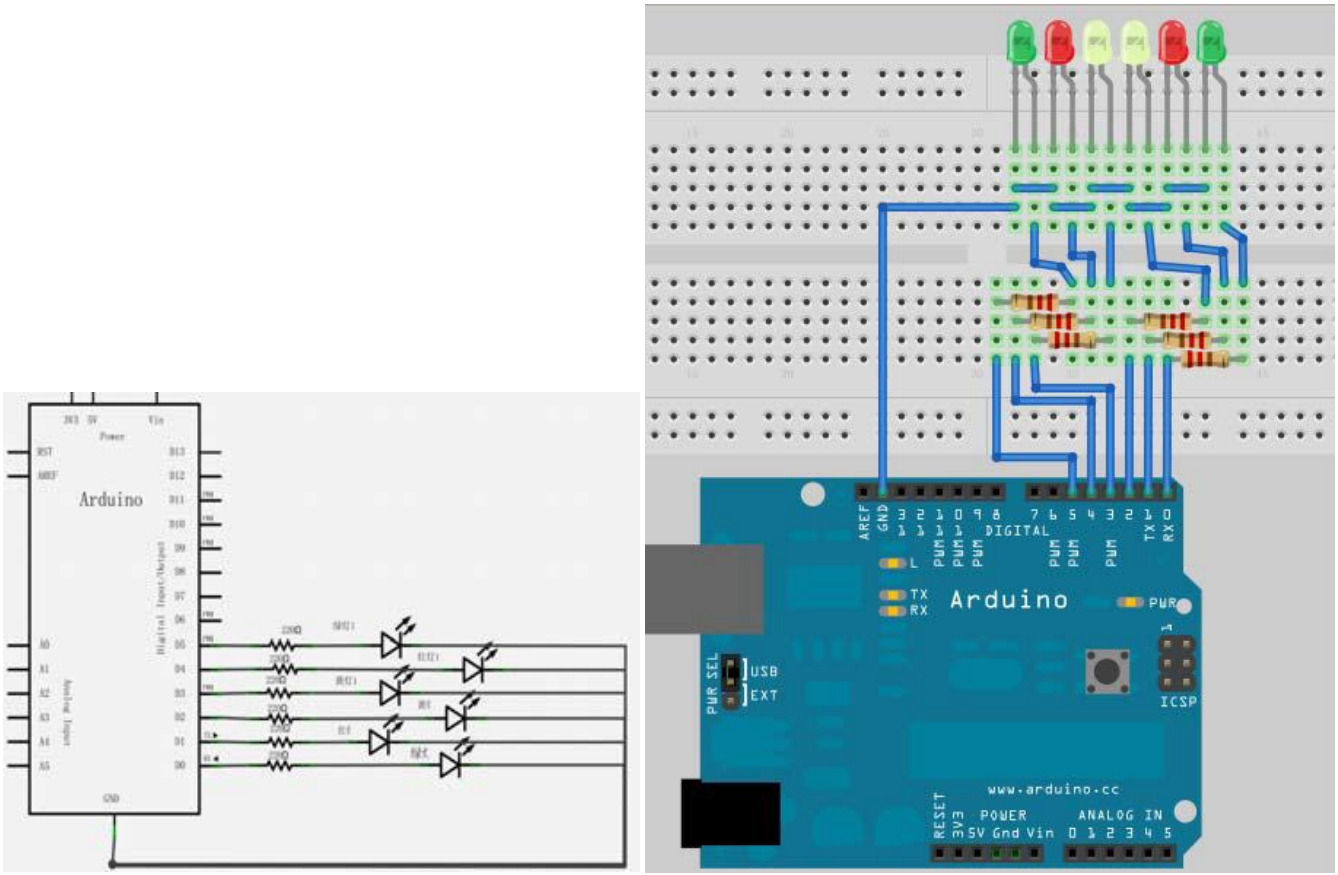
廣告流水燈實驗

1）實驗器件

Led 燈：6 個
220Ω的電阻：6 個
多彩麵包板實驗跳線：若干

2）實驗連線

按照上述方法將板子和數據線連好。然後按照二級管的接線方法，將六個 LED 燈依次接到數字 1~6 引腳上。如圖：



3）實驗原理

在生活中我們經常會看到一些由各種顏色的 led 燈組成的廣告燈，廣告燈上各個位置上的 led 燈不斷的亮滅變化，就形成各種不同的效果。本節實驗就是利用 led 燈編程模擬廣告燈的效果。在程序中我們設置 led 燈亮滅的次序和時間，這樣就可以組成不同的效果。樣式一子程序：led 首先從左邊的綠燈開始間隔 200ms 依次點亮六個 led 燈，如圖 1.6，接著從右邊的綠燈開始間隔 200ms 依次熄滅六個 led 燈。燈閃爍子程序：六個 led 燈首先全部點亮，接著延時 200ms，最後六個 led 燈全部熄滅，這個過程循環兩次就實現了閃爍的效果。樣式二子程序設置 k 和 j 的值讓中間的兩個黃燈亮先亮，接著讓挨著兩個黃燈兩邊的紅燈亮，最後讓兩邊的綠燈亮；執行一遍後改發 k 和 j 的值讓讓兩

邊的綠燈先熄滅，接著兩邊的紅燈熄滅，最後中間的兩個黃燈熄滅。樣式三子程序：設置 k 和 j 的值，讓兩邊的綠燈亮 400ms 後再熄滅，接著讓兩邊的紅燈亮 400ms 後再熄滅，最後讓中間的兩個黃燈亮 400ms 後再熄滅；執行一遍後改發 k 和 j 的值讓兩個紅燈亮 400ms 後熄滅，接著讓兩邊的綠燈亮 400ms 後熄滅。

4）程序代碼

```
//設置控制 Led 的數字 IO 腳
int Led1 = 1;
int Led2 = 2;
int Led3 = 3;
int Led4 = 4;
int Led5 = 5;
int Led6 = 6;
//led 燈花樣顯示樣式 1 子程序
void style_1(void)
{
unsigned char j;
for(j=1;j<=6;j++)//每隔 200ms 依次點亮 1~6 引腳相連的 led 燈
{
digitalWrite(j,HIGH);//點亮 j 引腳相連的 led 燈
delay(200);//延時 200ms
}
for(j=6;j>=1;j--)//每隔 200ms 依次熄滅 6~1 引腳相連的 led 燈
{
digitalWrite(j,LOW);//熄滅 j 引腳相連的 led 燈
delay(200);//延時 200ms
}
}
//燈閃爍子程序
void flash(void)
{
unsigned char j,k;
for(k=0;k<=1;k++)//閃爍兩次
{
for(j=1;j<=6;j++)//點亮 1~6 引腳相連的 led 燈
digitalWrite(j,HIGH);//點亮與 j 引腳相連的 led 燈
delay(200);//延時 200ms
for(j=1;j<=6;j++)//熄滅 1~6 引腳相連的 led 燈
digitalWrite(j,LOW);//熄滅與 j 引腳相連的 led 燈
delay(200);//延時 200ms
}
}
```

```
//led 燈花樣顯示樣式 2 子程序
void style_2(void)
{
unsigned char j,k;
k=1;//設置 k 的初值為 1
for(j=3;j>=1;j--)
{
digitalWrite(j,HIGH);//點亮燈
digitalWrite(j+k,HIGH);//點亮燈
delay(400);//延時 400ms
k +=2;//k 值加 2
}
k=5;//設置 k 值為 5
for(j=1;j<=3;j++)
{
digitalWrite(j,LOW);//熄滅燈
digitalWrite(j+k,LOW);//熄滅燈
delay(400);//延時 400ms
k -=2;//k 值減 2
}
}
//led 燈花樣顯示樣式 3 子程序
void style_3(void)
{
unsigned char j,k;//led 燈花樣顯示樣式 3 子程序
k=5;//設置 k 值為 5
for(j=1;j<=3;j++)
{
digitalWrite(j,HIGH);//點亮燈
digitalWrite(j+k,HIGH);//點亮燈
delay(400);//延時 400ms
digitalWrite(j,LOW);//熄滅燈
digitalWrite(j+k,LOW);//熄滅燈
k -=2;//k 值減 2
}
k=3;//設置 k 值為 3
for(j=2;j>=1;j--)
{
digitalWrite(j,HIGH);//點亮燈
digitalWrite(j+k,HIGH);//點亮燈
delay(400);//延時 400ms
digitalWrite(j,LOW);//熄滅燈
```

```
digitalWrite(j+k,LOW);//熄滅燈
k +=2;//k 值加 2
}
}
void setup()
{
unsigned char i;
for(i=1;i<=6;i++)//依次設置 1~6 個數字引腳為輸出模式
pinMode(i,OUTPUT);//設置第 i 個引腳為輸出模式
}
void loop()
{
style_1();//樣式 1
flash();//閃爍
style_2();//樣式 2
flash();//閃爍
style_3();//樣式 3
flash();//閃爍
}
```

程序代碼中用到的：

for(i=1;i<=6;i++)//依次設置 1~6 個數字引腳為輸出模式 pinMode(i,OUTPUT);//設置第 i 個引腳為輸出模式這是一個 for 循環。它的一般形式為: for(<初始化>; <條件表達式>; <增量>) 語句; 初始化總是一個賦值語句, 它用來給循環控制發量賦初值; 條件表達式是一個關係表達式, 它決定什麼時候退出循環; 增量定義循環控制發量每循環一次後按什麼方式變化。這三個部分之間用";"分開。例如: for(i=1; i<=10; i++) 語句; 上例中先給" i " 賦初值 1, 判斷" i " 是否小於等於 10, 若是則執行語句, 之後值增加 1。再重新判斷, 直到條件為假, 即 i>10 時, 結束循環。

5）上傳程序

按照 arduino 教程中的程序上傳方法將本程序上傳到實驗板中。

6）程序功能

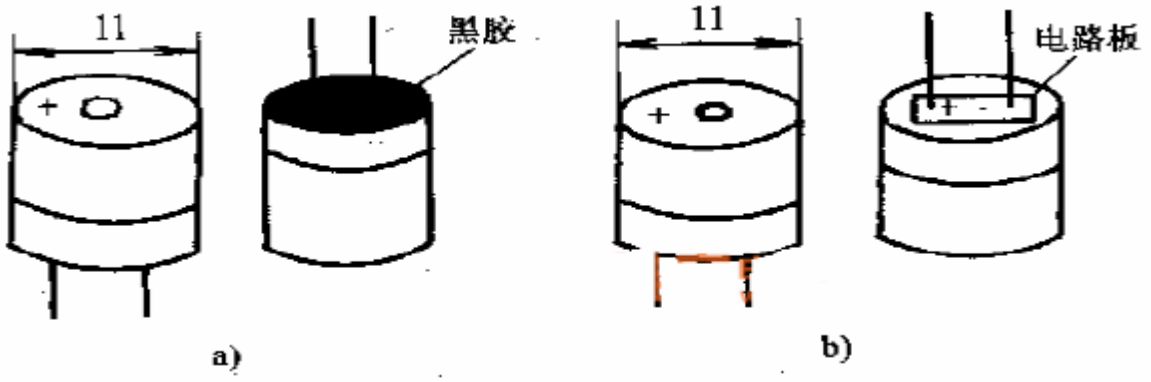
將程序下載到實驗板後我們可以觀察到，六個 led 不斷的循環執行樣式一子程序—>閃爍子程序—>樣式二子程序—>閃爍子程序—>樣式三子程序—>閃爍子程序。在掌握了以上兩個程序後，大家可以充分發揮自己的想像，編寫出自己想要的 led 燈效果，玩轉多彩 led 燈。

