Arduino 语言

Arduino 语言是建立在 C/C++基础上的，其实也就是基础的 C 语言，Arduino 语言只不过把 AVR 单片机（微控制器）相关的一些参数设置都函数化，不用我们去了解他的底层，让我们不了解 AVR 单片机（微控制器）的朋友也能轻松上手。

在与 Arduino DIYER 接触的这段时间里，发现有些朋友对 Arduino 语言还是比较难入手，那么这里我就简单的注释一下 Arduino 语言（本人也是半罐子水，有错的地方还请各位指正）。

基础 C 语言

关键字：

if...else

必须紧接着一个问题表示式(expression)，若这个表示式为真，紧连着表示式后的代码就会被执行。若这个表示式为假，则执行紧接着 else 之后的代码. 只使用 if 不搭配 else 是被允许的。

范例:

```
if (val == 1) {
digitalWrite(LED,HIGH);
}
```

for

用来明定一段区域代码重复指行的次数。

范例:

```
for (int i = 0; i < 10; i++) {
Serial.print("ciao");
}
```

switch case

if 叙述是程序里的分叉路口，switch case 是更多选项的路口。Swith case 根据变量值让程序有更多的选择，比起一串冗长的 if 叙述，使用 swith case 可使程序代码看起来比较简洁。

范例 :

```
switch (sensorValue) {
case 23:
digitalWrite(13,HIGH);
break;
case 46:
digitalWrite(12,HIGH);
break;
default: // 以上条件都不符合时，预设执行的动作
digitalWrite(12,LOW);
digitalWrite(13,LOW);
}
```

while

当 while 之后的条件成立时，执行括号内的程序代码。

范例 ：

```
// 当 sensor 值小于 512，闪烁 LED 灯
sensorValue = analogRead(1);
while (sensorValue < 512) {
digitalWrite(13,HIGH);
delay(100);
digitalWrite(13,HIGH);
delay(100);
sensorValue = analogRead(1);
}
```

do... while

和 while 相似，不同的是 while 前的那段程序代码会先被执行一次，不管特定的条件式为真或为假。因此若有一段程序代码至少需要被执行一次，就可以使用 do…while 架构。

范例 ：

```
do {
digitalWrite(13,HIGH);
delay(100);
digitalWrite(13,HIGH);
delay(100);
sensorValue = analogRead(1);
} while (sensorValue < 512);
```

break

让程序代码跳离循环，并继续执行这个循环之后的程序代码。此外，在 break 也用于分隔 switch case 不同的叙述。

范例 ：

```
//当 sensor 值小于 512，闪烁 LED 灯
do {
// 按下按钮离开循环
if (digitalRead(7) == HIGH)
break;
digitalWrite(13,HIGH);
delay(100);
digitalWrite(13,HIGH);
delay(100);
sensorValue = analogRead(1);
} while (sensorValue < 512);
```

continue

用于循环之内，它可以强制跳离接下来的程序，并直接执行下一个循环。

范例 :

```
for (light = 0; light < 255; light++)
{
// 忽略数值介于 140 到 200 之间
if ((x > 140) && (x < 200))
continue;
analogWrite(PWMpin, light);
delay(10);
}
```

return

函数的结尾可以透过 return 回传一个数值。

例如，有一个计算现在温度的函数叫 computeTemperature()，你想要回传现在的温度给 temperature 变量，你可以这样写：

```
int temperature = computeTemperature();
int computeTemperature() {
int temperature = 0;
temperature = (analogRead(0) + 45) / 100;
return temperature;
}
```

goto

语法符号：

; （分号）

Arduino 语言每一行程序都是以分号为结尾。这样的语法让你可以自由地安排代码，你可以将两个指令放置在同一行，只要中间用分号隔开。（但这样做可能降低程式的可读性。）

