DFRduino 2WDMiniQ Users Manual

Dreamfactory 梦工厂

DFRduino 2WDMiniQ Users Manual

TEL:（北京总部） 庄先生 010-82355005
（成都办事处）卫先生 15902808530
（上海办事处）桑先生 13774201234

DFRduino 2WDMiniQ

A. 注意！在没有认真阅读本说明之前，请勿给模块加电！错误接线将导致模块永久性损坏或烧毁微控制器。

B. 注意！请认真查看引脚功能说明，正确接线！请勿将电源反接，否则将导致模块永久性损坏。

C. 注意！请勿使用超出额定电压的电源！保证电源的稳定，如果出现高压脉冲，可能会导致微控制器永久性损坏。

D. 注意！本产品无防水防潮功能，请在干燥环境下保存或使用！不可将重物堆积在上面。

介绍

2WDMiniQ 机器人是一款只有手掌大小的小型轮式（车形）机器人。该款机器人具有外观小巧、功能强大等特点。非常适合初学者作为学习和娱乐的工具。

在四节镍氢电池供电的情况下，速度可达 79cm/s。整个结构包括四个直流电机，采用的是 mini 型金属齿轮电机；5 个红外一体反射式光电传感器；两个红外发射管、一个红外接收头；两个光敏电阻；5 个按键；一个蜂鸣器等等。有三种下载程序方式：无线下载（arduino 环境支持）、UART 口下载（arduino 环境支持）、ICSP 口下载（avr gcc 环境支持）。

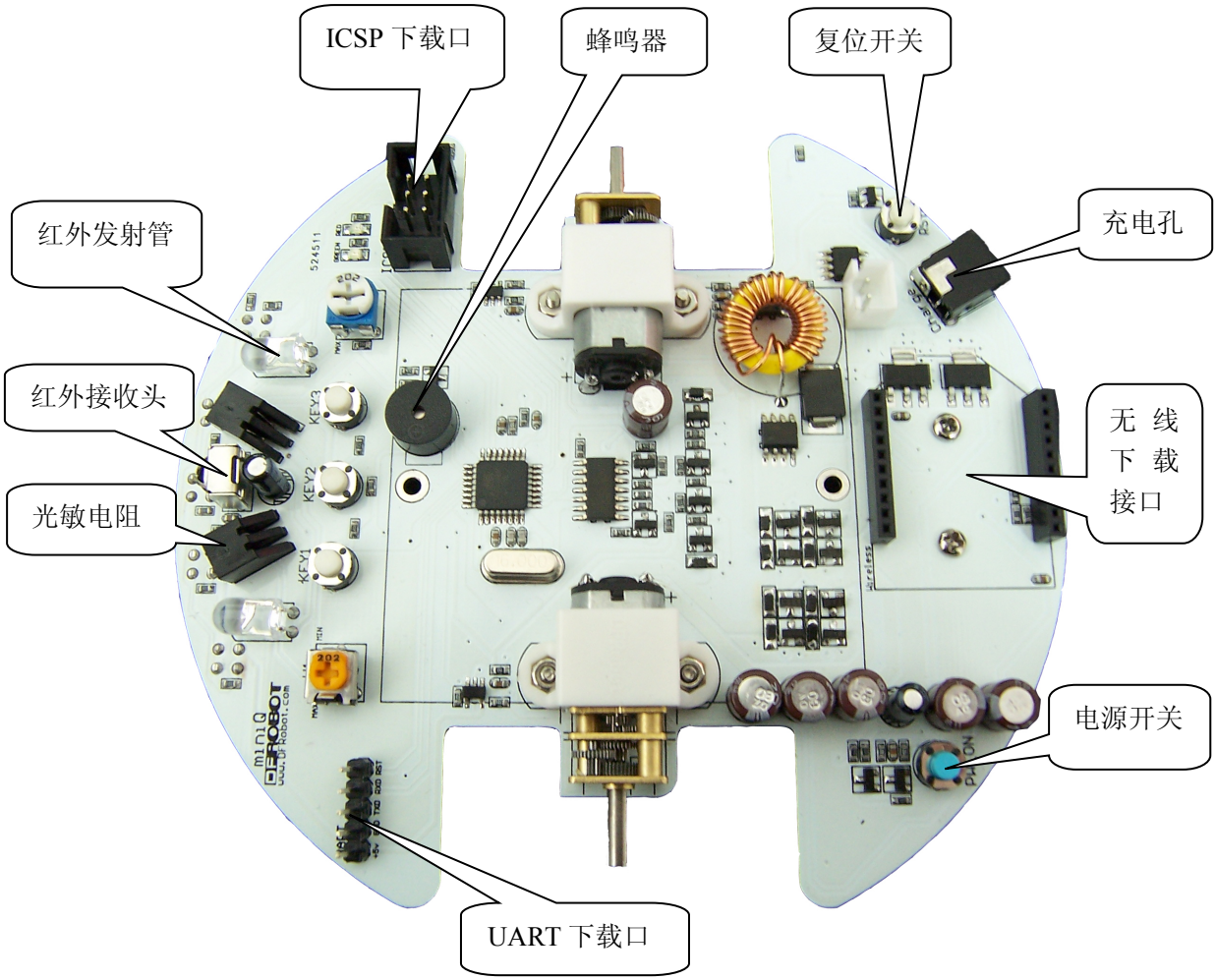
2WDMiniQ 是基于 Atmel ATmega328p 微处理器。ATmega328p 微处理器具有 32kB 系统可编程的 flash，1KB 的 EEPROM,2KB 的片内 SRAM.使用 ATmega328p 制作的 2WDMiniQ 教育机器人兼容 arduino 平台。初学者可以在 avr 开发环境下进行学习，也可以在 arduino 平台进行学习。

功能描述

1. 可以唱歌。
2. 可以在运行期间，实现避障功能。
3. 可以参加寻线比赛。
4. 可以跟随光源转动，类似趋光虫。
5. 支持遥控器控制 2WDMiniQ 转动。
6. 可以通过按键来切换各个模式。
7. 支持低电压检测。
8. 支持无线下载程序
9. 还具备简单的机器人行为。

DFRduino 2WDMiniQ 的使用篇

1. 认识 DFRduino 2WDMiniQ



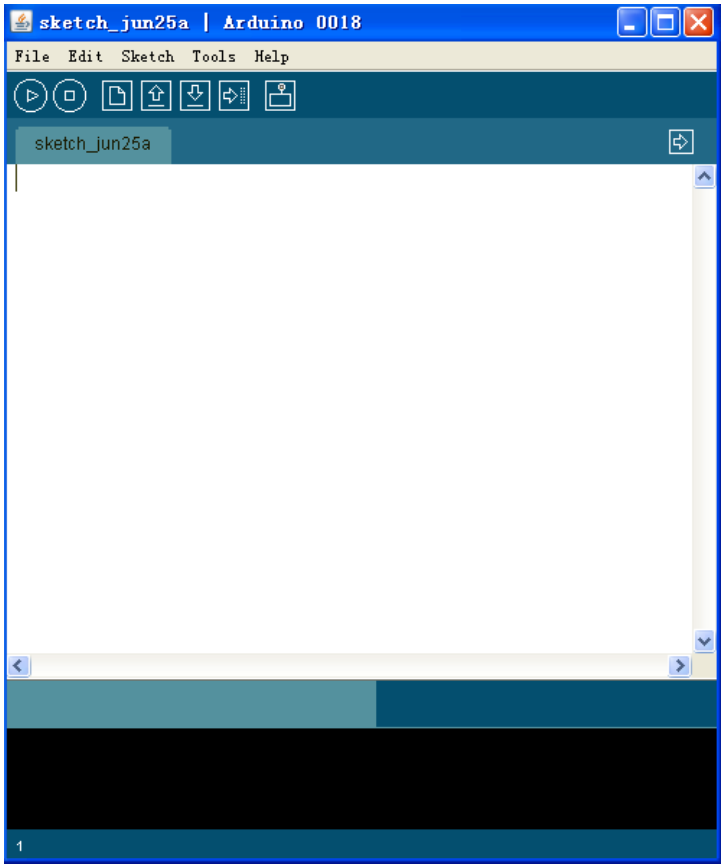
2. 集成开发环境介绍

2WDMiniQ 可直接使用 Arduino 0012 以上版本的集成开发环境，以下简称为 IDE，可到这里下载 http://arduino.cc/en/Main/SoftwareArduino 语言的语法请参考官方网站：http://arduino.cc/en/Reference/HomePage。IDE 的使用请参考：http://www.roboticfan.com/blog/user_2005/1229/archives/2008/20084292032.shtml

3、IDE 使用介绍

• 打开软件

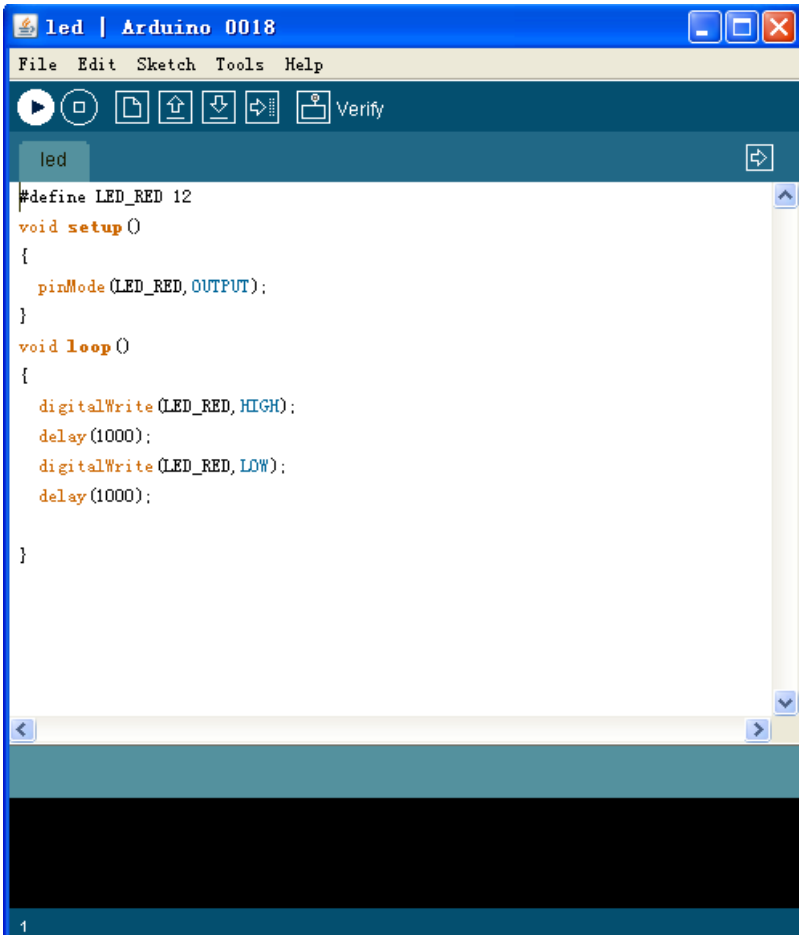
Arduino 无需安装，只要打开 arduino 软件包所在的文件夹，双击 arduino.exe 图标即可打开 arduino 编程环境。双击打开会出现如下界面：



