

<sup>2</sup><sub>0</sub><sup>1</sup><sub>1</sub>WK COST整理

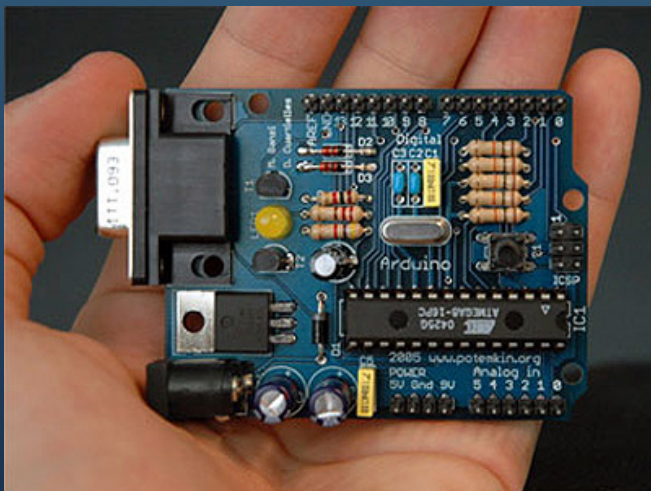
<http://hi.baidu.com/wkcost>

# 动手玩转Arduino (一)

Arduino北京俱乐部

# Arduino是什么？

- Arduino硬件介绍
- Arduino编程环境介绍

A screenshot of the Arduino IDE (version 0004 Alpha) window. The window title is 'Arduino - 0004 Alpha'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for running, saving, opening, and other sketch-related functions. The main text area displays the 'led\_blink' sketch, which is a simple program to blink an LED. The code is as follows:

```
/* Blinking LED
 * -----
 *
 * turns on and off a light emitting diode(LED) connected to a digital
 * pin, in intervals of 2 seconds. Ideally we use pin 13 on the Arduino
 * board because it has a resistor attached to it, needing only an LED
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
 *
 * based on an original by H. Barragan for the Wiring i/o board
 */

int ledPin = 13;           // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);                // waits for a second
}
```

# Arduino的安装

- Arduino IDE的安装
- USB驱动的安装

# Arduino程序结构: setup()



```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

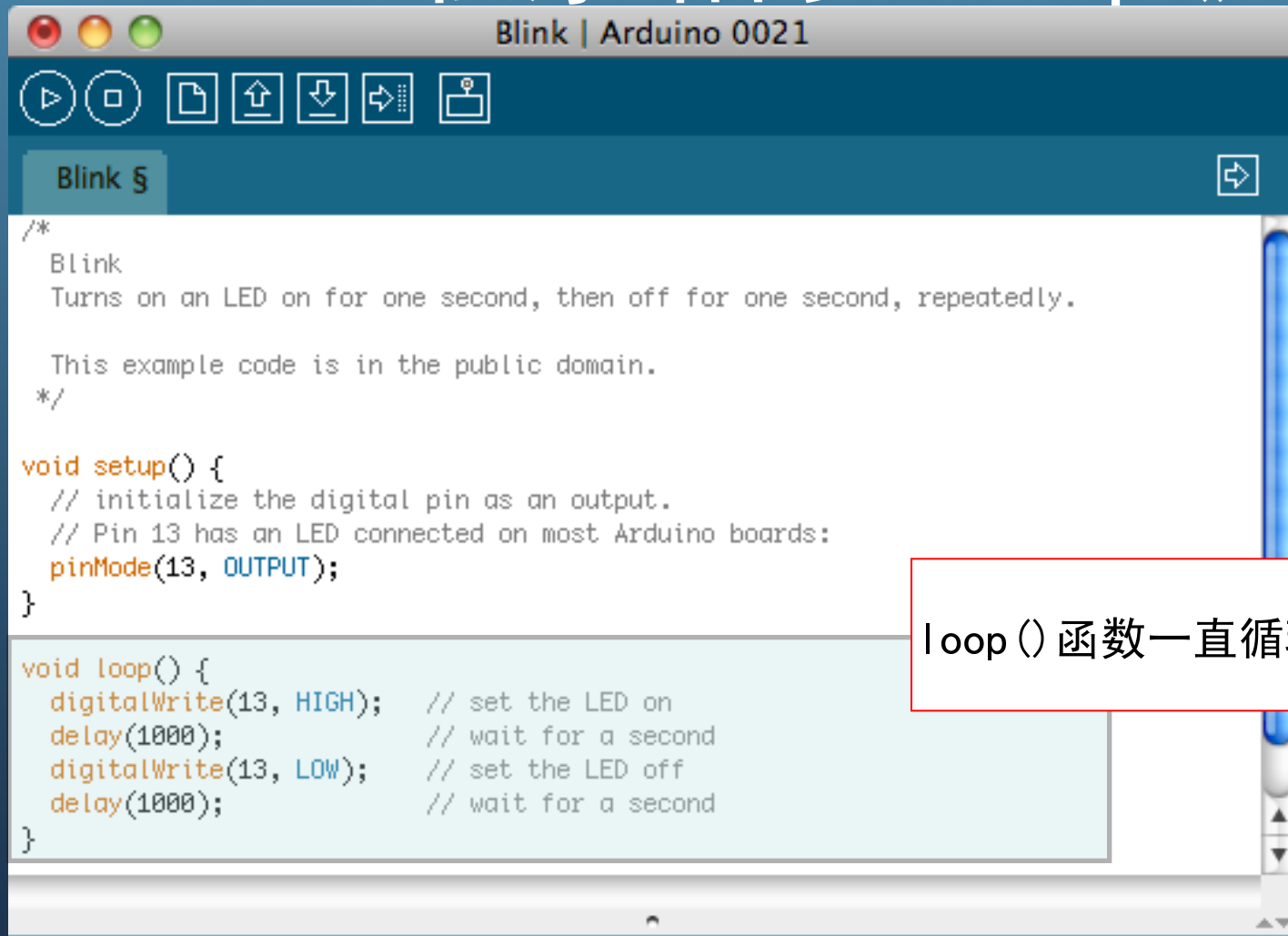
This example code is in the public domain.
*/

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

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

上电后setup()函数执行一次

# Arduino程序结构: loop()

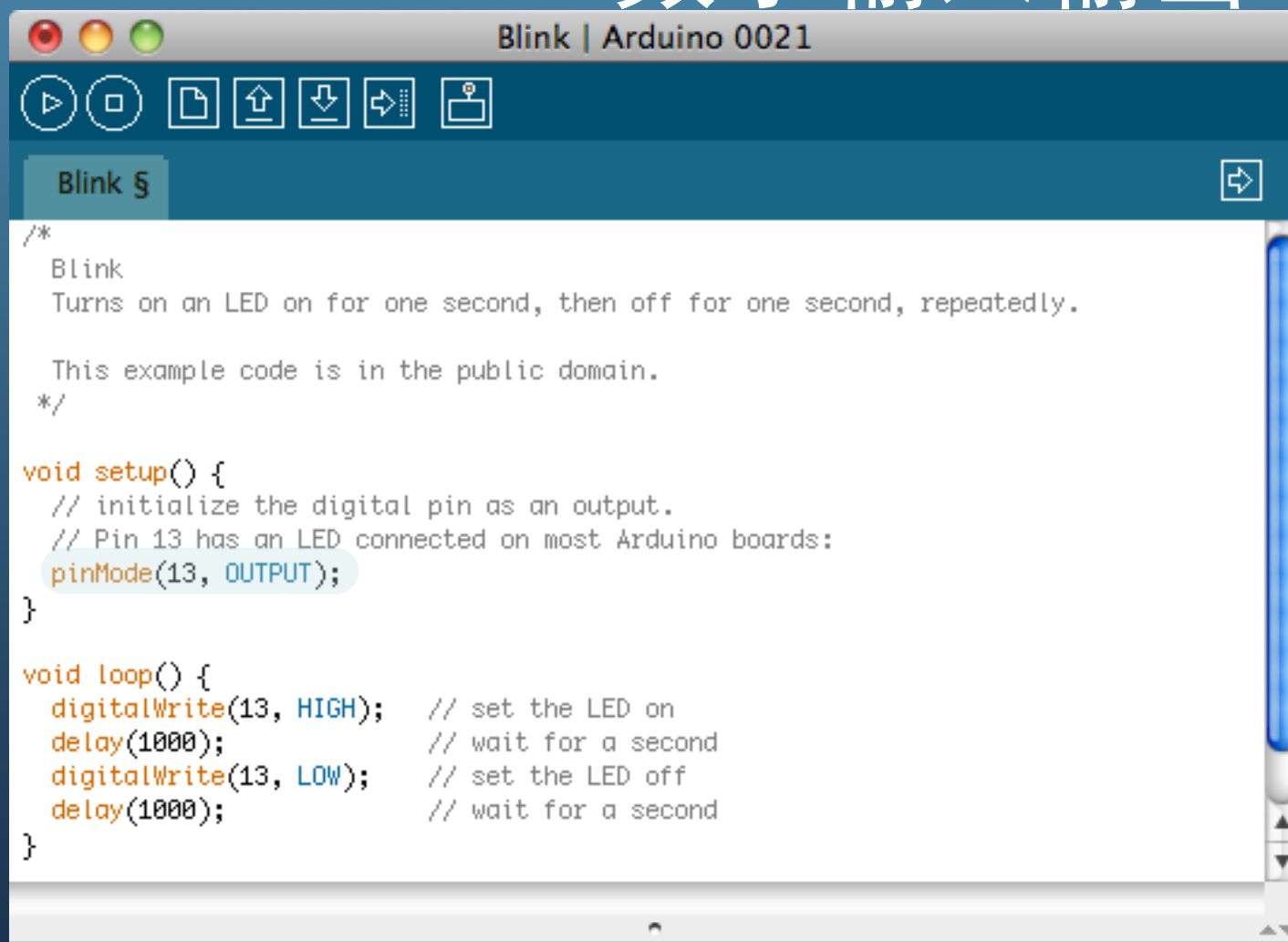


The screenshot shows the Arduino IDE interface with the 'Blink' program loaded. The title bar reads 'Blink | Arduino 0021'. The toolbar includes icons for running, stopping, saving, opening, and uploading. The code editor displays the following code:

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
  */  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH);  // set the LED on  
  delay(1000);             // wait for a second  
  digitalWrite(13, LOW);   // set the LED off  
  delay(1000);             // wait for a second  
}
```

loop() 函数一直循环运行

# Arduino数字输入输出



The image shows a screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 0021". The toolbar at the top includes icons for running, stopping, saving, opening, and other file operations. The file name "Blink §" is displayed in the top left of the editor. The code is as follows:

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
  */  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH);  // set the LED on  
  delay(1000);             // wait for a second  
  digitalWrite(13, LOW);   // set the LED off  
  delay(1000);             // wait for a second  
}
```

# Arduino程序下载

- HelloWorld程序

# 动手

- HelloWorld: 点亮一个LED灯

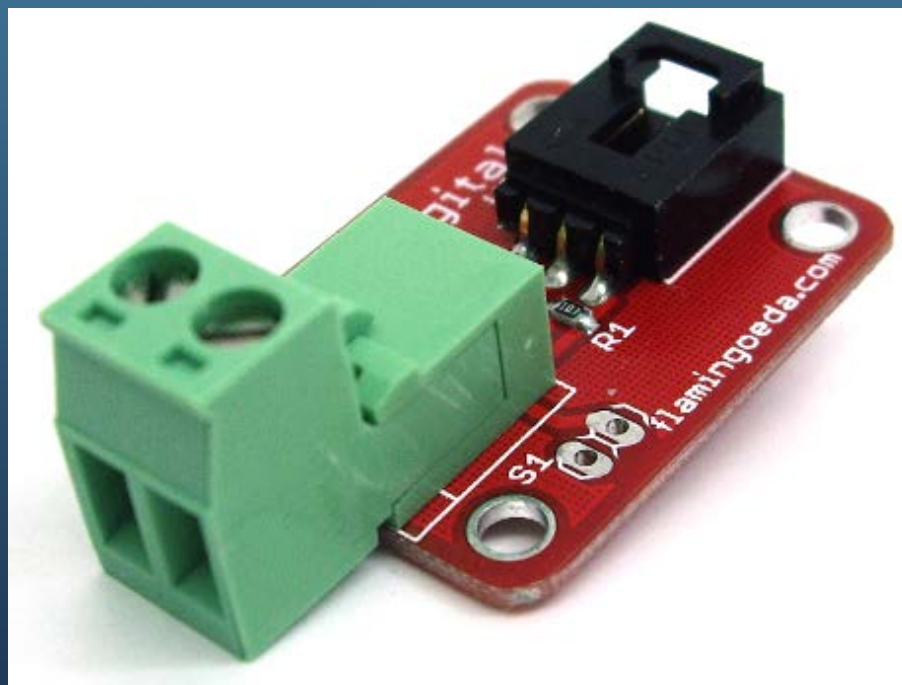


# Arduino与电子积木

- 什么是电子积木
- 电子积木如何与Arduino连接
  - 传感器扩展板
  - 连接线

# 数字型电子积木

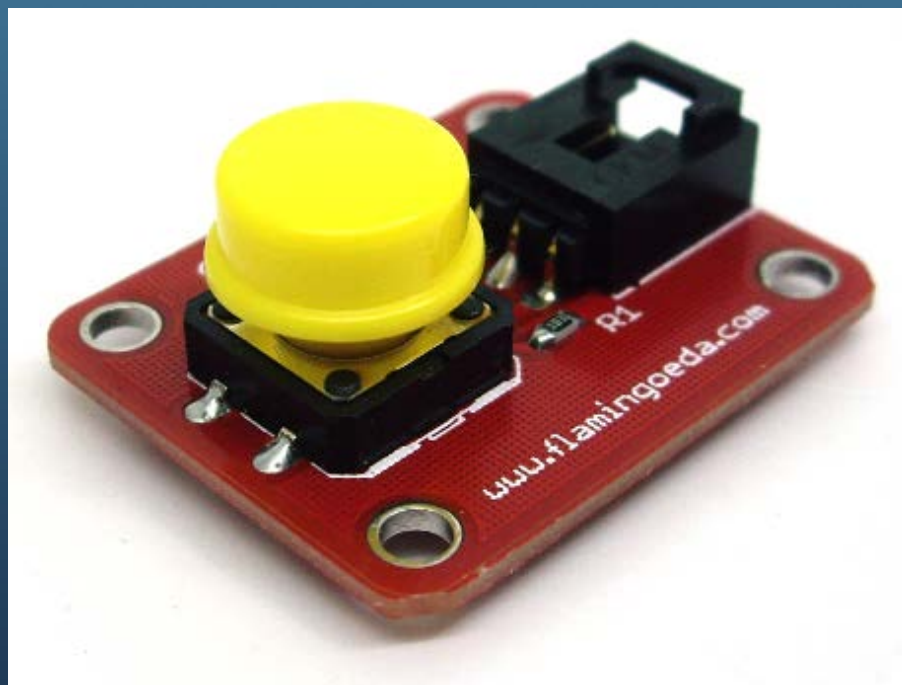
# 通用按钮模块



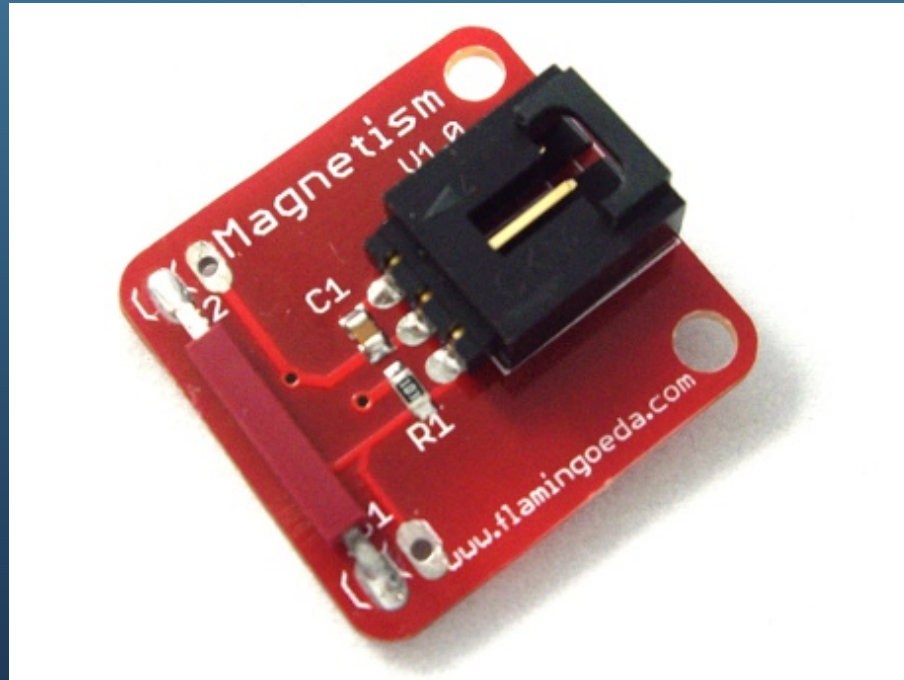
# 小按钮模块



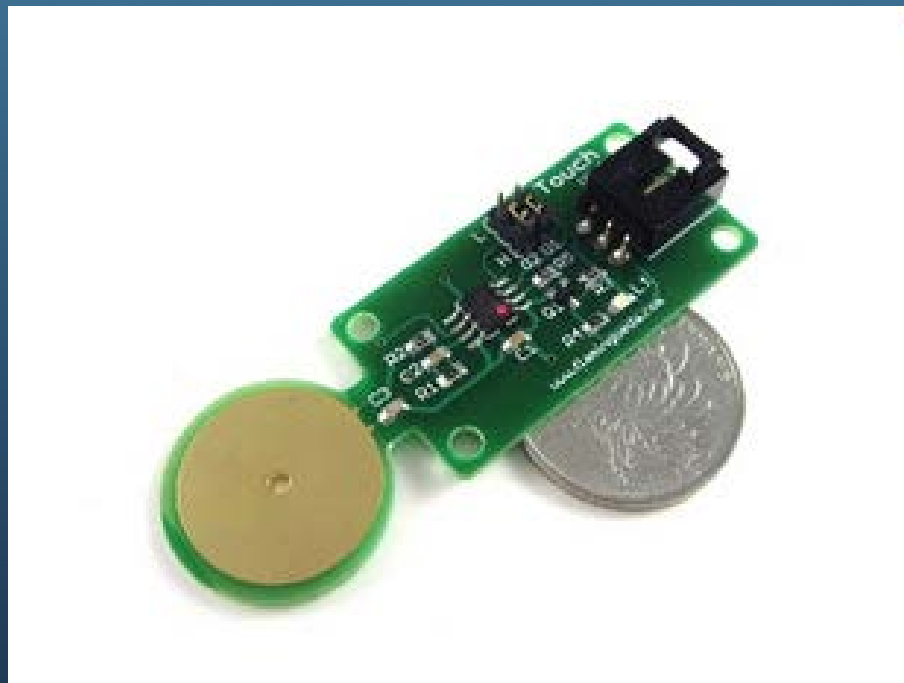
# 大按钮模块



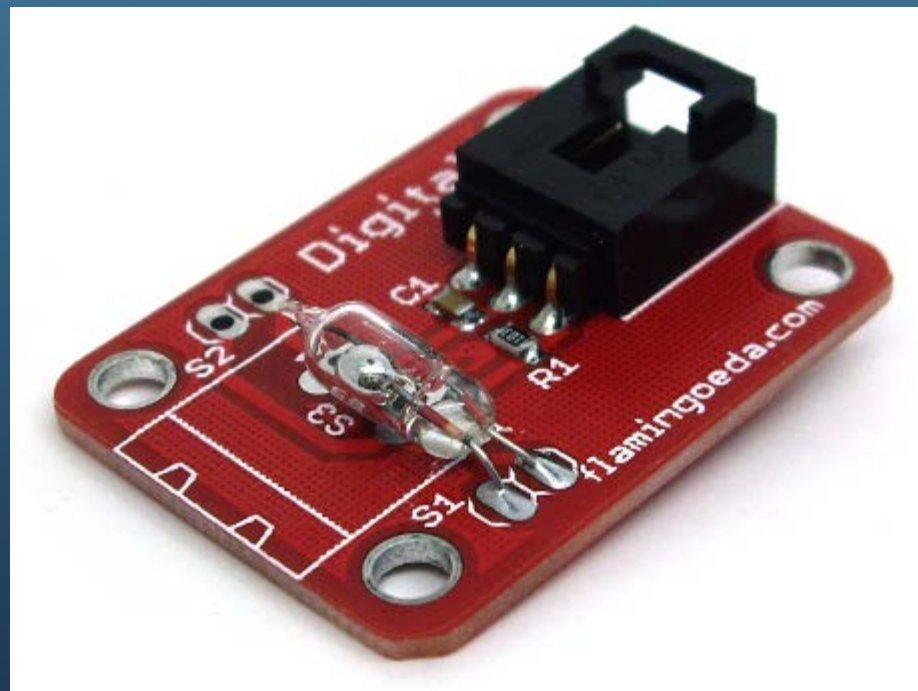
# 磁力开关



# 触摸传感器



# 倾斜传感器





# 人体运动 红外热释传感器



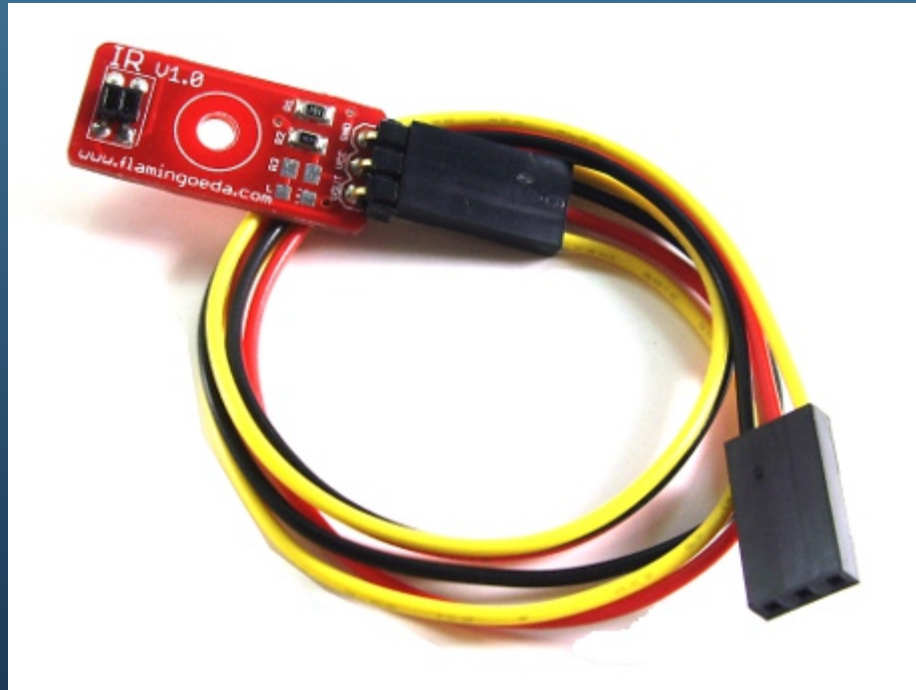
# 反射型红外开关 50cm~100cm



# 反射型红外开关 10cm



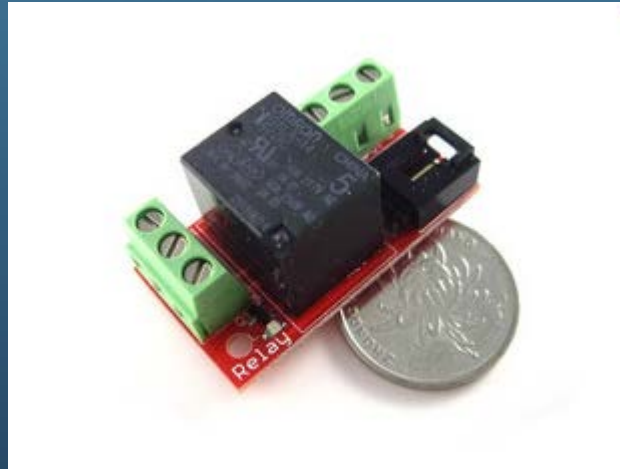
# 反射型红外开关 2cm



# 对射型 红外开关 8米



# 继电器模块



# 动手

- 通过按钮模拟来控制LED灯的亮灭

# 面包板使用方法介绍



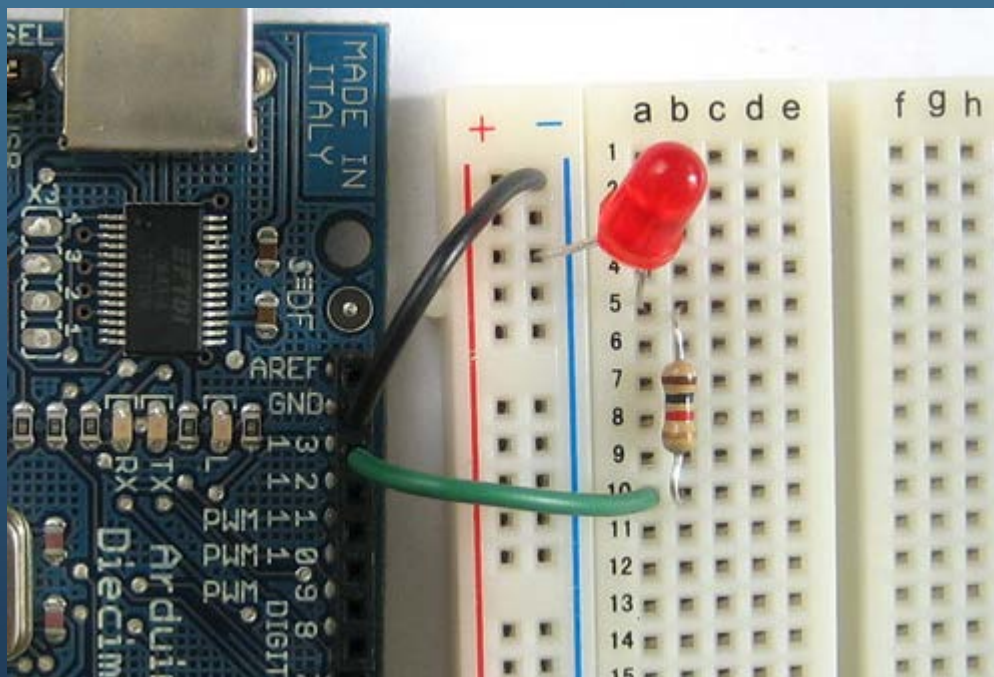
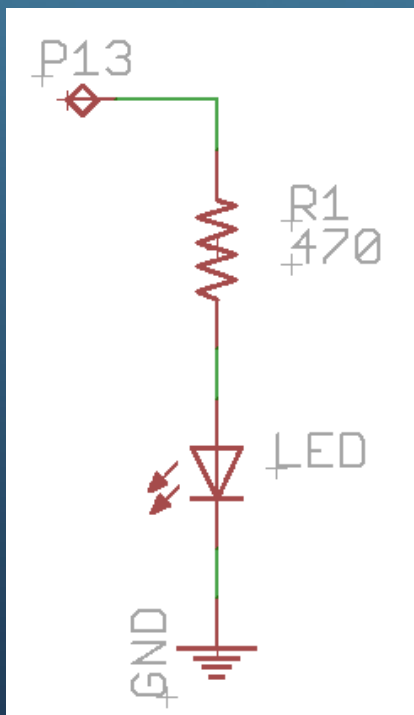
# 动手

- 用面包板完成LED模块的电路连接

# 动手玩转Arduino (二)

Arduino北京俱乐部

# LED基本电路



电阻



LED



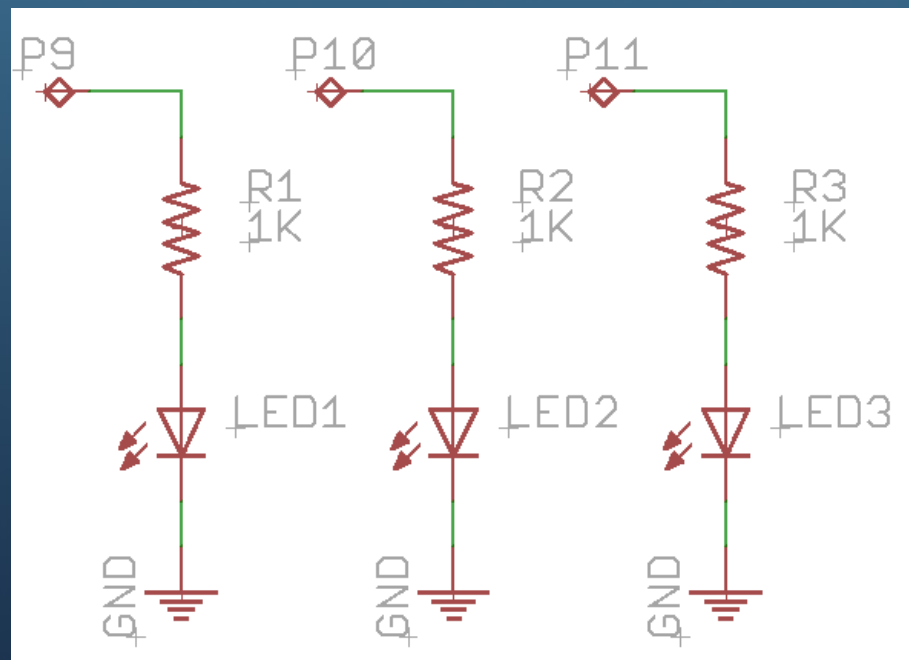
电源（正/负）

# LED控制代码

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

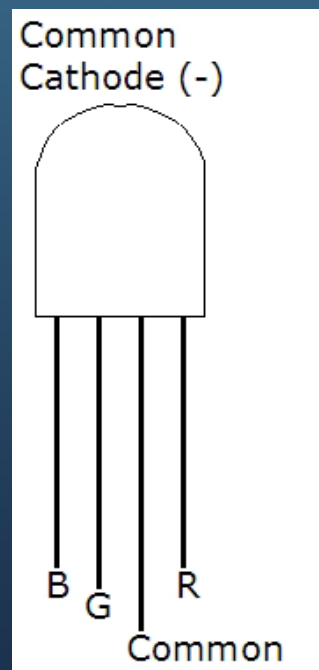
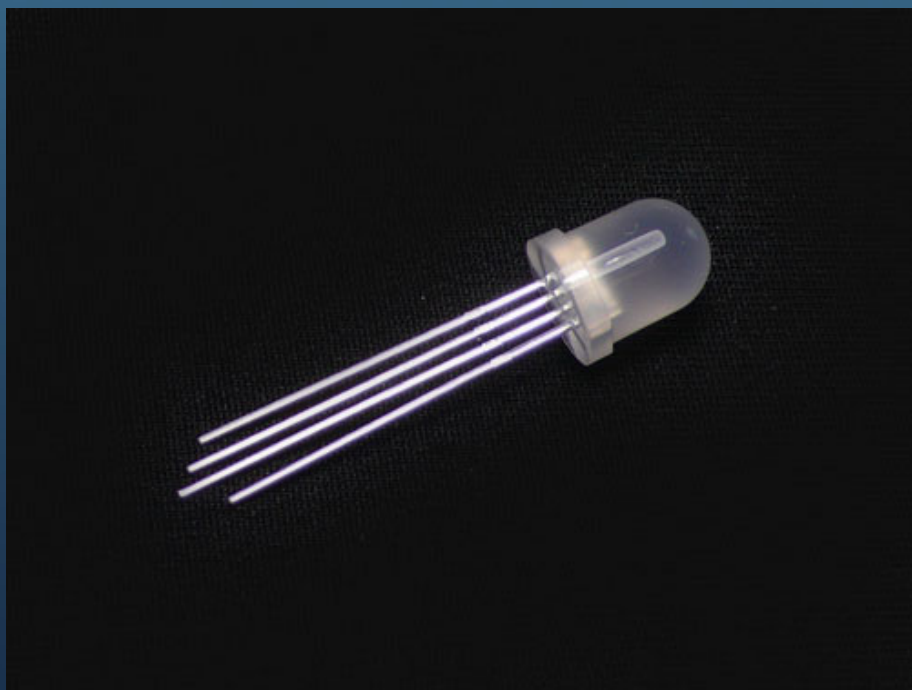
# 动手

- 使用Arduino和传感器扩展板
- 使用Arduino的9,10,11三个引脚
- 使用三个LED和1K的电阻
- 通过程序进行控制
- 讨论



# 彩色LED

- 红/绿/蓝 三原色
- 根据公共端的不同有共阳/共阴两种



# 动手

- 更换RGB LED
- 用Arduino程序混色
- 讨论

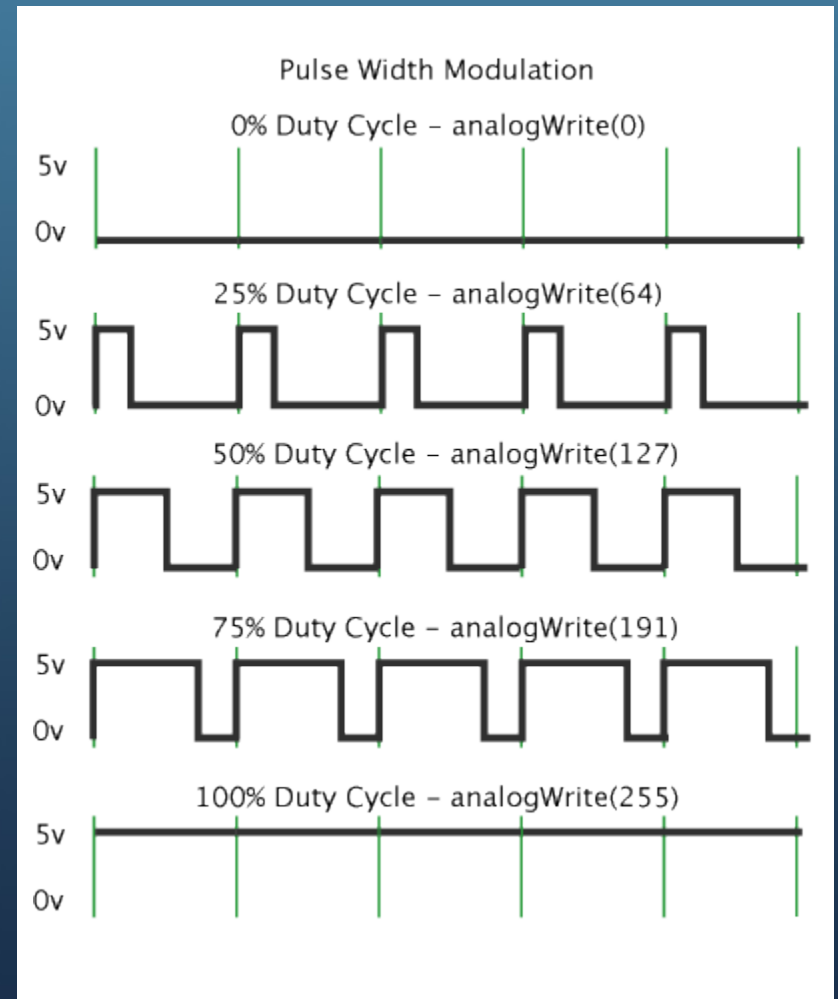
# 模拟信号

- 模拟信号的值可以连续变化
- 9V电池就是一种模拟器件，因为它的输出电压并不精确地等于9V，而是随时间发生变化，并可取任何实数值。
- 光照强度、温度、湿度等也都是模拟信号



# Arduino中的模拟输出

- `analogWrite(pin, value)`
- pin为Arduino上的PWM引脚
  - 3, 5, 6, 9, 10, 11
- value取值0-255
- PWM
  - 实现模拟信号的数字化方法



# 动手

- Arduino控制LED淡入淡出效果