蜂鳴器

認識蜂鳴器

蜂鳴器是一種一體化結構的電子訊響器，採用直流電壓供電，廣泛應用於計算機、打印機、複印機、報警器、電子玩具、汽車電子設備、電話機、定時器等電子產品中作發聲器件。

按其驅動方式的不同，可分為：有源蜂鳴器（內含驅動線路）和無源蜂鳴器（外部驅動）教你區分有源蜂鳴器和無源蜂鳴器,現在市場上出售的一種小型蜂鳴器因其體積小(直徑只有 llmm)、重量輕、價格低、結構牢靠，而廣泛地應用在各種需要發聲的電器設備、電子製作和單片機等電路中。有源蜂鳴器和無源蜂鳴器的外觀如圖 a、b 所示。 a)有源 b)無源。



從圖 a、b 外觀上看，兩種蜂鳴器好像一樣，但仔細看，兩者的高度略有區別，有源蜂鳴器 a，高度為 9mm，而無源蜂鳴器 b 的高度為 8mm。如將兩種蜂鳴器的引腳郡朝上放置時，可以看出有綠色電路板的一種是無源蜂鳴器，沒有電路板而用黑膠封閉的一種是有源蜂鳴器。進一步判斷有源蜂鳴器和無源蜂鳴器，還可以用萬用表電阻檔 Rxl 檔測試:用黑表筆接蜂鳴器"+"引腳，紅表筆在另一引腳上來回碰覺，如果覺發出咔、咔聲的且電阻只有 8Ω(或 16Ω)的是無源蜂鳴器;如果能發出持續聲音的，且電阻在幾百歐以上的，是有源蜂鳴器。

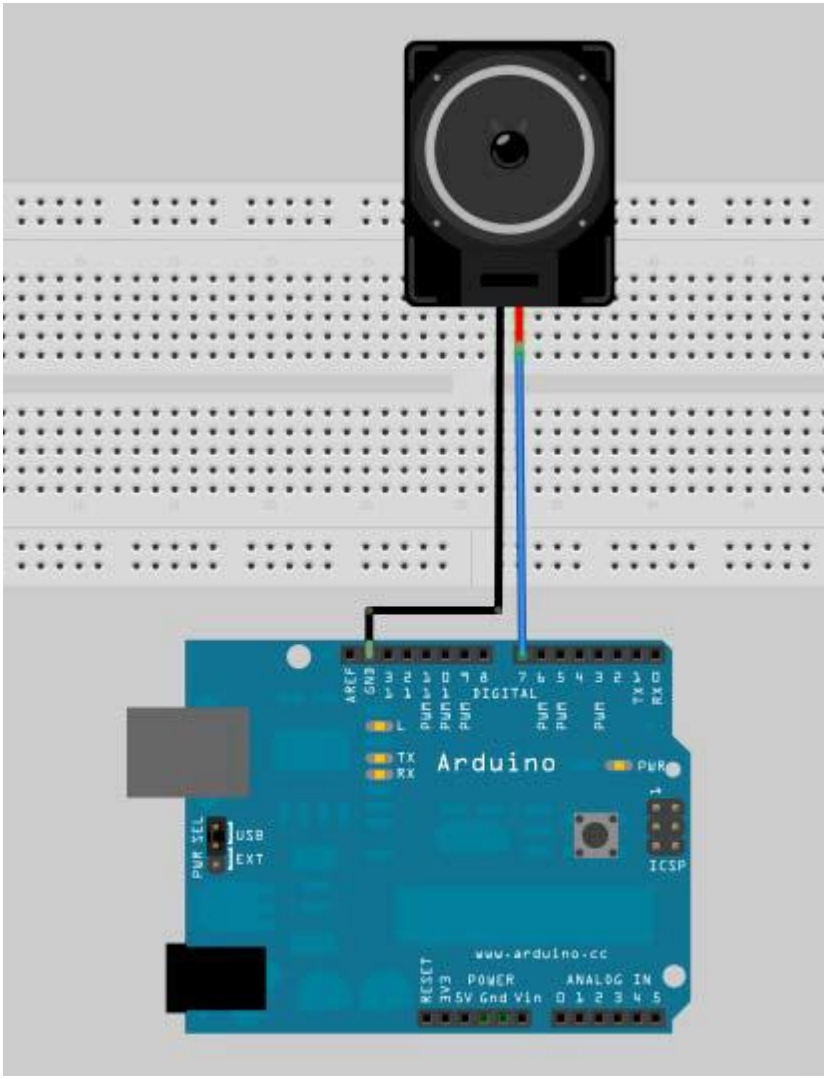
有源蜂鳴器直接接上額定電源(新的蜂鳴器在標籤上都有註明)就可連續發聲;而無源蜂鳴器則和電磁揚聲器一樣，需要接在音頻輸出電路中才能發聲。按構造方式的不同，可分為：電磁式蜂鳴器和壓電式蜂鳴器；壓電式蜂鳴器壓電式蜂鳴器主要由多諧振盪器、壓電蜂鳴片、阻抗匹配器及共鳴箱、外殼等組成。有的壓電式蜂鳴器外殼上還裝有發光二極管。多諧振盪器由晶體管或集成電路構成。當接通電源後（1.5~15V 直流工作電壓）,多諧振盪器起振,輸出 1.5~2.5kHZ 的音頻信號，阻抗匹配器推動壓電蜂鳴片發聲壓電蜂鳴片由鋯鈦酸鉛或鈮鎂酸鉛壓電陶瓷材料製成。在陶瓷片的兩面鍍上銀電極，經極化和老化處理後，再不黃銅片或不銹鋼片粘在一起。電磁式蜂鳴器由振盪器、電磁線圈、磁鐵、振動膜片及外殼等組成。接通電源後，振盪器產生的音頻信號電流通過電磁線圈，使電磁線圈產生磁場。振動膜片在電磁線圈和磁鐵的相互作用下，週期性地振動發聲。

工作原理

蜂鳴器發聲原理是電流通過電磁線圈，使電磁線圈產生磁場來驅動振動膜發聲的，因此需要一定的電流才能驅動它，本實驗用的蜂鳴器內部帶有驅動電路，所以可以直接使用。當不蜂鳴器連接的引腳為高電平時，內部驅動電路導通，蜂鳴器發出聲音；當不蜂鳴器連接的引腳為低電平，內部驅動電路截止，蜂鳴器不發出聲音。

蜂鳴器的連線

本實驗用的蜂鳴器內部帶有驅動電路，所以可以直接將蜂鳴器的正極連接到數字口，蜂鳴器的負極連接到 GND 插口中。如下圖：



1、 實驗器件

蜂鳴器：1 個
多彩麵包板實驗跳線：若干

2、 實驗連線

按照 Arduino 教程將控制板、麵包板連接好，下載線插好。然後按照蜂鳴器的接法將蜂鳴器連接到數字 7 口上，連線完畢。

3、 實驗原理

蜂鳴器發出聲音的時間間隔不同，頻率就不同，所以發出的聲音就不同。根據返一原理我們通過改發蜂鳴器發出聲音的時間間隔，來發出不同種聲音，來模擬各種聲音。本程序首先讓蜂鳴器間隔 1ms 發出一種頻率的聲音，循環 80 次；接著讓蜂鳴器間隔 2ms 發出另一種頻率的聲音，循環 100 次。

4、 程序代碼

```
int buzzer=7;//設置控制蜂鳴器的數字IO 腳
void setup()
{
pinMode(buzzer,OUTPUT);//設置數字IO 腳模式，OUTPUT 為輸出
}
void loop()
{
unsigned char i,j;//定義變量
while(1)
{
for(i=0;i<80;i++)//輸出一個頻率的聲音
{
digitalWrite(buzzer,HIGH);//發聲音
delay(1);//延時1ms
digitalWrite(buzzer,LOW);//不發聲音
delay(1);//延時ms
}
for(i=0;i<100;i++)//輸出另一個頻率的聲音
{
digitalWrite(buzzer,HIGH);//發聲音
delay(2);//延時2ms
digitalWrite(buzzer,LOW);//不發聲音
delay(2);//延時2ms
}
}
}
```

傾斜儀實驗

傾斜開關

本節實驗所使用的傾斜開關是內部帶有一個金屬滾珠的滾珠傾斜開關，如圖所示：

滾珠開關：也叫碰珠開關、搖珠開關、鋼珠開關、傾斜開關，倒順開關、角度傳感器。它主要是利用滾珠在開關內隨不同傾斜角度的變化，達到觸發電路的目的。目前滾珠開關在市場上使用的常用型號有 SW-200D、SW-460、SW-300DA 等，本節使用的是 SW-200D 型號的。這類開關不像傳統的水銀開關，它功效同水銀開關，但沒有水銀開關的環保及安全等問題。

工作原理

觀察傾斜開關我們可以發現，傾斜開關的一端為金色尋針，另一端為銀色尋針。金色一端為<ON>導通觸發端銀色一端為<OFF>開路端當受到外力搖晃而達到適當晃動力時或金色一端設置角度低於水平適當角度時導電接腳電氣特性會產生短時間導通或持續導通<ON>狀態 。而當電氣特性要恢復開路狀態<OFF>時開關設置環境必須為靜止，且銀色一端設置角度需低於水平 10 度。