Arduino 开发编译环境很简洁，各个功能键功能描述如下：

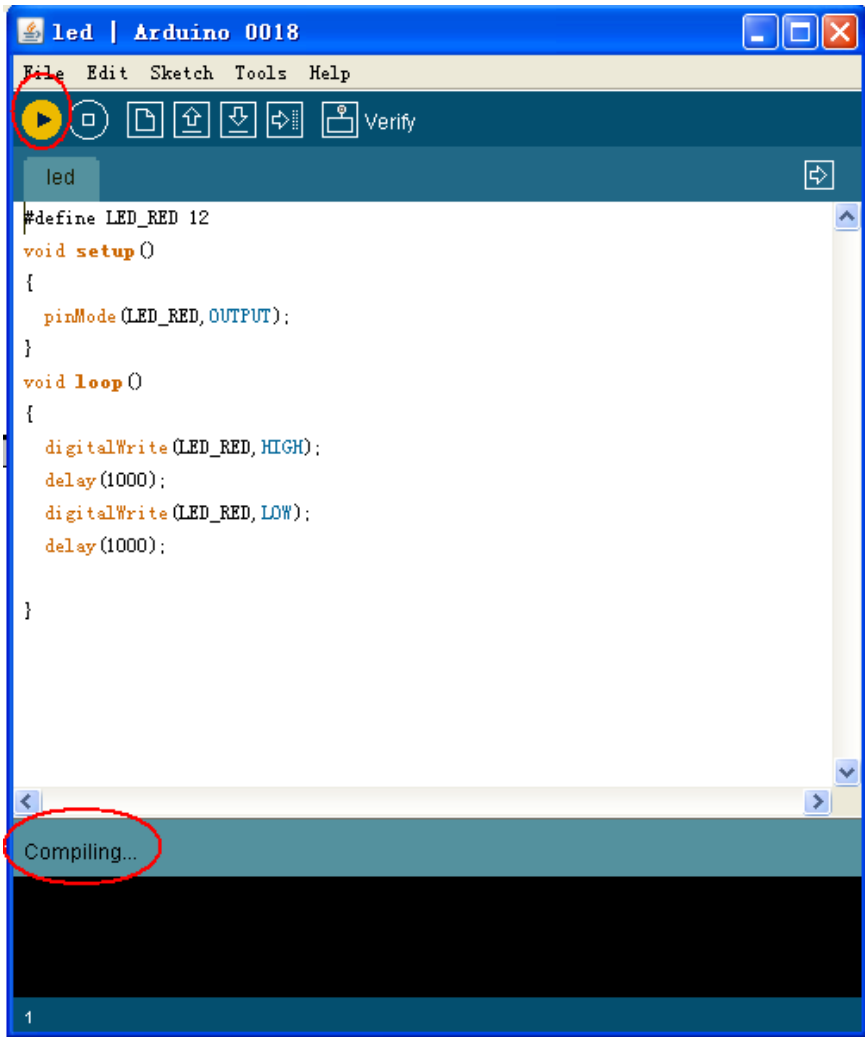
• 编写程序

打开软件后，我们就可以在窗口的空白处编写程序了。如下图所示：

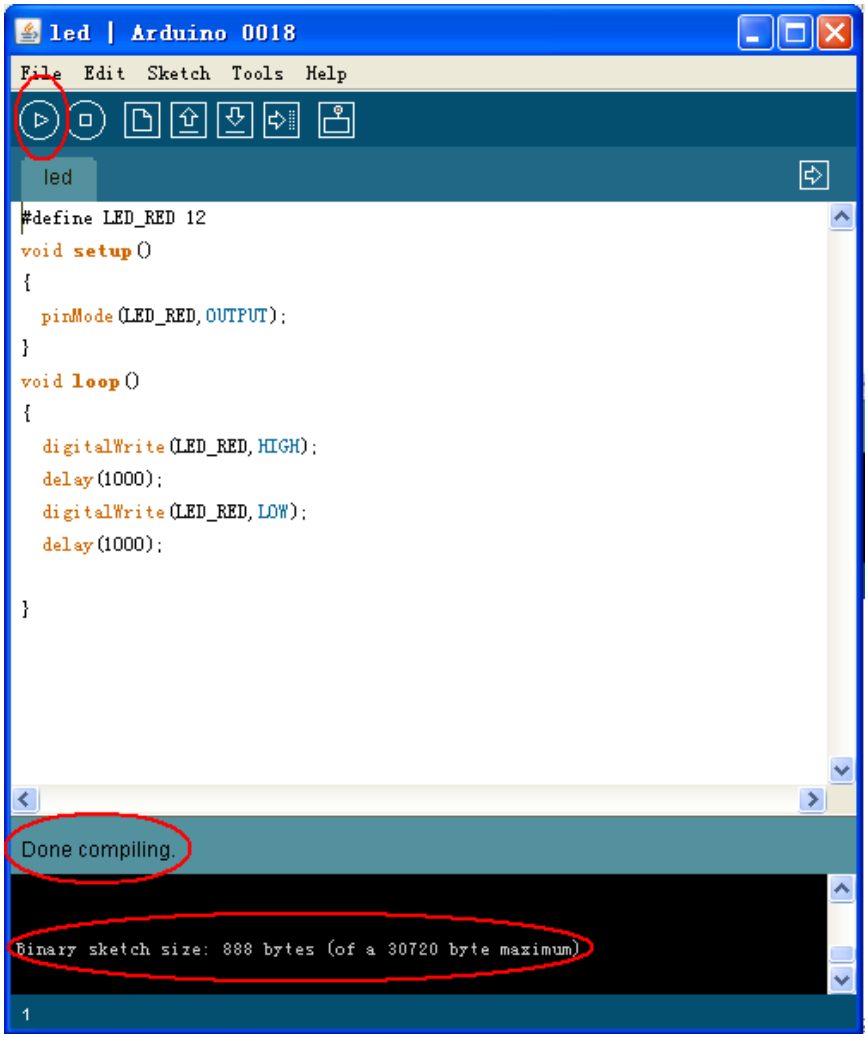


• 编译程序

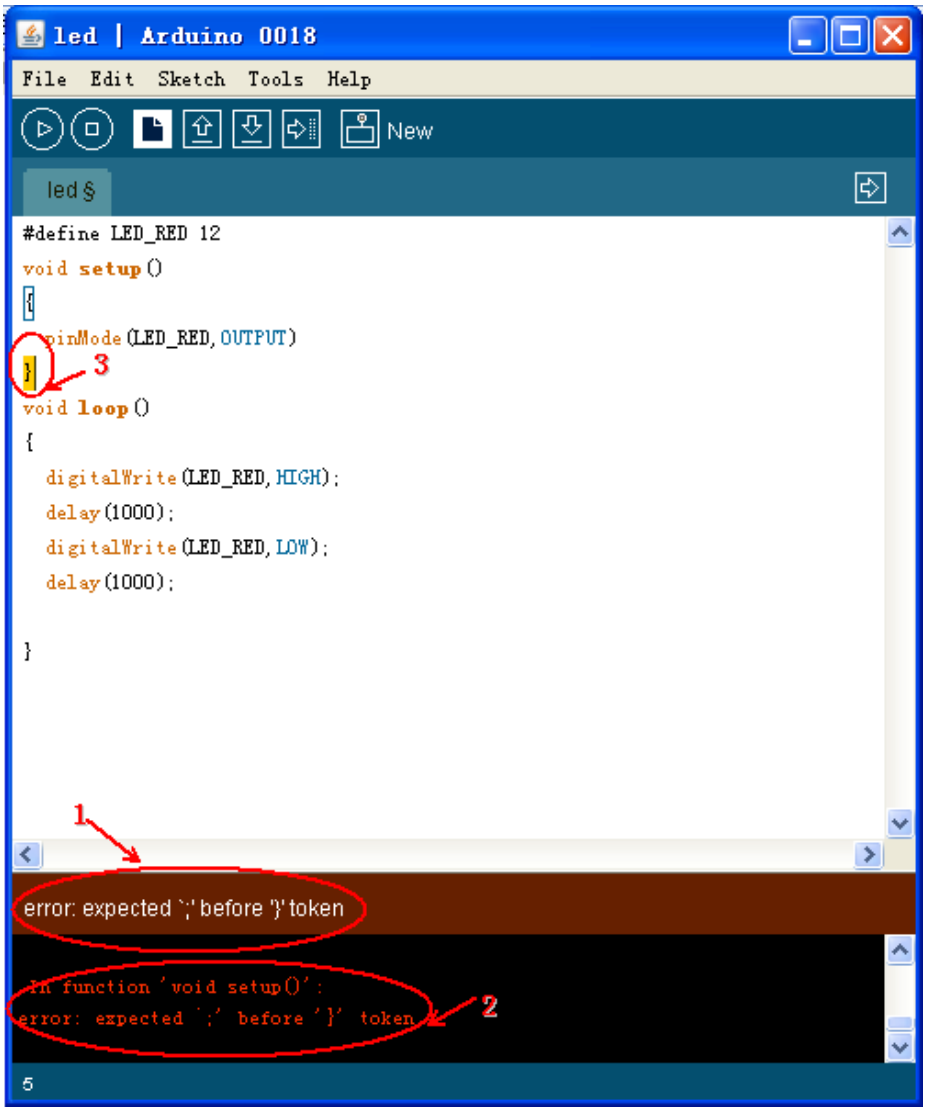
点击编译按钮，这时编译按钮会变成黄色，下面出现英文 compiling.....，这表示软件正在对你所写的程序进行编译，如下图所示：



等待一会，会看到编译按钮恢复原来的状态，下面出现 Done compiling，最下面一段文字说明编写的程序共有 888 字节数。这表明，程序编译成功，并且没有语法上的错误。如下图所示：



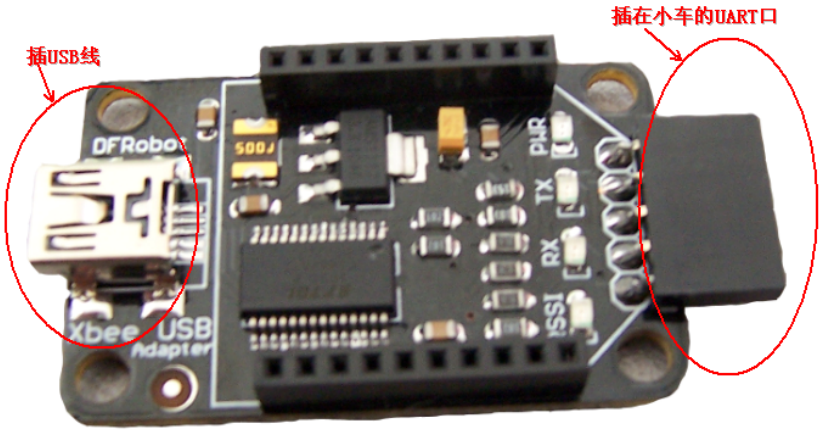
下面看看程序有语法错误时会出现什么状态，将程序中 pinMode（LED_RED,OUTPUT）后面的分号去掉，点击编译按钮，编译完成后会出现如下图所示状态：



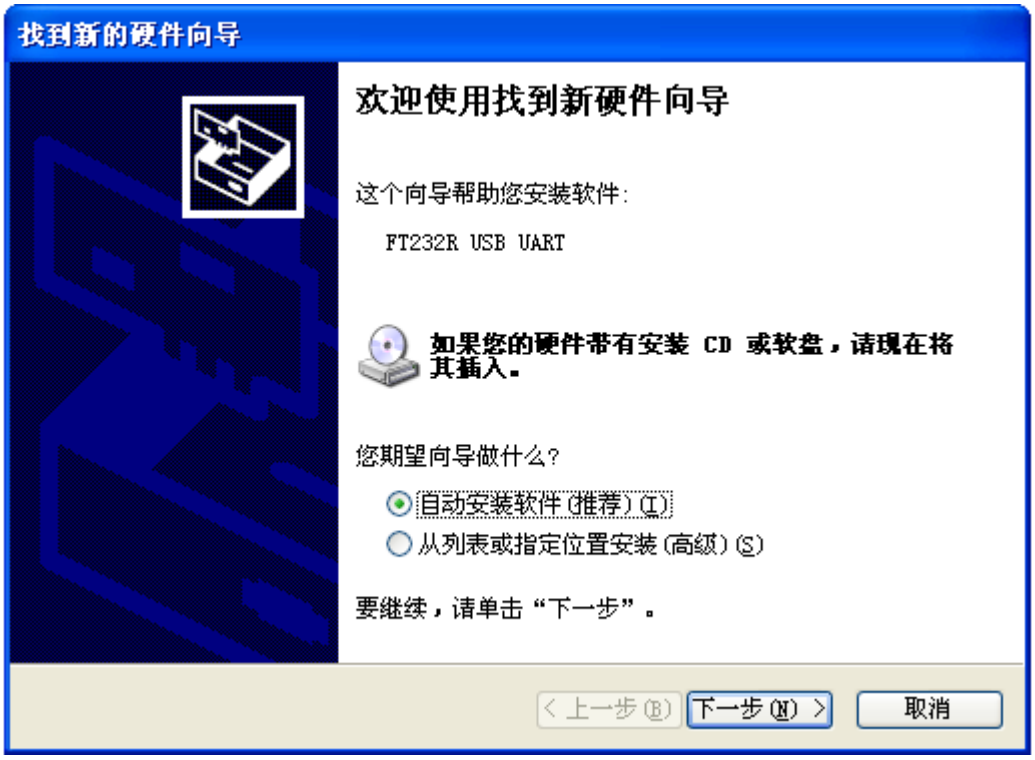
1 处告诉我们是因为在“}”附近缺少分号而出现的错误。2 处用文字告诉我们错误是出现在 void setup（）的一个“}”附近。3 处用黄颜色将“}”覆盖，表示错误就在这附近。从程序中看到错误确实在大括号附近，将分号添上后就会编译成功。以后编写程序出现错误时，就可以通过看下面信息栏里的提示调试程序。

• 安装驱动

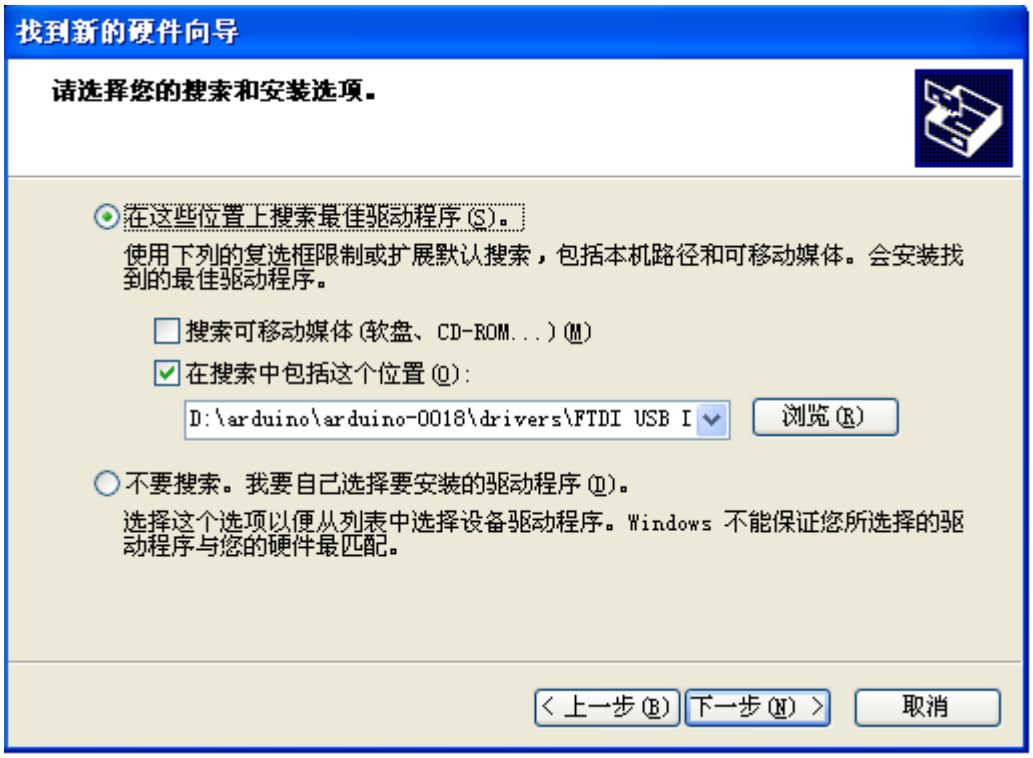
在第一次使用 arduino 软件下载程序，下载程序之前需要安装 USB 驱动。首先将 USB 线 一端插在电脑上，另一端插在 xbee usb 适配器上，最红将 xbee usb 适配器插在小车上的 UART 口（插的时候注意顺序，让 xbee usb 上的 5v 和小车上 UART 口的 5v 对应）。如下图所示：



