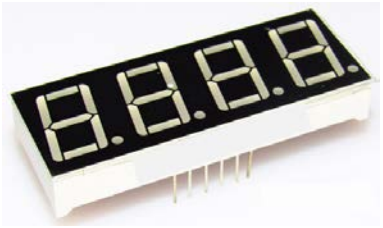
Chap 5
Arduino的I/O (2)

使用的 I/O 裝置

LED 燈

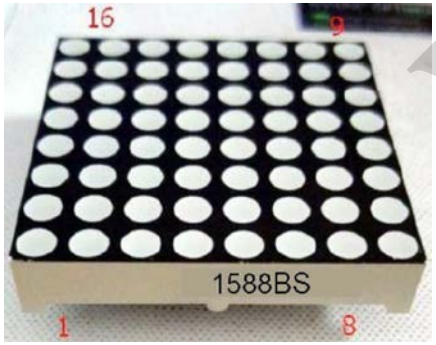
一位數七段

按鈕開關

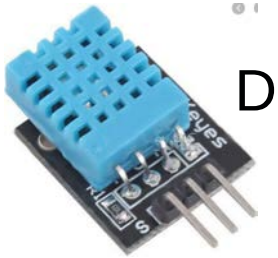


四位數七段

DS1302 RTC時鐘模組

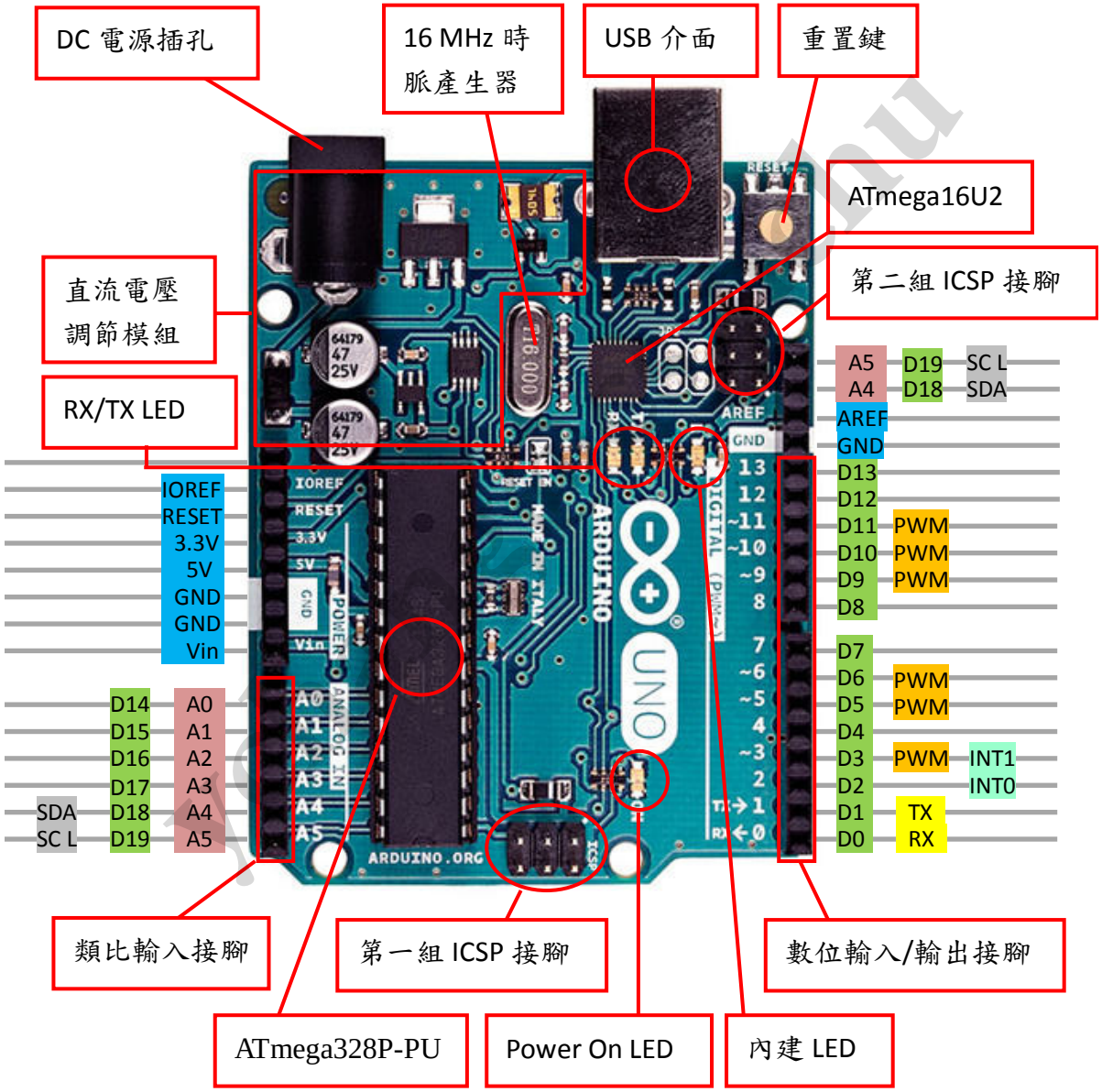


8x8 LED矩陣

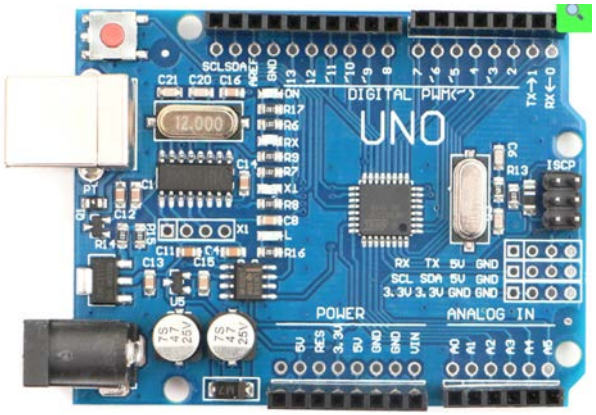


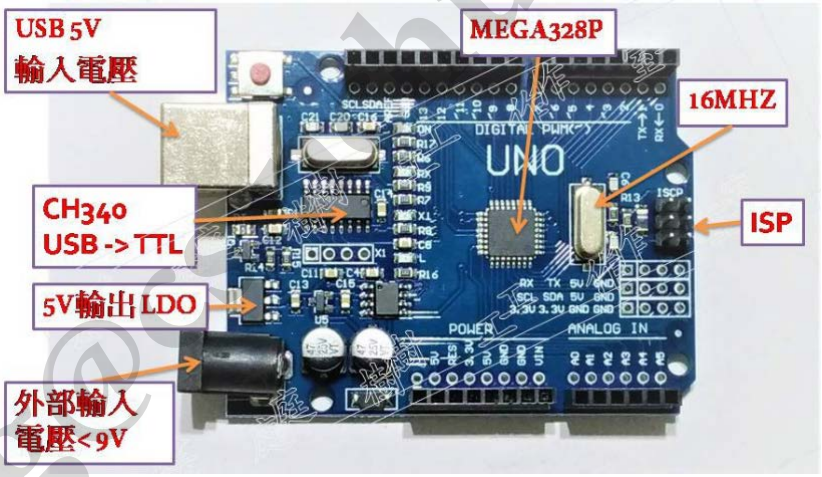
DHT11 溫濕度模組

Arduino UNO


DC 電源插孔
16 MHz 時脈產生器
USB 介面
重置鍵
ATmega16U2
第二組 ICSP 接腳
直流電壓調節模組
RX/TX LED
IOREF
RESET
3.3V
5V
GND
GND
Vin
A5 D19 SC L
A4 D18 SDA
AREF
