第5章
执行机构

5.1 LED 灯

LED 灯(如图 5-1 所示)是最简单、最基本的执行机构,通过它的亮灭,可以反映传感器的状态。本书前面好多关于传感器的实验,都是通过 LED 灯来显示传感器的状态的。

图 5-1 LED 灯

LED 灯模块的供电电压为 5V。

当输入高电平(HIGH)时,灯亮;当输入低电平(LOW)时,灯灭。

例程见前面的传感器模块。

5.2 蜂鸣器

蜂鸣器模块的作用与 LED 灯类似，LED 灯是使用灯的亮灭来显示状态，而蜂鸣器模块则是用声音来显示状态。

蜂鸣器模块分为有源蜂鸣器模块和无源蜂鸣器模块。这里所说的有源和无源并不是指电源，而是指震荡源。有源蜂鸣器模块内含震荡源，通电后能直接发出声音；而无源蜂鸣器模块通电后并不能发出声音，需要给它一个具有一定频率的方波信号才能发出声音。因此，有源蜂鸣器模块（如图 5-2 所示）使用起来比较简单。

图 5-2 有源蜂鸣器模块

蜂鸣器模块的工作电压为 5V，当输入高电平（HIGH）时，会发出声音。

例程

当按钮模块被按下时，有源蜂鸣器模块会发出声音。

```
int buttonPin = 53;        //定义按钮传感器接到 53 口
int buzzerPin = 40;        //40 口接蜂鸣器
int buttonState = 0;       //定义按钮传感器初始状态为 0
```

```
void setup()
{
  pinMode(buzzerPin, OUTPUT);       //定义 40 口为输出口
  pinMode(buttonPin, INPUT);        //定义 53 口为输入口
}

void loop()
{
 buttonState = digitalRead(buttonPin);  //读取 53 口状态
  if (buttonState == LOW)               //如果 53 口为低电平
{
    digitalWrite(buzzerPin, HIGH);//40 口输出高电平,蜂鸣器响
    }
  else {
    digitalWrite(buzzerPin, LOW); //反之,40 口输出低电平,蜂鸣
                                  //器不发声
  }
}
```

5.3　直流电动机驱动模块

本节主要介绍常用的 L298N 直流电动机驱动模块(如图 5-3 所示)。

主板上有两个输出口,可以分别接两个直流电动机。

有一个电源接口,左边的是电动机电源,输入范围是 7～12V;中间的接地;右边的是 5V 电源输入输出接口。

板上的“板载 5V 使能”跳线帽是使能板载的 5V 逻辑供电。当使用低于 12V 的电动机驱动电压时,可以利用它为控制板提供 5V 电源。当使用大于 12V 的电动机驱动电压时,为了避免稳压芯片损坏,首先要拔掉板载 5V 输出使能的跳线帽,然后在 5V 输入

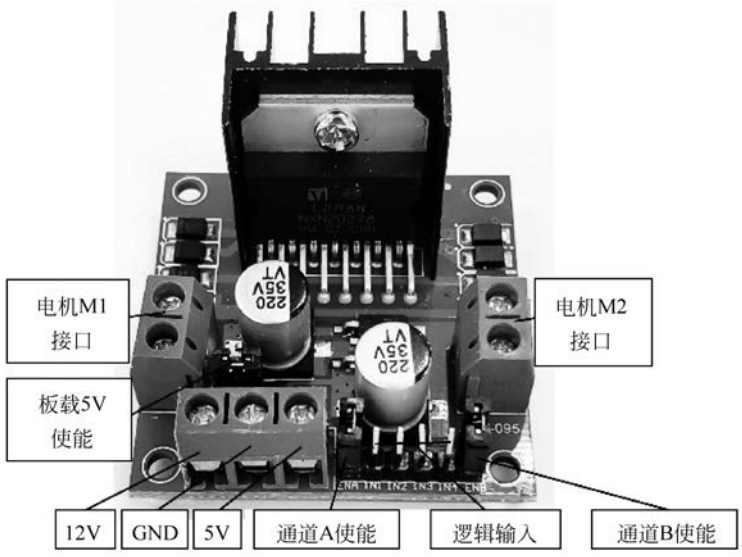

图 5-3　L298N 直流电动机驱动模块

端口外部接入 5V 电压，为 L298N 内部逻辑供电。

ENA 通道 A 使能，ENB 通道 B 使能，当不用 PWM 调速时，不需要拔掉跳线帽；当需要用 PWM 调速时，要拔掉跳线帽，接到 Arduino 上的 PWM 输出接口。