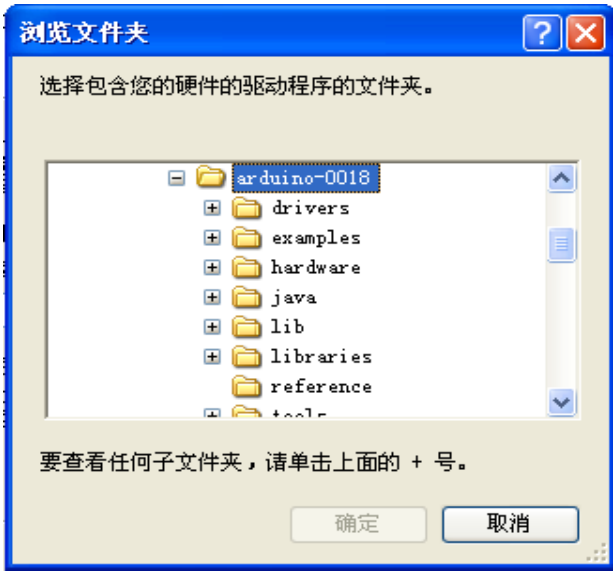
插好后电脑会弹出一个对话框如下图：



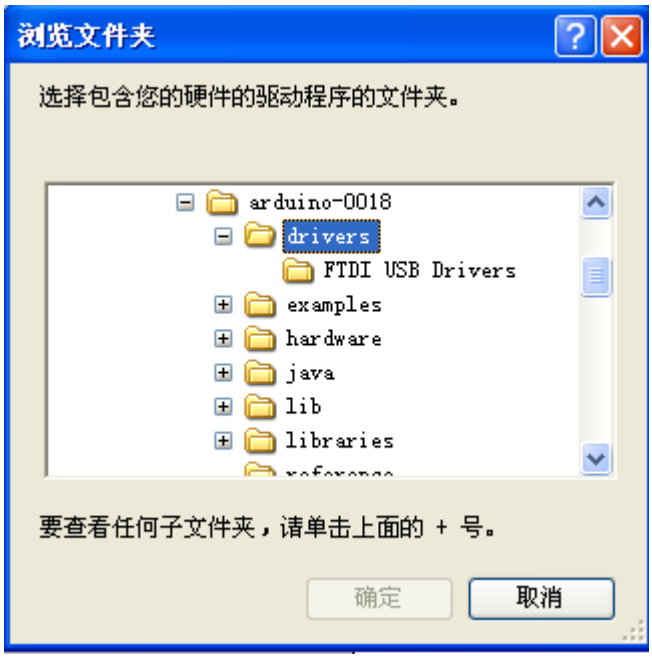
选择从列表或指定位置安装，点击下一步，出现如下图：



然后点击浏览，在出现的浏览文件夹对话框中点击光盘，在光盘下找到 arduino0018 文件夹，
点击打开，会看见有 drivers 文件夹如下图所示：



点击 drivers 文件夹，会看到 FTDI USB Drivers 文件夹，如图：



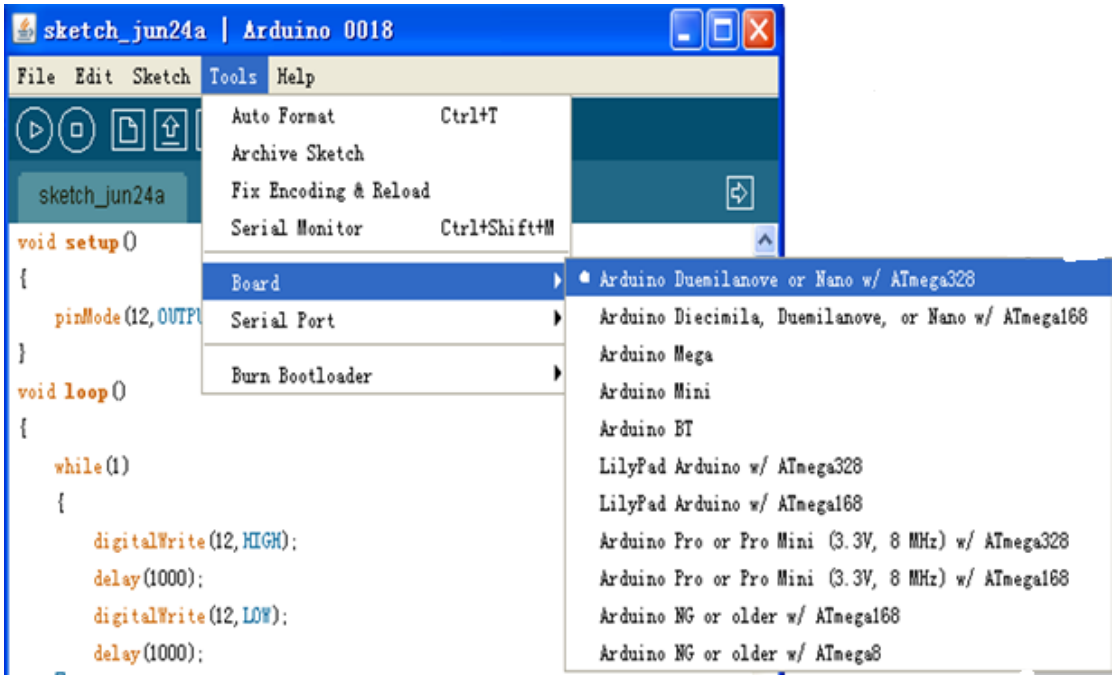
然后点击这个文件夹，接着点击确定，点击下一步，会出现如图对话框：

这时我们只要等待即可，稍后会出现如下图对话框：

点击完成，这样驱动就安装好了，下次再将数据线插到电脑就不会出现安装驱动对话框了，插上数据线就可以下载程序了。

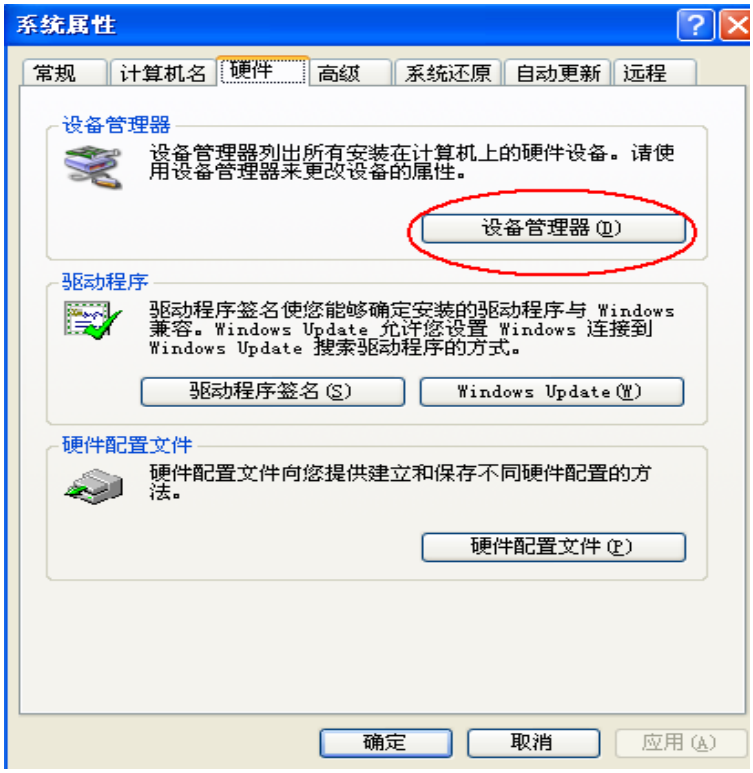
• 下载程序

下载程序前先将板子型号和 com 口选好。先点击 Tools->Board 选择开发板型号，如下图：

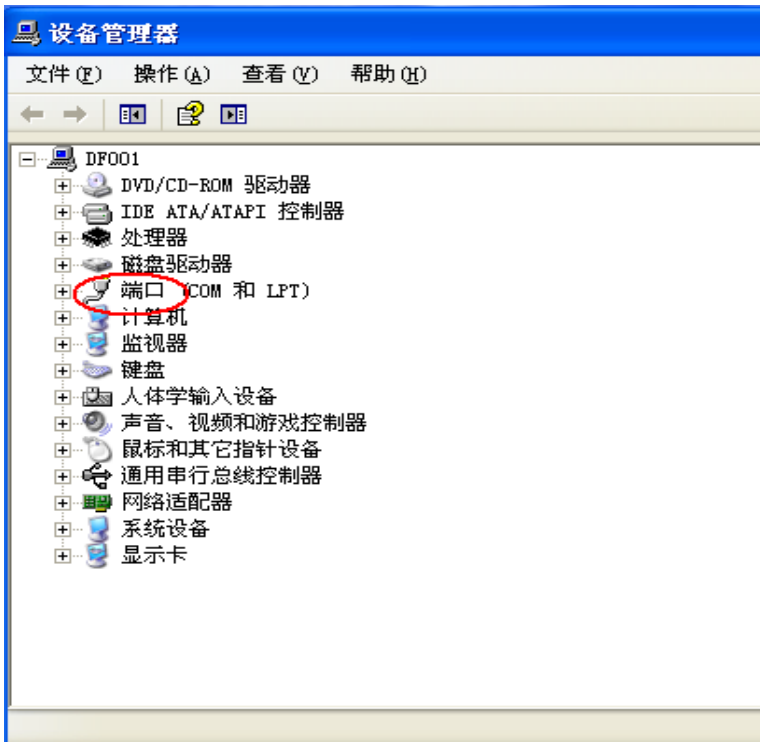


这里我们使用的是 ATmega328 控制板，所以点击第一个即可。接下来选择串口，首先看一下我们的串口是 COM 几，右键点击我的电脑的图标，选择属性，会出现如下对话

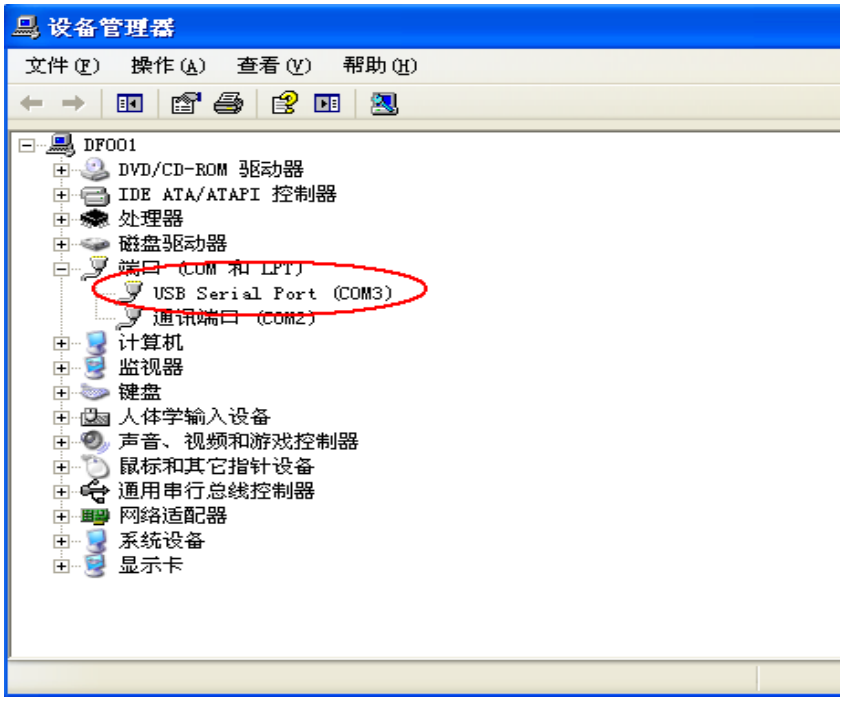
点击硬件，出现如图对话框：



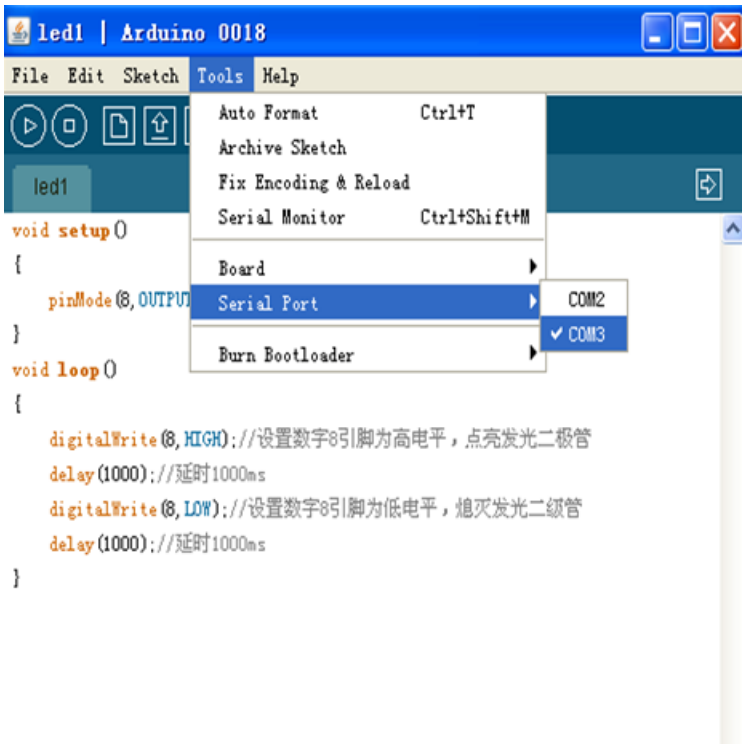
点击设备管理器，出现下图：


设备管理器
文件(F) 操作(A) 查看(V) 帮助(H)
DF001
DVD/CD-ROM 驱动器
IDE ATA/ATAPI 控制器
处理器
磁盘驱动器
端口 (COM 和 LPT)
计算机
监视器
键盘
人体学输入设备
声音、视频和游戏控制器
鼠标和其它指针设备
通用串行总线控制器
网络适配器
系统设备
显示卡