傾斜開關連線

將傾斜開關銀色的一端連接到 5V 插口，金色一端連接到模擬口。

1、實驗器件

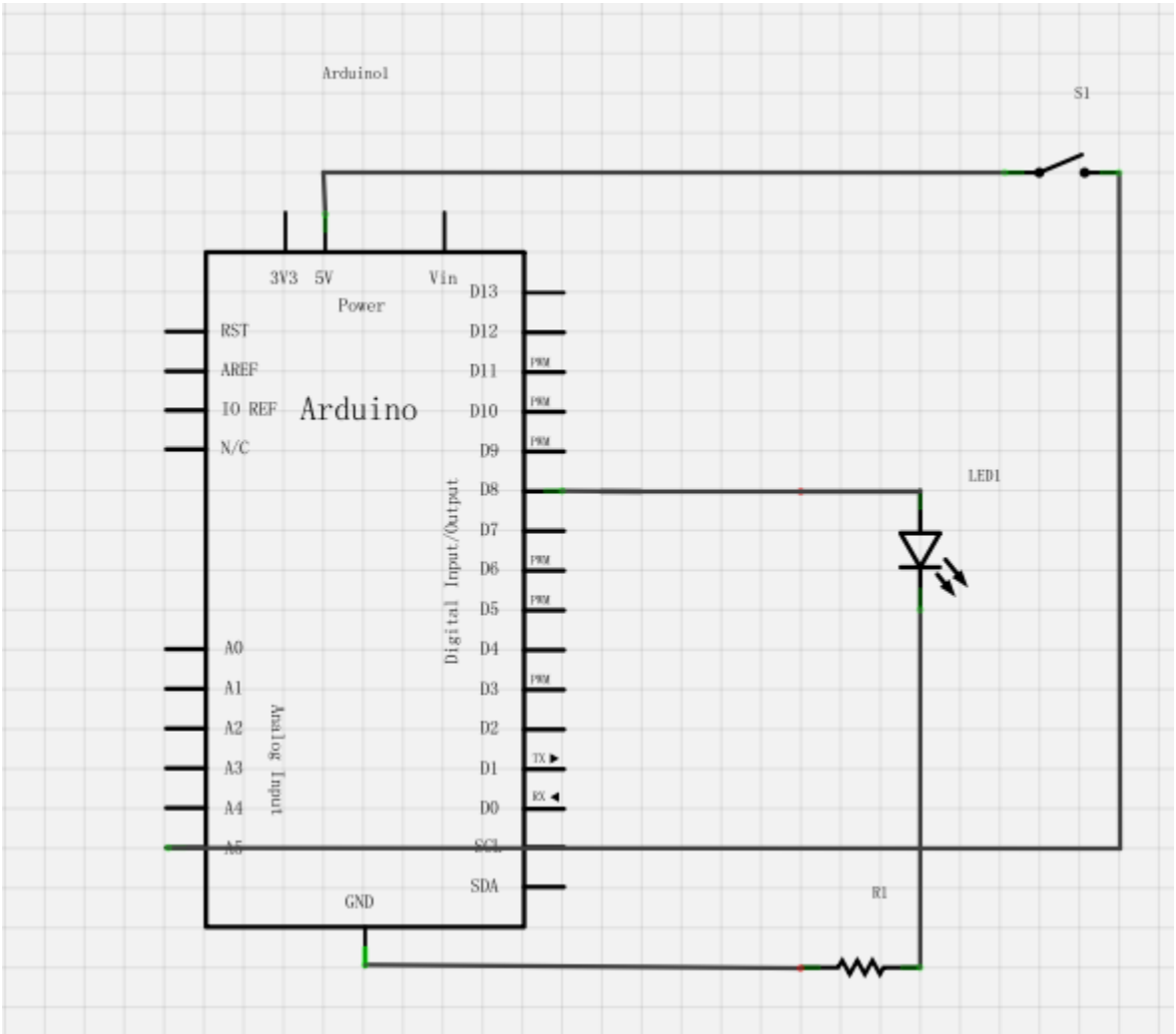
傾斜開關模塊：1 個
多彩麵包板實驗跳線：若干

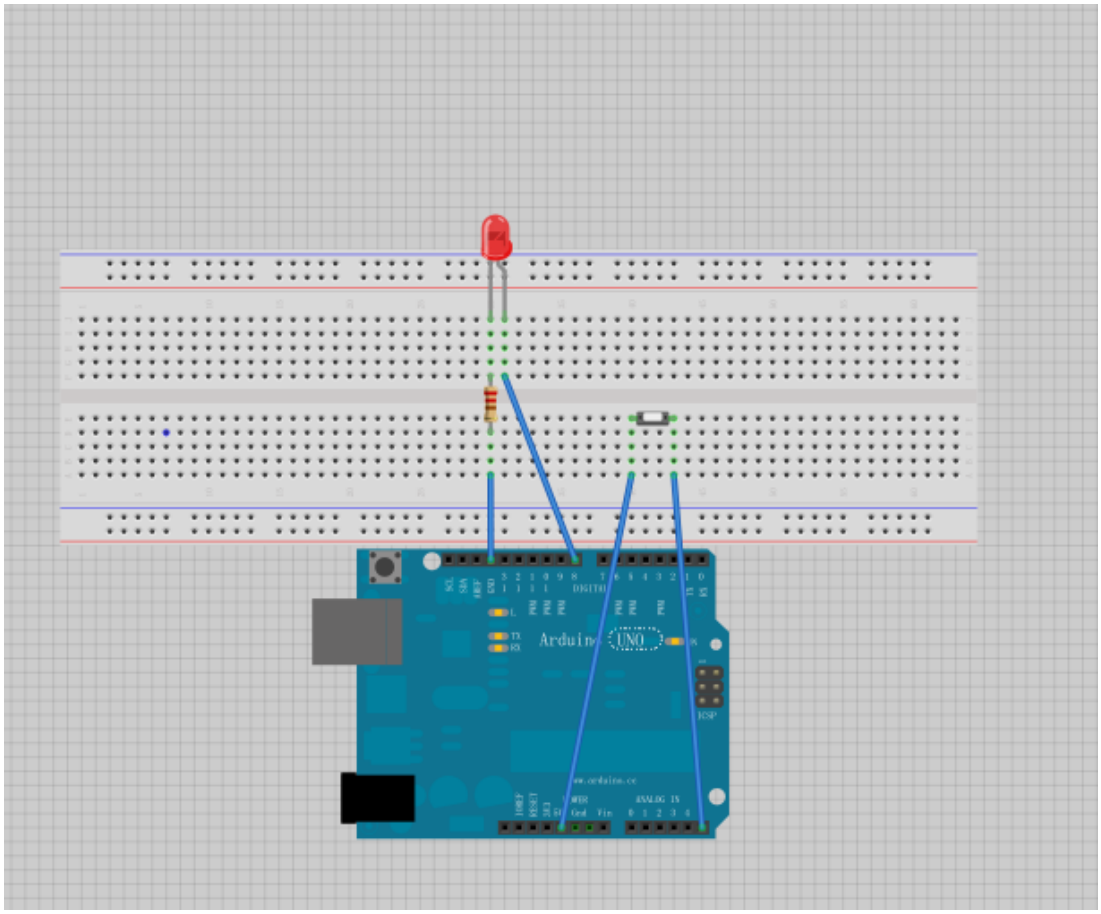
2、實驗連線

按照 Arduino 教程將控制板、麵包板連接好，下載線插好。然後將傾斜開關連接到模擬 5 引腳。

3、實驗原理

當金色一端低於水平位置傾斜，開關尋通，模擬口電壓值為 5V 左右（數字二進製表示為 1023），點亮 led 燈。當銀色一端低於水平位置傾斜，開關截止，模擬口電壓值為 0V 左右（數字二進製表示為 0），熄滅 led 燈。在程序中模擬口電壓值是否大於 2.5V 左右（數字二進製表示為 512），即可知道是否傾斜開關尋通了。





4、程序代碼

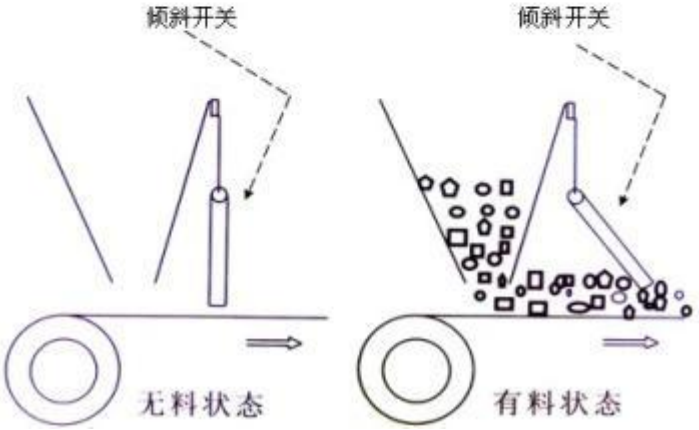
```
void setup()
{
pinMode(8,OUTPUT);//設置數字 8 引腳為輸出模式
}
void loop()
{
int i;//定義變量 i
while(1)
{
i=analogRead(5);//讀取模擬 5 口電壓值
if(i>200)//如果大於 512（2.5V）
{
digitalWrite(8,HIGH);//點亮 led 燈
}
else//否則
{
digitalWrite(8,LOW);//熄滅 led 燈
}
}
}
```

將程序下載到實驗板後大家可以將板子傾斜觀察 led 燈的狀態。當金色一端低於水平位置傾斜，開關尋通，點亮 led 燈；當銀色一端低於水平位置傾斜，開關截止，模擬口電壓值為 0V 左右（數字二進製表示為 0），熄滅 led 燈。掌握本程序後，大家可以按照自己的想法實驗，還可以控制其他器件例如蜂鳴器等。

技术参数:

MT-S 标准型	MT-P 轻便型
1.接点容量：220VAC/2A	1.接点容量：220VAC/2A
2.工作温度：-20~100℃（最高可做到 200℃）	2.工作温度：-20~100℃（最高可做到 200℃）
3.外壳材质：SUS 不锈钢	3.外壳材质：PP; PVC; PE
4.电缆长度：标准 6 米（可定制 1~20 米）	4.电缆长度：标准 6 米（可定制 1~20 米）
5.探头长度：180mm	5.探头长度：195mm
6.探头总长：210mm	6.探头总长：230mm
7.下端螺纹：M20X1.5X50	7.下端螺纹：M20X1.5X50

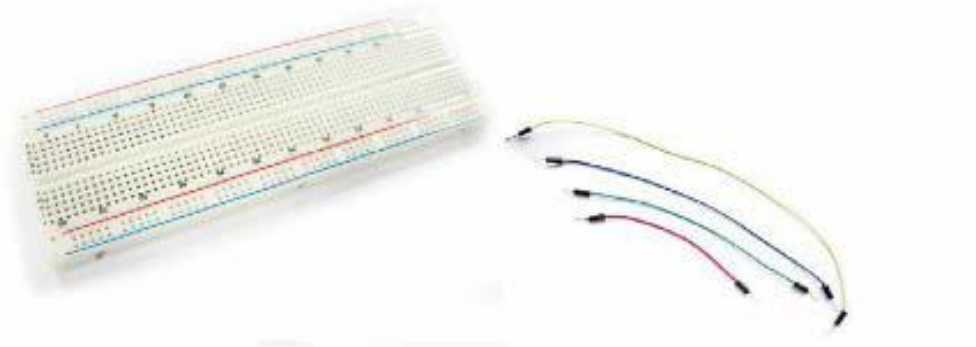
应用案例:

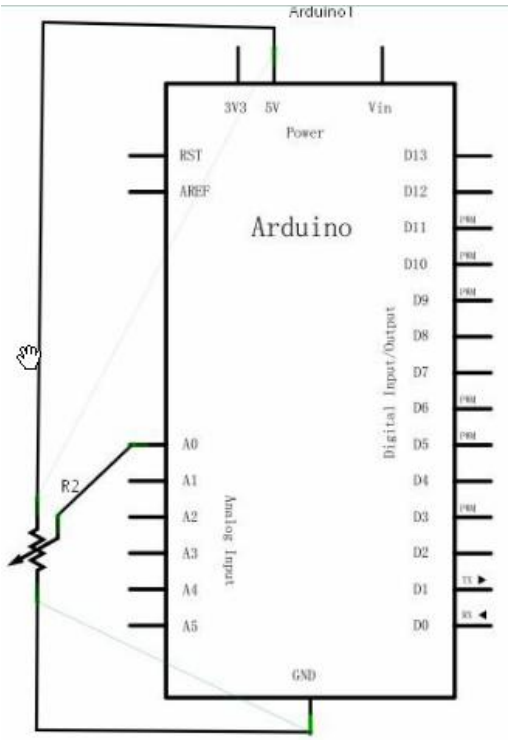


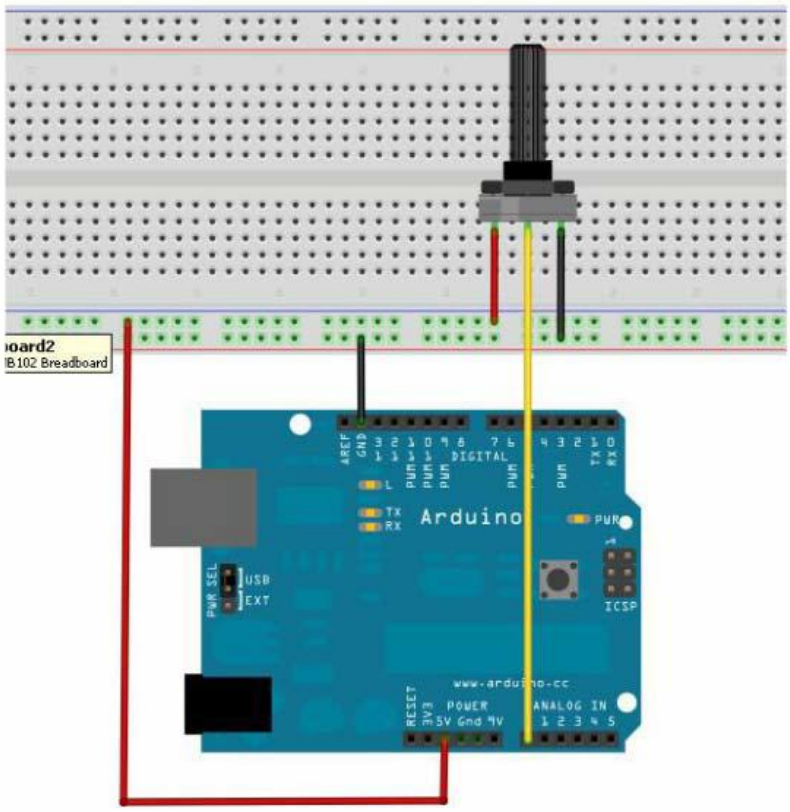
模擬讀值實驗

簡述

本章我們將使用模擬 I/O 口，Arduino 有模擬口一模擬 5 共計 6 個模擬接口，這 6 個接口也可以算作為接口功能複用，除模擬接口功能以外，這 6 個接口可作為數字接口使用，編號為數字 14 – 數字 19 。下面我們開始做實驗了。電位器（或者叫滑動電阻）是大家比較熟悉的典型的模擬值輸出元件，本實驗就用它來完成。







analogRead();語句就可以讀出模擬口的值，Arduino UNO 控制器是 10 位的 A/D 採集，所以讀取的模擬值範圍是 0-1023 。首先我們在 void setup()裡面設置波特率，顯示數值屬於 Arduino 與 PC 機通信，所以 Arduino 的波特率應與 PC 機軟件設置的相同才能顯示正確的數值，否則將會顯示亂碼或者不顯示。在 Arduino 軟件的串口工具監視窗口右下角有一個可以設置波特率的按鈕，選中與程序中設置的比特率語句相同的比特率，Serial.begin();括號中為比特率的值。

CODE

```
int potpin = 0 ; //定義模擬接口 0
int ledpin = 13 ; //定義數字接口 13
int val = 0 ; //將定義變量 val，並賦初值 0.
void setup()
{
pinMode(ledpin,OUTPUT);//設置數字 13 引腳為輸出模式
Serial.begin(9600);//設置波特率 9600
}
void loop()
{
digitalWrite(ledpin,HIGH);//點亮數字接口 13 的 LED
delay(50);//延時 0.05 秒
digitalWrite(ledpin,LOW);//熄滅數字接口 13 的 LED
delay(50);//延時 0.05 秒
val = analogRead(potpin);//讀取模擬接口 0 的值，並將其賦給 val
Serial.println(val) ; //顯示出 val 的值
}
```

每讀取一次值，arduino 自帶的 LED 小燈就會閃爍一下，下圖為讀出的值（注意只做參考值）。
本實驗中，當您旋轉電位計旋鈕的時候就可以看到屏幕上的數值變化了 。這種模擬值讀取是我們很常用的功能。因為在很多的傳感器，都是模擬值輸出，我們讀出模擬值後再進行相應的算法處理，就可以應用到我們需要實現的功能裡了。

光控實驗

認識光敏電阻

光敏電阻又稱光導管，常用的製作材料為硫化鎘，另外還有硒、硫化鋁、硫化鉛和硫化鉍等材料。這些製作材料具有在特定波長的光照下，其阻值迅速減小的特性。這是由於光照產生的載流子都參與導電，在外加電場的作用下漂移運動，從而使光敏電阻的阻值迅速下降。

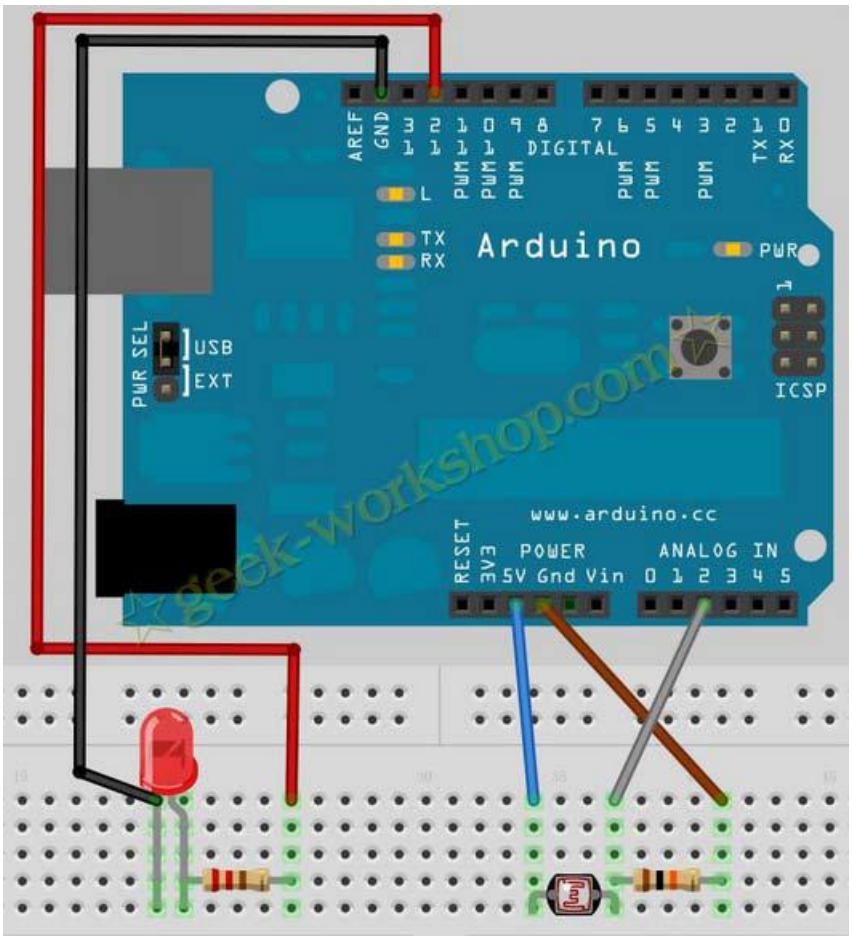
光敏電阻的工作原理是基於內光電效應。在半尋體光敏材料兩端裝上電極引線，將其封裝在帶有透明窗的管殼裡就極成光敏電阻，為了增加靈敏度，兩電極常做成梳狀。在有光照射時入射光強，電阻減小，入射光弱，電阻增大。

實驗器件

光敏電阻：1 個
蜂鳴器：1 個
10 K 歐電阻 ： 1 個
220 歐電阻 ： 1 個
多彩麵包板實驗跳線：若干

光敏電阻的連線

光敏電阻在使用中可以直接與所控制器件相連。

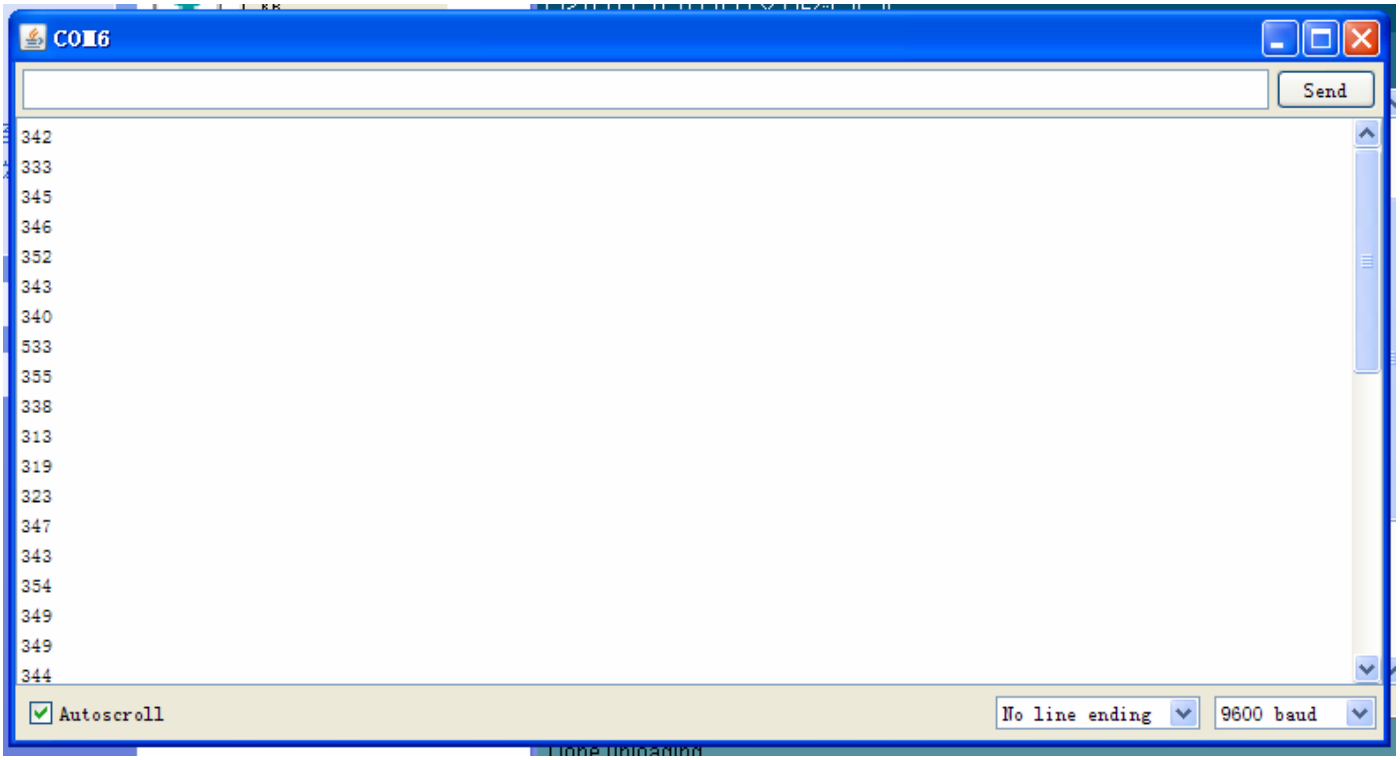


光敏電阻是根據光強度改變阻值的元件，本章實驗我們接著上一章學過的知識，使用模擬口讀取模擬值。

CODE

```
int photocellPin = 2; //定義變量 photocellsh=2，為電壓讀取端口。
int ledPin = 12; //定義變量 ledPin=12，為 led 電平輸出端口
int val = 0; //定義 val 變量的起始值
void setup() {
pinMode(ledPin, OUTPUT); //使 ledPin 為輸出模式
}
void loop() {
val = analogRead(photocellPin); //從傳感器讀取值
if(val<=512){
//512=2.5V，想讓傳感器敏感一些的時候，把數值調高，想讓傳感器遲鈍的時候把數值調低。
digitalWrite(ledPin, HIGH); //當 val 小於 512(2.5V)的時候，led 亮。
}
else{
digitalWrite(ledPin, LOW);
}
}
```