PWM 就是脉宽调制器，通过调制器给电动机提供一个具有一定频率的脉冲宽度可调的脉冲电流。脉冲宽度越大即占空比越大，提供给电动机的平均电压就越大，电动机转速也就越高。反之，脉冲宽度越小，则占空比越小，提供给电动机的平均电压就越小，电动机转速也就越低。

IN1、IN2、IN3、IN4 是逻辑输入口。其中：IN1、IN2 控制一个电动机 M1 的转动；IN3、IN4 控制另一个电动机 M2 的转动。只要一个置高一个置低，就可以让电动机转动起来。

信号控制方式如表 5-1 所示。

表 5-1　信号控制方式

直流电动机	旋转方式	IN1	IN2	IN3	IN4	调速 PWM 信号	
						调速端 A	调速端 B
M1	正转	高	低	/	/	高	/
	反转	低	高	/	/	高	/
	停止	低	低	/	/	高	/
M2	正转	/	/	高	低	/	高
	反转	/	/	低	高	/	高
	停止	/	/	低	低	/	高

例程

下面例子实现的目标是：小车以 100 的速度前进 5s，左转 5s，右转 5s，最后停止。小车的速度可调，赋的值越大，速度越快（本实验通过编程产生 PWM 脉冲控制电动机速度，PWM 值的范围为 0～255。）

如果程序运行后，电动机的转动方向与设定的不一致，那么将电动机控制板上的电动机接线端子上的两根电线颠倒一下即可。

注意：在接线的时候，主控器和电动机驱动板之间一定要共地，以保证控制信号的传输。

```
int ENA = 2;              //左电动机 M1 速度控制,控制板 PWM 接口
int ENB = 3;              //右电动机 M2 速度控制,控制板 PWM 接口
int IN1 = 50;             //左电动机 M1 正反转控制
int IN2 = 51;             //左电动机 M1 正反转控制
int IN3 = 52;             //右电动机 M2 正反转控制
int IN4 = 53;             //右电动机 M2 正反转控制

void setup()
{
  pinMode(IN1, OUTPUT);   //设定各接口为输出口
```

```
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  digitalWrite(IN1, LOW);//设定各接口初始值为低电平,小车停止
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

void loop()
{
    forward(100,100);
    delay(5000);
    stopp();
    delay(10);
    turnleft(100,100);          //左转
    delay(5000);
    stopp();
    delay(10);
    turnright(100,100);         //右转
    delay(5000);
    stopp();
    while(1);
  }

void turnleft(int b,int c)      //定义左转函数
{
  analogWrite(ENA,b);           //给 ENA 赋一个速度值
  analogWrite(ENB,c);           //给 ENB 赋一个速度值
  digitalWrite(IN1, LOW);       //给 IN1 低电平
  digitalWrite(IN2, HIGH);      //给 IN2 高电平
  digitalWrite(IN3, HIGH);      //给 IN3 高电平
```

```
  digitalWrite(IN4, LOW);       //给 IN4 低电平
}

void turnright(int b,int c)     //定义右转函数
{
  analogWrite(ENA,b);
  analogWrite(ENB,c);
  digitalWrite(IN1, HIGH);      //给高电平
  digitalWrite(IN2, LOW);       //给低电平
  digitalWrite(IN3, LOW);       //给低电平
  digitalWrite(IN4, HIGH);      //给高电平
}

void forward(int b,int c)       //定义前进函数
{
  analogWrite(ENA,b);
  analogWrite(ENB,c);
  digitalWrite(IN1,HIGH );      //给 IN1 高电平
  digitalWrite(IN2, LOW);       //给 IN2 低电平
  digitalWrite(IN3,HIGH );      //给 IN3 高电平
  digitalWrite(IN4, LOW);       //给 IN4 低电平
}

void stopp()                    //定义停止函数
{
  analogWrite(ENA,0);
  analogWrite(ENB,0);
  digitalWrite(IN1, LOW);       //给低电平
  digitalWrite(IN2, LOW);       //给低电平
  digitalWrite(IN3, LOW);       //给低电平
  digitalWrite(IN4, LOW);       //给低电平
}
```