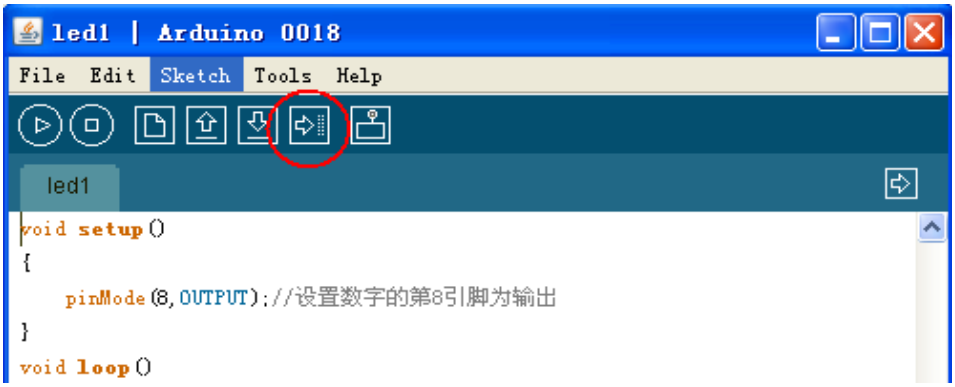
双击端口，出现下图：


设备管理器
文件(F) 操作(A) 查看(V) 帮助(H)
DF001
DVD/CD-ROM 驱动器
IDE ATA/ATAPI 控制器
处理器
磁盘驱动器
端口 (COM 和 LPT)
USB Serial Port (COM3)
通讯端口 (COM2)
计算机
监视器
键盘
人体学输入设备
声音、视频和游戏控制器
鼠标和其它指针设备
通用串行总线控制器
网络适配器
系统设备
显示卡

会看到有一个写着 USB Serial Port（COM3），COM3 这个就是我们的串口号。把这个号记住，关闭窗口，回到 arduino 软件窗口，点击 Serial Port，选择刚才记住的 COM 口号——COM3，如图：



这样板子型号和 COM 口就选好了。接下来点击 arduino 软件上的下载按钮，如图：



点击之后下载按钮变成橙色，软件下方出现 Uploading to I/O Board，如图所示：

程序下载完毕后，下载按钮恢复原来的颜色，下面出现 Done Uploading，如图：

如果没有显示 Done Uploading，而是出现了红色的字，表示下载失败，可以检查一下 USB 线是否连接好、电源开关是否打开、COM 口是否选对等等。如果出现上图，这样程序就下载成功了，如果你看到 2WDMiniQ 的 led 灯亮 1s、灭 1s 的在闪烁，恭喜你，你的 2WDMiniQ 开始工作啦！

• 下载程序方式介绍

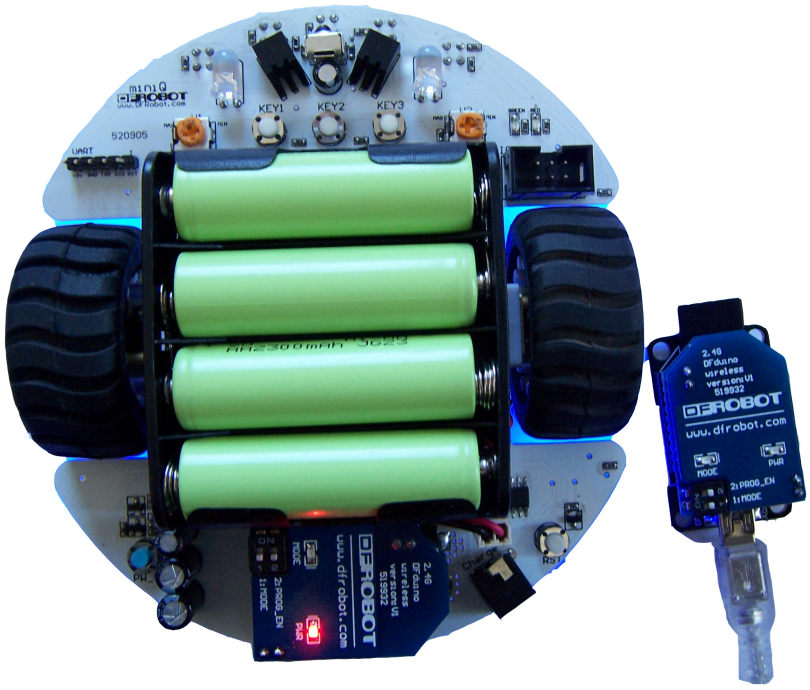
在 arduino 环境下有两种下载程序的方式：

1、利用 UART 口下载程序

首先将 USB 线 一端插在电脑上，另一端插在 xbee usb 适配器上，最红将 xbee usb 适配器插在小车上的 UART 口（插的时候注意顺序，让 xbee usb 上的 5v 和小车上 UART 口的 5v 对应）。如前面的图。

2、利用 wireless 无线下载口下载程序

准备一对 wireless 无线下载模块、一个 xbee usb 适配器、和一条迷你 USB 数据线，将 wireless 无线下载模块中的一个插在小车上的 wireless 口，另一个插在 xbee usb 适配器上，在 xbee usb 适配器上插上 USB 数据线，usb 数据线的另一端插在电脑上。即可下载程序。如下图所示：

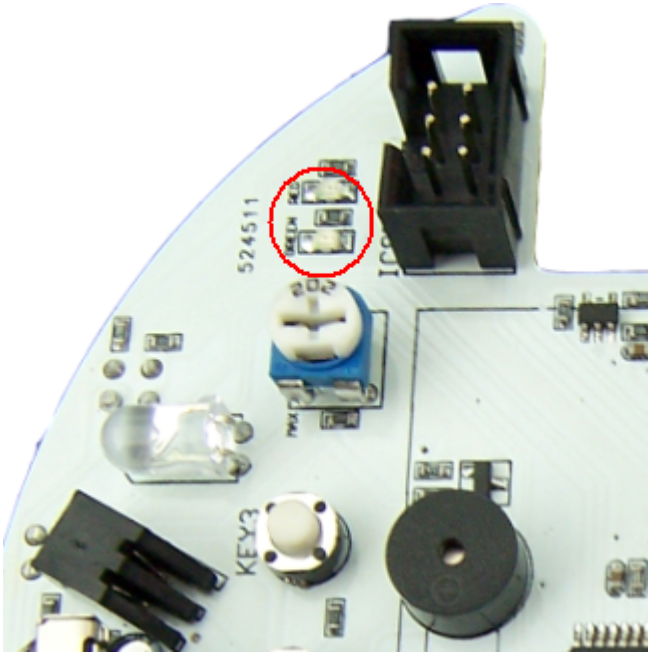


实验篇

1、LED 实验：

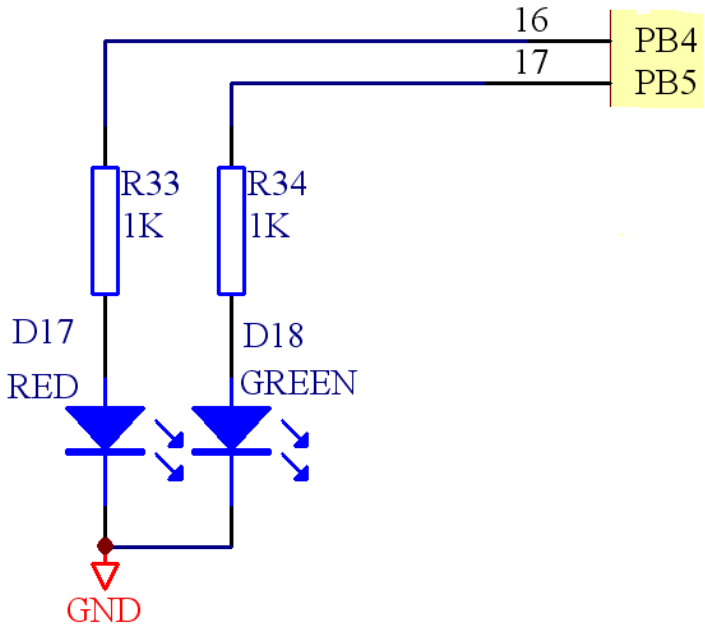
1)、介绍

2WDMiniQ 机器人共有六个贴片形式的 led 灯，其中小车低盘有四个蓝色的灯，小车上电后即亮。还有一个红色的灯和一个绿色的灯分别与 PB4、PB5 引脚相连，下载程序时作为状态指示灯用，还可以供大家编程使用。灯的位置如图：



2)、工作原理

下图是从 2WDMiniQ 的原理图中截取的红色和绿色 LED 灯的原理图连接部分，图中 R33、R34 为限流电阻。如果流过 led 的电流过大，led 会被烧毁的。将 PB4 或 PB5 引脚置 1 时，相应的发光二极管导通，灯亮。将 PB4 或 PB5 引脚清零时，相应的发光二极管截止，灯灭。



3)、演示代码：

```
#define LED_RED 12//定义红色的 led 灯的引脚
void setup()
{
  pinMode(LED_RED,OUTPUT);//设置 LED 灯引脚的模式为输出
}
void loop()
{
  digitalWrite(LED_RED,HIGH);//LED 灯引脚置高，点亮 LED 灯
  delay(1000);//延时 1s
  digitalWrite(LED_RED,LOW);//LED 灯引脚置低，熄灭 LED 灯
  delay(1000);//延时 1s
}
```

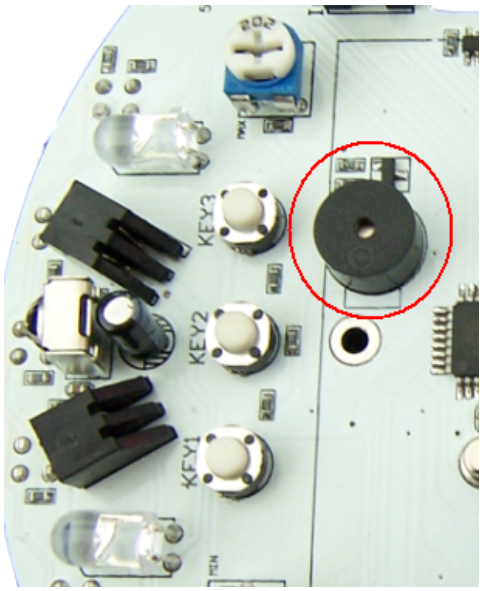
程序中用到的函数，大家可以到 arduino 教程里面找到，从那里可以了解到相应函数的功能和如何使用。因为 arduino 语言已经把 AVR 单片机（微控制器）相关的一些参数设置都函数化，所以使用起来非常的方便。

4)、程序功能：红灯以间隔时间为 1s 亮灭闪烁。了解了本代码后，大家可以根据自己的思路对 led
灯编程。

2、蜂鸣器实验：

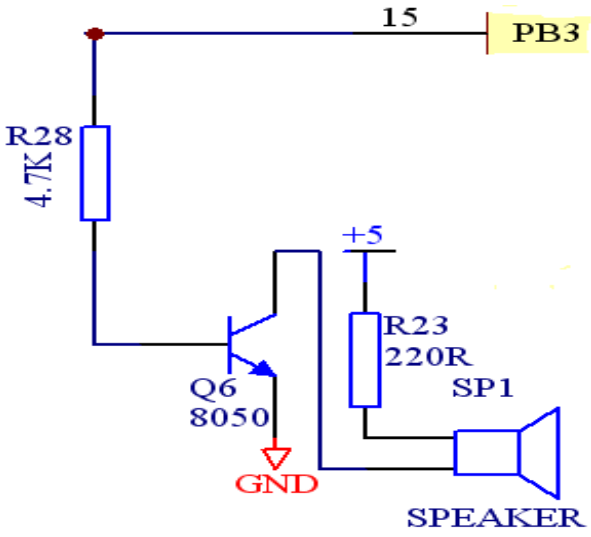
1)、介绍

2WDMiniQ 具有一个蜂鸣器，它与 atmega328p 的 PB3 引脚相连。具体位置如图：



2)、工作原理

蜂鸣器是用一个 NPN 三极管驱动的， 当 PB3 引脚置为 1 时三极管导通蜂鸣器响，当 PB3 引脚置为 0 时三极管截止蜂鸣器不响。图中 R28 为基极限流电阻，R23 起分压作用。



3)、演示代码：

```
#define BUZZER 11//定义蜂鸣器的引脚
void setup()
{
  pinMode(BUZZER,OUTPUT);//设置蜂鸣器的引脚为输出模式
}
```

```
void loop()
{
  unsigned char i,j;//定义变量
  while(1)
  {
    //输出一个频率的声音
    for(i=0;i<80;i++)
    {
      digitalWrite(BUZZER,HIGH);//发声音
      delay(1);//延时 1ms
      digitalWrite(BUZZER,LOW);//不发声音
      delay(1);//延时 1ms
    }
    //输出另一个频率的声音
    for(i=0;i<100;i++)
    {
      digitalWrite(BUZZER,HIGH);//发声音
      delay(2);//延时 2ms
      digitalWrite(BUZZER,LOW);//不发声音
      delay(2);//延时 2ms
    }
  }
}
```

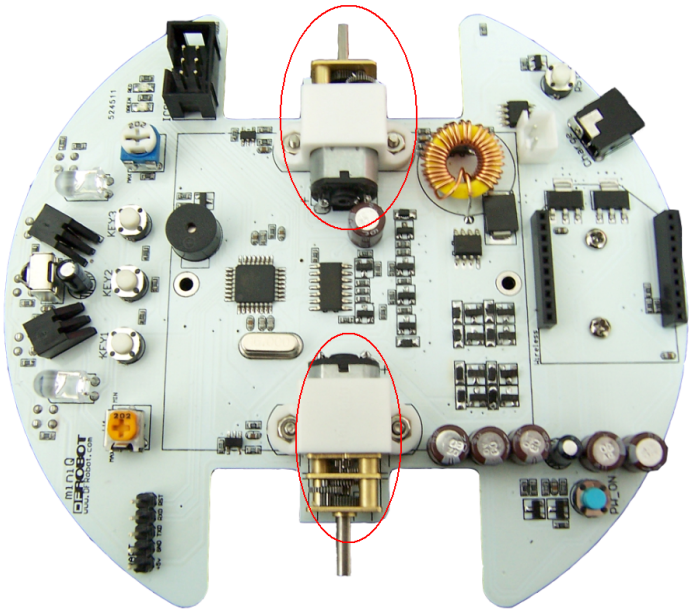
对比 avr gcc 语言的程序和 arduino 语言的程序，大家可以发现 arduino 语言可以对 AVR 芯片的引脚进行单独操作，而 avr gcc 语言只能对以 8 或者 7 个引脚为一组的端口进行操作。