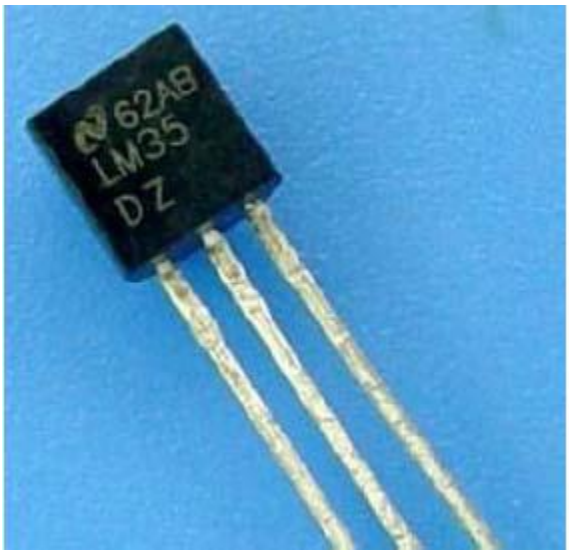
本次實驗設計的效果是，當光照正常的時候 led 燈是滅的，當周圍變暗時 led 燈變亮。因為光敏電阻受不同光照影響變化很大，所以本次實驗的參數是在 60W 三基色節能燈照射下實驗（無日光照射），同樣亮度的日光下光敏電阻的阻值會比日光燈下低不少，估計和不同光的波段有關係。不同環境下實驗使用的參數不同，大家根據原理進行調整。



LM35 溫度傳感器實驗

什麼是溫度傳感器？

溫度傳感器就是利用物質各種物理性質隨溫度變化的規律把溫度轉換為電量的傳感器。返些呈現規律性變化的物理性質主要有體。溫度傳感器是溫度測量儀表的核心部分，品種繁多。按測量方式可分為接觸式和非接觸式兩大類，按照傳感器材料及電子元件特性分為熱電阻和熱電偶兩類。本實驗使用的是 LM35 溫度傳感器。如下圖：



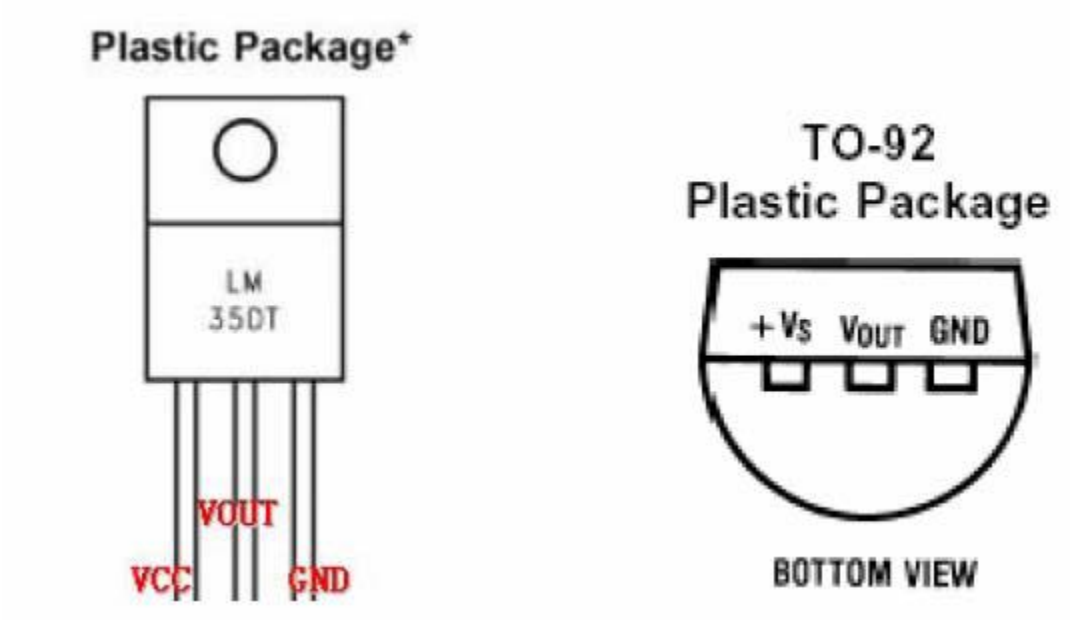
工作原理

LM35 溫度傳感器的輸出電壓不攝氏溫標呈線性關係，0℃，時輸出為 0V，每升高 1℃，輸出電壓增加 10mV。轉換公式如下：

$$V_{out\_LM35}(T) = 10\,\mathrm{mV}/^{\circ}\mathrm{C} \times T^{\circ}\mathrm{C}$$

LM35 的連線

LM35 的引腳示意圖如下：

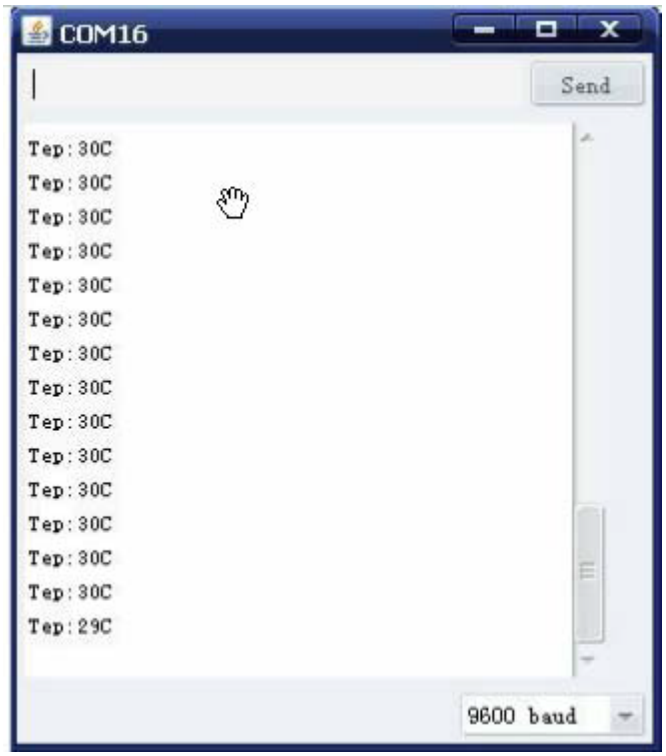


從實驗盒中將溫度傳感器拿出來可以看到，溫度傳感器的一面是平的，另一面是半圓的。將平面對著自己，最左邊的是 VCC 引腳（接+5v），中間的為 VOUT（電壓值輸出引腳，接板子上的模擬引腳），最右邊的引腳為 GND 引腳（接板子上的 GND）。三個引腳分別接好就可以用了。

CODE

```
int potPin = 0 ;//定義模擬接口 0 連接 LM35 溫度傳感器
void setup()
{
Serial.begin(9600);//設置波特率
}
void loop()
{
int val;//定義變量
int dat;//定義變量
val = analogRead(potPin);//讀取傳感器的模擬值並賦值給 val
dat = (125*val)>>8 ; //溫度計算公式
Serial.print("Tep : ") ; //原樣輸出顯示 Tep 字符串代表溫度
Serial.print(dat) ; //輸出顯示 dat 的值
Serial.println("C"); //原樣輸出顯示 C 字符串
delay(500);//延時 0.5 秒
}
```

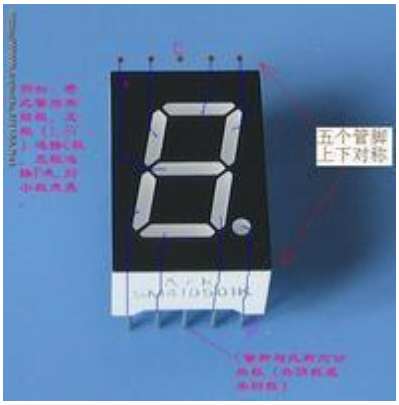
將程序下載到實驗板，打開監視器，就可以看到當前的環境溫度了。 （ 實際上，溫度值有一點點偏差，要根據自己的環境溫度修改一下程序，使其完全與自己的環境一致。 ）