```
int ledPin = 9;

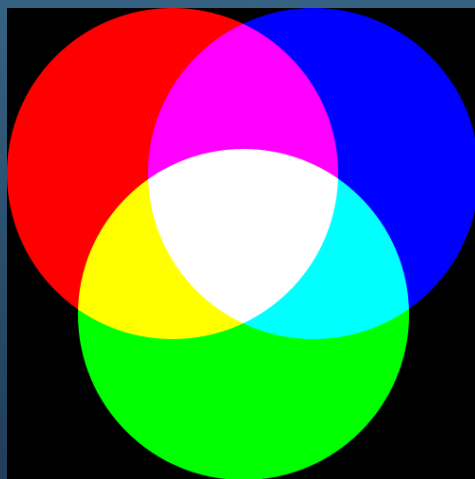
void setup() {
}

void loop() {
    for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
        analogWrite(ledPin, fadeValue);
        delay(30);
    }

    for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
        analogWrite(ledPin, fadeValue);
        delay(30);
    }
}
```

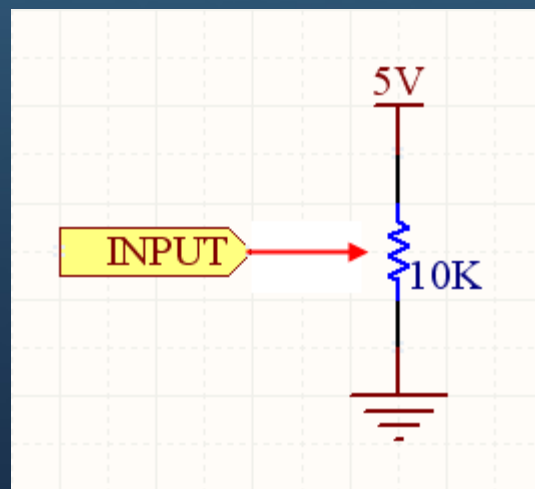
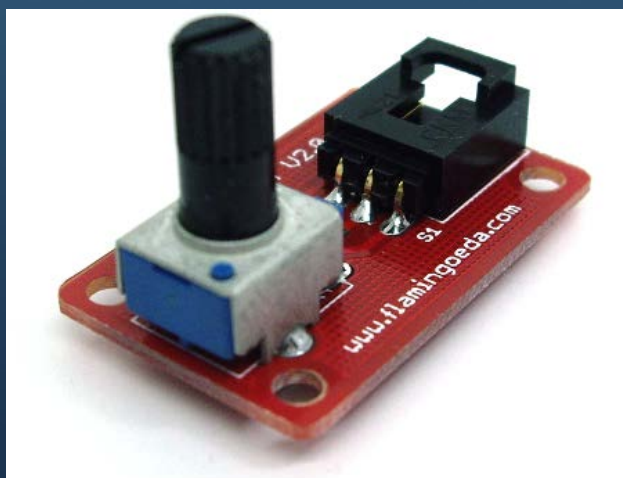
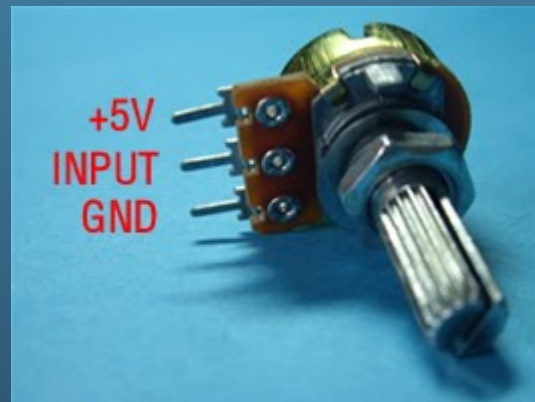
# 动手

- 实现全彩的混色



# 模拟输入

- 通常基于分压原理
- 电位器
- 电子积木
  - 300度 旋转角度传感器



# Arduino中的模拟输入

- `analogRead(pin)`
  - `pin`: 模拟输入引脚 A0, A1, A2, A3, A4, A5
  - 返回值: 0-1024 (10位精度)

# 动手

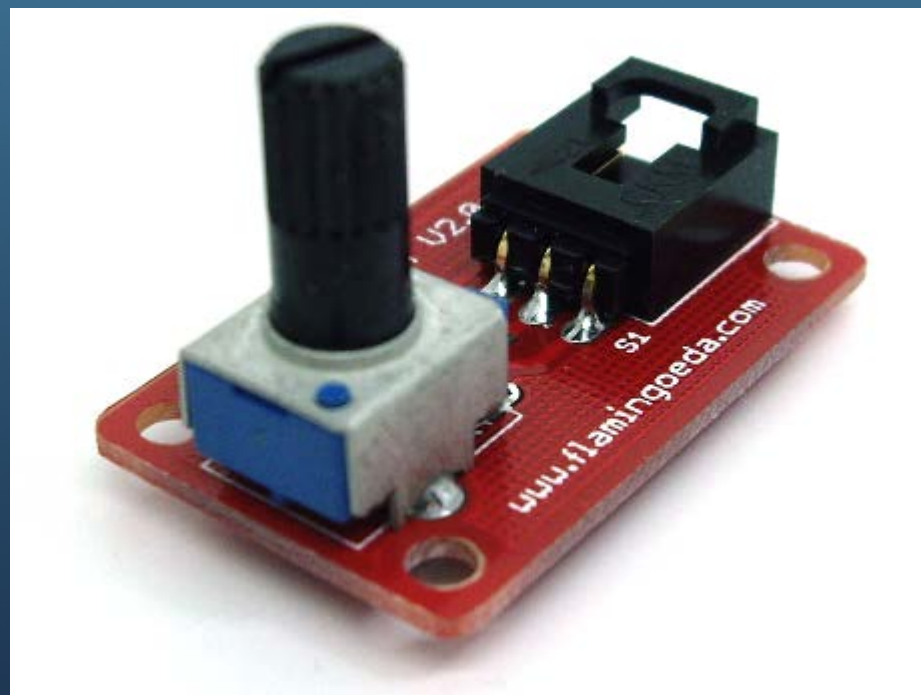
- 通过读取电位器的值来控制LED灯的亮度

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 9;        // select the pin for the LED
int sensorValue = 0;   // variable to store the value coming from the sensor

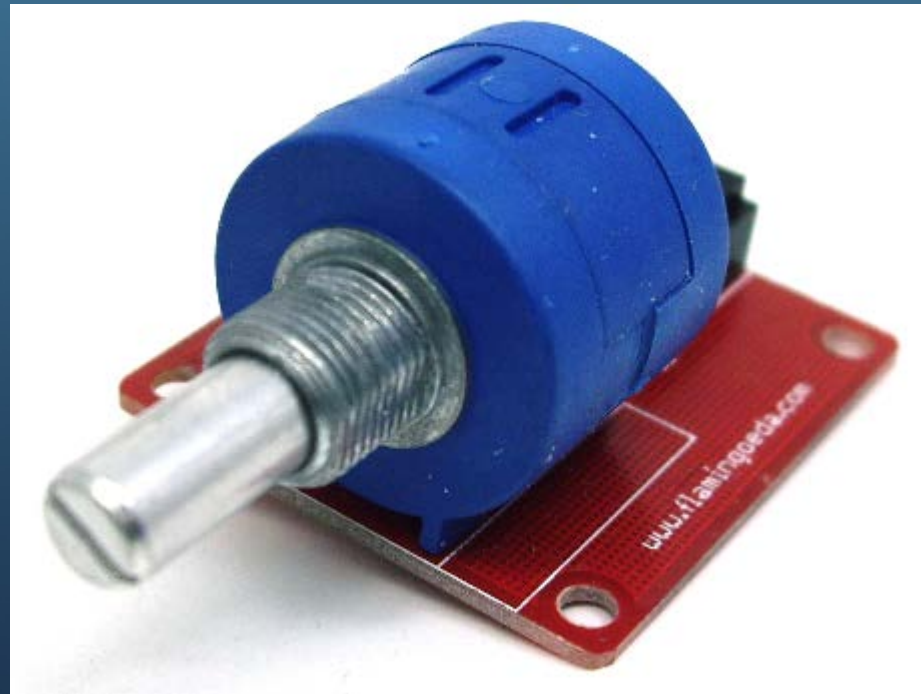
void setup() {
}

void loop() {
    // read the value from the sensor:
    sensorValue = analogRead(sensorPin);
    analogWrite(ledPin, sensorValue / 4);
    delay(30);
}
```