4）、程序功能：将程序代码下载到 2WDMiniQ 后，类似警笛声音。在此程序的基础上，大家可以修改延时时间，调试出不同种的声音。

3、2WDMiniQ 转动实验

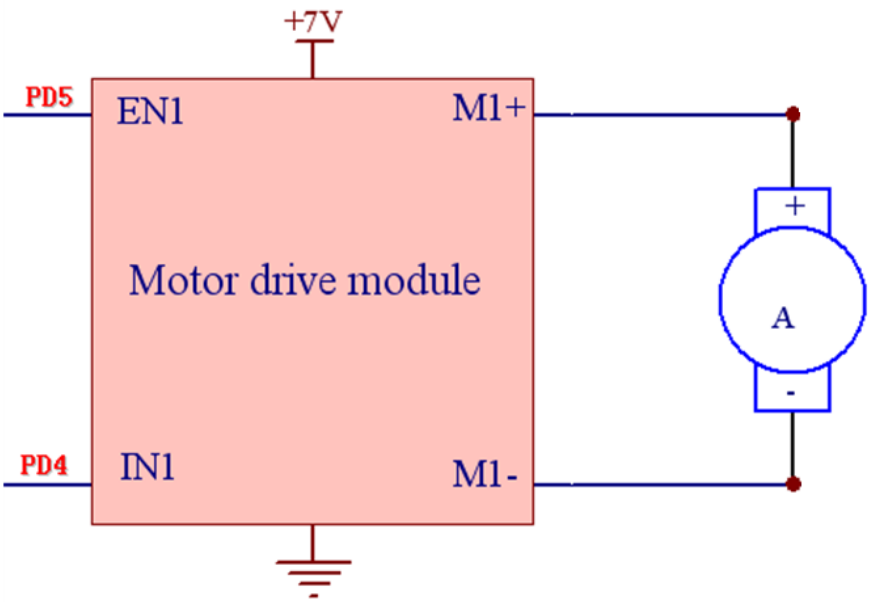
1)、介绍

体验了前两个实验的乐趣之后，让我们继续了解 2WDMiniQ，来控制 2WDMiniQ 转动。2WDMiniQ 的两个直流电机是分为两组的，左侧为一组、右侧为另一组。每组有两个信号控制，一个使能信号、一个方向信号。具体位置如图：



2)、原理

两组电机的控制信号线分别为 PD4(方向信号)、PD7(方向信号)、PD5(使能信号)、PD6(使能信号)。从下面的原理图中可以看到左侧这组电机的连接，EN1 为使能信号，置 1 则使能电机转动、清零则电机不动；IN1 为方向信号，置 1 则向前转、清零则向后转。



2WDMiniQ 使用的是 pwm 脉冲控制电机速度的方法脉冲宽度调制（PWM）是英文“Pulse Width Modulation”的缩写，简称脉宽调制。它是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用于测量，通信，功率控制与变换等许多领域。 Pwm 调速是通过调节脉冲宽度的占空比来调节的，即在固定周期 T 内，高电平和低电平各占多少时间。如果高电平占 1/4T，则低电平占 3/4T。占空比=高电平所占时间/整个周期=1：4。所以在一个周期内，高电平所占时间越长（即电机通电时间越长），加在电机上的平均电压值越大，速度越快。根据以上原理只要有占空比可
调的脉冲波加在使能端，就可以调节电机的速度了。由于 2WDMiniQ 的微处理器采用的是资源丰富的 atmega328p，它的内部定时器即可产生 PWM 脉冲，可以方便的供大家使用。具体怎么设置定时器产生 pwm 脉冲可参考 atmega328p 手册。如果是 avr gcc 语言就需用到定时器来产生 PWM 脉冲，而 arduino 语言中有一个函数：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。可用于电机 PWM 调速。

3)、演示代码：

```
#define EN1 6//右侧电机使能引脚
#define IN1 7//右侧电机方向引脚
#define EN2 5//左侧电机使能引脚
#define IN2 4//左侧电机方向引脚

#define FORW 1//前进
#define BACK 0//后退
void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)//控制电机转动
{
  //////////M1////////////////////////
  if(M1_DIR==FORW)//M1 电机的方向
    digitalWrite(IN1,FORW); //设置方向向前
  else
    digitalWrite(IN1,BACK);//设置方向向后
  if(M1_EN==0)//M1 电机的速度
    analogWrite(EN1,LOW);//置低，停止
  else
    analogWrite(EN1,M1_EN);//设置相应的数值

  //////////M2////////////////////////
  if(M2_DIR==FORW)//M2 电机的方向
    digitalWrite(IN2,FORW);//设置方向向前
  else
    digitalWrite(IN2,BACK);//设置方向向后
  if(M2_EN==0)//M2 电机的速度
    analogWrite(EN2,LOW);//置低，停止
```

```
  else
    analogWrite(EN2,M2_EN);//设置为给定的值
}
void setup()
{
  unsigned char i;
  for(i=4;i<=7;i++)//设置控制两组电机的四个引脚为输出
    pinMode(i,OUTPUT);
}
void loop()
{
  Motor_Control(FORW,100,FORW,100);//前进
  delay(500);//500ms
  Motor_Control(FORW,0,FORW,0);//停止
  delay(500);//500ms
  Motor_Control(BACK,150,BACK,150);//后退
  delay(500);//500ms
   Motor_Control(FORW,0,FORW,0);//停止
  delay(500);//500ms
}
```

4)、程序功能：2WDMiniQ 前进 500ms——>停止 500ms——>后退 500ms——>停止 500ms。大家可以通过改变程序来改变 2WDMiniQ 的速度和方向，从而灵活的控制 2WDMiniQ 运动。

4、蜂鸣器唱歌实验

1)、原理

每个音符都有一个对应的频率值 f，频率值的倒数就是周期值 T。让蜂鸣器在这个 T 周期内，半个周期的时间给蜂鸣器高电平，发出声音；半个周期时间给蜂鸣器低电平，不发出声音。蜂鸣器按照这个周期动作，就会听到跟这个周期相对应的音符。而在实际实验中，定时是由定时器来完成的，所以得将每个音符对应的半个周期的计时初值计算出来。这样每首歌的乐谱就可以翻译成对应的计时值数组。每个计时值后面的数值是让这个音符响多长时间。最后，在程序中按照翻译过来的数组中的数值的先后顺序播放，就可以听到我们想要的音乐了。

1)、演示代码：

```
//        音符        计时值       频率值 hz
#define DO_L         E2           //262
#define DOA_L        E4           //277
#define RE_L         E5           //294
```

```
#define REA_L       E7          //311
#define MI_L        E8          //330
#define FA_L        EA          //349
#define FAA_L       EB          //370
#define SO_L        EC          //392
#define SOA_L       ED          //415
#define LA_L        EE          //440
#define LAA_L       EF          //466
#define TI_L        F0          //494
#define DO          F1          //523
#define DOA         F2          //554
#define RE          F3          //587

#define REA         F3          //622
#define MI          F4          //659
#define FA          F5          //698
#define FAA         F5          //740
#define SO          F6          //784
#define SOA         F7          //831
#define LA          F7          //880
#define LAA         F8          //932
#define TI          F8          //988
#define DO_H        F9          //1046
#define DOA_H       F9          //1109
#define RE_H        F9          //1175
#define REA_H       FA          //1245
#define MI_H        FA          //1318
#define FA_H        FA          //1397
#define FAA_H       FB          //1480
#define SO_H        FB          //1568
#define SOA_H       FB          //1661
#define LA_H        FC          //1760
#define LAA_H       FC          //1865
#define TI_H        FC          //1976
#define ZERO        0           //休止符

#define BUZZER 11//定义蜂鸣器的引脚

int     initial_value;  //发出每个音符时定时器的初值
char    time;           //每个音符演奏的时间
char    ptr = 0x00;     //指向 music 数组的指针
char    flag=0;
int     music[] = {
```

```
  0XF1, 2, 0XF3, 2, 0XF4, 2, 0XF1, 1, 0, 1,     //两只老虎
  0XF1, 2, 0XF3, 2, 0XF4, 2, 0XF1, 1, 0, 1,
  0XF4, 2, 0XF5, 2, 0XF6, 2, 0, 2,
  0XF4, 2, 0XF5, 2, 0XF6, 2, 0, 2,
  0XF6, 1, 0XF7, 1, 0XF6, 1, 0XF5, 1, 0XF4, 2, 0XF1, 2,
  0XF6, 1, 0XF7, 1, 0XF6, 1, 0XF5, 1, 0XF4, 2, 0XF1, 2,
  0XF3, 2, 0XEC, 2, 0XF1, 2, 0, 2,
  0XF3, 2, 0XEC, 2, 0XF1, 2, 0, 2,0xff};

void timer2_init(void)//定时器 2 初始化
{
  TCCR2A = 0X00;
  TCCR2B = 0X07; //时钟源 1024 分频
  TCNT2 = initial_value;

  TIMSK2 = 0X01; //允许中断
}
ISR(TIMER2_OVF_vect)//定时器 2 中断
{
  TCNT2 = initial_value;//给定时器计时初值
  flag=~flag;
  if(flag)
  {
     digitalWrite(BUZZER,HIGH);//置高，蜂鸣器响
  }
  else
  {
    digitalWrite(BUZZER,LOW);//置低，蜂鸣器不响
  }
}
void play_music(void)//播放音乐
{
  if (music[ptr] != 0xFF && music[ptr] != 0x00) //判断是否是正常音符
  {
    TCCR2B = 0X07;//定时器开始工作
    initial_value = music[ptr]; //取设置定时器初值
    time = music[ptr + 1]; //取发声时间
    delay(time*200);//延时
    ptr += 2;//指向下一个音符
  }
  else if (music[ptr] == 0x00) //判断是否是休止符
  {
    time = music[ptr + 1];//取发声时间
```

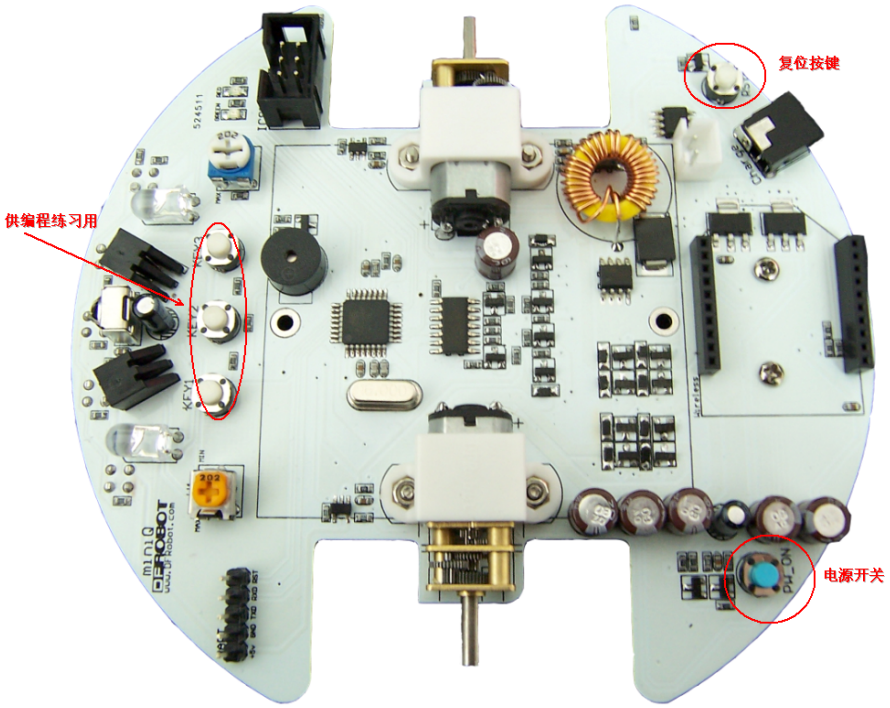
```
      delay(time*200);//延时
      ptr += 2;//指向下一个音符
   }
   else//是结束符
   {
      TCCR2B = 0X00;//定时器停止工作
      digitalWrite(BUZZER,LOW);//置低，蜂鸣器不响
      delay(time*200);//延时
      ptr = 0;//清零，便于重新开始
   }
}
void setup()
{
    pinMode(BUZZER,OUTPUT);//设置连接蜂鸣器的引脚为输出模式
    timer2_init();//定时器初始化
    sei();//全局中断开启

}
void loop(void)
{
   while (1)
   {
      play_music();//播放音乐
   }

}
```