范例：

```
delay(100);
```

{} （大括号）

大括号用来将程式代码分成一个又一个的区块，如以下范例所示，在 loop()函数的前、后，必须用大括号括起来。

范例：

```
void loop(){
  Serial.pritln("cial");
}
```

程式的注释就是对代码的解释和说明，编写注释有助于程式设计师(或其他人)了解代码的功能。

Arduino 处理器在对程式码进行编译时会忽略注释的部份。

Arduino 语言中的编写注释有两种方式

//单行注释：这整行的文字会被处理器忽略

/*多行注释：

在这个范围内你可以

写一整首诗
*/
运算符：
=
+ 相加
- 相减
* 相乘
/ 相除
% 余数除法
== 等于
!=不等于
< 小于
> 大于
<= 小于等于
>= 大于等于
&& 交集
|| 联集
! 反相
++ 累加
-- 递减
+=
-=
*=
/=
数据类型：
boolean 布林
布尔变数的值只能为真(true)或是假(false)
char 字符
单一字符例如 A，和一般的计算机做法一样 Arduino 将字符储存成一个数字，即使你看到的明明就是一个文字。
用数字表示一个字符时，它的值有效范围为 -128 到 127。
注意：有两种主流的计算机编码系统 ASCII 和 UNICODE。ASCII 表示了 127 个字符， 用来在序列终端机和分时计算器之间传输文字。
UNICODE 可表示的字符量比较多，在现代计算机操作系统内它可以用来表示多国语言。
在位数需求较少的信息传输时，例如意大利文或英文这类由拉丁文，阿拉伯数字和一般常见符号构成的语言，ASCII 仍是目前主要用来交换信息的编码法。
byte 字节类型
储存的数值范围为 0 到 255。如同字符一样字节型态的变量只需要用一个字节(8 位)的内存空间储存。
int 整数

整数数据型态用到 2 字节的内存空间，可表示的整数范围为 -32,768 到 32,767；整数变量是 Arduino 内最常用到的数据型态。

unsigned int　　无符号整数(绝对值)

无号整数同样利用 2 字节的内存空间，无号意谓着它不能储存负的数值，因此无号整数可表示的整数范围为 0 到 65,535。

long　　长整数

长整数利用到的内存大小是整数的两倍，因此它可表示的整数范围从 -2,147,483,648 到 2,147,483,647。

unsigned long　　无符号长整数

无号长整数可表示的整数范围为 0 到 4,294,967,295。

float　　浮点数

浮点数就是用来表达有小数点的数值，每个浮点数会用掉四字节的 RAM，注意芯片内存空间的限制，谨慎的使用浮点数

double　　双字节浮点

也叫双精度浮点数，可表达最大值为 1.7976931348623157 x 10308。

string　　字符串

字符串用来表达文字信息，它是由多个 ASCII 字符组成(你可以透过序串端口发送一个文字讯息或者将之显示在液晶显示器上)。字符串中的每一个字符都用一个组元组空间储存，并且在字符串的最尾端加上一个空字符以提示 Ardunio 处理器字符串的结束。下面两种宣告方式是相同的。

例如:

```
char string1[] = "Arduino";//7 字符+1 空字符
char string2[8] = "Arduino"; // 与上行相同
```

array　　数组

一串变量可以透过索引去直接取得。假如你想要储存不同程度的 LED 亮度时，你可以宣告六个变量 light01, light02, light03, light04, light05, light06, 但其实你有更好的选择，例如宣告一个整数数组变量如下:

```
int  light[6] = {0 , 20 , 50 , 75 , 100}
```

"array" 这个字为没有直接用在变量宣告，而是[]和{}宣告数组。

控制指令

数据类型转换:

char()

byte()

int()

long()

float()

常量: 在 Arduino 语言中事先定义了一些具特殊用途的保留字。

HIGH | LOW

表示数字 IO 口的电平，HIGH 表示高电平（1），LOW 表示低电平（0）。 HIGH 和 LOW 也用来表示你开启或是关闭了一个 Arduino 的脚位(pin)

INPUT | OUTPUT

表示数字 IO 口的方向，INPUT 表示输入（高阻态），OUTPUT 表示输出（AVR 能提供 5V 电压 40mA 电流）。

true | false

true 表示真（1），false 表示假（0）。

变数：

变量用来指定 Arduino 内存中的一个位置，变量可以用来储存数据，程序人员可以透过脚本代码去不限次数的操作变数的值。

因为 Arduino 是一个非常简易的微处理器，但你要宣告一个变量时必须先定义他的数据型态，好让微处理器知道准备多大的空间以储存这个变量值。

以上为基础 c 语言的关键字和符号，有 c 语言基础的都应该了解其含义，这里也不作过多的解释。

Arduino 语言

结构

1、声明变量及接口名称（int val;int ledPin=13;）。

2、void setup()