數碼管顯示器實驗

認識數碼管

數碼管是一種半尋體發光器件，其基本單元是収光二糸管.數碼管按段數分為七段數碼管和八段數碼管，八段數碼管比七段數碼管多一個發光二極管單元（多一個小數點顯示）；



按能顯示多少個“8”可分為 1 位、2 位、4 位等等數碼管。

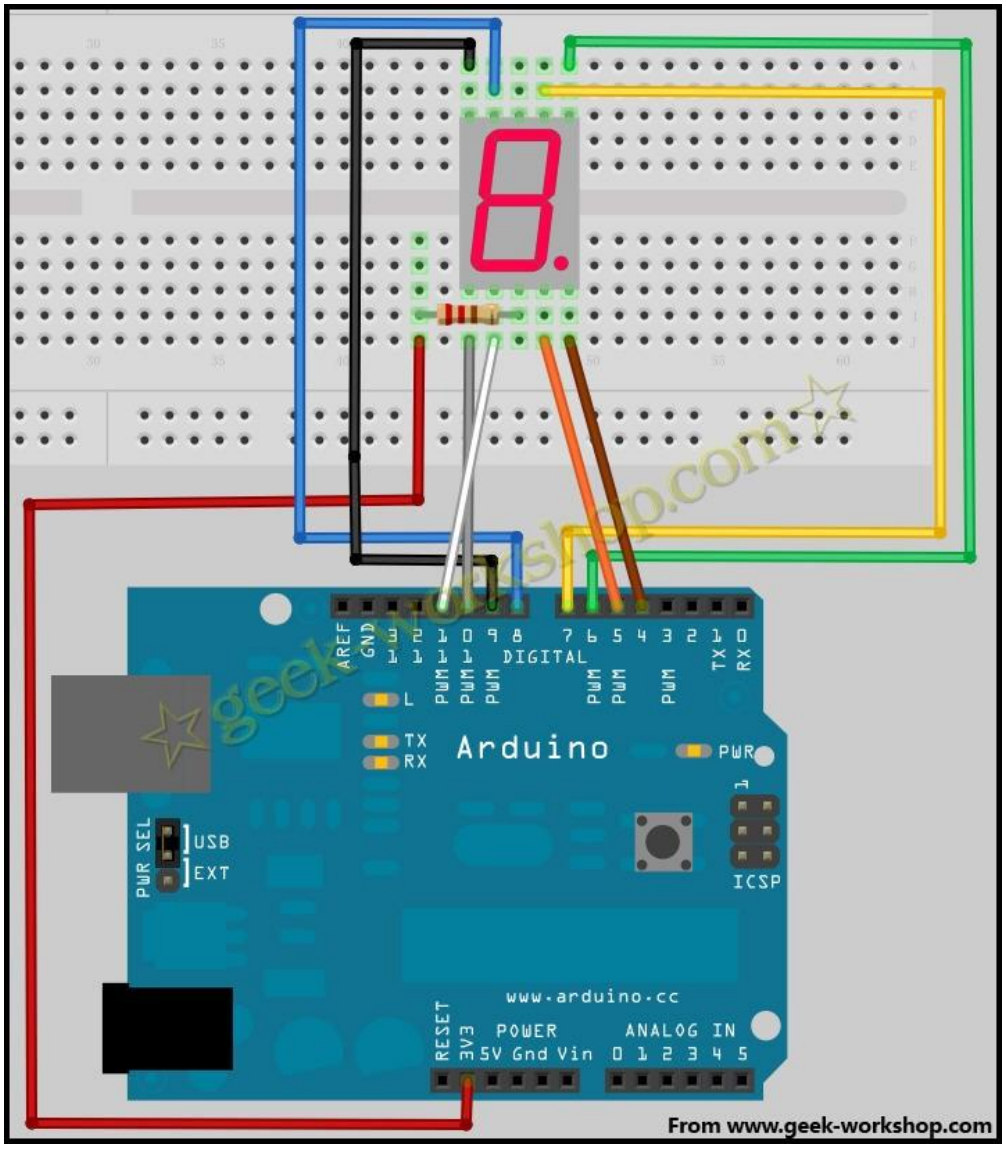
將限流電阻的一端捏到數字 I/O 中，另一端數碼管的字段引腳相連，剩下的六個字段和一個小數點依次按照返種方法接。將公共 COM 如果是共陽極的就接到+5V，如果是共陰極的就接到 GND。

實驗器件

數碼管：1 個
220Ω的電阻：8 個
多彩麵包板實驗跳線：若干

實驗連線

按照 Arduino 教程將控制板、麵包板連接好，下載線捏好。按數碼管的接法將數碼管 g 段通過限流電阻不數字的 9 引腳相連，如圖 3.4 中的（a）圖，f 段通過限流電阻不數字 8 引腳相連，共陽極與 5V 接口相連，同樣的接法 a、b 分別接 7、6 引腳，e、d 分別接 10、11 引腳，第二個共陽極可以不接，c、DP 分別接 5、4 引腳，連線完畢。如下圖：



數碼管共有七段顯示數字的段，還有一個顯示小數點的段。當讓數碼管顯示數字時，另要將相應的段點亮即可。例如：讓數碼管顯示數字 1，則將 b、c 段點亮即可。將每個數字寫成一個子程序。在主程序中每隔 2s 顯示一個數字，讓數碼管循環顯示 1～8 數字。每一個數字顯示的時間由延時時間來決定，時間設置的大些，顯示的時間就長些，時間設置的小些，顯示的時間就短。

CODE

```
//設置控制各段的數字 IO 腳
int a=7;
int b=6;
int c=5;
int d=11;
int e=10;
int f=8;
int g=9;
int dp=4;
//顯示數字 1
void digital_1(void)
```

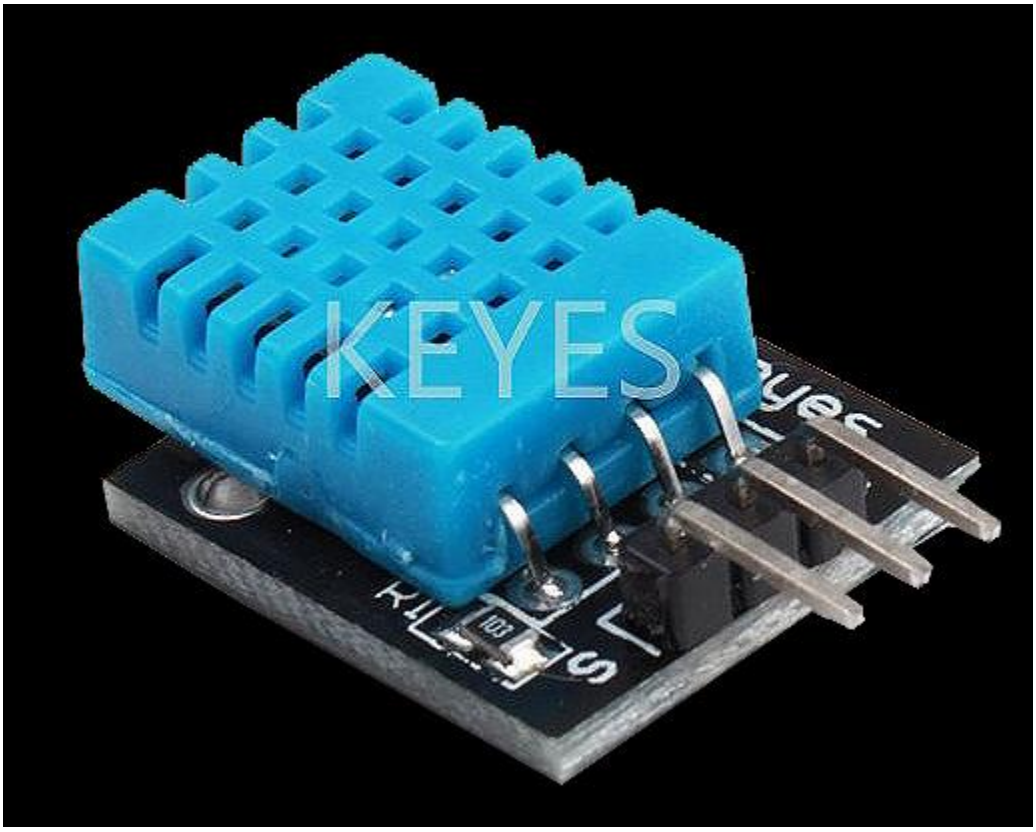
```
{
unsigned char j;
digitalWrite(c,LOW);//給數字 5 引腳低電平，點亮 c 段
digitalWrite(b,LOW);//點亮 b 段
for( j=7;j<=11;j++)//熄滅其餘段
digitalWrite(j,HIGH);
digitalWrite(dp,HIGH);//熄滅小數點 DP 段
}
//顯示數字 2
void digital_2(void)
{
unsigned char j;
digitalWrite(b,LOW);
digitalWrite(a,LOW);
for( j=9;j<=11;j++)
digitalWrite(j,LOW);
digitalWrite(dp,HIGH);
digitalWrite(c,HIGH);
digitalWrite(f,HIGH);
}
//顯示數字 3
void digital_3(void)
{
unsigned char j;
digitalWrite(g,LOW);
digitalWrite(d,LOW);
for( j=5;j<=7;j++)
digitalWrite(j,LOW);
digitalWrite(dp,HIGH);
digitalWrite(f,HIGH);
digitalWrite(e,HIGH);
}
//顯示數字 4
void digital_4(void)
{
digitalWrite(c,LOW);
digitalWrite(b,LOW);
digitalWrite(f,LOW);
digitalWrite(g,LOW);
digitalWrite(dp,HIGH);
digitalWrite(a,HIGH);
digitalWrite(e,HIGH);
```

```
digitalWrite(d,HIGH);
}
//顯示數字 5
void digital_5(void)
{
unsigned char j;
for( j=7;j<=9;j++)
digitalWrite(j,LOW);
digitalWrite(c,LOW);
digitalWrite(d,LOW);
digitalWrite(dp,HIGH);
digitalWrite(b,HIGH);
digitalWrite(e,HIGH);
}
//顯示數字 6
void digital_6(void)
{
unsigned char j;
for( j=7;j<=11;j++)
digitalWrite(j,LOW);
digitalWrite(c,LOW);
digitalWrite(dp,HIGH);
digitalWrite(b,HIGH);
}
//顯示數字 7
void digital_7(void)
{
unsigned char j;
for( j=5;j<=7;j++)
digitalWrite(j,LOW);
digitalWrite(dp,HIGH);
for( j=8;j<=11;j++)
digitalWrite(j,HIGH);
}
//顯示數字 8
void digital_8(void)
{
unsigned char j;
for( j=5;j<=11;j++)
digitalWrite(j,LOW);
digitalWrite(dp,HIGH);
}
```

```
void setup()
{
int i;//定義發量
for(i=4;i<=11;i++)
pinMode(i,OUTPUT);//設置 4～11 引腳為輒出模式
}
void loop()
{
while(1)
{
digital_1();//數字 1
delay(2000);//延時 2s
digital_2();
delay(2000);
digital_3();
delay(2000);
digital_4();
delay(2000);
digital_5();
delay(2000);
digital_6();
delay(2000);
digital_7();
delay(2000);
digital_8();
delay(2000);
}
}
```

在 setup()前面定義了一系列的數字顯示子程序，返些子程序的定義可以方便在 loop()中使用，使用時另需將子程序的名寫上即可。將程序下載到實驗板後我夫可以看到，數碼管循環顯示數字 1~8，每隔數字顯示兩秒鐘。掌握本程序後，大家可以収揮自己的想像，做出各種數碼管實驗。

DHT11 溫溼度感應模塊



DHT11 數字溫濕度傳感器是一款含有已校準數字信號輸出的溫濕度複合傳感器，它應用專用的數字模塊採集技術和溫濕度傳感技術，確保產品具有極高的可靠性和卓越的長期穩定性。該產品具有品質卓越、超快響應、抗干擾能力強、性價比極高等優點。單線製串行接口，使系統集成變得簡易快捷。超小的體積、極低的功耗，信號傳輸距離可達 20 米以上，使其成為給類應用甚至最為苛刻的應用場合的最佳選擇。產品為 4 針單排引腳封裝，連接方便。