# 300度 旋转角度传感器



# 多圈 旋转角度传感器

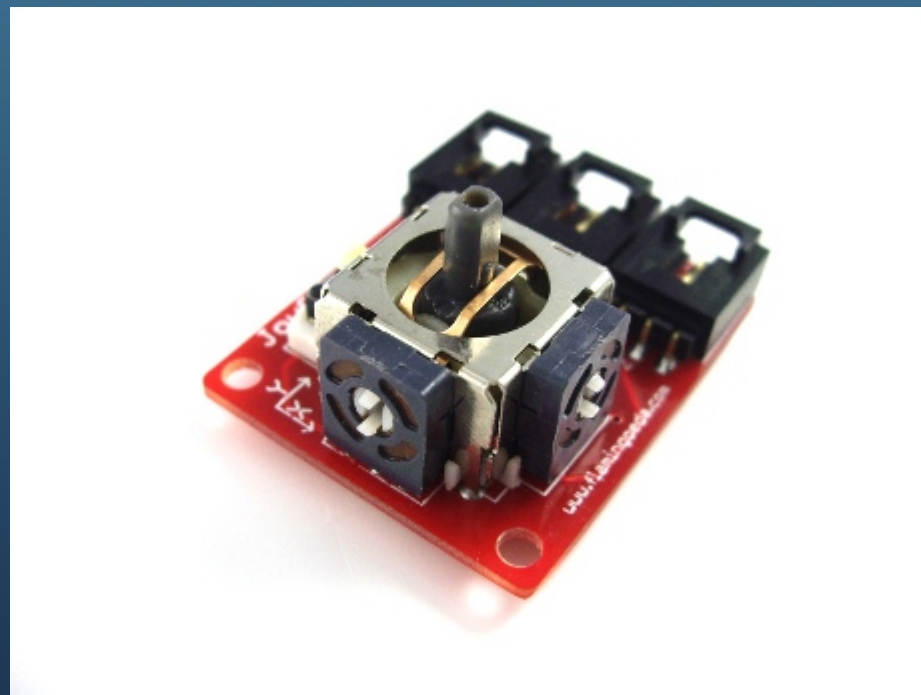




# 60行程 滑动电位器 推子



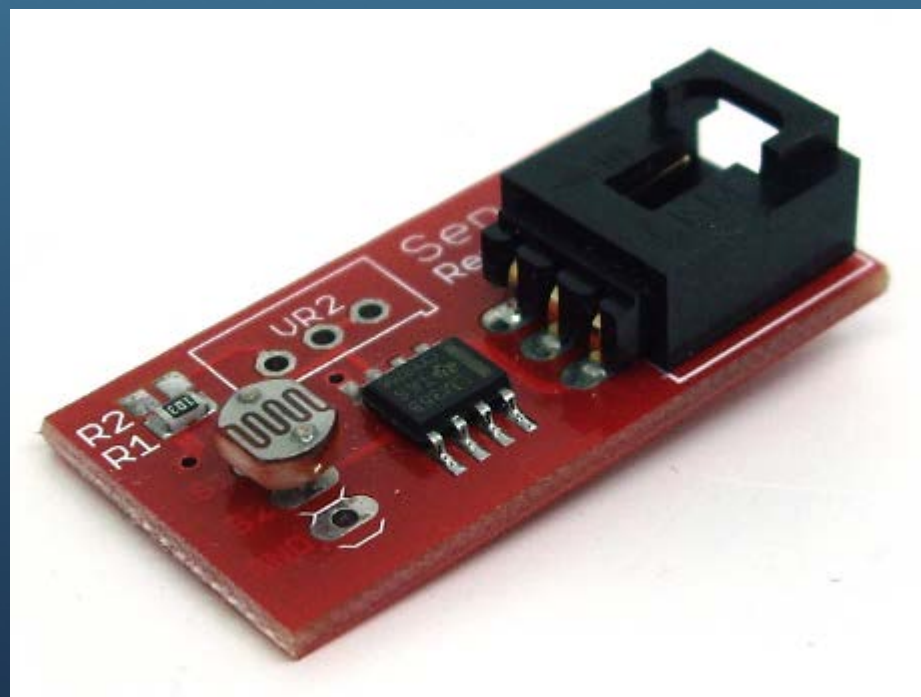
# 游戏杆 控制杆 JoyStick



# 模拟温度传感器



# 光线传感器



# 声音传感器

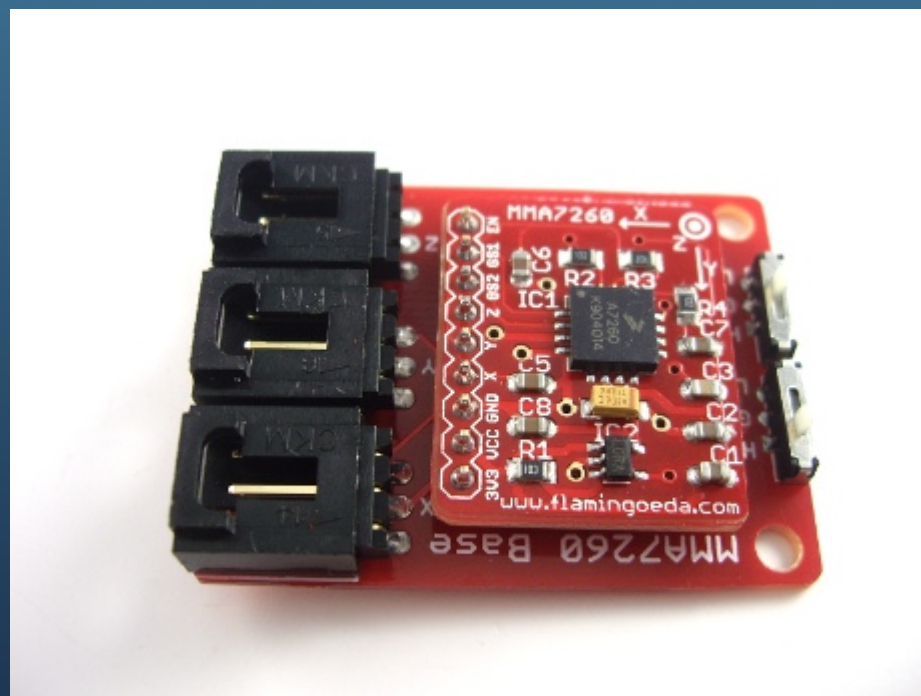


# Sharp 红外距离传感器

- 30cm, 80cm, 150cm, 5.5m



# 三轴 加速度传感器



# FlexiForce 压力传感器

- 1 lb, 25 lb, 100 lb





# FSR系列压力传感器

- 压力1-10kG
- 尺寸和外形不同



# 弯曲传感器

- 单向，双向



# 拉伸 传感器

- 4 inch, 6 inch



# 动手

- 实验各种模拟传感器
- 对RGB LED进行控制
- 分享

Thank you

# 动手玩转Arduino

## (三)

串口输入输出

Arduino北京俱乐部

# 串口通信

- 串口通信（Serial Communication）是Arduino和计算机间按位进行数据传输的一种最基本的方式。
- 使用3根线完成
  - 地线，GND
  - 发送，Tx
  - 接收，Rx
- 主要参数
  - 波特率：通信速度，表示每秒钟传送的位（bit）的个数
  - 数据位
  - 停止位
  - 奇偶校验位

# Arduino 串口

- USB转串口
  - PC端：串口
  - Arduino端：USB
- 串口引脚
  - RX: Pin 0
  - TX: Pin 1
- 串口数目
  - Arduino MEGA和Arduino 2560: 4个
  - 其余: 1个
- 串口初始化函数
  - `Serial.begin(speed)`
    - speed: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200
  - 通常在 `setup()` 函数里调用



# Arduino 串口输出

- 将数据从Arduino传输到PC
  - TX → 串口转USB → PC端软件串口监视软件
  - Arduino IDE自带串口监视软件
  - 也可以用其它软件进行接收：Flash, Processing, Director, vvvv等
- 串口输出函数
  - Serial.print(val), Serial.println(val)
    - 输出ASCII码（后者多一个回车换行）
  - Serial.print(val, format) , Serial.println(val, format)
    - 按指定格式输出（后者多一个回车换行）

# Serial.print(val) 实例

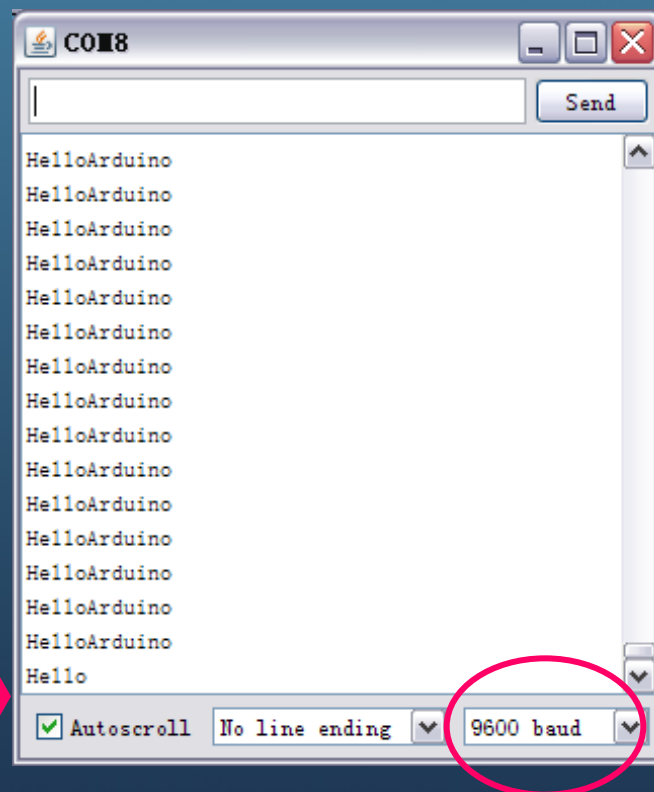
- Serial.print(78)
  - 输出"78"
- Serial.print(1.23456)
  - 输出"1.23"
- Serial.print(byte(78))
  - 输出"N"（N的ASCII码值为78）
- Serial.print('N')
  - 输出"N"
- Serial.print("Hello world.")
  - 输出"Hello world."

# Serial.print(val, format) 实例

- Serial.print(78, BYTE)
  - 输出"N"
- Serial.print(78, BIN)
  - 输出"1001110"
- Serial.print(78, OCT)
  - 输出"116"
- Serial.print(78, DEC)
  - 输出"78"
- Serial.print(78, HEX)
  - 输出"4E"
- Serial.print(1.23456, 0)
  - 输出"1"
- Serial.print(1.23456, 2)
  - 输出"1.23"
- Serial.print(1.23456, 4)
  - 输出"1.2346"

# 动手

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  // print labels  
  Serial.print("Hello");  
  Serial.println("Arduino");  
}
```

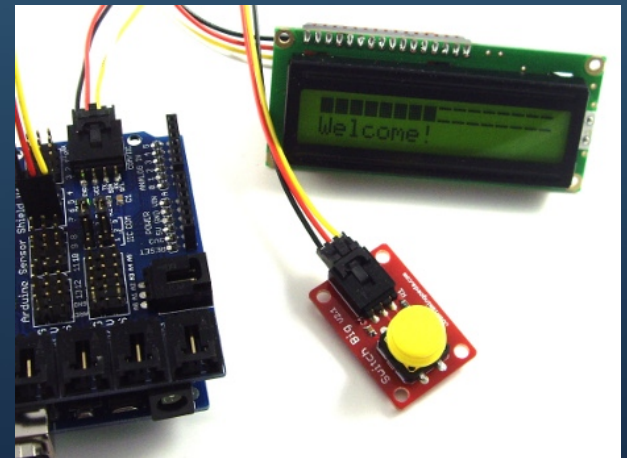


# 动手

- 尝试输出更多格式的数据到计算机
  - `Serial.print(78)`
  - `Serial.print(1.23456)`
  - `Serial.print(byte(78))`
  - `Serial.print('N')`
  - `Serial.print("Hello world.")`
  - `Serial.print(78, BYTE)`
  - `Serial.print(78, BIN)`
  - `Serial.print(78, DEC)`
  - `Serial.print(78, HEX)`
  - `Serial.print(1.23456, 2)`

# 串口液晶模块

- 波特率9600
- 命令格式
  - 所有对该液晶屏进行控制的串口命令都以字符”\$”开始，以回车换行”\r\n”结束，两者之间是相应的命令和参数，不同的命令具有不同的参数。
- 操作命令
  - 光标移动：GO<空格>行<空格>列
    - 行和列均从1开始
    - GO 1 1
  - 在当前光标位置上显示字符串 PRINT<空格>字符串
    - PRINT Hello Arduino
  - 清屏：CLEAR
  - 将光标移回到屏幕左上角的初始位置：HOME
  - 设置光标效果：CURSOR<空格>显示<空格>闪烁
    - 第一个参数为是否显示光标（1和0）
    - 第二个参数为是否闪烁（1和0）
    - CURSOR 1 1
- 一个完整命令的例子
  - Serial.print("\$PRINT Flamingo EDA\r\n");
  - Serial.println("\$PRINT Flamingo EDA");



# 动手

- 要求：清屏后在屏幕上的第二行第二列开始显示字符串 Arduino Club
- 提示
  - 在**loop**的最后加**delay**防止闪烁
  - 在下载代码的时候不能接显示屏

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.print("$CLEAR\r\n");  
    Serial.print("$GO 2 2\r\n");  
    Serial.print("$PRINT Arduino Club\r\n");  
    delay (2000);  
}
```