GND
D13
D12
D11 PWM
D10 PWM
D9 PWM
D8
D7
D6 PWM
D5 PWM
D4
D3 PWM INT1
D2 INT0
D1 TX
D0 RX
D14 A0
D15 A1
D16 A2
D17 A3
SDA D18 A4
SC L D19 A5
類比輸入接腳
第一組 ICSP 接腳
數位輸入/輸出接腳
ATmega328P-PU
Power On LED
內建 LED

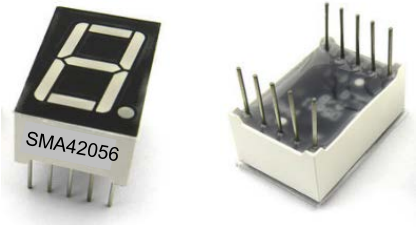
副廠Arduino Uno R3 經濟版

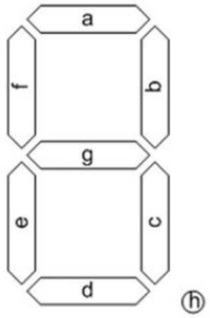


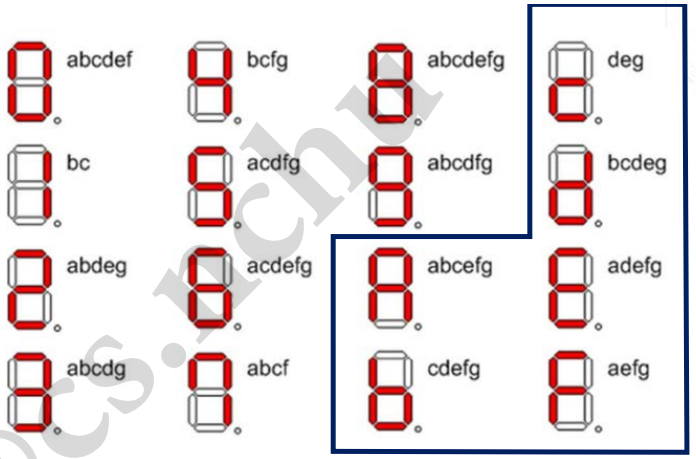


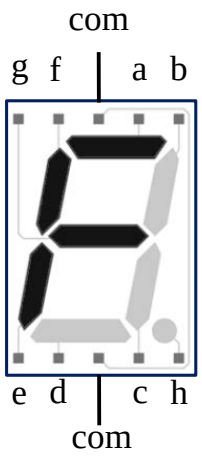
- 副廠Arduino Uno R3 經濟版，為了降低成本，ATMEGA328P晶片改為貼片封裝，ATmega16U2 改以 CH340G USB 晶片取代。並與原始Arduino Uno R3 100%相容。