3)、程序功能：播放两只老虎乐曲。快快行动起来，把自己喜欢的音乐编辑出来，让 2WDMiniQ 为你
唱你喜欢的歌。

5、按键实验：

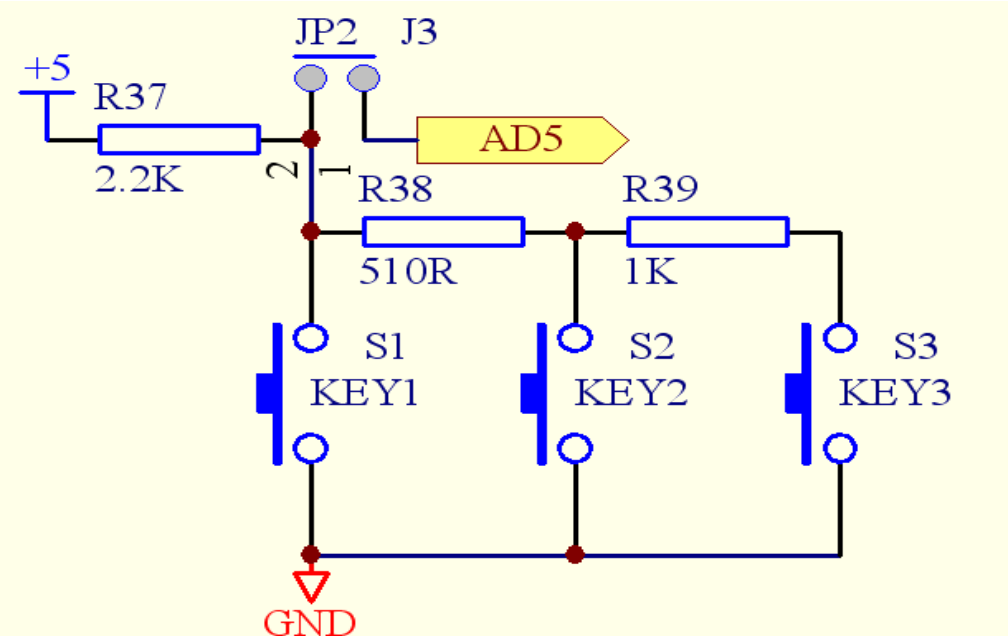
1)、介绍

2WDMiniQ 共有 5 个按键，其中有一个是电源开关按键，还有一个复位键，剩下的三个按键可供我们学习编程使用。具体位置如图：



2)、原理

这三个按键的原理图连接如下图：



用万用表测量可知，当没有按键按下时 AD5 端口电压为 5.04V 左右；当 key1 按下时 AD5

端口电压为 0.00V 左右；当 key2 按下时 AD5 端口电压为 0.95V 左右；当 key3 按下时 AD5 端口电压为 2.06V 左右。做实验时大家可以实际动手测量一下。

由于电压值为模拟值，所以接到 atmega328p 后得经过 A/D(模拟/数字)转换，atmega328p 内部资源丰富已经自带有 A/D 转换功能，所以编程时写好 A/D 转换功能即可得到 AD5 端口电压值，根据测得的 AD5 端口电压值来判断是否有按键按下、是哪一个按键按下的。实际编程时，按照事先测好的电压值来判断是哪一个按键按下可能不准，因为在程序中对接收到的经过 A/D 转换的数字值进行了计算转换，转换成了模拟值，所以得到的模拟值可能会有误差，大家可以在程序中不断修改这几个值，直至按键好使。

3)、演示代码：

```
#define BUZZER 11//设置控制蜂鸣器数字 IO 脚
#define LED_RED 12//设置控制红色 LED 灯数字 IO 脚
#define LED_GREEN 13//设置控制绿色 LED 灯数字 IO 脚
#define Vr     5//参考电压值

float data=0xff;/存储模拟 电压值,初值设置为不为零的数就可以。如初值设置为 0，会影响判断按键 1 是否按下。
void buzzer(void)//蜂鸣器发出声音
{
  digitalWrite(BUZZER,HIGH);//置高，蜂鸣器响
  delay(1);
  digitalWrite(BUZZER,LOW);//置低，蜂鸣器不响
  delay(10);
}
void Read_Value(void)//读取电压值
{
  data=analogRead(5);//读取模拟 5 口电压值
  data = ((data * Vr) / 1024);//将数字值转换成模拟值
}
void key_scan(void)//按键扫描
{
  if(data>4.50&&data<6.00)//没有按键按下
    return;//返回
  else
  {
    if(data>=0.00&&data<0.50)//按键 1 按下
    {
      delay(10);//延时消抖
      if(data>=0.00&&data<0.50)//按键 1 确实按下
      {
        buzzer();//蜂鸣器响
```

```
            while(data>=0.00&&data<0.50)
            {
              Read_Value();//读取电压值
              digitalWrite(LED_RED,HIGH);//红灯亮
            }
          }
        }
        else if(data>=0.50&&data<1.5)
        {
          delay(10);
          if(data>=0.50&&data<1.5)
          {
            buzzer();
            while(data>=0.50&&data<1.5)
            {
              Read_Value();//读取电压值
              digitalWrite(LED_GREEN,HIGH);//绿灯亮
            }
          }

        }
        else if(data>=1.5&&data<2.5)
        {
          delay(10);
          if(data>=1.5&&data<2.5)
          {
            buzzer();
            while(data>=1.5&&data<2.5)
            {
              Read_Value();//读取电压值
              digitalWrite(LED_RED,LOW);//红灯灭
              digitalWrite(LED_GREEN,LOW);//绿灯灭
            }
          }
        }
    }
}
void setup()
{
  pinMode(BUZZER,OUTPUT);//蜂鸣器的端口为输出模式
  pinMode(LED_RED,OUTPUT);//红灯的端口为输出模式
  pinMode(LED_GREEN,OUTPUT);//绿灯的端口为输出模式
}
```

```
void loop()
{
  Read_Value();//读取电压值
  key_scan();//执行按键扫描函数
}
```

4)、程序功能：按键 1 按下时，红灯亮、绿灯灭；按键 2 按下时，绿灯亮、红灯灭；按键 3 按下时，红灯和绿灯均灭。这节实验中的 AD 转换很重要噢，后面的寻线、寻光实验都要用到 AD 转换的，熟练掌握，便于后面的学习。

6、避障实验

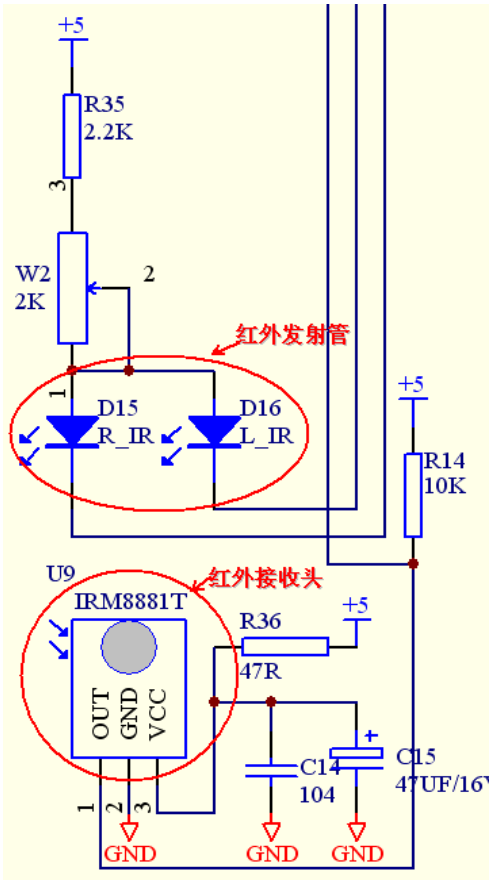
1)、介绍

2WDMiniQ 红外避障传感器是，用两个红外发射管作为发送器，接收部分采用一个一体红外接收 IC 实现。具体位置如图：

2)、原理

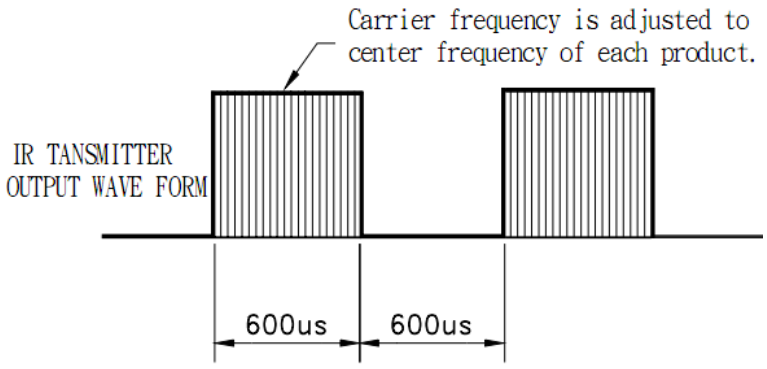
红外避障传感器主要用于检测道路周围的障碍物，从而达到小车自动躲避障碍物继续前行的功能。基于这种技术的传感器通常包括红外发送器和接收器两个部分。只要传感器能发射脉冲和接收脉冲即可，具体避障可在程序中实现。

通过红外发射管发射出一系列脉冲信号。当有障碍物时，脉冲经物体反射回红外线接收器。当没有障碍物时，则无脉冲反射回红外线接收器。所以，通过检测红外线接收器是否有脉冲返回，即可判断是否有障碍物。为了在有其他漫反射红外源存在的情况下改善传感器系统的工作性能，发送器输出通常使用载波调制，载波的频率范围大致为 38khz 至 56khz（噪声源不可能被调制到这种频率段）。具体原理图如下：



图中 W2 电位器的作用：通过调整电位器的电阻来调整发光二极管发光的强弱，进而达到调整检测障碍物的距离。

红外管发射波形：



3)、演示代码：

```
#define EN1 6//右侧电机使能引脚
#define IN1 7//右侧电机方向引脚
#define EN2 5//左侧电机使能引脚
#define IN2 4//左侧电机方向引脚

#define FORW 1//前进
#define BACK 0//后退

#define IR_IN   8//红外接收
#define L_IR 9//左红外发送
#define R_IR 10//右红外发送

int count;//对返回的脉冲进行计数

void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)//控制电机转
动
{
  //////////M1////////////////////////
  if(M1_DIR==FORW)//M1 电机方向
    digitalWrite(IN1,FORW);//设置方向向前
  else
    digitalWrite(IN1,BACK);//设置方向向后
  if(M1_EN==0)//M1 电机速度
    analogWrite(EN1,LOW);//置低，停止
  else
    analogWrite(EN1,M1_EN);//设置给定的数值

  //////////M2////////////////////
  if(M2_DIR==FORW)//M2 电机的方向
    digitalWrite(IN2,FORW);//设置方向向前
  else
    digitalWrite(IN2,BACK);//设置方向向后
  if(M2_EN==0) //M2 电机的速度
```

```
    analogWrite(EN2,LOW);//置低，停止
  else
    analogWrite(EN2,M2_EN); //设置给定的数值
}

void L_Send40KHZ(void)//左发射管发射频率为 40kHZ 脉冲
{
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(L_IR,LOW);//置低，导通
    delayMicroseconds(8);//延时
    digitalWrite(L_IR,HIGH);//置高，截止
    delayMicroseconds(8);//延时
  }
}
void R_Send40KHZ(void)//右发射管发射频率为 40kHZ 脉冲
{
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(R_IR,LOW);//置低，导通
    delayMicroseconds(8);//延时
    digitalWrite(R_IR,HIGH);//置低，截止
    delayMicroseconds(8);//延时
  }
}
void pcint0_init(void)//引脚变化中断初始化
{
  PCICR = 0X01;//使能第 0 组引脚变化中断
  PCMSK0 = 0X01;//使能第 0 组的第 0 个引脚变化中断
}

ISR(PCINT0_vect)//PB0 引脚变化中断
{
  count++;//对接收到的脉冲计数
```

```
}
void Obstacle_Avoidance(void)
{
  char i;
  count=0;
  for(i=0;i<20;i++)//左发射管发射 20 个脉冲
  {
    L_Send40KHZ();
    delayMicroseconds(600);
  }
  if(count>20)//如果接收超过了 10 个脉冲，判断有障碍物
  {
    Motor_Control(BACK,100,BACK,100);//后退
    delay(300);//延时
    Motor_Control(BACK,100,FORW,100);//右转
    delay(500);//延时
  }
  else
  {
    count=0;
    for(i=0;i<20;i++)//右发射管发射 20 个脉冲
    {
      R_Send40KHZ();
      delayMicroseconds(600);
    }
    if(count>20)
    {
      Motor_Control(BACK,100,BACK,100);//后退
      delay(300);//延时
      Motor_Control(FORW,100,BACK,100);//左转
      delay(500);//延时
    }
    else
    {
      Motor_Control(FORW,100,FORW,100);//前进
    }
```

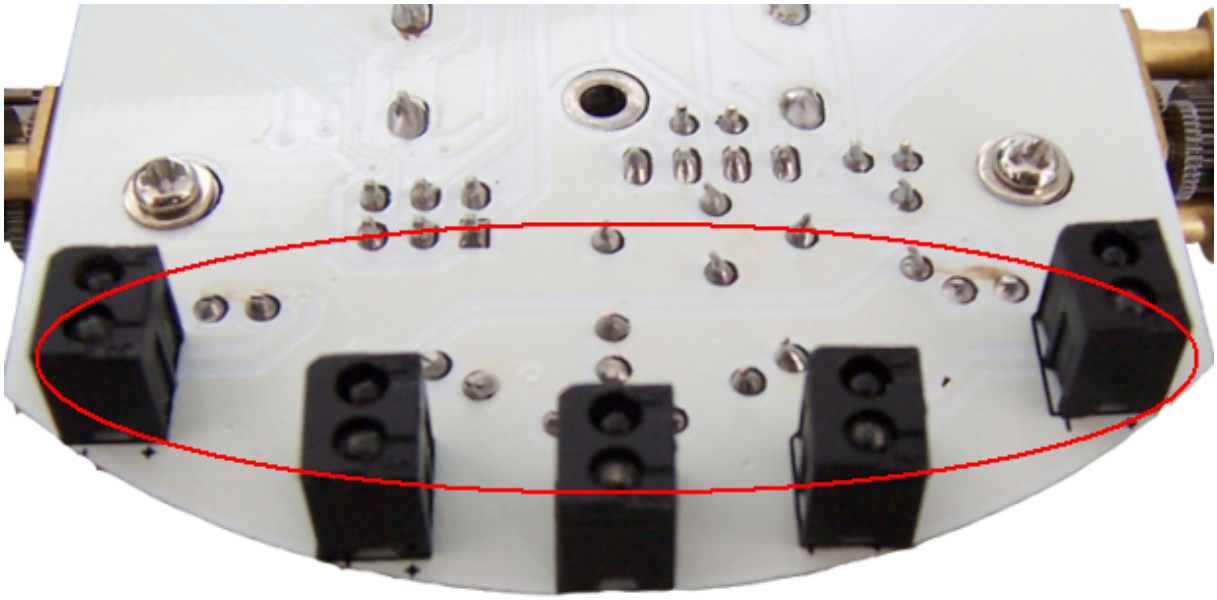
```
    }
}
void setup()
{
    char i;
    for(i=4;i<=7;i++)//设置控制两组电机的四个引脚为输出
    {
        pinMode(i,OUTPUT);
    }
    pinMode(L_IR,OUTPUT);//连接左发射管的引脚为输出
    pinMode(R_IR,OUTPUT);//连接右发射管的引脚为输出
    pinMode(IR_IN,INPUT);/连接红外 线接收的引脚为输入
    digitalWrite(R_IR,HIGH);
    digitalWrite(L_IR,HIGH);
    pcint0_init();//引脚变化中断初始化
    sei();          //全局中断使能
}
void loop()
{
    Motor_Control(FORW,100,FORW,100);//设置小车的速度为 100
    while(1)
    {
        Obstacle_Avoidance();//避障子函数
    }
}
```