5.4 直流电动机

直流电动机(如图 5-4 所示)作为机器人系统的主要执行单元,为机器人实现灵活运动提供动力来源。直流电动机是将电能转换为机械能的转动装置,因其良好的调速性能而得到广泛应用。

图 5-4 直流电动机

直流电动机由定子和转子两部分组成。运行时静止不动的部分称为定子,定子的主要作用是产生磁场,由机座、主磁极、换向极、端盖、轴承和电刷装置等组成。运行时转动的部分称为转子,其主要作用是产生电磁转矩和感应电动势,是直流电动机进行能量转换的枢纽。

直流电动机是根据通电流的导体在磁场中会受力的原理来工作的,即电工基础中的左手定则。电动机的转子上绕有线圈,通入电流,永磁电动机的定子为永磁铁,产生定子磁场,通电流的转子线圈在定子磁场中就会产生电动力,推动转子旋转。转子电流通过整流子上的碳刷连接到直流电源。

我们用得最多的是直流减速电动机。

直流减速电动机是市场上最普及的齿轮减速电动机。它具有体积小、重量轻、力矩大、控制能力强、结构紧凑、运行可靠等优点，被广泛应用在很多行业。

直流减速电动机也被称为齿轮减速电动机，它是在普通直流电动机的基础上，加上配套齿轮减速箱组成的。齿轮减速箱的作用是可提供较低的转速和较大的力矩。同时，齿轮箱不同的减速比可以提供不同的转速和力矩，这大大提高了直流电动机在自动化行业中的使用率。减速电动机是减速机和电动机的集成体。

直流电动机的选型如下。

（1）明确电动机安装的空间大小，以确定电动机的大小。

（2）确定电动机可用的工作电压和电流。

（3）确定电动机的转速。

（4）确定电动机的转矩。

直流减速电动机在实际使用中，可以利用直流电动机驱动板来控制电动机的转速和旋转方向。

5.5　舵机

在机器人机电控制系统中，舵机的控制效果非常重要，它作为基本的输出执行机构，简单的控制操作使其在使用时非常方便。

舵机（如图 5-5 所示）是一种位置（角度）伺服的驱动器，适用于那些保持一定角度的控制系统。它主要由外壳、电路板、电动机、齿轮、位置检测器等构成。

舵机常用的控制信号是一个周期为 20ms 左右，宽度为 1ms 到 2ms 的脉冲信号。当舵机收到该信号后，会马上激发出一个与

图 5-5　单轴舵机

之相同的，宽度为 1.5ms 的负向标准的中位脉冲。之后两个脉冲在一个加法器中进行相加，得到所谓的差值脉冲。输入信号脉冲如果宽于负向的标准脉冲，得到的就是正的差值脉冲。如果输入脉冲比标准脉冲窄，相加后得到的肯定是负的脉冲。此差值脉冲放大后，就是驱动舵机正反转动的动力信号。舵机电动机的转动通过齿轮组减速后，同时驱动转盘和标准脉冲宽度调节电位器转动，直到标准脉冲与输入脉冲宽度完全相同，差值脉冲消失时，才会停止转动。

舵机的工作电压一般为 5V。

Arduino 控制舵机有自带的函数库“Servo.h”，在程序中可以很方便地进行调用。

其中几个常用的函数如下。

(1) attach(接口)，设定舵机与控制器相连接的接口。舵机要与控制器的 PWM 接口相连接。

(2) write(角度),设定舵机旋转的角度,范围是0°～180°。

(3) read(),读取舵机的角度。

例程

目标:利用Arduino控制器控制舵机从0°转到180°,然后再从180°转到0°,如此循环。

```
#include <Servo.h>          //调用舵机库
Servo myservo;              //定义舵机变量
int pos = 0;                //给整型变量pos赋值0,设定舵机初始
                            //角度为0°


void setup()
{
  myservo.attach(9);        //舵机接到控制板9号口
}

void loop()
{
 for (pos = 0; pos <= 180; pos += 1)  //pos值从0到180
 {
    myservo.write(pos);     //把pos值发送给舵机
    delay(15);              //等待15ms
  }
  for (pos = 180; pos >= 0; pos -= 1) //从180到0
{
    myservo.write(pos);     //把pos值发送给舵机
    delay(15);              //等待15ms
  }
}
```