# Arduino 串口输入

- 串口队列（**Buffer**）
  - PC和Arduino间的缓冲区
- 串口输入函数
  - **Serial.available()**
    - 返回值：当前可读的数据数目
  - **Serial.read()**
    - 如果串口没有数据可读，返回 -1
    - 如果串口有数据可读，返回第一个字符，并从串口队列中取出
  - **Serial.peek()**
    - 如果串口没有数据可读，返回-1
    - 如果串口有数据可读，返回第一个字符，但不从串口队列取出，因此下次还能读到
  - **Serial.flush()**
    - 清空串口队列

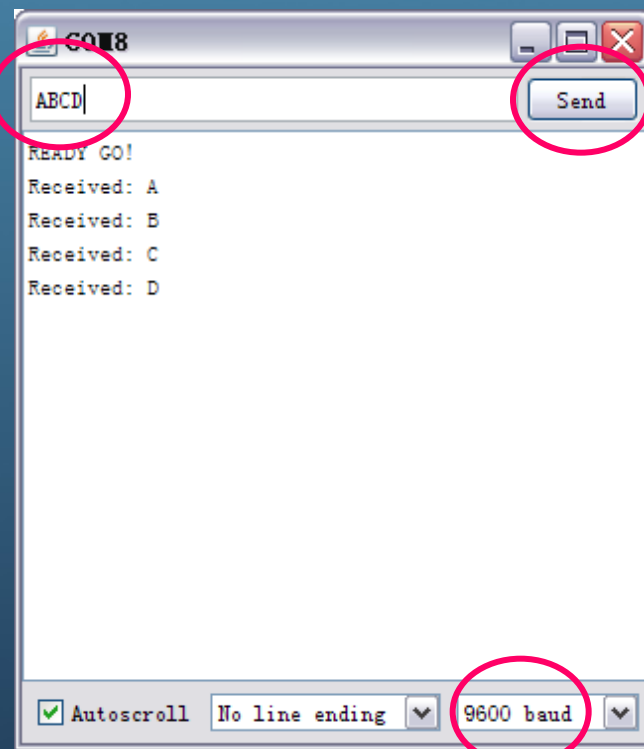


# 动手

```
int incomingByte = 0;

void setup() {
  Serial.begin(9600);
  Serial.print("READY GO!\r\n");
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    Serial.print("Received: ");
    Serial.println(incomingByte, BYTE);
  }
}
```



# 作业

- 从串口输入以\$开始，以回车换行（\r\n）的命令，对其进行解析，
  - 控制指定数字I/O引脚上的LED，
    - 第一个参数为引脚号，第二个参数为亮灭
    - \$D 2 1\r\n
    - \$D 3 0\r\n
  - 控制指定PWM端口上的LED亮度
    - 第一个参数为引脚号，第二个参数为亮度值
    - \$P 6 128\r\n

# 基于串口的电子积木（一）



串行液晶显示屏  
字符型 1602 LCD



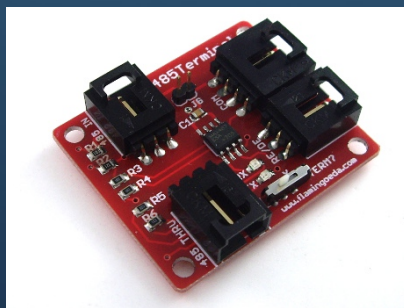
无线数据传输模块  
APC220



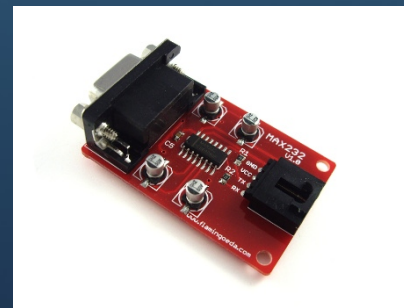
蓝牙串口模块



USB转串口适配器

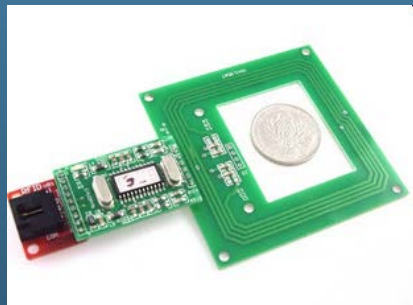


RS485串口模块

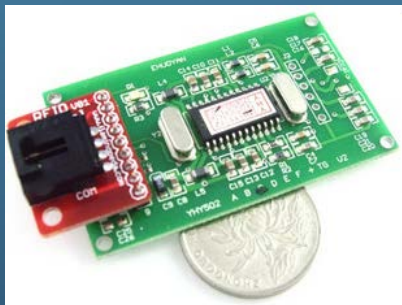


MAX232 串口模块

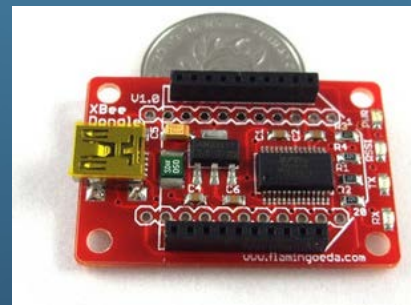
# 基于串口的电子积木（二）



串口RFID模块 10cm



串口RFID模块 6cm



XBee USB 适配器

谢谢!

# 动手玩转Arduino (四)

运动控制（直流电机/步进电机/舵机）

Arduino北京俱乐部

# 直流电机

- 将电能转换为机械能的一种装置
- 两个电源接头
- 在适当的电压下给予足够的电流时将连续旋转，旋转方向由电流方向决定
- 普通直流电机转速高力矩小，适用于对力矩要求小的场合



# 直流减速电机

- 直流减速电机（齿轮减速电机）是在普通直流电机的基础上，加上配套齿轮减速箱。
- 齿轮减速箱可以提供较低的转速和较大的力矩，不同的减速比可以提供不同的转速和力矩





# 常用参数

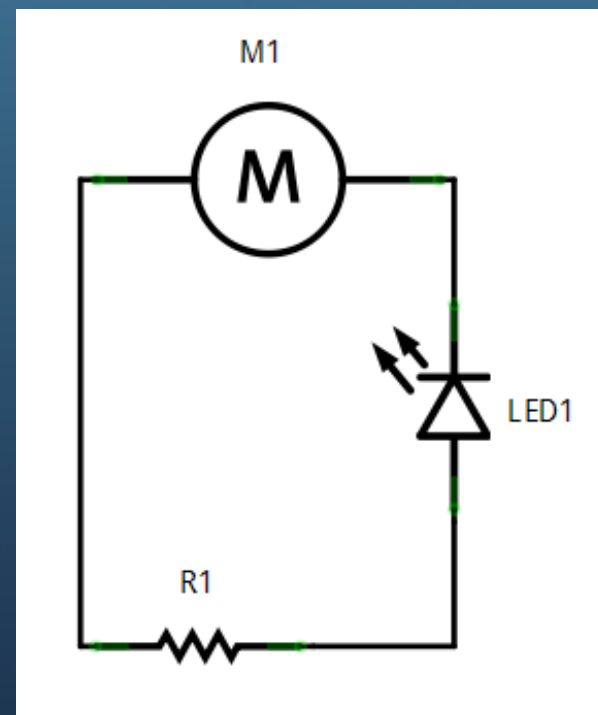
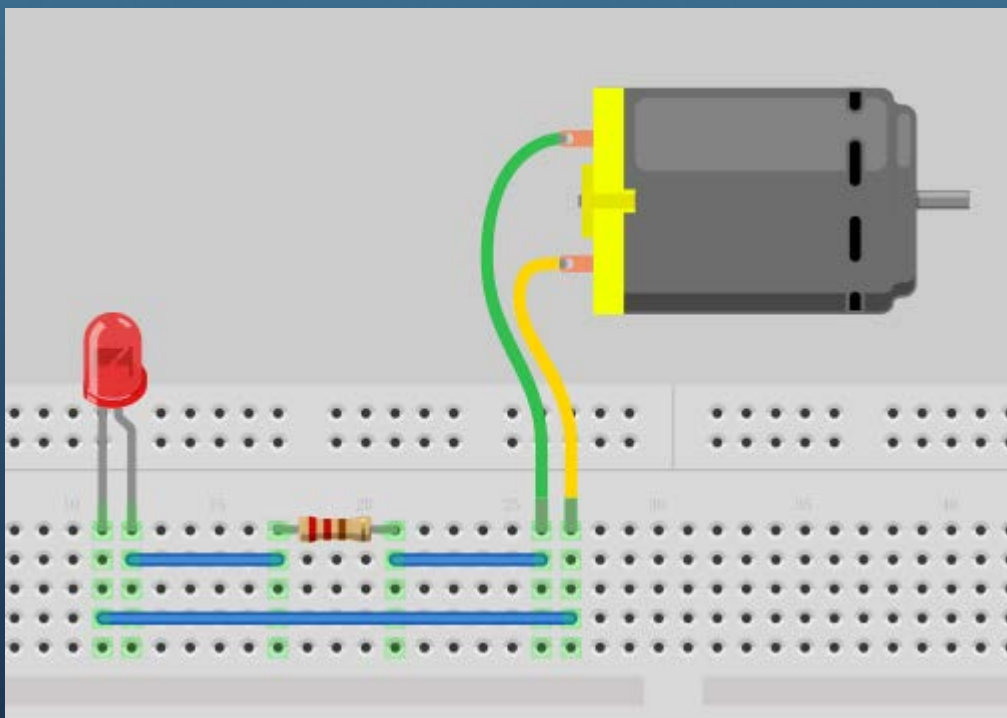
- 工作电压（额定电压）
  - 驱动电机推荐使用的电压
  - 高于或者低于工作电压时电机也能工作
  - 实际电压  $<$  额定电压，输出功率变小
  - 实际电压  $>$  额定电压，会影响电机的寿命
- 工作电流
  - 电机工作电流越大，输出功率越大
  - 空载运行时，电机的电流最小（空载电流）
  - 负载增大到使电机停止转动时（堵转电流）
- 转矩
  - 电机的转动力
- 转速
  - 每分钟旋转的圈数（转/分，RPM）

# 注意！

- 电机属于大电流设备，无法用Arduino引脚直接控制
  - 区别于LED
- 电机电压高于Arduino的工作电压，注意隔离和接线
  - 出错可能导致Arduino烧毁
- 电机在不通电的情况下旋转将产生逆电流（逆电压）
  - 发电机的工作原理
  - 逆电流的方向与电机工作电流的方向相反
  - 逆电流会造成电子设备的损坏