4)、**程序功能**：2WDMiniQ 上电后向前行驶，遇到障碍物时先后退、然后转弯、前进。如果 2WDMiniQ 避障距离太近，可以适当的调节 W2 的大小，向大的方向调节。

7、寻线实验

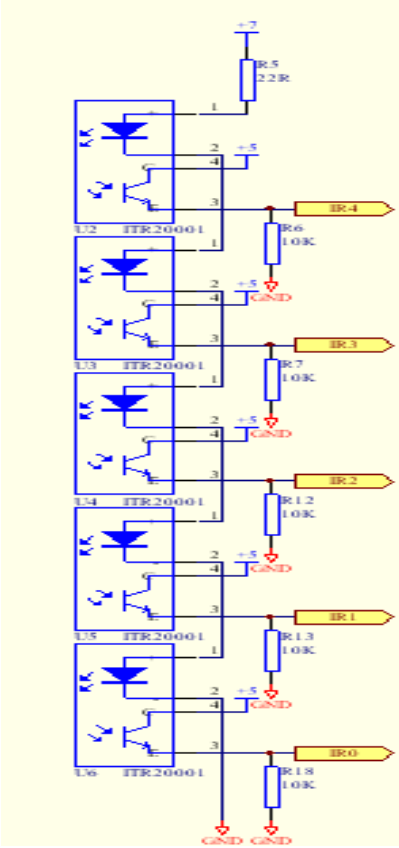
1)、介绍

2WDMiniQ 共有 5 个红外寻线传感器，位置如图：



2)、原理

红外寻线功能的实现是利用红外一体反射式光电传感器。这种光电传感器的基本原理是，自带一个光源和一个光接收装置，光源发出的光经过待测物体的反射被光敏元件接收，再经过相关电路的处理得到所需要的电压值。所以就可以根据不同场地的反光率不同电压值也不同，来判断哪是正确的赛道。原理图如下：



因为每个地面的反光率不同，程序中使用的临界电压值不同，为了方便，程序中实现了用按键调整电压值。在用 2WDMiniQ 寻线前，先将 2WDMiniQ 放到场地中不是赛道的部分，这时 AD 转换会测出场地的反光率电压值，用这个值跟数组 value[i]中的各值进行比较。按键 1 功能：选择传感器；第一次按下选择传感器 1，第二次按下选择传感器 2，依次类推。按键 2 功能：对 value[i]值加 1；每按下一次，value[i]的值就增 1。按键 3 功能：对 value[i]值减 1；每按下一次，value[i]的值就减 1。最后将调整好的值存入 eeprom，起到掉电保护作用。

3)、演示代码：

```
#include <EEPROM.h>

#define EN1 6//右侧电机使能端
#define IN1 7//右侧电机方向端
#define EN2 5//左侧电机使能端
#define IN2 4//左侧电机方向端

#define FORW 1//前进
```

```
#define BACK 0//后退

#define BUZZER 11//控制蜂鸣器的数字 IO 脚
#define LED_RED 12//控制红色 LED 灯的数字 IO 脚
#define LED_GREEN 13//控制绿色 LED 灯的数字 IO 脚
#define Vr     5//参考电压值

float data[8]={
  0X00,0X00,0X00,0X00,0x00,0xff,0x00,0x00};//存储 8 个通道 ad 转换的值
unsigned char value[5]={
  0x00,0x00,0x00,0x00,0x00};//五个寻线传感器的电压值
unsigned char key_1=0x00,key_2=0x00;/对按键  1 按下的次数进行计数



void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)//控制电机转
动
{
  //////////M1//////////////////////
  if(M1_DIR==FORW) //M1 电机的方向
    digitalWrite(IN1,FORW); //设置方向向前
  else
    digitalWrite(IN1,BACK);//设置方向向后
  if(M1_EN==0) //M1 电机的速度
    analogWrite(EN1,LOW);//置低，停止
  else
    analogWrite(EN1,M1_EN);//设置相应的数值

  //////////M2//////////////////////
  if(M2_DIR==FORW)//M2 电机的方向
    digitalWrite(IN2,FORW);//设置方向向前
  else
    digitalWrite(IN2,BACK);//设置方向向后
  if(M2_EN==0) //M2 电机的速度
    analogWrite(EN2,LOW);//置低，停止
  else
    analogWrite(EN2,M2_EN);//设置给定的数值
}
void Read_Value(void)//读取模拟值
{
  char i;
  for(i=0;i<8;i++)
  {
```

```
    data[i]=analogRead(i);//读取模拟 i 口的电压值
    data[i]= ((data[i]*Vr)/1024); //转换成模拟值
  }
}
void huntline_deal(void)//寻线
{
  if(key_2==1)
  {
    int i;
    //读取存在 EEPROM 中的电压值
    for(i=0;i<5;i++)
      value[i]=EEPROM.read(0x0a+i);

if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]<(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))//测一下实际值
    {
      Motor_Control(FORW,100,FORW,100);//前进
    }
    else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]<(value[2]-1)&&data[3]<(v
alue[3]-1)&&data[7]>(value[4]-1))
    {
      Motor_Control(BACK,50,FORW,100);//右转
    }
    else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]<(v
alue[3]-1)&&data[7]>(value[4]-1))
    {
      Motor_Control(BACK,100,FORW,100);//快速右转
    }
    else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]<(v
alue[3]-1)&&data[7]<(value[4]-1))
    {
      Motor_Control(BACK,100,FORW,100);//快速右转
    }
    else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]<(value[4]-1))
    {
      Motor_Control(BACK,100,FORW,100);//快速右转
    }
    else
```

```
if(data[0]>(value[0]-1)&&data[1]<(value[1]-1)&&data[2]<(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
      {
        Motor_Control(FORW,100,BACK,50);//左转
      }
      else
if(data[0]>(value[0]-1)&&data[1]<(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
      {
        Motor_Control(FORW,100,BACK,100);//快速左转
      }
      else
if(data[0]<(value[0]-1)&&data[1]<(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
      {
        Motor_Control(FORW,100,BACK,100);//快速左转
      }
      else
if(data[0]<(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
      {
        Motor_Control(FORW,100,BACK,100);//快速左转
      }
   }
}
void key_scan(void)//按键扫描
{
   int i;
   if(data[5]>4.50&&data[5]<6.00)//没有按键按下
      return;//返回
   else
   {
      if(data[5]>=0.00&&data[5]<0.50)//按键 1 按下
      {
        delay(10);//延时消抖
        if(data[5]>=0.00&&data[5]<0.50)//按键 1 确实按下
        {
          buzzer();//蜂鸣器响
          while(data[5]>=0.00&&data[5]<0.50)
            Read_Value();
          key_1++;//按键 1 计数
          if(key_1>=1&&key_1<=5) value_adjust(key_1);//寻线传感器的值调整
          if(key_1==6)
```

```
            {
              digitalWrite(LED_RED,LOW);//红灯灭
              digitalWrite(LED_GREEN,LOW);//绿灯灭
              //将设置好的电压值存入 EEPROM，做到掉电保护
              for(i=0;i<5;i++)
                EEPROM.write(0x0a+i,value[i]);
            }
          }
        }
        else if(data[5]>=0.50&&data[5]<2.00)
        {
          delay(10);//延时消抖
          if(data[5]>=0.50&&data[5]<2.00)
          {
            buzzer();//蜂鸣器响
            while(data[5]>=0.50&&data[5]<2.00)
              Read_Value();
            if(key_1>=1&&key_1<=5)//如果 key1 的按键值在 1~5 之间
            {
              value[key_1-1]++;//传感器的分辨轨迹的界限值加加（调整）
              value_adjust(key_1);//跟实际值对比
            }
            else
              key_2++;
          }
        }
        else if(data[5]>=2.00&&data[5]<3.00)
        {
          delay(10);//延时消抖
          if(data[5]>=2.00&&data[5]<3.00)
          {
            buzzer();//蜂鸣器响
            while(data[5]>=2.00&&data[5]<3.00)
              Read_Value();
            if(key_1>=1&&key_1<=5)//如果按键 key1 的值在 1~5 之间
            {
              value[key_1-1]--;//传感器的分辨轨迹的界限值减减（调整）
              value_adjust(key_1);//跟实际值对比
            }
          }
        }
      }
    }
```

```
void value_adjust(unsigned char num)//调整寻线传感器的值
{
  if(num==1)//调节第一个寻线传感器
  {
    if(data[0]>value[0])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==2)//调节第二个寻线传感器
  {
    if(data[1]>value[1])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==3)//调节第三个寻线传感器
  {
    if(data[2]>value[2])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==4)//调节第四个寻线传感器
  {
```

```
    if(data[3]>value[3])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==5)//调节第五个寻线传感器
  {
    if(data[4]>value[4])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
}
void buzzer(void)//蜂鸣器发出声音
{
  digitalWrite(BUZZER,HIGH);//置高，蜂鸣器响
  delay(1);
  digitalWrite(BUZZER,LOW);//置低，蜂鸣器不响
  delay(10);
}
void setup()
{
  char i;
  for(i=4;i<=7;i++)//设置连接两组电机的端口为输出模式
  {
    pinMode(i,OUTPUT);
  }
  pinMode(BUZZER,OUTPUT);//设置控制蜂鸣器的数字 IO 脚模式，OUTPUT 为
输出
  pinMode(LED_RED,OUTPUT);//设置控制红色 LED 灯的数字 IO 脚模式，
OUTPUT 为输出
```

```
  pinMode(LED_GREEN,OUTPUT);// 设置控制绿色 LED 灯的数字 IO 脚模式,OUTPUT 为输出
}
void loop()
{
  Motor_Control(FORW,0,FORW,0);//小车停止
  while(1)
  {
    Read_Value();//读取模拟口的值
    key_scan();//扫描按键
    huntline_deal();//寻线处理子函数
  }
}
```

4)程序功能：将 2WDMiniQ 放到场地的非赛道部分，按下按键 key_1 按键 1 次选择左边第一个传感器，红灯亮，按下按键 key_2 按键 value[0]值加 1，直到绿灯亮，重复上述过程，换下一个传感器调整，直到五个都调整好。第六次按下按键 1，两个灯均灭，将 2WDMiniQ 放到赛道开始寻线。

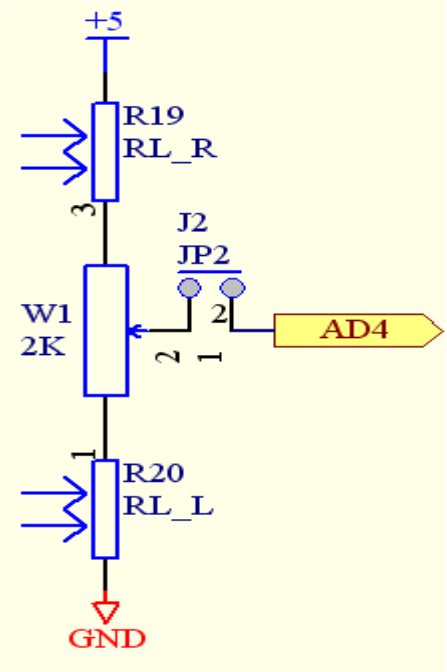
8、寻光实验

1)、介绍

2WDMiniQ 共有两个寻光传感器，寻光传感器位置如图：

2)、原理

本设计使用的寻光传感器是光敏电阻。当光照射到光敏电阻上，它的阻值就会变小，这是由于光照产生的载流子都参与导电，在外加电场的作用下作漂移运动，电子奔向电源的正极，空穴奔向电源的负极，从而使光敏电阻器的阻值迅速下降。原理图如下：



用外用表测量可知，在右侧光敏电阻有光源照射的情况下，AD4 端口的电压值比无光照时的电压值大；左侧光敏电阻有光照射的情况下， AD4 端口的电压值比无光照时的电压值小。程序中利用按 key_1 键来控制是否采集环境光的电压值。按下 key_1 键 1 次开始采集环境光电压值 10 次，并求和，将电压值的和存进 eeprom，做到掉电保护。不按 key_按键时，就读取 eeprom 里的电压和，然后求取平均值，将这个值作为临界值，判断是哪一侧有光照。