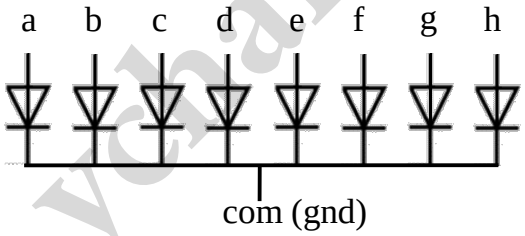
5-1 七段顯示器

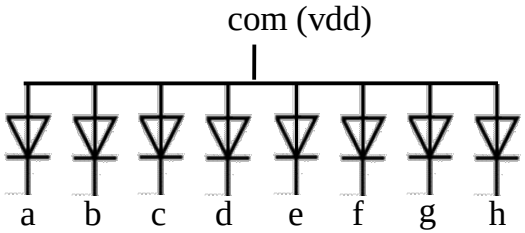












(a)共陰極電路 **(b)共陽極電路**

圖 5.5.2 七段顯示器的種類

5-1 一位數七段顯示器

表 5.5.1 共陰極七段顯示器數字 0~9 的編碼

	a	b	c	d	e	f	g	h
0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	0	0
2	1	1	0	1	1	0	1	0
3	1	1	1	1	0	0	1	0
4	0	1	1	0	0	1	1	0
5	1	0	1	1	0	1	1	0
6	1	0	1	1	1	1	1	0
7	1	1	1	0	0	1	0	0
8	1	1	1	1	1	1	1	0
9	1	1	1	1	0	1	1	0

```
const byte d0=B11111100;  // 數字0的定義，B表示二進位的格式
if(d0&B10000000) digitalWrite(2, HIGH); else digitalWrite(2, LOW);  //輸出a段的值1
if(d0&B01000000) digitalWrite(3, HIGH); else digitalWrite(3, LOW);  //輸出b段的值1
if(d0&B00100000) digitalWrite(4, HIGH); else digitalWrite(4, LOW);  //輸出c段的值1
if(d0&B00010000) digitalWrite(5, HIGH); else digitalWrite(5, LOW);  //輸出d段的值1
if(d0&B00001000) digitalWrite(6, HIGH); else digitalWrite(6, LOW);  //輸出e段的值1
if(d0&B00000100) digitalWrite(7, HIGH); else digitalWrite(7, LOW);  //輸出f段的值1
if(d0&B00000010) digitalWrite(8, HIGH); else digitalWrite(8, LOW);  //輸出g段的值0
if(d0&B00000001) digitalWrite(9, HIGH); else digitalWrite(9, LOW);  //輸出h段的值0
```

因為以上的程式碼具有高度的規律性，我們可以使用迴圈的方式改寫如下，可大幅的減少程式碼的數量

```
for(i=0; i<8; i++)  {
  mask=B10000000>>i;  //將1右移到正確的位置
  if(d0&mask) digitalWrite(2+i, HIGH); else digitalWrite(2+i, LOW);  //從a段到h段
}
```

5-1 一位數七段顯示器

```
/*** seg7.h ***/
const byte seg7_digit[10]={ B11111100,  //數字 0
                            B01100000,  //數字 1
                            B11011010,  //數字 2
                            B11110010,  //數字 3
                            B01100110,  //數字 4
                            B10110110,  //數字 5
                            B10111110,  //數字 6
                            B11100100,  //數字 7
                            B11111110,  //數字 8
                            B11110110}; //數字 9
int seg7_x1_FirstPin;  //一位數七段顯示器的第一隻接腳
int seg7_x4_FirstPin;  //四位數七段顯示器的第一隻接腳

//*** 初始化一位數七段顯示器 8 隻連續的接腳
void seg7_x1_init(int pin) {
seg7_x1_FirstPin=pin;  //設定一位數七段顯示器的第一隻接腳
for(int i=0; i<8; i++) pinMode(seg7_x1_FirstPin+i, OUTPUT);
}

//*** 一位數七段顯示器顯示數字
void seg7_x1_display(int num) {
byte mask;

  for(int i=0; i<8; i++)  {
    mask=B10000000>>i;  //將 1 右移到正確的位置
    if(seg7_digit[num] & mask) digitalWrite(seg7_x1_FirstPin+i, HIGH);
    else digitalWrite(seg7_x1_FirstPin+i, LOW);
  }
}
```