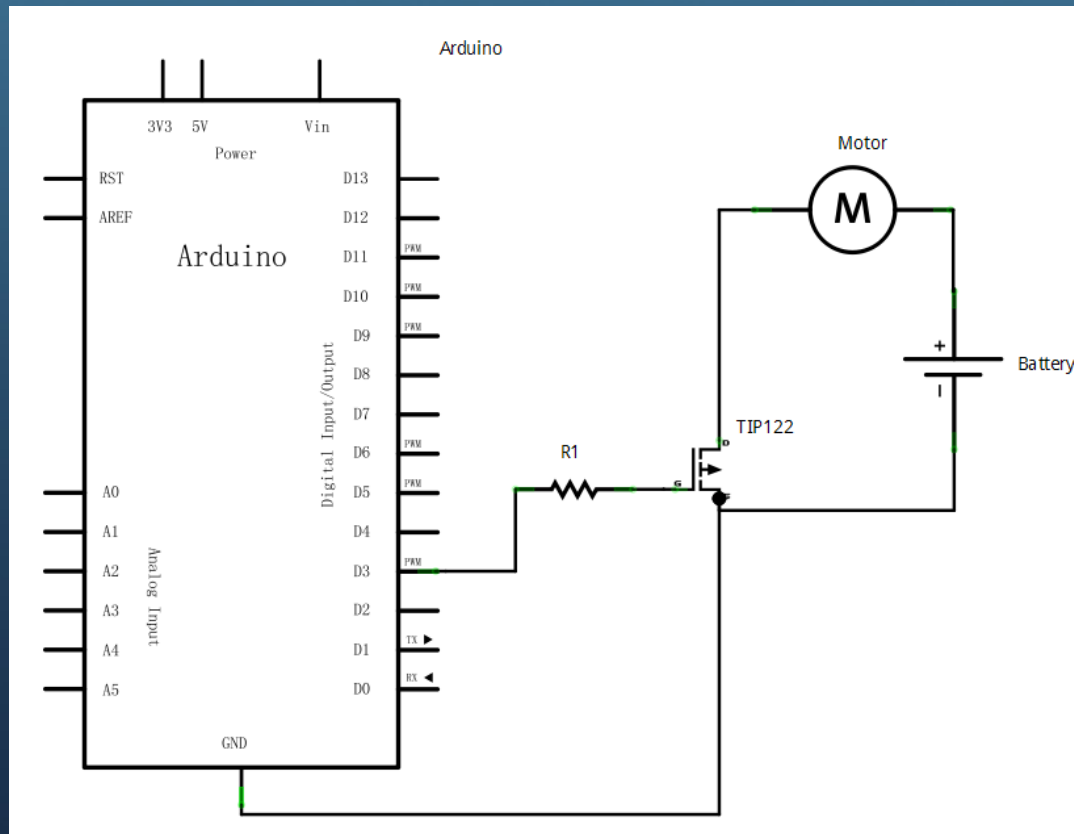
# 动手

- 观察逆电压——手工发电机



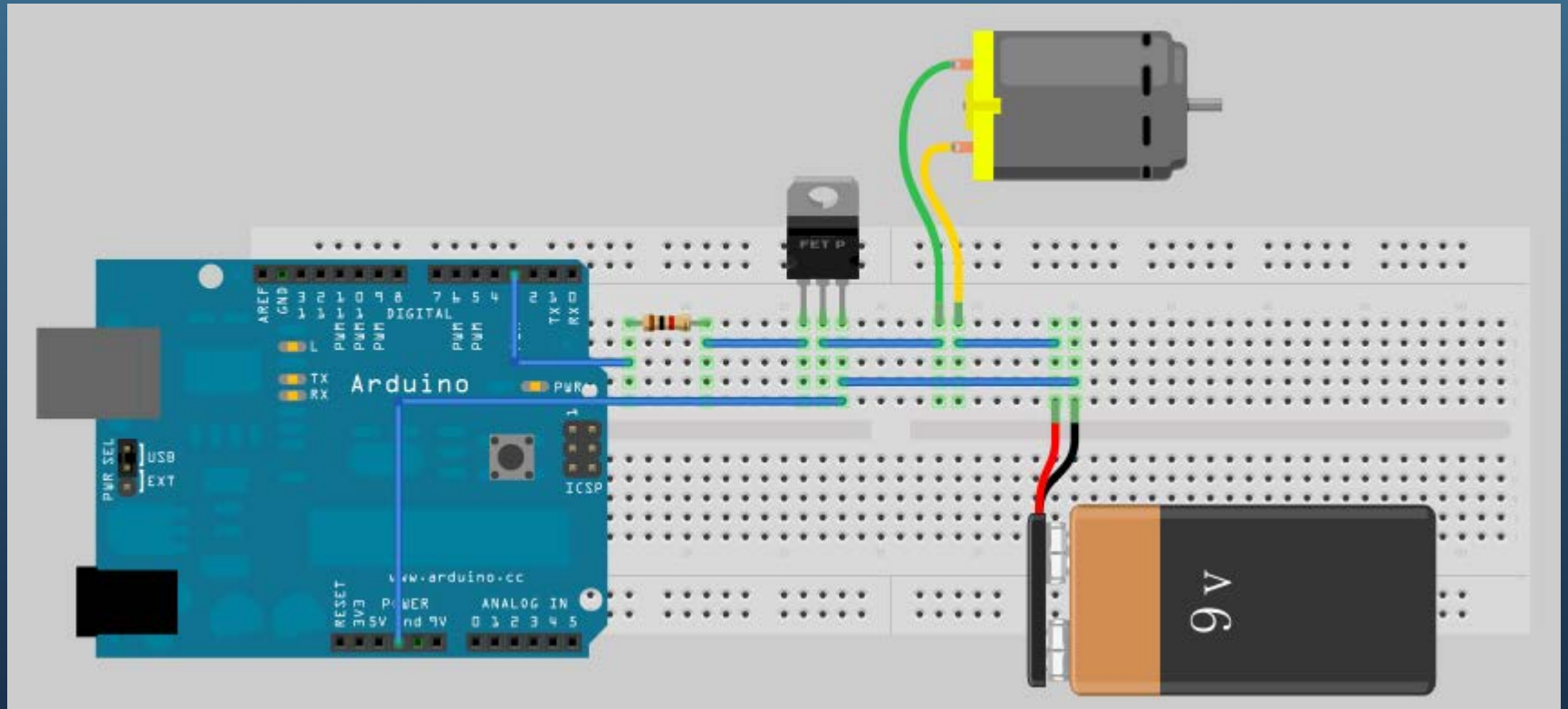
# 基本控制电路

- 用三极管或者MOS管驱动电机



# 动手

- 用Arduino控制电机的转动



# 代码

- 与LED灯控制代码相同

```
void setup() {  
    pinMode(3, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(3, HIGH);  
    delay(2000);  
    digitalWrite(3, LOW);  
    delay(2000);  
}
```

# 电机速度控制

- 复习PWM
  - 高速地开关三极管来实现对电机的控制
  - 占空比（打开时间:关闭时间）决定了输出给电机的能量份额
  - Arduino的`analogWrite`函数用来产生PWM信号

# 动手

- 用Arduino控制电机转动的速度
- 如何让速度变化更容易观察到？

```
int motorPin = 3;

void setup() {
}

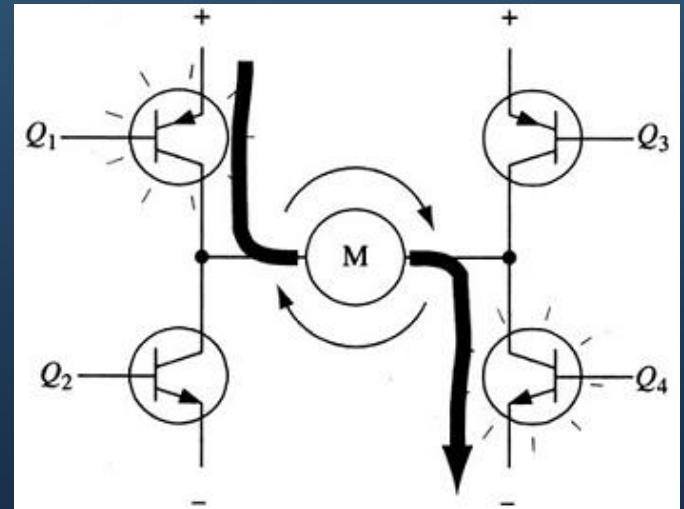
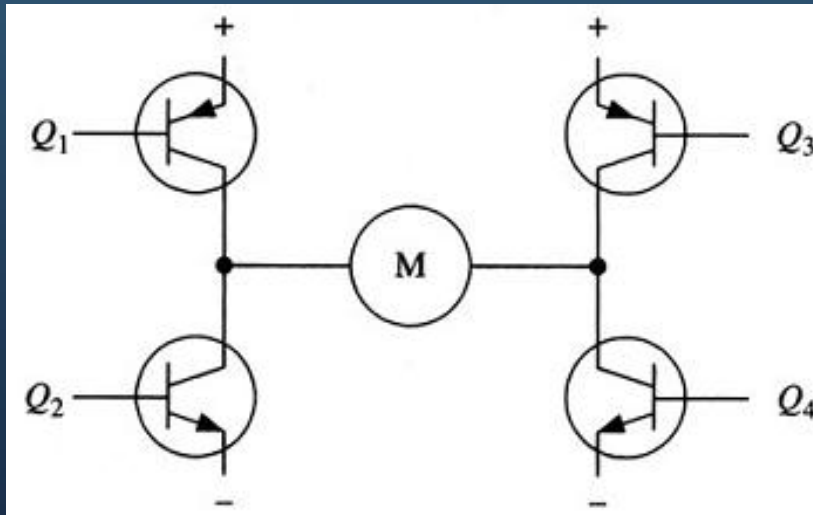
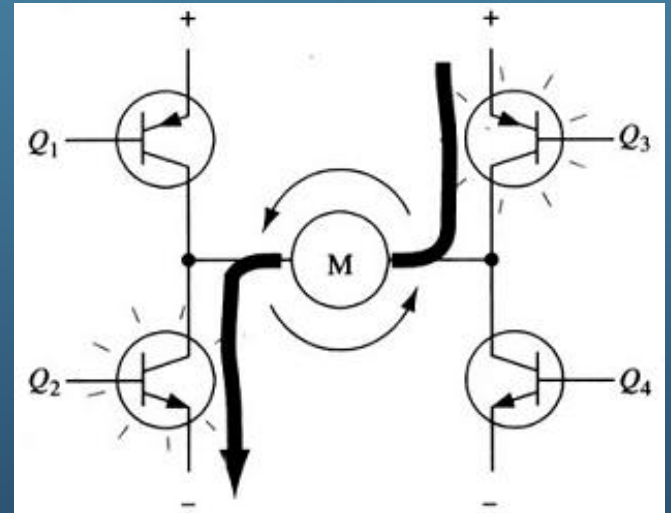
void loop() {
    for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=50) {
        analogWrite(motorPin, fadeValue);
        delay(2000);
    }

    for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=50) {
        analogWrite(motorPin, fadeValue);
        delay(2000);
    }
}
```



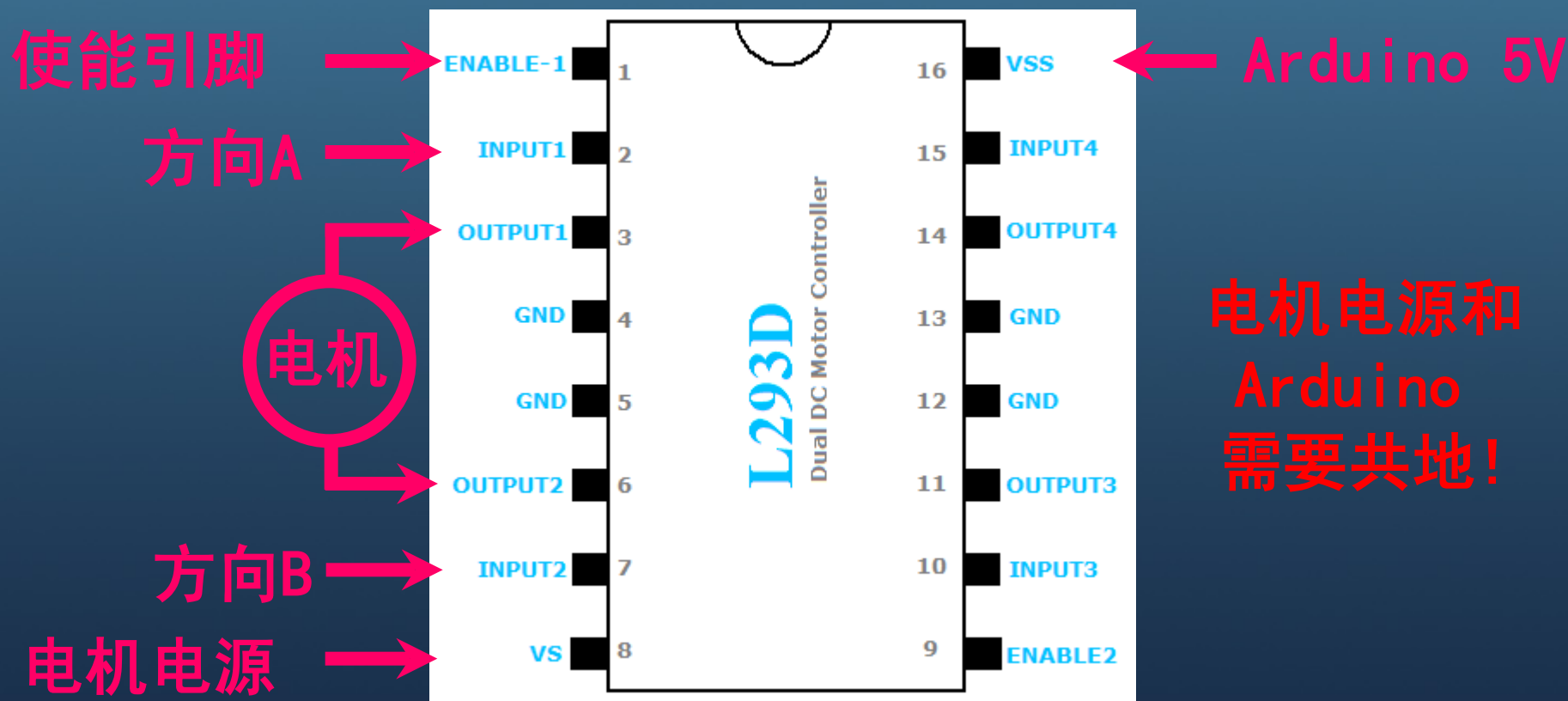
# H桥式驱动电路

- 4个三极管控制一个直流电机
- 只让对角线上的一对三极管导通
  - 否则会短路，烧坏三极管
- 能同时控制速度和方向



# 电机驱动芯片L293D

- 包含两个H桥式驱动电路，可以用来驱动两个直流电机



# Arduino控制L293D

- 速度控制
  - 使用Arduino模拟输出引脚
  - 将产生的PWM信号连接到L293D的使能引脚
- 方向控制
  - 使用Arduino的数字输出引脚
  - 将产生的数字信号连接到L293D的A、B两个方向引脚
  - 正转：A为高，B为低
  - 反转：A为低，B为高
  - 制动：A、B同时为高或者低