5.6 机械手臂

机械手臂是机械手和机械臂的合体。

作为机器人的抓取机构，机械手臂为机器人实现精准抓取提供了可能，是机器人系统中的重要组成部分。

机械手主要由爪子和舵机组成，通过对舵机的控制，可以实现爪子的收紧和打开，以便对物体进行抓取和放下。

机械臂的主要作用是将爪子移到所需位置，以便对物体进行抓取。

机械臂的种类比较多，图 5-6、图 5-7 这两种是我们常用的，它们结构简单，容易控制。

图 5-6 只有两个自由度，它由两个舵机来完成机械臂的前后伸缩和升降运动。

图 5-6 两个自由度的机械手臂

图 5-7 有三个自由度，增加了机械臂在水平方向的转动，这个转动的动力也是来自舵机。这样，这个手臂就有三种运动：伸缩、升降和旋转。

图 5-7　三个自由度的机械手臂

例程

目标：三个自由度的机械手臂在原地旋转、伸缩或升降，将指定位置的物体抓起保持，然后机械手臂回到原位。

```
#include <Servo.h>          //调用舵机库
Servo myservoA;             //定义舵机变量底座
Servo myservoB;             //大臂
Servo myservoC;             //小臂
Servo myservoD;             //爪子
int pos;                    //定义舵机初始位置
int pos1 = 82;              //底座
int pos2 = 90;              //大臂
int pos3 = 100;             //小臂
int pos4 = 70;              //爪子已经打开

void setup()
```

```
{
  myservoA.attach(2);    //定义舵机接口底座
  myservoB.attach(3);    //大臂
  myservoC.attach(4);    //小臂
  myservoD.attach(5);    //爪子
  myservoA.write(pos1);  //写入舵机初始值
  myservoB.write(pos2);
  myservoC.write(pos3);
  myservoD.write(pos4);
}

void loop()              //抓指定位置的物体
{
  servoCr(45);           //小臂上
  servoAr(62);           //向右旋转底座
  servoBr(35);           //大臂下
  servoCr(11);           //小臂上
  servoBr(10);           //大臂下
  servoDl(120);          //收爪子
  servoBl(50);           //大臂上
  servoAl(62);           //底座左转
  servoCl(65);           //小臂下
}

void servoAl(int a)      //底座左转
{
  for (i = 0; i <= a; i += 1)
  {
   pos1 += 1;
   myservoA.write(pos1);
   delay(20);
  }
}
void servoAr(int a)      //向右旋转底座
```

```
{
  for (i = 0; i <= a; i += 1)
  {
   pos1 -= 1;
   myservoA.write(pos1);
   delay(20);
  }
}
void servoBl(int a)        //大臂上
{
  for (i = 0; i <= a; i += 1)
  {
   pos2 += 1;
   myservoB.write(pos2);
   delay(20);
  }
}
void servoBr(int a)        //大臂下
{
  for (i = 0; i <= a; i += 1)
  {
   pos2 -= 1;
   myservoB.write(pos2);
   delay(20);
  }
}
void servoCl(int a)        //小臂下
{
  for (i = 0; i <= a; i += 1)
  {
   pos3 += 1;
   myservoC.write(pos3);
   delay(20);
  }
```

```
}
void servoCr(int a)        //小臂上
{
  for (i = 0; i <= a; i += 1)
  {
   pos3 -= 1;
   myservoC.write(pos3);
   delay(20);
  }
}
void servoDl(int a)        //收爪子
{
  for (i = 0; i <= a; i += 1)
  {
   pos4 += 1;
   myservoD.write(pos4);
   delay(20);
  }
}
void servoDr(int a)        //打开爪子
{
  for (i = 0; i <= a; i += 1)
  {
   pos4 -= 1;
   myservoD.write(pos4);
   delay(20);
  }
}
```