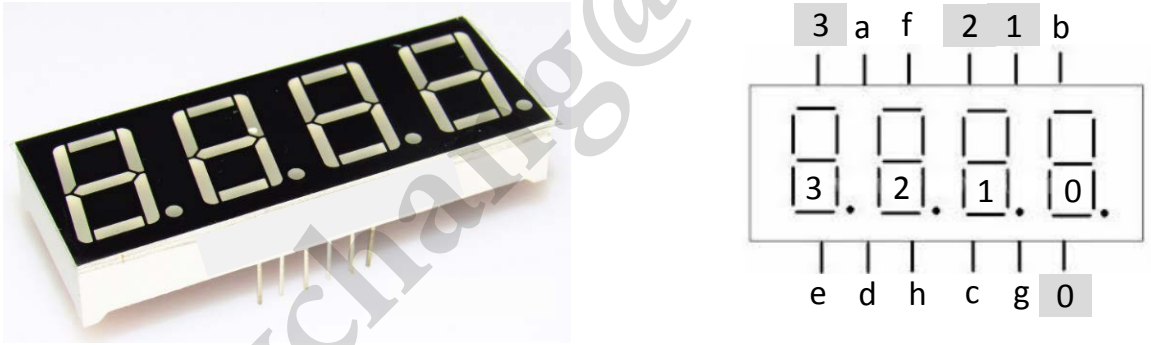
範例 5-1

- 使用一位數的七段顯示器來計數按鈕開關按下的次數，以按一下就加1的方式顯示，從數字0開始一直到數字9，然後又從0開始。

```
1  /*** 一位數七段顯示器範例 ***/
2  #include  “seg7.h”
3  #define Button 10  //指定按鈕開關的接腳為 D10
4  int num=0, flag=0;
5
6  void setup() {
7    pinMode(Button,INPUT_PULLUP); //設定 Button 接腳，並啟用內建的上拉電阻
8    seg7_x1_init(2);      //初始化七段顯示器，第一隻接腳為 D2，到 D9
9    seg7_x1_display(0);  //顯示數字 0
10 }
11
12 void loop() {
13   if(digitalRead(Button)==LOW)  //讀取 Button 接腳的電位是否為 LOW
14     { num=++num%10; flag=1; }   //若是，就代表按下開關，num+1 後取 10 的餘數
15   if(flag==1) {
16     seg7_x1_display(num);  //顯示數字 num
17     flag=0;
18   }
19 }
20
```

5-1 四位數七段顯示器

- 一顆七段顯示器需要8隻接腳控制顯示，四顆豈不是要32隻接腳，可是Arduino UNO所有數位接腳的總量也才14隻，即使全部用上了也不夠，那要如何顯示四個數字呢？

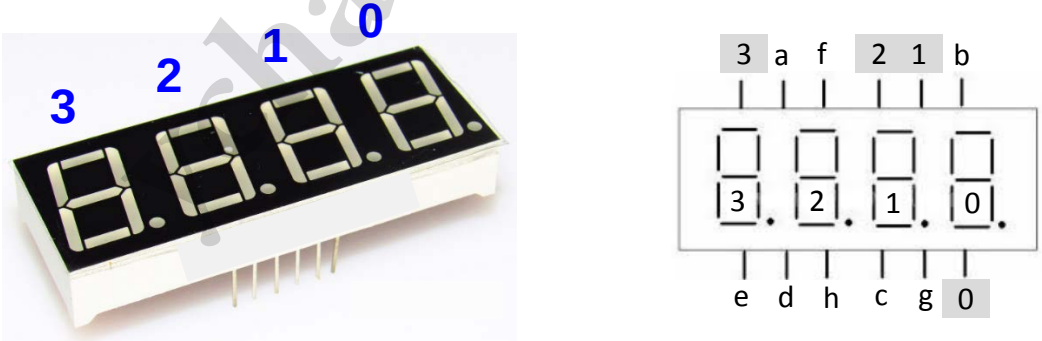


(a) (b)

圖 5.5.3 共陰極四位數七段顯示器及其腳位

5-1 四位數七段顯示器

- 其實這個問題的解法也不難，我們還是一次只顯示一個數字，從左到右（或從右到左）輪流顯示，只要二次掃描的間隔時間夠短，通常是小於1/16秒，就會因為眼睛視覺暫留的效應，而達到四個數字同時顯示的視覺效果，再次強調這不是真正的同時顯示，只是快速輪動所造成的視覺假象。
- 經由以上的解釋，除了原來8段LED的控制針腳a，b，c，d，e，f，g，h之外，還需要4隻針腳來控制哪一顆七段顯示器要動作，所以總共會有12隻針腳，如圖5.5.3(b)所示，其中0，1，2，3針腳分別控制四顆七段顯示器，因為是共陰極的電路，所以邏輯 **0** 代表致能，而邏輯 **1** 則為禁用。



(a) (b)