在程序开始时使用，在这个函数范围内放置初始化 Arduino 板子的程式，主要程式开始撰写前， 使 Arduino 板子装置妥当的指令可以初始化变量、管脚接口模式、启用库等（例如：pinMode(ledPin,OUTPUT);）。

3、void loop()

在 setup()函数之后，即初始化之后，loop() 让你的程序循环地被执行。使用它来运转 Arduino。连续执行函数内的语句，这部份的程式会一直重复的被执行，直到 Arduino 板子被关闭。

功能

数字 I/O

pinMode(pin, mode)

数字 IO 口输入输出模式定义函数，将接口定义为输入或输出接口，用在 setup()函数里，pin 表示为 0～13 接口名称， mode 表示为 INPUT 或 OUTPUT。即“ pinMode(接口名称,OUTPUT 或 INPUT)”。

范例 :

```
pinMode(7,INPUT); // 将脚位 7 设定为输入模式
```

digitalWrite(pin, value)

数字 IO 口输出电平定义函数，将数字接口值至高或低、开或关，pin 表示为 0～13，value 表示为 HIGH 或 LOW，即 digitalWrite(接口名称，HIGH 或 LOW)。但脚

位必须先透过 pinMode 明示为输入或输出模式 digitalWrite 才能生效。比如定义 HIGH 可以驱动 LED。

范例 :

```
digitalWrite(8,HIGH); //将脚位 8 设定输出高电位
```

int digitalRead(pin)

数字 IO 口读输入电平函数，读出数字接口的值，pin 表示为 0～13，value 表示为 HIGH 或 LOW，即 digitalRead（接口名称）。比如可以读数字传感器。当感测到脚位处于高电位时时回传 HIGH，否则回传 LOW。

范例 :

```
val = digitalRead(7); // 读出脚位 7 的值并指定给 val
```

模拟 I/O

int analogRead(pin)

模拟 IO 口读函数，从指定的模拟接口读取值，Arduino 对该模拟值进行 10-bit 的数字转换，这个方法将输入的 0-5 电压值转换为 0 到 1023 间的整数值。pin 表示为 0～5（Arduino Diecimila 为 0～5，Arduino nano 为 0～7）。即“analogRead(接口名称)”，比如可以读模拟传感器（10 位 AD，0～5V 表示为 0～1023）。

范例 :

```
val = analogRead(0); //读出类比脚位 0 的值并指定给 val 变数
```

analogWrite(pin, value)

数字 IO 口 PWM 输出函数，给一个接口写入模拟值（PWM 波）。改变 PWM 脚位的输出电压值。对于 ATmega168 芯片的 Arduino（包括 Mini 或 BT），该函数可以工作于 3，5，6，9，10 和 11 号接口，即“analogWrite(接口名称，数值)”,pin 表示 3，5，6，9，10，11，value 表示为 0～255。比如可用于电机 PWM 调速或音乐播放。

例如：输出电压 2.5 伏特（V），该值大约是 128。

范例 :

```
analogWrite(9,128); // 输出电压约 2.5 伏特（V）
```

扩展 I/O

shiftOut(dataPin, clockPin, bitOrder, value)

SPI 外部 IO 扩展函数，通常使用带 SPI 接口的 74HC595 做 8 个 IO 扩展，把资料传给用来延伸数位输出的暂存器，此函式通常使用在延伸数位的输出。函式使用一个脚位表示资料、一个脚位表示时脉。dataPin 为数据口，clockPin 为时钟口，bitOrder 用来表示位元间移动的方式，为数据传输方向（MSBFIRST 高位在前，LSBFIRST 低位在前），value 会以 byte 形式输出，表示所要传送的数据（0～255），另外还需要一个 IO 口做 74HC595 的使能控制。

范例 :

```
shiftOut(dataPin, clockPin, LSBFIRST, 255);
```

unsigned long pulseIn(pin, value)