技術參數

供電電壓： 3.3~5.5V DC
輸出： 單總線數字信號

測量範圍：
- 濕度 20-90%RH
- 溫度 0~50℃

測量精度：
- 濕度+-5%RH
- 溫度+-2℃

分辨率：
- 濕度 1% RH
- 溫度 1℃

長期穩定性： <±1%RH/年

注意事項

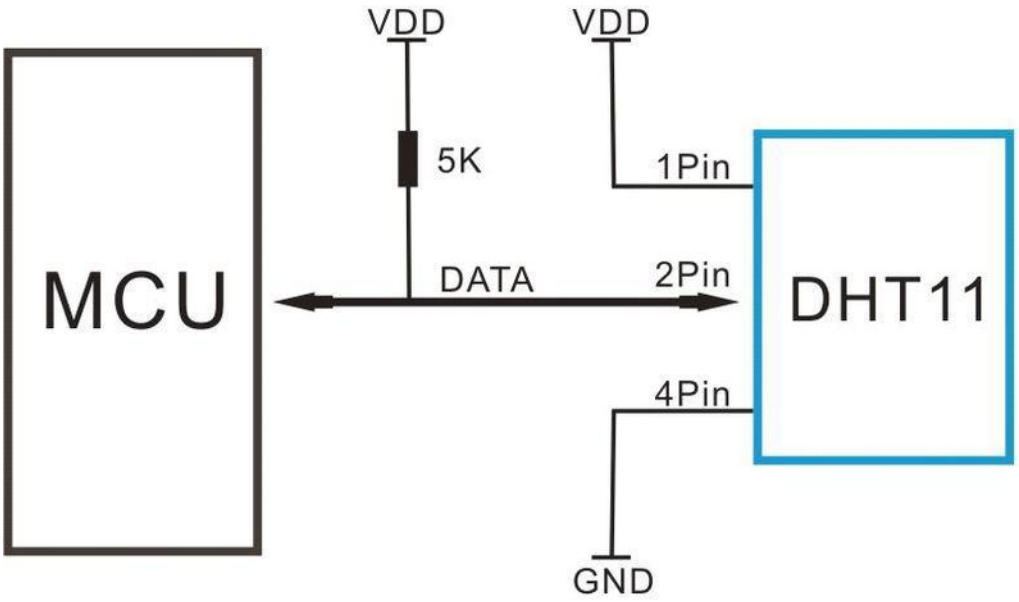
1、避免在結露情況下使用

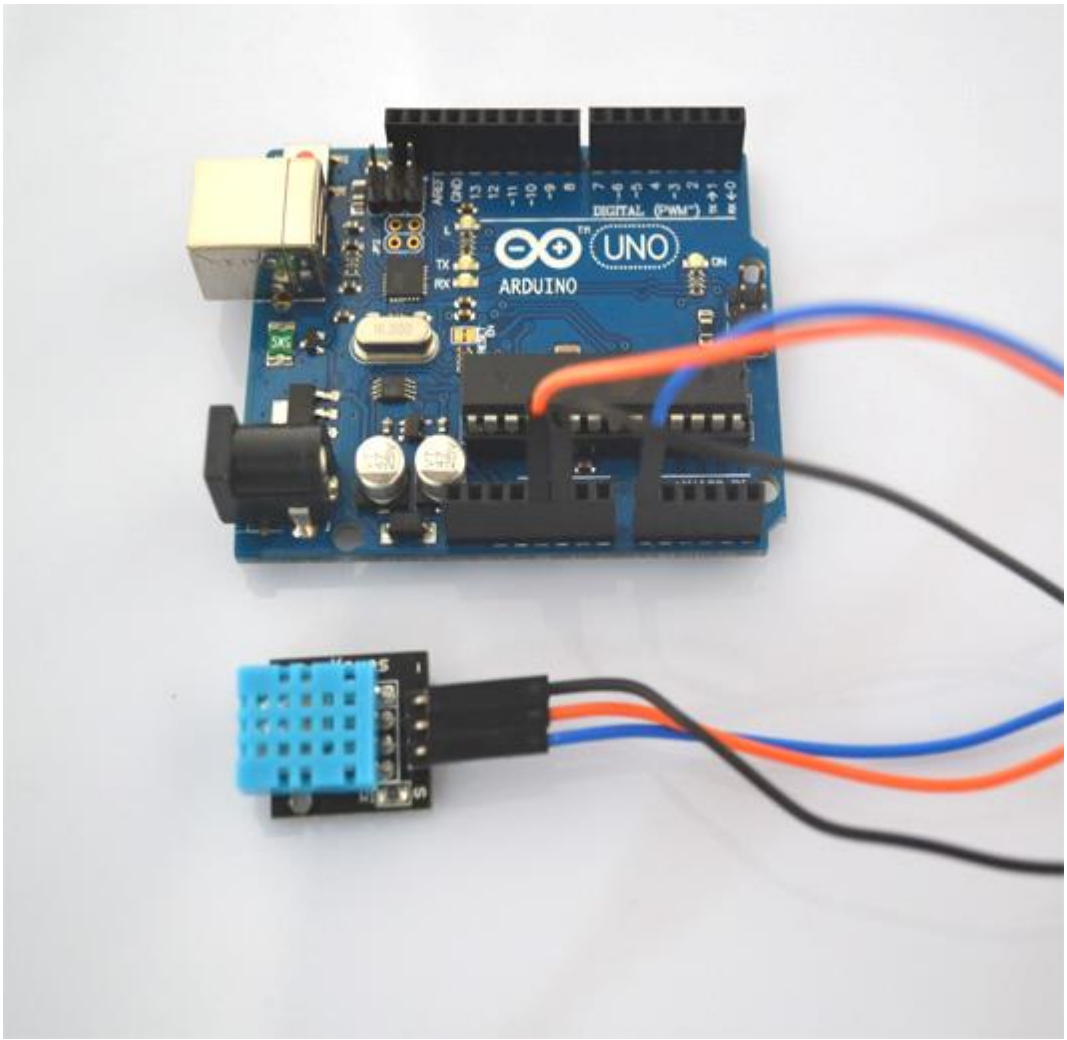
2、長期保存溫度 10-40℃，濕度 60%以下

3、使用時電源和地接法要正確，以免損壞傳感器

實驗

下面為模塊如何與 Arduino 板連接





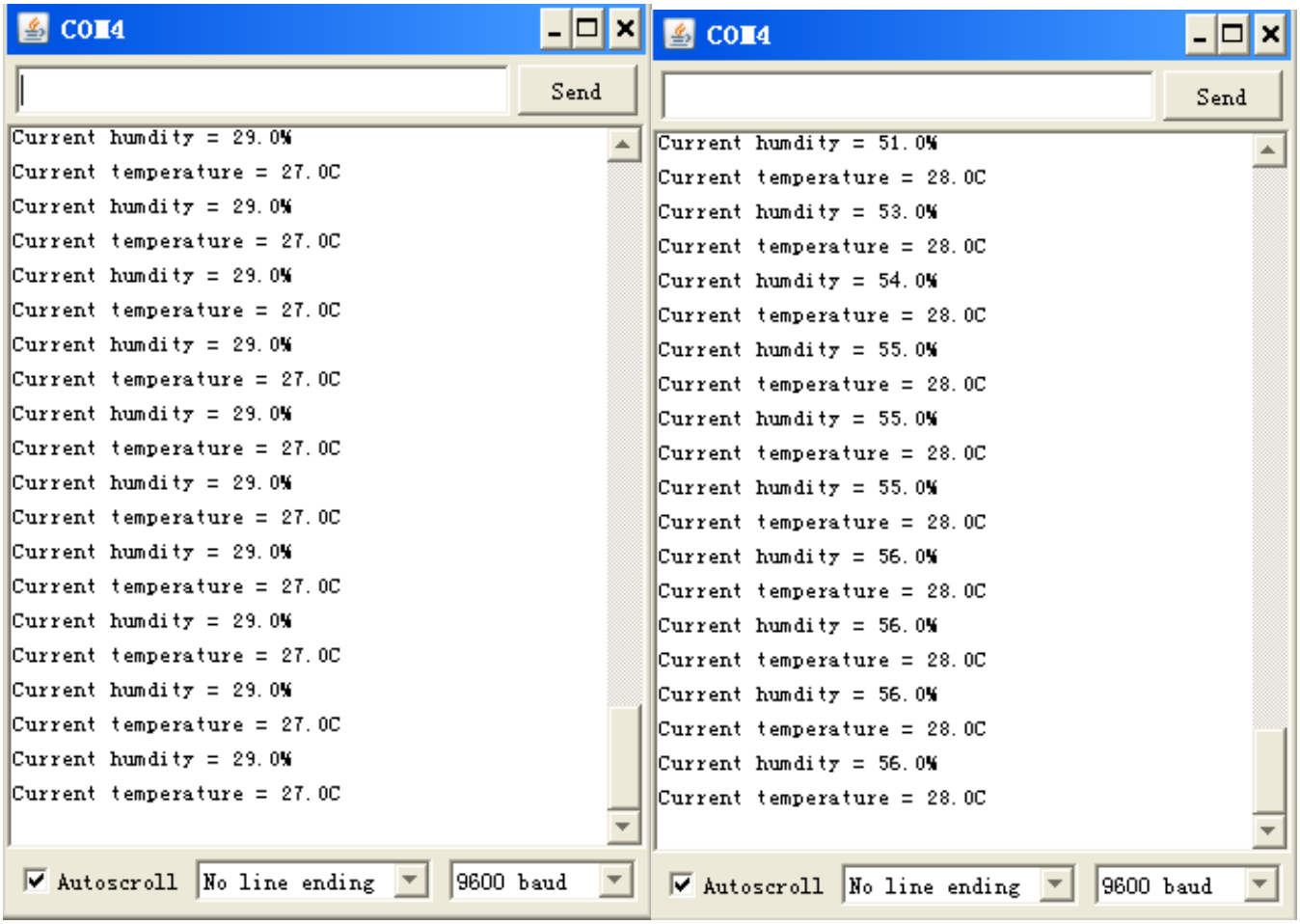
CODE

```
#define dht_dpin A0
byte bGlobalErr;
byte dht_dat[5];

void setup(){
InitDHT();
Serial.begin(9600);
delay(300);
Serial.println("Humidity and temperature\n\n");
delay(700);
}

void loop(){
   ReadDHT();
   switch (bGlobalErr){
        case 0:
  Serial.print("Current humdity = ");
  Serial.print(dht_dat[0], DEC);
  Serial.print(".");
  Serial.print(dht_dat[1], DEC);
  Serial.print("%    ");
  Serial.print("temperature = ");
  Serial.print(dht_dat[2], DEC);
  Serial.print(".");
  Serial.print(dht_dat[3], DEC);
  Serial.println("C    ");
            break;
        case 1:
            Serial.println("Error 1: DHT start condition 1 not met.");
            break;
        case 2:
            Serial.println("Error 2: DHT start condition 2 not met.");
            break;
        case 3:
            Serial.println("Error 3: DHT checksum error.");
            break;
        default:
            Serial.println("Error: Unrecognized code encountered.");
            break;
         }
   delay(800);
```

```
}

void InitDHT(){
    pinMode(dht_dpin,OUTPUT);
        digitalWrite(dht_dpin,HIGH);
}

void ReadDHT(){
bGlobalErr=0;
byte dht_in;
byte i;
digitalWrite(dht_dpin,LOW);
delay(20);

digitalWrite(dht_dpin,HIGH);
delayMicroseconds(40);
pinMode(dht_dpin,INPUT);
//delayMicroseconds(40);
dht_in=digitalRead(dht_dpin);

if(dht_in){
    bGlobalErr=1;
    return;
    }
delayMicroseconds(80);
dht_in=digitalRead(dht_dpin);

if(!dht_in){
    bGlobalErr=2;
    return;
    }
delayMicroseconds(80);
for (i=0; i<5; i++)
    dht_dat[i] = read_dht_dat();
pinMode(dht_dpin,OUTPUT);
digitalWrite(dht_dpin,HIGH);
byte dht_check_sum =
        dht_dat[0]+dht_dat[1]+dht_dat[2]+dht_dat[3];
if(dht_dat[4]!= dht_check_sum)
    {bGlobalErr=3;}
};
```

```
byte read_dht_dat(){
   byte i = 0;
   byte result=0;
   for(i=0; i< 8; i++){
          while(digitalRead(dht_dpin)==LOW);
          delayMicroseconds(30);
          if (digitalRead(dht_dpin)==HIGH)
        result |=(1<<(7-i));
       while (digitalRead(dht_dpin)==HIGH);
       }
   return result;
}
```