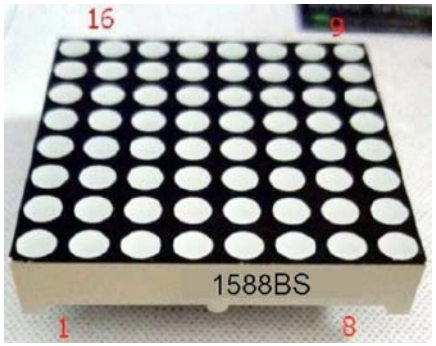
圖 5.5.3 共陰極四位數七段顯示器及其腳位

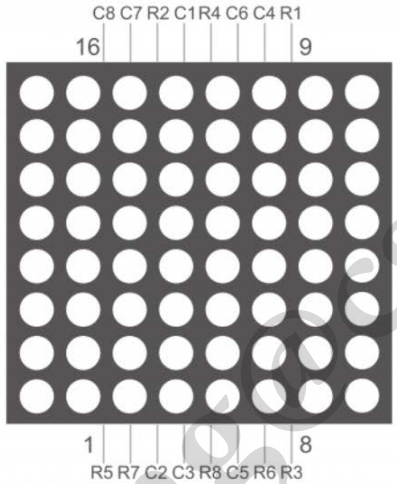
5-1 四位數七段顯示器

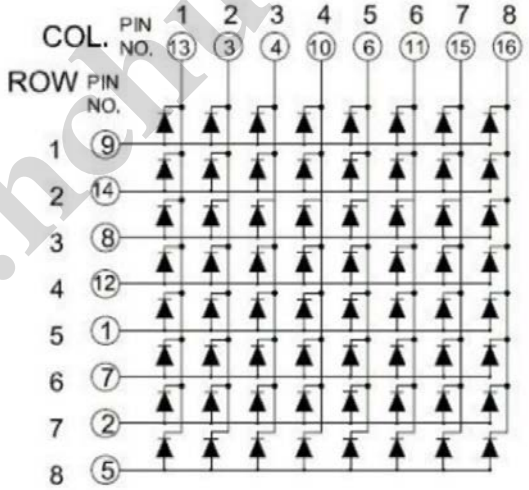
- 因為擴充了四位數的七段顯示器，所以我們必須把下列的程式碼加入seg7.h才夠使用，特別注意在數字顯示的函式中，必須指定四顆當中要在哪一顆顯示

```
1  //*** 初始化四位數七段顯示器 12 隻連續的接腳
2  void seg7_x4_init(int pin) {
3  seg7_x4_FirstPin=pin;  //設定四位數七段顯示器的第一隻接腳
4  for(int i=0; i<12; i++) pinMode(seg7_x4_FirstPin+i, OUTPUT);
5  }
6
7  //*** 四位數七段顯示器顯示數字
8  void seg7_x4_display(int digit, int num) {
9  byte mask;
10
11   //先關掉四顆七段顯示器
12   for(int i=8; i<12; i++) digitalWrite(seg7_x4_FirstPin+i, HIGH);
13   //再致能指定的七段顯示器
14   switch(digit) {
15     case 0: digitalWrite(seg7_x4_FirstPin+8, LOW); break;
16     case 1: digitalWrite(seg7_x4_FirstPin+9, LOW); break;
17     case 2: digitalWrite(seg7_x4_FirstPin+10, LOW); break;
18     case 3: digitalWrite(seg7_x4_FirstPin+11, LOW); break;
19   }
20   //顯示數字
21   for(i=0; i<8; i++)  {
22     mask=B10000000>>i;
23     if(seg7_digit[num] & mask) digitalWrite(seg7_x4_FirstPin+i, HIGH);
       else digitalWrite(seg7_x4_FirstPin+i, LOW);
     }
   }
```

5-2 8x8 LED 矩陣







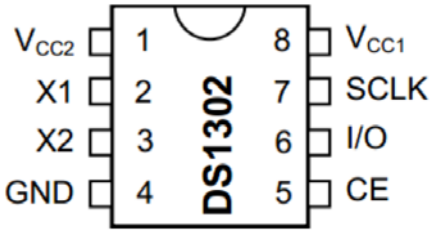
- 程式碼5-2.1
- 程式碼5-2.2

5-3 DS1302 RTC時鐘模組

- DS1302是DALLAS（後來被MAXIM收購）所推出的一款涓流充電(trickle-charge)時鐘晶片。DS1302實時時鐘晶片廣泛應用於電話、傳真、可攜式裝置等產品，它的主要性能指標如下

1. DS1302 是一個實時時鐘晶片，可以提供秒、分、小時、日期、月、年等信息，並且還有軟件自動調整的能力，可以通過配置 AM/PM 來決定採用24小時格式還是12小時格式。
2. 使用SPI(Serial Peripheral Interface）串列 I/O 通信方式，相對並行來說比較節省 IO 接腳的使用。
3. DS1302 的工作電壓比較寬，在 2.0～5.5 V 的範圍內都可以正常工作。
4. DS1302 這種時鐘晶片功耗一般都很低，它在工作電壓 2.0 V 的時候，工作電流小於 300 nA。
5. 擁有31字節數據存儲 RAM。