脉冲长度记录函数，设定读取脚位状态的持续时间，返回时间参数（us），例如使用红外线、加速度感测器测得某一项数值时，在时间单位内不会改变状态。pin 表示

为 0～13，value 为 HIGH 或 LOW。比如 value 为 HIGH，那么当 pin 输入为高电平时，开始计时，当 pin 输入为低电平时，停止计时，然后返回该时间。
范例 :
time = pulsein(7,HIGH); // 设定脚位 7 的状态在时间单位内保持为 HIGH
时间函数
unsigned long millis()
返回时间函数（单位 ms），回传晶片开始执行到目前的毫秒，该函数是指，当程序运行就开始计时并返回记录的参数，该参数溢出大概需要 50 天时间。
范例:
duration = millis()-lastTime; // 表示自"lastTime"至当下的时间
delay(ms) 延时函数（单位 ms），延时一段时间，暂停晶片执行多少毫秒，delay(1000)为一秒。
范例:
delay(500); //暂停半秒（500 毫秒）
delayMicroseconds(us) 延时函数（单位 us）暂停晶片执行多少微秒。
delayMicroseconds(1000); //暂停 1 豪秒
数学函数
min(x, y)
求最小值 ，回传两数之间较小者
范例:
val = min(10,20); // 回传 10
max(x, y)
求最大值 ，回传两数之间较大者
范例:
val = max(10,20); // 回传 20
abs(x)
计算绝对值 ，回传该数的绝对值，可以将负数转正数。
范例:
val = abs(-5); // 回传 5
constrain(x, a, b) 约束函数，下限 a，上限 b，判断 x 变数位于 a 与 b 之间的状态。x 若小于 a 回传 a；介于 a 与 b 之间回传 x 本身；大于 b 回传 b
范例:
val = constrain(analogRead(0), 0, 255); // 忽略大于 255 的数
map(value, fromLow, fromHigh, toLow, toHigh)
约束函数，value 必须在 fromLow 与 toLow 之间和 fromHigh 与 toHigh 之间。将 value 变数依照 fromLow 与 fromHigh 范围，对等转换至 toLow 与 toHigh 范围。时常使用于读取类比讯号，转换至程式所需要的范围值。
例如:
val = map(analogRead(0),0,1023,100, 200); // 将 analog0 所读取到的讯号对等转换至 100，200 之间的数值。
pow(base, exponent)

开方函数，base 的 exponent 次方。回传一个数(base)的指数(exponent)值。
范例：
```
double x = pow(y, 32); // 设定 x 为 y 的 32 次方
```
sq(x)　平方
sqrt(x)　开根号
回传 double 型态的取平方根值。
范例：
```
double a = sqrt(1138); // 回传 1138 平方根的近似值 33.73425674438
```
三角函数
sin (rad)
回传角度（radians）的三角函数 sine 值。
范例：
```
double sine = sin(2); // 近似值 0.90929737091
```
cos(rad)
回传角度（radians）的三角函数 cosine 值。
范例：
```
double cosine = cos(2); //近似值-0.41614685058
```
tan(rad)
回传角度（radians）的三角函数 tangent 值。
范例：
```
double tangent = tan(2); //近似值-2.18503975868
```
随机数函数
randomSeed(seed)
随机数端口定义函数，seed 表示读模拟口 analogRead(pin)函数 。
事实上在 Arduino 里的乱数是可以被预知的。所以如果需要一个真正的乱数，可以呼叫此函式重新设定产生乱数种子。你可以使用乱数当作乱数的种子，以确保数字以随机的方式出现，通常会使用类比输入当作乱数种子，藉此可以产生与环境有关的乱数（例如：无线电波、宇宙雷射线、电话和萤光灯发出的电磁波等）。
范例：
```
randomSeed(analogRead(5)); // 使用类比输入当作乱数种子
```
long random(max)
随机数函数，返回数据大于等于 0，小于 max。
范例：
```
long randnum = random(11);          // 回传 0 -10 之间的数字
```
long random(min, max)
随机数函数，返回数据大于等于 min，小于 max。
范例：
```
long randnum = random(0, 100); // 回传 0 - 99 之间的数字
```
外部中断函数
attachInterrupt(interrupt, , mode)

外部中断只能用到数字 IO 口 2 和 3，interrupt 表示中断口初始 0 或 1，表示一个功能函数，mode：LOW 低电平中断，CHANGE 有变化就中断，RISING 上升沿中断，FALLING 下降沿中断。

detachInterrupt(interrupt)