我們把測試代碼編譯一下，編譯上傳通過我們就可以看結果了。

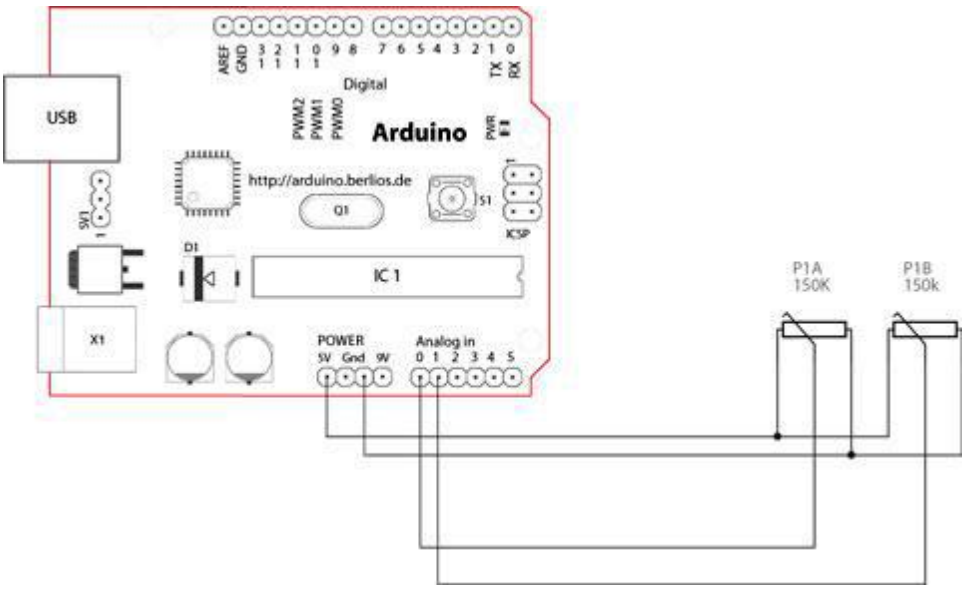


蘑菇頭控制器實驗

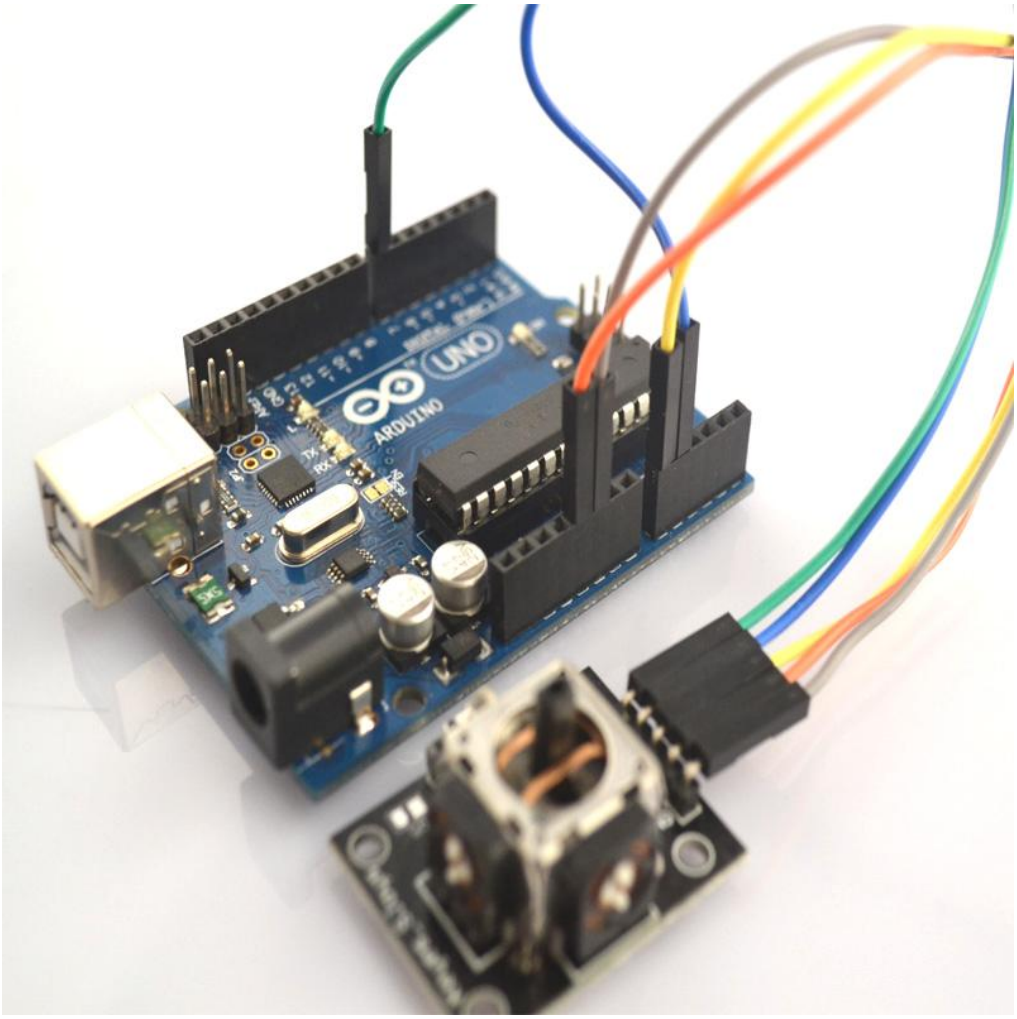
原理

它就像一個在遊戲控制台中操縱桿，你可以控制輸入這個操縱桿模塊的 x、y、z 的值以及在特定的值下實現某種功能，它可以被視為一個按鈕和電位計的組合。數據類型的 x,y 維為模擬輸入信號而 z 維是數字輸入信號，因此,x 和 y 端口連接到模擬插腳傳感器端,而 z 端口連接到數字端口。

在介紹如何使用前，我們先來看下它的工作原理吧，那樣我們也知道它裡面到底是怎麼回事，這對我們對它的使用很有幫助，下面有一個功能示意圖，我們一起來看看

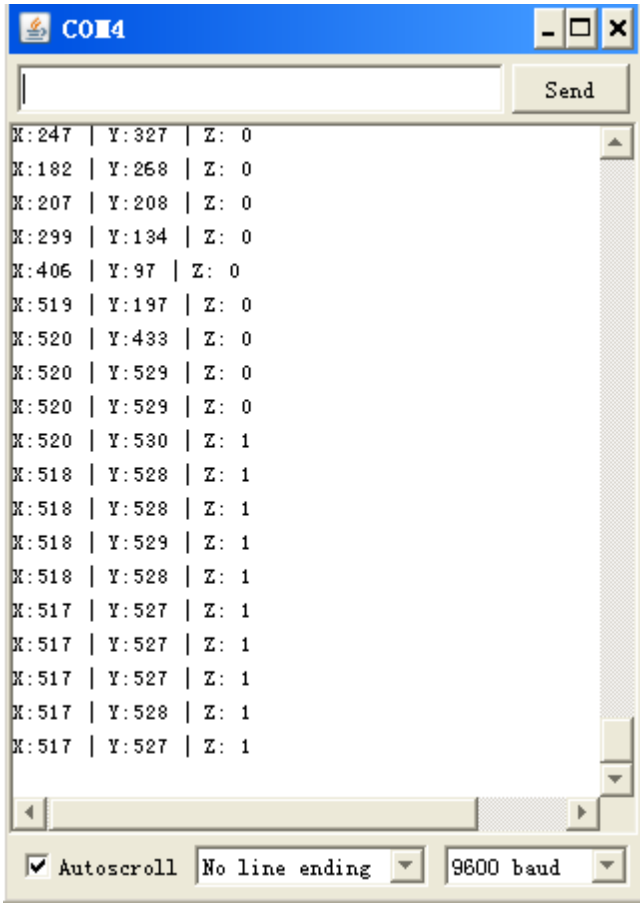


現在大家應該一目了然了吧，其實它就是一電位器嘛，x、y 維的數據輸出就是模擬端口讀出的電壓值，是不是有點意外。當然這上面沒有畫出 z 維的數據輸出，其實它更簡單，我們知道 z 維只輸出 0 和 1，那麼就通過一按鍵就能實現的吧。現在就應了我們上面說的一句話，它就是電位器和按鍵的組合體（說句實話，如果你對它不了解剛看到那句話是不是有點雲裡霧裡呢？）。看完上圖相信大家都知道如何在 Arduino 下使用它了吧，x、y 維我們接到兩個模擬端口去讀它們的值，而 z 維我們則接到數字口，這樣就行了，在加上電源和地，這樣就好了。



CODE

```
int sensorPin = 5;
int value = 0;
void setup() {
pinMode(7, OUTPUT);
Serial.begin(9600);
}
void loop() {
value = analogRead(0);
Serial.print("X:");
Serial.print(value, DEC);
value = analogRead(1);
Serial.print(" | Y:");
Serial.print(value, DEC);
value = digitalRead(7);
Serial.print(" | Z: ");
Serial.println(value, DEC);
delay(100);
}
```


COM4
Send
X:247 | Y:327 | Z: 0
X:182 | Y:268 | Z: 0
X:207 | Y:208 | Z: 0
X:299 | Y:134 | Z: 0
X:406 | Y:97 | Z: 0
X:519 | Y:197 | Z: 0
X:520 | Y:433 | Z: 0
X:520 | Y:529 | Z: 0
X:520 | Y:529 | Z: 0
X:520 | Y:530 | Z: 1
X:518 | Y:528 | Z: 1
X:518 | Y:528 | Z: 1
X:518 | Y:529 | Z: 1
X:518 | Y:528 | Z: 1
X:517 | Y:527 | Z: 1
X:517 | Y:527 | Z: 1
X:517 | Y:527 | Z: 1
X:517 | Y:528 | Z: 1
X:517 | Y:527 | Z: 1
Autoscroll
No line ending
9600 baud