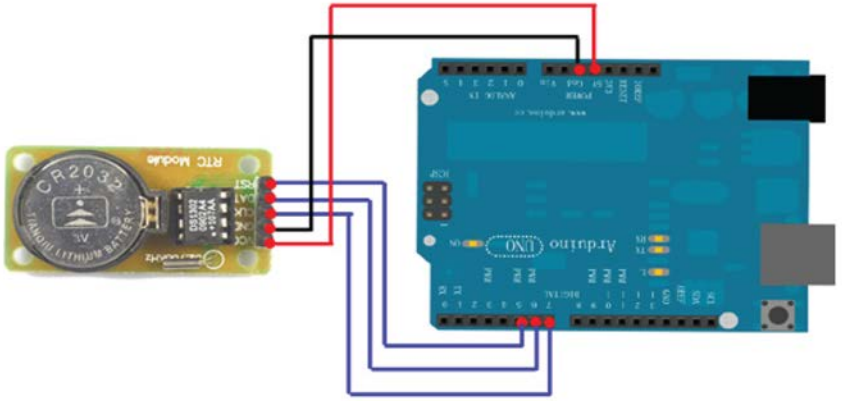
5-3 DS1302 硬體規格



引脚编号	引脚名称	引脚功能
1	Vcc2	主电源引脚，当 Vcc2 比 Vcc1 高 0.2V 以上时，DS1302 由 Vcc2 供电，当 Vcc2 低于 Vcc1 时，由 Vcc1 供电。
2	X1	这两个引脚需要接一个 32.768K 的晶振，给 DS1302 提供一个基准。特别注意，要求这个晶振的引脚负载电容必须是 6pF，而不是要加 6pF 的电容。如果使用有源晶振的话，接到 X1 上即可，X2 悬空。
3	X2	
4	GND	接地。
5	CE	DS1302 的使能输入引脚。当读写 DS1302 的时候，这个引脚必须是高电平，DS1302 这个引脚内部有一个 40k 的下拉电阻。
6	I/O	这个引脚是一个双向通信引脚，读写数据都是通过这个引脚完成。DS1302 这个引脚的内部含有一个 40k 的下拉电阻。
7	SCLK	输入引脚。SCLK 是用来作为通信的时钟信号。DS1302 这个引脚的内部含有一个 40k 的下拉电阻。
8	Vcc1	备用电源引脚。

DS1302		Arduino
RST	->	Arduino D5
DAT	->	Arduino D6
CLK	->	Arduino D7
GND	->	Arduino GND
VCC	->	Arduino +5v

如下图：



5-3 Open source Lib.

```
#include <stdio.h>
#include <DS1302.h>

namespace {

const int kCePin   = 5;  // Chip Enable
const int kIoPin   = 6;  // Input/Output
const int kSclkPin = 7;  // Serial Clock

// Create a DS1302 object.
DS1302 rtc(kCePin, kIoPin, kSclkPin);

String dayAsString(const Time::Day day) {
  switch (day) {
    case Time::kSunday: return "Sunday";
    case Time::kMonday: return "Monday";
    case Time::kTuesday: return "Tuesday";
    case Time::kWednesday: return "Wednesday";
    case Time::kThursday: return "Thursday";
    case Time::kFriday: return "Friday";
    case Time::kSaturday: return "Saturday";
  }
  return "(unknown day)";
}

void printTime() {
  // Get the current time and date from the chip.
  Time t = rtc.time();

  // Name the day of the week.
  const String day = dayAsString(t.day);
  char buf[50];
  snprintf(buf, sizeof(buf), "%s %04d-%02d-%02d %02d:%02d:%02d",
           day.c_str(),
           t.yr, t.mon, t.date,
           t.hr, t.min, t.sec);

  // Print the formatted string to serial so we can see the time.
  Serial.println(buf);
}
}  // namespace
```

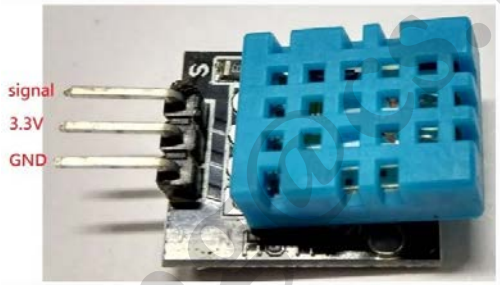
```
void setup() {
  Serial.begin(9600);
  rtc.writeProtect(false);
  rtc.halt(false);
  Time t(2013, 9, 22, 1, 38, 50, Time::kSunday);
  rtc.time(t);
}

// Loop and print the time every second.
void loop() {
  printTime();
  delay(1000);
}
```

程式碼

5-4 DHT11 溫濕度模組

- DHT11是一款經過校準過且直接以數字訊號輸出的溫濕度感測器。內含一個電阻式感濕元件和一個NTC測溫元件，並與一個8bit單晶片相連接。體積小、功耗低，傳輸距離最遠可達20公尺以上。



型號	測試範圍	測濕精度	測溫精度	分辨力
DHT11	20-90% RH 0-50°C	±5% RH	±2°C	1

DHT11 PIN腳說明

Pin	名稱	說明
S	Signal	信號線
	VCC	3~5.5V
–	GND	接地

5-4 DHT11 資料傳輸

4、串行接口 （单线双向）

DATA 用于微处理器与 DHT11之间的通讯和同步,采用单总线数据格式,一次通讯时间4ms左右,数据分小数部分和整数部分,具体格式在下面说明,当前小数部分用于以后扩展,现读出为零.操作流程如下:

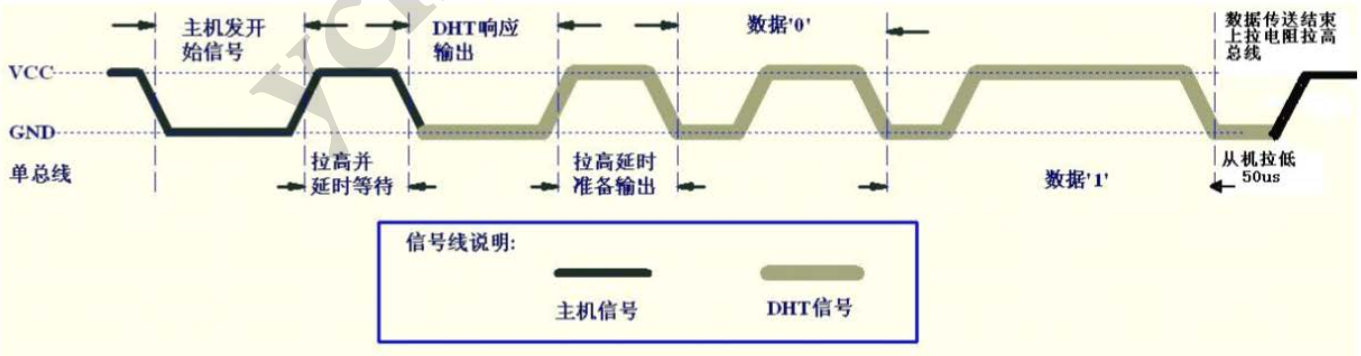
一次完整的数据传输为40bit,高位先出。

数据格式:8bit湿度整数数据+8bit湿度小数数据
+8bi温度整数数据+8bit温度小数数据
+8bit校验和

数据传送正确时校验和数据等于“**8bit湿度整数数据+8bit湿度小数数据+8bi温度整数数据+8bit温度小数数据**”所得结果的末8位。

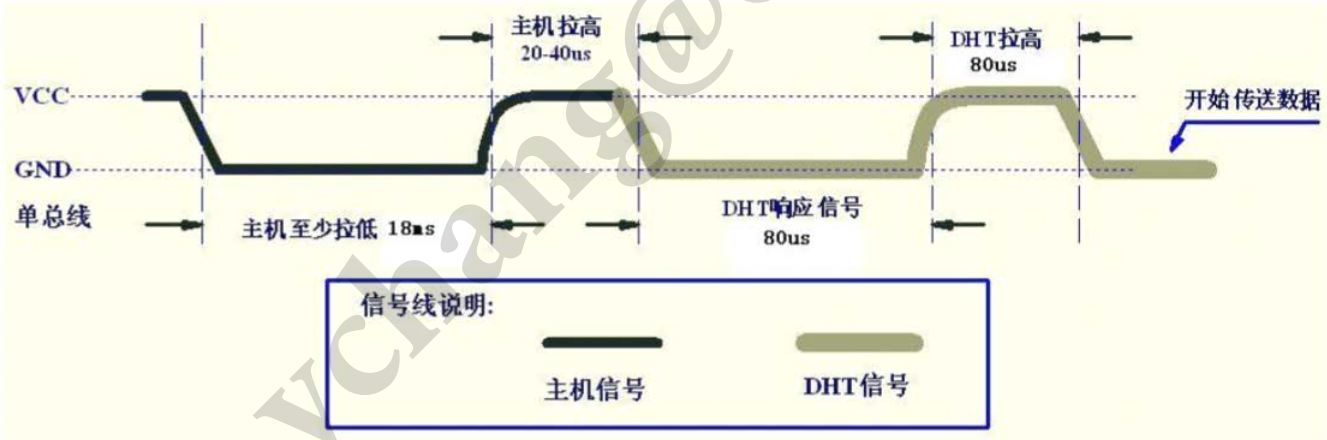
用户MCU发送一次开始信号后,DHT11从低功耗模式转换到高速模式,等待主机开始信号结束后,DHT11发送响应信号,送出40bit的数据,并触发一次信号采集,用户可选择读取部分数据.从模式下,DHT11接收到开始信号触发一次温湿度采集,如果没有接收到主机发送开始信号,DHT11不会主动进行温湿度采集.采集数据后转换到低速模式。