# 动手

- 用L293D控制电机

```
int dir1PinA = 13;
int dir2PinA = 12;
int speedPinA = 10;

unsigned long time;
int speed;
int dir;

void setup() {
    pinMode(dir1PinA, OUTPUT);
    pinMode(dir2PinA, OUTPUT);
    pinMode(speedPinA, OUTPUT);

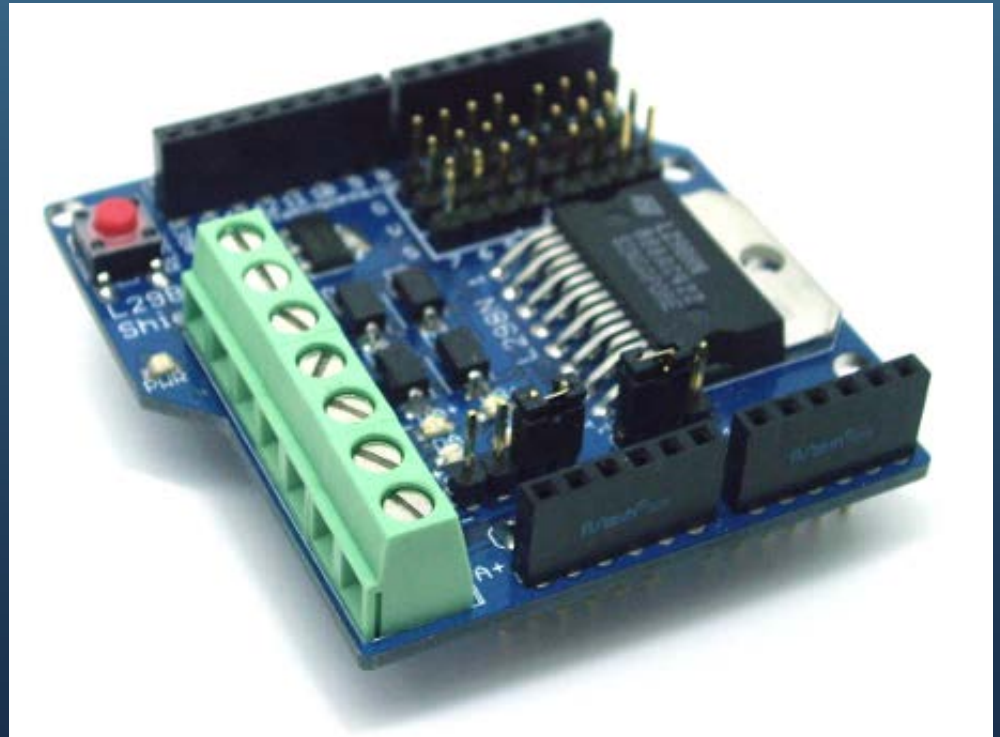
    time = millis();
    speed = 0;
    dir = 1;
}
```

```
void loop() {
    analogWrite(speedPinA, speed);
    // set direction
    if (1 == dir) {
        digitalWrite(dir1PinA, LOW);
        digitalWrite(dir2PinA, HIGH);
    } else {
        digitalWrite(dir1PinA, HIGH);
        digitalWrite(dir2PinA, LOW);
    }

    if (millis() - time > 5000) {
        time = millis();
        speed += 20;
        if (speed > 255) {
            speed = 0;
        }
        if (1 == dir) {
            dir = 0;
        } else {
            dir = 1;
        }
    }
}
```

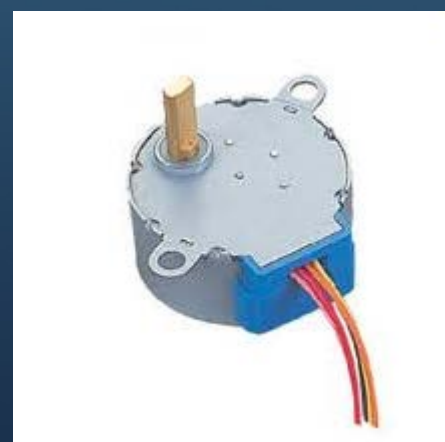
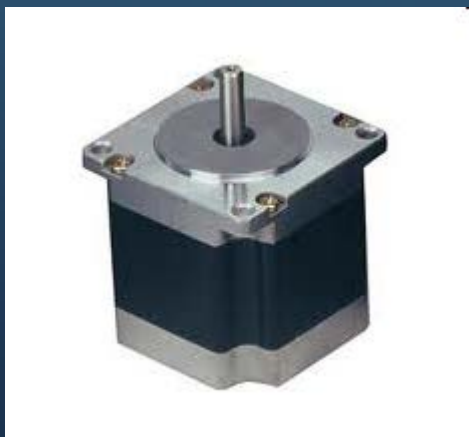
# Arduino电机驱动扩展板

- 基于L298N，工作原理与L293D一样
- 提供更大工作电流，可以驱动大的直线电机
- 逆电压消除电路



# 步进电机

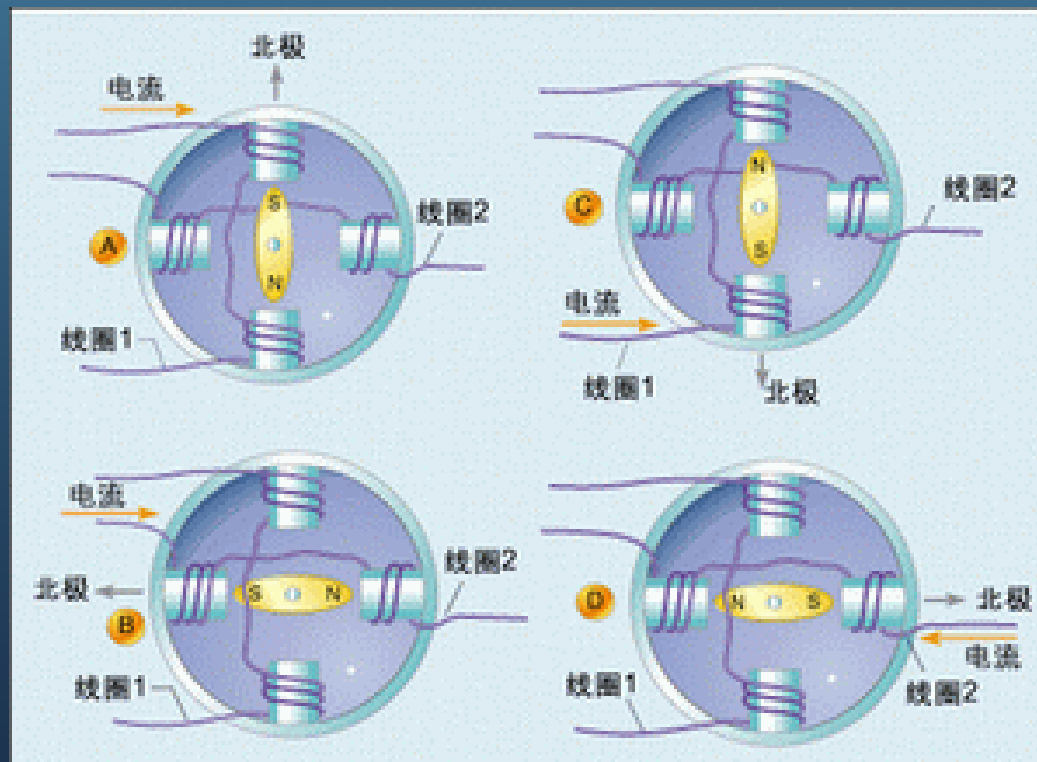
- 一种可以精确控制的交流电机
- 由驱动脉冲控制，每个脉冲让电机前进一个固定的角度（步进角）
- 电机速度由单位时间内脉冲的个数决定（脉冲频率）
- 功率小，负载能力低，控制相对复杂



# 双极性步进电机

- 每个线圈都可以两个方向通电
- 四根引线，每个线圈两条
- 使用数字万用表确定线圈分组
  - 某两根引线之间能够测量到阻值就属于一组
- 双极性步进电机的步距通常是 $1.8^{\circ}$ 
  - 转一圈需要200步

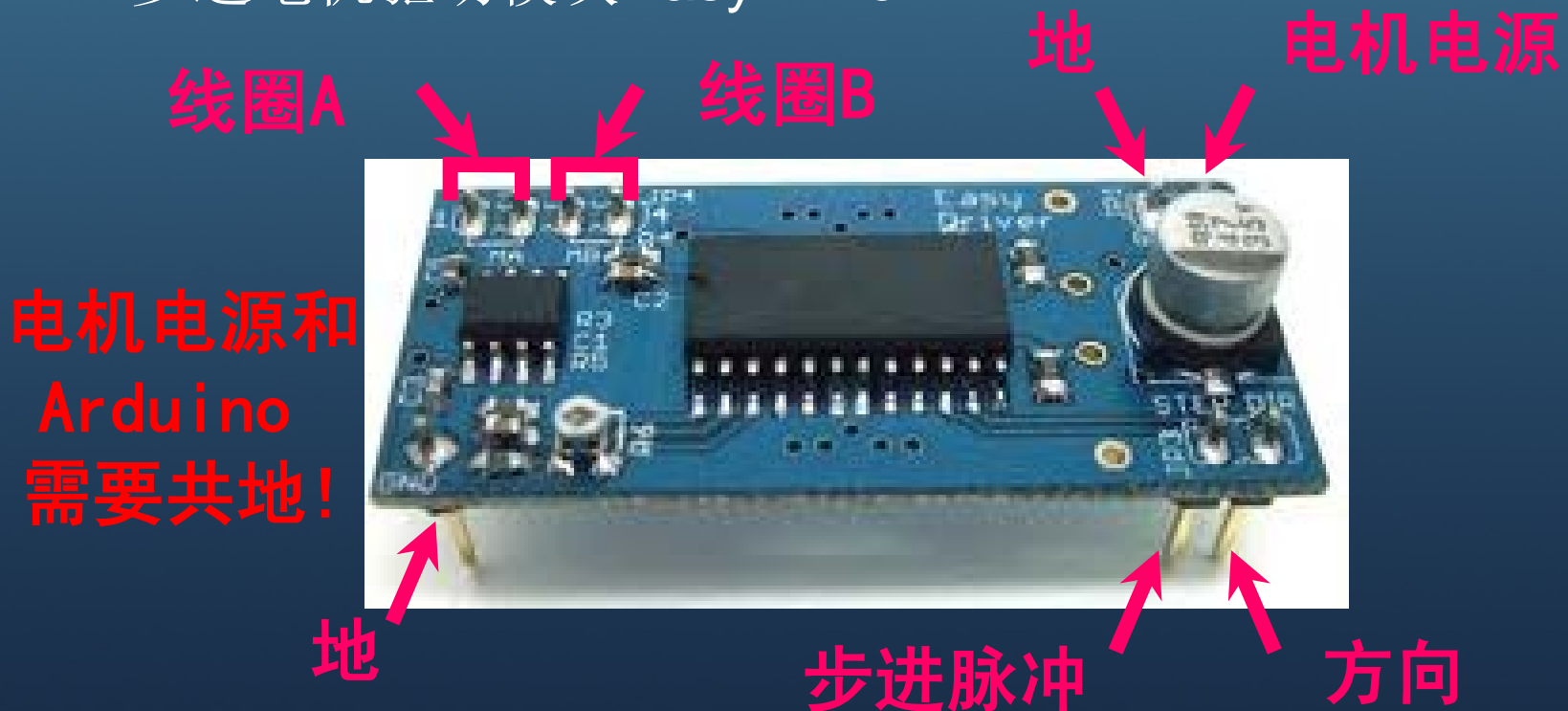
# 工作原理





# 步进电机驱动器

- 需要不断改变线圈中电流的方向
  - 通过H桥式驱动电路完成
- 步进电机驱动模块EasyDriver



# Arduino控制EasyDriver

- 方向控制
  - 使用Arduino的数字输出引脚
- 速度控制
  - 使用Arduino的数字输出引脚
  - 产生脉冲信号

# 动手

- 用EasyDriver驱动双极性步进电机

```
int stepperPin = 5;

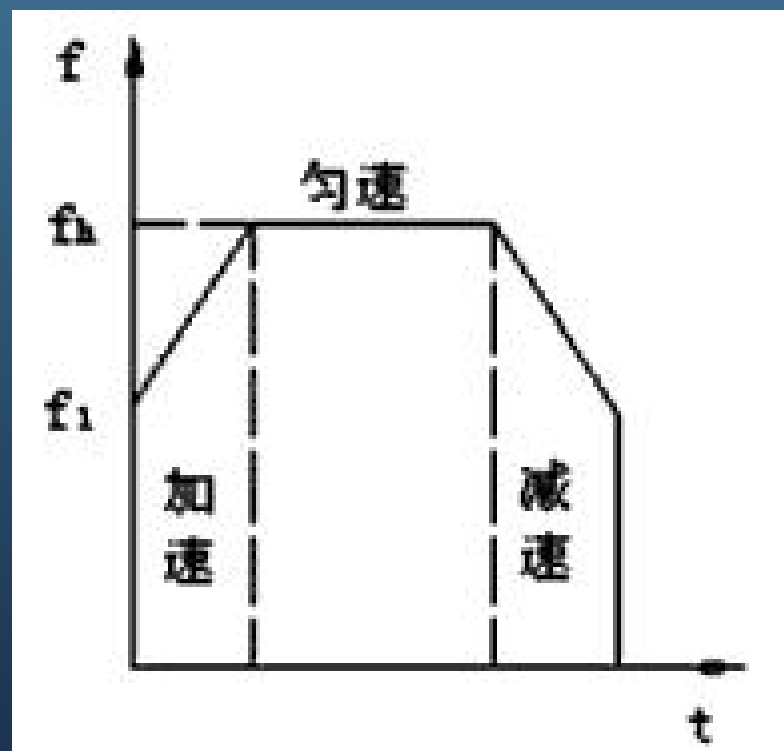
void setup() {
    pinMode(dirPin, OUTPUT);
    pinMode(stepperPin, OUTPUT);
}

void step(boolean dir, int steps){
    digitalWrite(dirPin, dir);
    delay(50);
    for(int i=0; i<steps; i++){
        digitalWrite(stepperPin, HIGH);
        delayMicroseconds(100);
        digitalWrite(stepperPin, LOW);
        delayMicroseconds(100);
    }
}
```

```
void loop(){
    step(true, 1600);
    delay(500);
    step(false, 1600*5);
    delay(500);
}
```

# 升降速曲线

- 步进电机在拖动负载高速移动一定距离并精确定位时一般来说都应包括五个阶段
  - 启动
  - 加速
  - 高速运行（匀速）
  - 减速
  - 停止
- 不同阶段的脉冲频率应不同