中断开关，interrupt=1 开，interrupt=0 关。

中断使能函数

interrupts() 使能中断

noInterrupts() 禁止中断

串口收发函数

Serial.begin(speed)

串口定义波特率函数，设置串行每秒传输数据的速率（波特率），可以指定 Arduino 从电脑交换讯息的速率，通常我们使用 9600 bps。，speed 表示波特率，如 9600，19200 等。在同计算机通讯时，使用下面这些值：300，1200，2400，4800，9600，14400，19200，28800，38400，57600 或 115200 bps（每秒位元组）。。你也可以在任何时候使用其它的值，比如，与 0 号或 1 号插口通信就要求特殊的波特率。用在 setup()函数里

范例：

```
Serial.begin(9600)
```

int Serial.available()

判断缓冲器状态。回传有多少位元组（bytes）的资料尚未被 read()函式读取，如果回传值是 0 代表所有序列埠上资料都已经被 read()函式读取。

范例：

```
int count = Serial.available();
```

int Serial.read()

读串口并返回收到参数。Serial.read()——读取持续输入的数据。读取 1byte 的序列资料

范例：

```
int data = Serial.read();
```

Serial.flush()

清空缓冲器。 有时候因为资料速度太快，超过程式处理资料的速度，你可以使用此函式清除缓冲区内的资料。经过此函式可以确保缓冲区(buffer)内的资料都是最新的。

范例：

```
Serial.flush();
```

Serial.print(data)

从串行端口输出数据。Serial.print(数据)默认为十进制等于 Serial.print(数据，DEC)。

Serial.print(data, encoding)

经序列埠传送资料，提供编码方式的选项。Serial.print(数据，数据的进制)如果没有指定，预设以一般文字传送。

范例：

```
Serial.print(75);              // 列印出 "75"
Serial.print(75, DEC); //列印出 "75"
Serial.print(75, HEX); // "4B" (75 的十六进位)
Serial.print(75, OCT); // "113" (75 in 的八进位)
Serial.print(75, BIN); // "1001011" (75 的二进位)
Serial.print(75, BYTE); // "K" (以 byte 进行传送，显示以 ASCII 编码方式)
```

Serial.println(data)

从串行端口输出数据，跟随一个回车和一个换行符。这个函数所取得的值与 Serial.print()一样。

Serial.println(data, encoding)

与 Serial.print()相同，但会在资料尾端加上换行字元（ ）。意思如同你在键盘上打了一些资料后按下 Enter。

范例:

```
Serial.println(75);              //列印出"75 "
Serial.println(75, DEC); //列印出"75 "
Serial.println(75, HEX); // "4B "
Serial.println(75, OCT); // "113 "
Serial.println(75, BIN); // "1001011 "
Serial.println(75, BYTE); // "K "
```

以上几个函数是常用基本函数，还有很多以后会慢慢学习

Arduino 语言库文件

官方库文件

- EEPROM - EEPROM 读写程序库
- Ethernet - 以太网控制器程序库
- LiquidCrystal - LCD 控制程序库
- Servo - 舵机控制程序库
- SoftwareSerial - 任何数字 IO 口模拟串口程序库
- Stepper - 步进电机控制程序库
- Wire - TWI/I2C 总线程序库
- Matrix - LED 矩阵控制程序库
- Sprite - LED 矩阵图象处理控制程序库

非官方库文件

- DateTime - a library for keeping track of the current date and time in software.
- Debounce - for reading noisy digital inputs (e.g. from buttons)

- Firmata - for communicating with applications on the computer using a standard serial protocol.
- GLCD - graphics routines for LCD based on the KS0108 or equivalent chipset.
- LCD - control LCDs (using 8 data lines)
- LCD 4 Bit - control LCDs (using 4 data lines)
- LedControl - for controlling LED matrices or seven-segment displays with a MAX7221 or MAX7219.
- LedControl - an alternative to the Matrix library for driving multiple LEDs with Maxim chips.
- Messenger - for processing text-based messages from the computer
- Metro - help you time actions at regular intervals
- MsTimer2 - uses the timer 2 interrupt to trigger an action every N milliseconds.
- OneWire - control devices (from Dallas Semiconductor) that use the One Wire protocol.
- PS2Keyboard - read characters from a PS2 keyboard.
- Servo - provides software support for Servo motors on any pins.
- Servotimer1 - provides hardware support for Servo motors on pins 9 and 10
- Simple Message System - send messages between Arduino and the computer
- SSerial2Mobile - send text messages or emails using a cell phone (via AT commands over software serial)
- TextString - handle strings
- TLC5940 - 16 channel 12 bit PWM controller.
- X10 - Sending X10 signals over AC power lines