1.通讯过程如图1所示



5-4 DHT11 資料傳輸

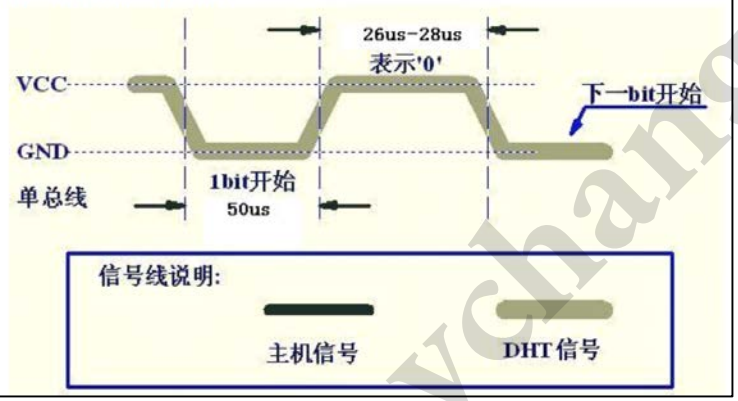
总线空闲状态为高电平，主机把总线拉低等待DHT11响应，主机把总线拉低必须大于18毫秒，保证DHT11能检测到起始信号。DHT11接收到主机的开始信号后，等待主机开始信号结束，然后发送80us低电平响应信号.主机发送开始信号结束后，延时等待20-40us后， 读取DHT11的响应信号，主机发送开始信号后，可以切换到输入模式，或者输出高电平均可， 总线由上拉电阻拉高。



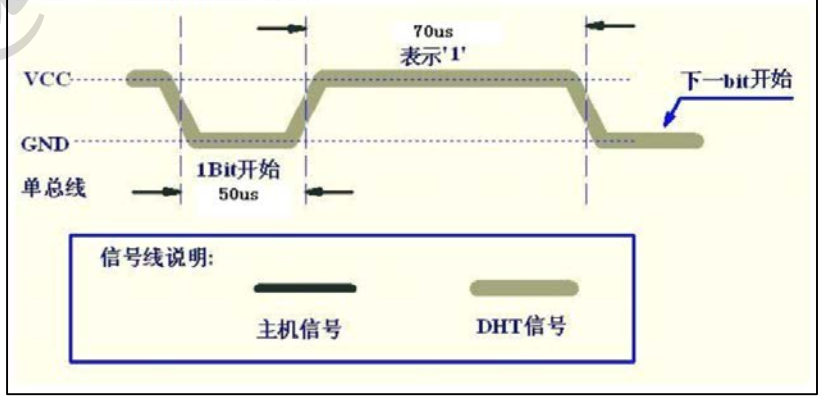
5-4 DHT11 資料傳輸

总线为低电平，说明DHT11发送响应信号，DHT11发送响应信号后，再把总线拉高80us，准备发送数据，每一bit数据都以50us低电平时隙开始，高电平的长短定了数据位是0还是1. 格式见下面图示. 如果读取响应信号为高电平，则DHT11没有响应，请检查线路是否连接正常. 当最后一bit数据传送完毕后，DHT11拉低总线50us，随后总线由上拉电阻拉高进入空闲状态。

数字0信号表示方法如图4所示



数字1信号表示方法. 如图5所示



```
int DHpin = 8;
byte dat[5];

byte read_data()
{
  byte data;
  for(int i=0; i<8; i++)
  {
    if(digitalRead(DHpin) == LOW)
    {
      while(digitalRead(DHpin) == LOW);
      delayMicroseconds(30);
      if(digitalRead(DHpin) == HIGH) data |= (1<<(7-i));
      while(digitalRead(DHpin) == HIGH);
    }
  }
  return data;
}

void start_test()
{
  digitalWrite(DHpin,LOW);
  delay(30);
  digitalWrite(DHpin,HIGH);
  delayMicroseconds(40);
  pinMode(DHpin,INPUT);
  while(digitalRead(DHpin) == HIGH);
  delayMicroseconds(80);
  if(digitalRead(DHpin) == LOW);
  delayMicroseconds(80);
  for(int i=0;i<4;i++) dat[i] = read_data();
  pinMode(DHpin,OUTPUT);
  digitalWrite(DHpin,HIGH);
}
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(DHpin,OUTPUT);
}

void loop()
{
  start_test();
  Serial.print("Current humdity = ");
  Serial.print(dat[0], DEC);
  Serial.print('.');
  Serial.print(dat[1],DEC);
  Serial.println('%');
  Serial.print("Current temperature = ");
  Serial.print(dat[2], DEC);
  Serial.print('.');
  Serial.print(dat[3],DEC);
  Serial.println('C');
  delay(700);
}
```

5-4 Open source Lib.

```
#include "DHT.h"
#define dhtPin 8       //讀取DHT11 Data
#define dhtType DHT11 //選用DHT11

DHT dht(dhtPin, dhtType); // Initialize DHT sensor

void setup() {
  Serial.begin(9600);//設定鮑率9600
  dht.begin();//啟動DHT
}

void loop() {
  float h = dht.readHumidity();//讀取濕度
  float t = dht.readTemperature();//讀取攝氏溫度
  float f = dht.readTemperature(true);//讀取華氏溫度
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("無法從DHT傳感器讀取！");
    return;
  }
  Serial.print("濕度: ");
  Serial.print(h);
  Serial.print("%\t");
  Serial.print("攝氏溫度: ");
  Serial.print(t);
  Serial.print("*C\t");
  Serial.print("華氏溫度: ");
  Serial.print(f);
  Serial.print("*F\n");
  delay(5000);//延時5秒
}
```


程式碼