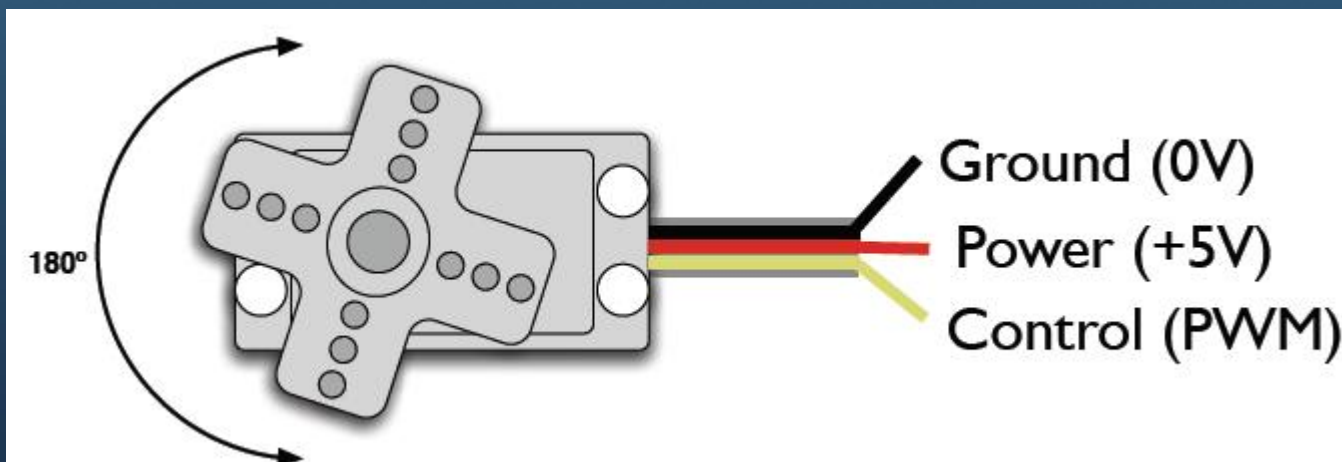
# 舵机

- 舵机也叫伺服电机（**Servo**），工作电压一般4.6 – 7.2V
- 由直流电机、减速齿轮组、传感器和控制电路组成
- 主要应用在定位控制上



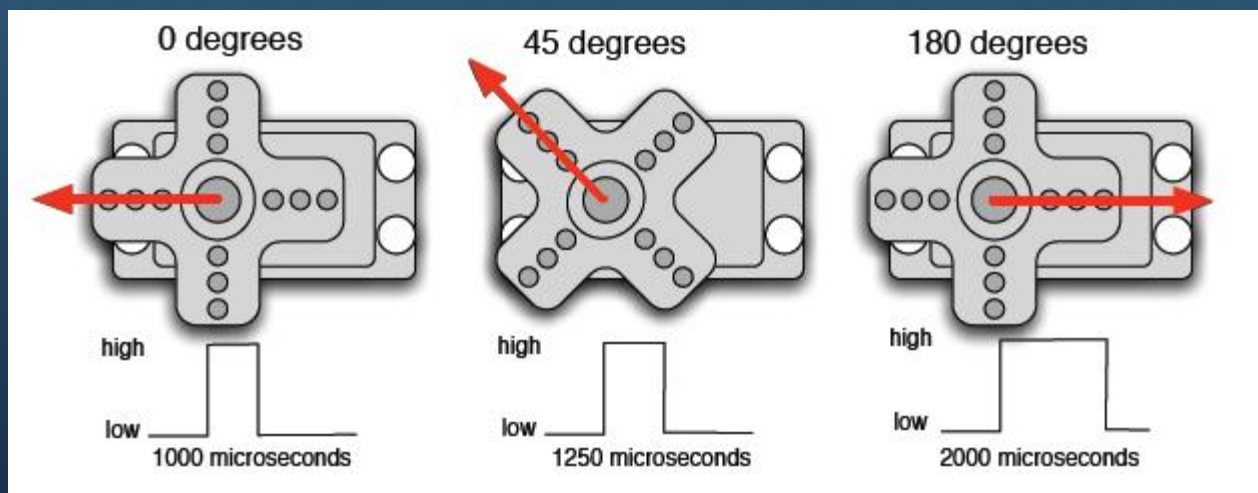
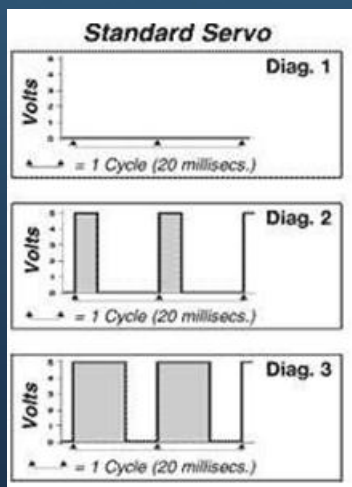
# 舵机接口

- 标准三线接口
  - 电源线
  - 地线
  - 控制线



# 控制信号

- 舵机的控制信号也是一种PWM信号
- 脉冲周期20毫秒
- 脉冲高电平持续1毫秒（1000微秒）到2毫秒（2000微秒）
- 用来控制的脉冲信号必须持续产生，否则很难稳定



# 注意！

- 舵机对控制脉冲高电平的宽度非常敏感
  - 抖动：控制信号每个脉冲的高电平宽度不稳定
- 两个控制脉冲的间隔时间不太敏感
  - 只要在可以接受的范围内都可以（14ms到20ms）
- 只有机械约束，没有电路约束
  - 控制脉冲宽度大于920us或者2120us，对电机寿命会有影响
  - 驱动电机到达物理上不可能到达的位置
- 千万不能接反电源
  - 大部分舵机都可能会烧毁
- 通过实验来找出电机的运动范围
  - 大部分舵机的运动区间都在150度左右



# Arduino如何控制舵机

- Servo库
  - servo.attach(pin)
    - 在指定的引脚上连接舵机
  - servo.write(analog)
    - 将舵机旋转 to 指定的角度位置
  - servo.writeMicroseconds(us)
    - 生成指定宽度（微秒）的控制脉冲

# 动手

- 利用Arduino为舵机找到中心位置
- 以中心位置左右摆动

谢谢!

# 参考

- <http://zhouxiaoxi198906.blog.163.com/blog/static/1290140362010431238402/>
- <http://www.guokr.com/article/5292>

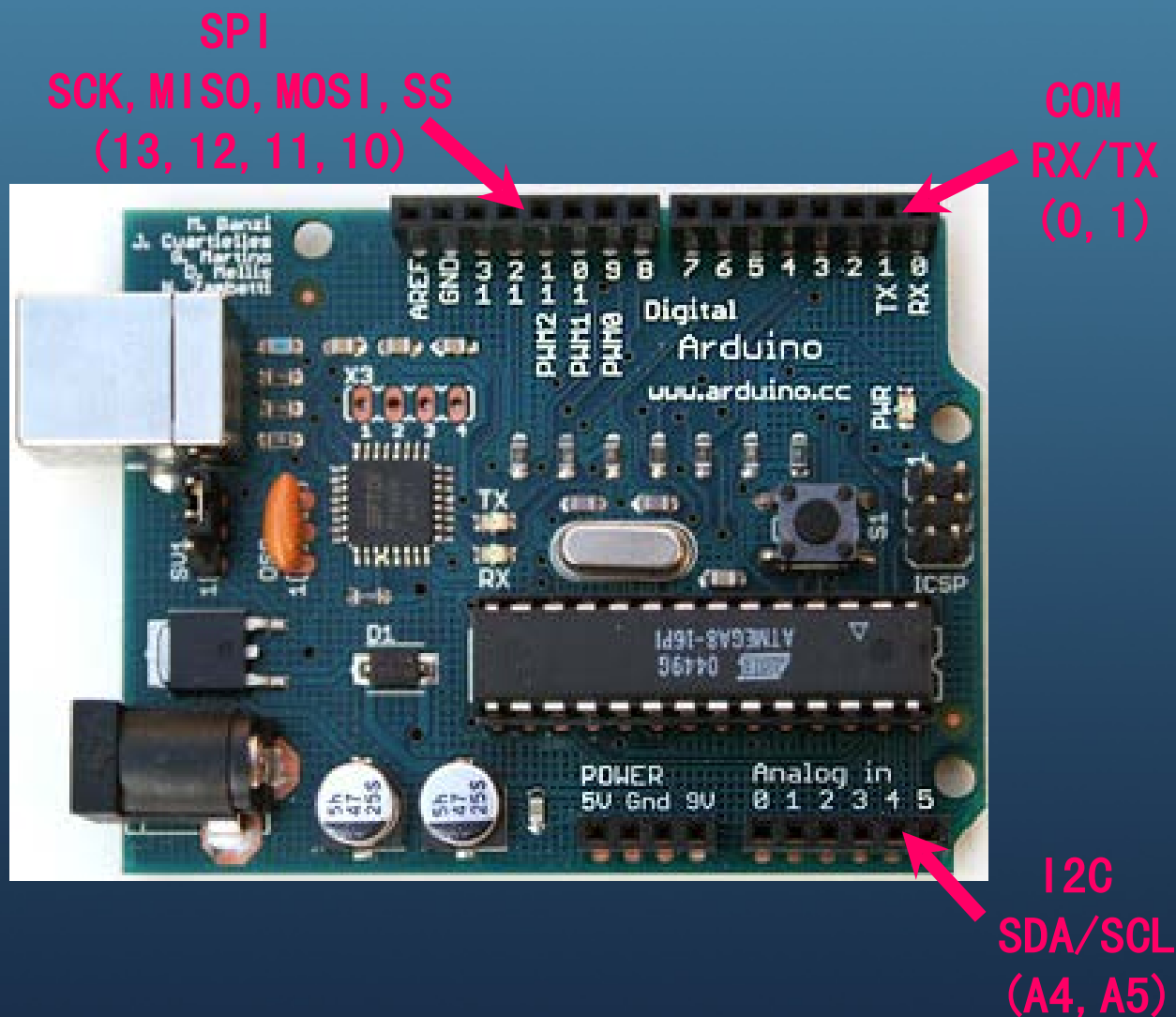
# 动手玩转Arduino (五)

RF数据通信

Arduino北京俱乐部

# Arduino如何与外界沟通

- Pin
- COM
- SPI
- IIC



# RF遥控

- 无线射频（Radio Frequency）
  - 在空气中传播的电磁波
- 主要包括无线收/发机
  - 发射模块
  - 接收模块
- 应用场景
  - 车辆/家庭防盗
  - 遥控玩具



# 4路RF模块

- 无线发射模块对应于数字输出（Digital Out）
  - 将相应引脚设置成高，发出数据
- 无线接收模块对应于数字输入（Digital In）
  - 若相应引脚读出为高，接收数据
- 一共4个通道：C1、C2、 C3、 C4



无线发射模块



无线接收模块



# 动手

- 两台Arduino一组
  - 发射组
    - RF发射模块一个
    - 按钮模块一个
  - 接收组
    - RF接收模块一个
    - LED模块一个
- 目标
  - 通过按钮来控制对方LED灯的亮灭

# 参考代码

```
int buttonPin = 3;    // pushbutton pin
int rfPin = 7;        // RF send pin
int state = 0;

void setup() {
  pinMode(rfPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop(){
  state = digitalRead(buttonPin);
  if (state == HIGH) {
    digitalWrite(rfPin, HIGH);
  } else {
    digitalWrite(rfPin, LOW);
  }
}
```

发射端代码

```
int ledPin = 3;        // led pin
int rfPin = 7;         // RF recv pin
int state = 0;

void setup() {
  pinMode(rfPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop(){
  state = digitalRead(rfPin);
  if (state == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

接收端代码

# 问题讨论

- 各个组之间如何避免冲突
  - 可以尝试使用不同的通道
  - 为RF模块配置不同的地址



# 动手

- 两台Arduino一组，交换发射组和接收组
  - 发射组
    - RF发射模块一个
    - 按钮模块两个
  - 接收组
    - RF接收模块一个
    - 舵机
- 目标
  - 多通道无线数据收发
  - 模拟遥控汽车，用按钮控制舵机的左右转向

# RF数据传输

- 4路RF模块传输的是数字信号
  - 简单的开/关、1/0信号
- 无线数传模块可以实现各种数据的传输
  - 数字量：开/关
  - 模拟量：光线、温度、声音等

# APC220

- 通过串口（COM）与Arduino进行沟通
- 数据收发一体，即可以发送数据又可以接收数据
  - 数据发送：写串口 `Serial.write()`
  - 数据接收：读串口 `Serial.read()`
- 传输距离：开阔地1200m



# Arduino与COM模块的连接

- 传感器扩展板
- COM/I2C连接线



COM

# 动手

- 两台Arduino一组
  - 发射组
    - APC220一个
    - 按钮模块一个
  - 接收组
    - APC220一个
    - LED模块一个
- 目标
  - 发射组在按钮按下时通过APC220模块发送数据'A'
  - 接收组在收到数据'A'时点亮LED



# 参考代码

```
int buttonPin = 3;
int state = 0;

void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop(){
  state = digitalRead(buttonPin);
  if (state == HIGH) {
    delay(100);
    state = digitalRead(buttonPin);
    if (state == HIGH) {
      Serial.print('A');
    }
  }
}
```

发射端代码

```
int ledPin = 7;
int val = 0;

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

void loop(){
  val = Serial.read();
  if (-1 != val) {
    if ('A' == val || 'a' == val) {
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
    }
  }
}
```

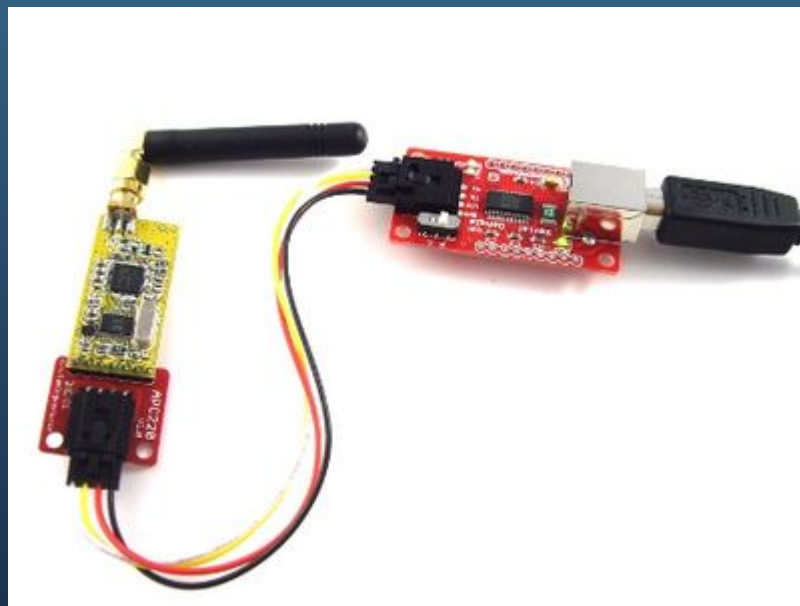
接收端代码

# 问题讨论

- 按钮的处理
  - 消抖处理：发送端`delay(100)`改成`delay(10)`呢？
  - 如何保证按一下按钮，只发送一次？

# APC与PC间的无线连接

- USB转串口适配器



# 动手

- Arduino作为数据收集器采集传感器数据
- 通过APC220模块传递给PC机
- 通过串口监视软件观察数据变化
- 如何使数据更加可读？

谢谢!