/**********/

Arduuino 复合运算符

+= , -= , *= , /=
对一个变量和另一个参数或变量完成一个数学运算。+=（以及其他）可以缩短语法长度。
Syntax 语法

```
x += y;                          // 等价于 x = x + y;
x -= y;                          // 等价于 x = x - y;
x *= y;                          // 等价于 x = x * y;
x /= y;                          // 等价于 x = x / y;
```

Parameters 参数
x：任何变量类型
y：任何变量类型或常数
Examples 范例

```
x = 2;
x += 4;            // x 现在为 6
x -= 3;            // x 现在为 3
x *= 10;           // x 现在为 30
x /= 2;            // x 现在为 15
```

Syntax 语法

```
x++; // increment x by one and returns the old value of x
          // 将 x 的值加 1 并返回原来的 x 的值。        ++x; // increment x
by one and returns the new value of x          // 将 x 的值加 1 并返回现在
的 x 的值。
x-- ;     // decrement x by one and returns the old value of
x             // 将 x 的值减 1 并返回原来的 x 的值。
--x ;     // decrement x by one and returns the new value of
x               // 将 x 的值减 1 并返回现在的 x 的值。
```

Parameters 参数
x: an integer or long (possibly unsigned)
x：一个整数或长整数（可以无符号）

Returns 返回
The original or newly incremented / decremented value of the variable.
返回变量原始值或增加/消耗后的新值。
Examples 范例

```
x = 2;
y = ++x;            // x now contains 3, y contains
3                           // x 现在为 3，y 为 3
y = x--;            // x contains 2 again, y still contains
3                           // x 现在仍然为 2，y 将为 3
```

Arduuino 常见语法

常量：

- HIGH | LOW　　表示数字 IO 口的电平，HIGH 表示高电平（1），LOW 表示低电平（0）。
- INPUT | OUTPUT　表示数字 IO 口的方向，INPUT 表示输入（高阻态），OUTPUT 表示输出（AVR 能提供5V 电压 40mA 电流）。
- true | false　　true 表示真（1），false 表示假（0）。

结构

- void setup()　初始化变量，管脚模式，调用库函数等
- void loop()　连续执行函数内的语句

数字 I/O

- pinMode(pin, mode)　　数字 IO 口输入输出模式定义函数,pin 表示为0～13，mode 表示为 INPUT 或 OUTPUT。
- digitalWrite(pin, value)　　数字 IO 口输出电平定义函数，pin 表示为0～13，value 表示为 HIGH 或 LOW。比如定义 HIGH 可以驱动 LED。
- int digitalRead(pin)　　数字 IO 口读输入电平函数，pin 表示为0～13，value 表示为 HIGH 或 LOW。比如可以读数字传感器。

模拟 I/O

- int analogRead(pin)　　模拟 IO 口读函数，pin 表示为0～5（Arduino Diecimila 为0～5，Arduino nano 为0～7)。比如可以读模拟传感器（10位 AD，0～5V 表示为0～1023)。
- analogWrite(pin, value) - *PWM*　　数字 IO 口 PWM 输出函数，Arduino 数字 IO 口标注了 PWM 的 IO 口可使用该函数，pin 表示3, 5, 6, 9, 10, 11，value 表示为0～255。比如可用于电机 PWM 调速或音乐播放。

扩展 I/O

- shiftOut(dataPin, clockPin, bitOrder, value)　　SPI 外部 IO 扩展函数，通常使用带 SPI 接口的74HC595做8个 IO 扩展，dataPin 为数据口，clockPin 为时钟口，bitOrder 为数据传输方向(**MSBFIRST** 高位在前，**LSBFIRST** 低位在前)，value 表示所要传送的数据（0～255)，另外还需要一个 IO 口做74HC595的使能控制。
- unsigned long pulseIn(pin, value)　　脉冲长度记录函数，返回时间参数（us)，pin 表示为0～13，value 为 HIGH 或 LOW。比如 value 为 HIGH，那么当 pin 输入为高电平时，开始计时，当 pin 输入为低电平时，停止计时，然后返回该时间。

时间函数

- unsigned long millis()　　返回时间函数（单位 ms)，该函数是指，当程序运行就开始计时并返回记录的参数，该参数溢出大概需要50天时间。
- delay(ms)　　延时函数（单位 ms)。
- delayMicroseconds(us)　　延时函数（单位 us)。

数学函数

- min(x, y)　求最小值
- max(x, y)　求最大值
- abs(x)　计算绝对值
- constrain(x, a, b)　约束函数，下限 a，上限 b，x 必须在 ab 之间才能返回。
- map(value, fromLow, fromHigh, toLow, toHigh)　　约束函数，value 必须在 fromLow 与 toLow 之间和 fromHigh 与 toHigh 之间。
- pow(base, exponent)　开方函数，base 的 exponent 次方。
- sq(x)　　平方
- sqrt(x)　开根号

三角函数

- sin(rad)
- cos(rad)
- tan(rad)

随机数函数

- randomSeed(seed)　　随机数端口定义函数，seed 表示读模拟口 analogRead(pin)函数 。
- long random(max)　随机数函数，返回数据大于等于0，小于 max。
- long random(min, max)　随机数函数，返回数据大于等于 min，小于 max。

外部中断函数

- attachInterrupt(interrupt, , mode)　　外部中断只能用到数字 IO 口2和3，interrupt 表示中断口初始0或1，表示一个功能函数，mode：**LOW** 低电平中断，**CHANGE** 有变化就中断，**RISING** 上升沿中断，**FALLING** 下降沿中断。
- detachInterrupt(interrupt)　　中断开关，interrupt=1 开，interrupt=0 关。

中断使能函数

- interrupts()　使能中断
- noInterrupts()　禁止中断

串口收发函数

- Serial.begin(speed)　串口定义波特率函数，speed 表示波特率，如9600，19200等。
- int Serial.available()　判断缓冲器状态。
- int Serial.read()　读串口并返回收到参数。
- Serial.flush()　清空缓冲器。
- Serial.print(data)　串口输出数据。
- Serial.println(data)　串口输出数据并带回车符。

/**********Arduino 语言库文件**********/

官方库文件

- EEPROM - EEPROM 读写程序库
- Ethernet - 以太网控制器程序库
- LiquidCrystal - LCD 控制程序库
- Servo - 舵机控制程序库
- SoftwareSerial - 任何数字 IO 口模拟串口程序库
- Stepper - 步进电机控制程序库

- Wire - TWI/I2C 总线程序库
- Matrix - LED 矩阵控制程序库
- Sprite - LED 矩阵图象处理控制程序库

非官方库文件

- DateTime - a library for keeping track of the current date and time in software.
- Debounce - for reading noisy digital inputs (e.g. from buttons)
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- GLCD - graphics routines for LCD based on the KS0108 or equivalent chipset.
- LCD - control LCDs (using 8 data lines)
- LCD 4 Bit - control LCDs (using 4 data lines)
- LedControl - for controlling LED matrices or seven-segment displays with a MAX7221 or MAX7219.
- LedControl - an alternative to the Matrix library for driving multiple LEDs with Maxim chips.
- Messenger - for processing text-based messages from the computer
- Metro - help you time actions at regular intervals
- MsTimer2 - uses the timer 2 interrupt to trigger an action every N milliseconds.
- OneWire - control devices (from Dallas Semiconductor) that use the One Wire protocol.
- PS2Keyboard - read characters from a PS2 keyboard.
- Servo - provides software support for Servo motors on any pins.
- Servotimer1 - provides hardware support for Servo motors on pins 9 and 10
- Simple Message System - send messages between Arduino and the computer
- SSerial2Mobile - send text messages or emails using a cell phone (via AT commands over software serial)
- TextString - handle strings
- TLC5940 - 16 channel 12 bit PWM controller.
- X10 - Sending X10 signals over AC power lines

/**********/