3)、演示代码：

```
#include <EEPROM.h>

#define EN1 6//右侧电机使能引脚
#define IN1 7//右侧电机方向引脚
#define EN2 5//左侧电机使能引脚
#define IN2 4//左侧电机方向引脚

#define FORW 1//前进
#define BACK 0//后退
#define BUZZER 11//设置控制蜂鸣器数字 IO 脚
#define Vr     5//参考电压值

float data[10]={
   0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};
unsigned char sum=0;
float data_1=0x01,data_2=0x01;//存储模拟电压值
float val;
```

```
int     key_1=0,key_2=0,key_3=0;
void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)//控制电机转
动
{
  //////////M1//////////////////////
  if(M1_DIR==FORW)//M1 电机的方向
    digitalWrite(IN1,FORW); //设置方向向前
  else
    digitalWrite(IN1,BACK);//设置方向向后
  if(M1_EN==0) //M1 电机的速度
    analogWrite(EN1,LOW);//置低，停止
  else
    analogWrite(EN1,M1_EN);//设置相应的数值

  //////////M2//////////////////////
  if(M2_DIR==FORW)//M2 电机的方向
    digitalWrite(IN2,FORW);//设置方向向前
  else
    digitalWrite(IN2,BACK);//设置方向向后
  if(M2_EN==0)//M2 电机的速度
    analogWrite(EN2,LOW);//置低，停止
  else
    analogWrite(EN2,M2_EN);//设置为给定的数值
}
void Read_Value(void)//读取电压值
{
  data_1=analogRead(4);//读取模拟 4 口电压值
  data_1 = ((data_1 * Vr) / 1024);//将数字值转换成模拟值
  data_2=analogRead(5);//读取模拟 5 口电压值
  data_2 = ((data_2 * Vr) / 1024);//将数字值转换成模拟值
}
void buzzer(void)//蜂鸣器发出声音
{
  digitalWrite(BUZZER,HIGH);//置高，蜂鸣器响
  delay(1);
  digitalWrite(BUZZER,LOW);//置低，蜂鸣器不响
  delay(10);
}
void key_scan(void)//按键扫描
{
  if(data_2>4.50&&data_2<6.00)//没有按键按下
    return;//返回
```

```
  else
  {
    if(data_2>=0.00&&data_2<0.50)//按键 1 按下
    {
      delay(10);//延时消抖
      if(data_2>=0.00&&data_2<0.50)//按键 1 确实按下
      {
        buzzer();//蜂鸣器响
        while(data_2>=0.00&&data_2<0.50)
          Read_Value();//读取电压值
        key_1++;
      }
    }
    else if(data_2>=0.50&&data_2<1.5)
    {
      delay(10);
      if(data_2>=0.50&&data_2<1.5)
      {
        buzzer();
        while(data_2>=0.50&&data_2<1.5)
          Read_Value();//读取电压值
          key_2++;
      }
    }
    else if(data_2>=1.5&&data_2<2.5)
    {
      delay(10);
      if(data_2>=1.5&&data_2<2.5)
      {
        buzzer();
        while(data_2>=1.5&&data_2<2.5)
          Read_Value();//读取电压值
          key_3++;
      }
    }
  }
}
void hunt_light(void)//寻光子函数
{
  int i;
  if(key_1==1)
  {
```

```
    key_1=0;
    //读取 10 次环境光的电压值，并求和
    for(i=0;i<10;i++)
    {
      Read_Value();
      data[i]=data_1;
      sum +=data[i];
    }
    //将环境光电压值的和存放在 EEPROM 里，这样掉电可以保护数据不丢失
    EEPROM.write(0x0a,sum);
  }
  else
  {
    val=EEPROM.read(0x0a);//读取电压值的和
    val=val/10;/求平均值
    Read_Value();
    if (data_1<val-0.5)
      Motor_Control(FORW,100,BACK,100);//左转
    else if (data_1>val+1)
      Motor_Control(BACK,100,FORW,100);//右转
    else
      Motor_Control(FORW,0,FORW,0);//停止
  }
}
void setup()
{
  unsigned char j;
  for(j=4;j<=7;j++)//设置连接两组电机的端口为输出模式
    pinMode(j,OUTPUT);
}
void loop()
{
  Motor_Control(FORW,0,FORW,0);//小车前进
  while(1)
  {
    Read_Value();//读取电压值
    key_scan();//按键扫描
    hunt_light();//寻光子函数
  }
}
```

4)、程序功能：无光源时 2WDMiniQ 停止，左侧有光源 2WDMiniQ 左转，右侧有光源右

转。2WDMiniQ 的行为类似一个趋光虫。

9、红外遥控实验

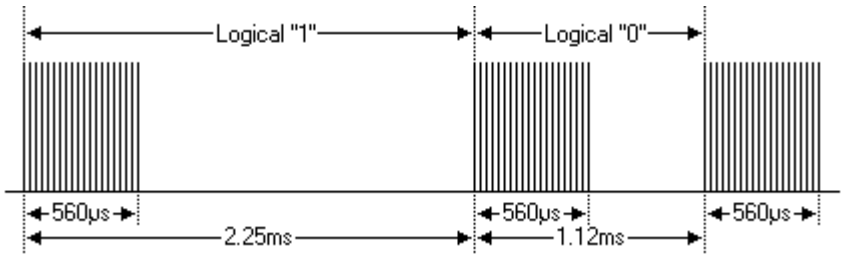
1)、介绍

2WDMiniQ 的红外接收头与避障传感器的红外接收头利用的是同一个。

2)、原理

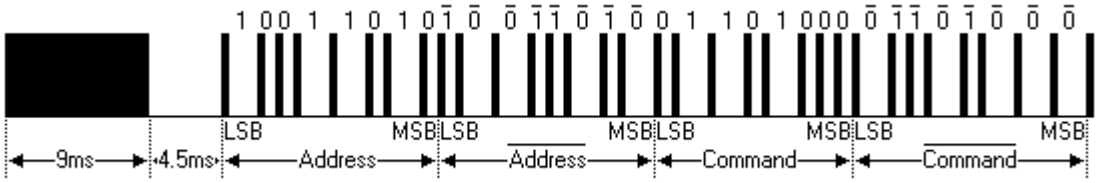
要想对某一遥控器进行解码必须要了解该遥控器的编码方式，这就叫知己知彼，百战不殆 。本产品使用的遥控器的编码方式为：NEC 协议。下面就介绍一下 NEC 协议：

- NEC 协议介绍：特点：（1）8 位地址位，8 位命令位
  （2）为了可靠性地址位和命令位被传输两次
  （3）脉冲位置调制
  （4）载波频率 38khz
  （5）每一位的时间为 1.125ms 或 2.25ms
- 逻辑 0 和 1 的定义如下图：



协议如下：

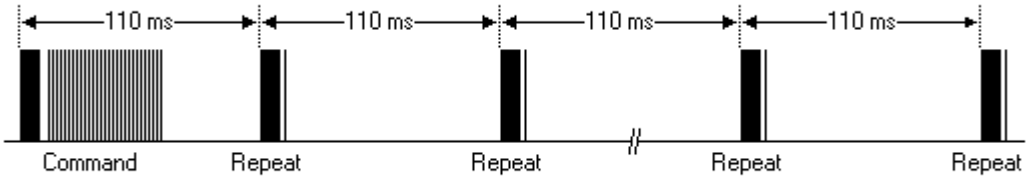
- 按键按下立刻松开的发射脉冲：



上面的图片显示了 NEC 的协议典型的脉冲序列。注意：这是首先发送 LSB（最低位）的协议。在上面的脉冲传输的地址为 0x59 命令为 0x16。一个消息是由一个 9ms 的高电平开始，随后有一个 4.5ms 的低电平，（这两段电平组成引导码）然后由地址码和命令码。地址和命令传输两次。第二次所有位都取反，可用于对所收到的消息中的确认使用。总传输时间是恒

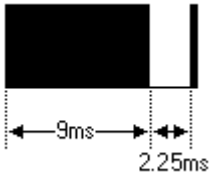
定的，因为每一点与它取反长度重复。如果你不感兴趣，你可以忽略这个可靠性取反，也可以扩大地址和命令，以每 16 位！

• 按键按下一段时间才松开的发射脉冲：



一个命令发送一次，即使在遥控器上的按键仍然按下。当按键一直按下时，第一个 110ms 的脉冲与上图一样，之后每 110ms 重复代码传输一次。这个重复代码是由一个 9ms 的高电平脉冲和一个 2.25ms 低电平和 560μs 的高电平组成。

• 重复脉冲



本介绍是参考 http://www.sbprojects.com/knowledge/ir/nec.htm

注意： 脉冲波形进入一体化接收头以后，因为一体化接收头里要进行解码、信号放大和整形，故要注意：在没有红外信号时，其输出端为高电平，有信号时为低电平，故其输出信号电平正好和发射端相反。接收端脉冲大家可以通过示波器看到，结合看到的波形理解程序。

• 本实验编程思想

根据 NEC 编码的特点和接收端的波形，本实验将接收端的波形分成四部分：引导码（9ms 和 4.5ms 的脉冲）、地址码 16 位（包括 8 位的地址位和 8 位的地址的取反）、命令码 16 位（包括 8 位命令位和 8 位命令位的取反）、重复码（9ms、2.25ms、560us 脉冲组成）。利用定时器对接收到的波形的高电平段和低电平段进行测量，根据测量到的时间来区分：逻辑“0”、逻辑“1”、引导脉冲、重复脉冲。引导码和地址码只要判断是正确的脉冲即可，不用存储，但是命令码必须存储，因为每个按键的命令码都不同，根据命令码来执行相应的动作。设置遥控器上的几个按键 VOL+：作为前进键；VOL-：作为后退键；两个向左的三角组成的符号键：作为左转键；两个向右的三角组成的符号键：作为右转键

3)、演示代码：

```
#define EN1 6//右侧电机使能引脚
#define IN1 7//右侧电机方向引脚
#define EN2 5//左侧电机使能引脚
#define IN2 4//左侧电机方向引脚

#define FORW 1//前进
#define BACK 0//后退

#define IR_IN   8   //红外接收

int Pulse_Width=0;//存储脉宽
int   ir_code=0x00; //命令值
//控制电机转动
void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)
{
  //////////M1////////////////////////
  if(M1_DIR==FORW)//M1 方向为 FORW
       digitalWrite(IN1,FORW); //向前
  else
     digitalWrite(IN1,BACK);//向后
  if(M1_EN==0) //M1 速度为 0
       analogWrite(EN1,LOW);//置低，停止
  else
     analogWrite(EN1,M1_EN);//设置给定的数值

  //////////M2////////////////////////
  if(M2_DIR==FORW) //M2 方向为 FORW
       digitalWrite(IN2,FORW);//向前
  else
     digitalWrite(IN2,BACK);//向后
  if(M2_EN==0) //M2 速度为 0
       analogWrite(EN2,LOW);//置低，停止
  else//否则
  analogWrite(EN2,M2_EN);//设置给定的数值
}
//定时器 1 初始化函数
void timer1_init(void)
{
```

```
    TCCR1A = 0X00;
    TCCR1B = 0X05;//给定时器时钟源
    TCCR1C = 0X00;
    TCNT1 = 0X00;
    TIMSK1 = 0X00;  //禁止定时器溢出中断
}
//遥控器命令执行函数
void remote_deal(void)
{
    switch(ir_code)
    {
        case 0xff00:
            Motor_Control(FORW,0,FORW,0);//停止
            break;
        case 0xfe01:
             Motor_Control(FORW,100,FORW,100);//前进
             break;
        case 0xf609:
             Motor_Control(BACK,100,BACK,100);//后退
             break;
        case 0xfb04:
            Motor_Control(FORW,100,BACK,100);//左转
            break;
        case 0xf906:
            Motor_Control(BACK,100,FORW,100);//右转
            break;
    }
}
char logic_value()
{
        while(!(digitalRead(8))); //低等待
         Pulse_Width=TCNT1;
         TCNT1=0;
         if(Pulse_Width>=7&&Pulse_Width<=10)//低电平 560us
         {
               while(digitalRead(8));//是高就等待
               Pulse_Width=TCNT1;
               TCNT1=0;
               if(Pulse_Width>=7&&Pulse_Width<=10)//接着高电平 560us
                  return 0;
               else if(Pulse_Width>=25&&Pulse_Width<=27) //接着高电平 1.7ms
                  return 1;
```

```
        }
        return -1;
}
//遥控器解码函数
void pulse_deal()
{
   int i;

   //执行 8 个 0
   for(i=0; i<8; i++)
   {
      if(logic_value() != 0) //不是 0
         return;
   }
   //执行 6 个 1
   for(i=0; i<6; i++)
   {
      if(logic_value()!= 1) //不是 1
         return;
   }
   //执行 1 个 0
   if(logic_value()!= 0) //不是 0
         return;
   //执行 1 个 1
   if(logic_value()!= 1) //不是 1
         return;


   //解析遥控器编码中的 command 指令
   ir_code=0x00;//清零
   for(i=0; i<16;i++ )
   {
      if(logic_value() == 1)
      {
         ir_code |=(1<<i);
      }
   }
}
//遥控器解码函数
void remote_decode(void)
{
        TCNT1=0X00;
```

```
    while(digitalRead(8))//是高就等待
    {
        if(TCNT1>=1563)   //当高电平持续时间超过 100ms，表明此时没有按键
按下
        {
            ir_code = 0xff00;
            return;
        }
    }

    //如果高电平持续时间不超过 100ms
    TCNT1=0X00;

    while(!(digitalRead(8))); //低等待
    Pulse_Width=TCNT1;
    TCNT1=0;
    if(Pulse_Width>=140&&Pulse_Width<=141)//9ms
    {

        while(digitalRead(8));//是高就等待
        Pulse_Width=TCNT1;
        TCNT1=0;
        if(Pulse_Width>=68&&Pulse_Width<=72)//4.5ms
        {
            pulse_deal();
            return;
        }
        else if(Pulse_Width>=34&&Pulse_Width<=36)//2.25ms
        {
            while(!(digitalRead(8)));//低等待
            Pulse_Width=TCNT1;
            TCNT1=0;
            if(Pulse_Width>=7&&Pulse_Width<=10)//560us
            {
                return;
            }
        }
    }

}

void setup()
```

```
{
  unsigned char i;
  for(i=4;i<=7;i++)
  {
    pinMode(i,OUTPUT);
  }
  pinMode(IR_IN,INPUT);
}
void loop()
{
  timer1_init();//定时器函数初始化
  while(1)
  {
    remote_decode();   //译码
    remote_deal();     //执行译码结果
  }
}
```

4)、程序功能：对遥控器发射出来的编码脉冲进行解码，根据解码结果执行相应的动作。

按下前进键 2WDMiniQ 前进，松开 2WDMiniQ 停止；按下后退键 2WDMiniQ 后退，松

开 2WDMiniQ 停止；按下左转键 2WDMiniQ 左转，松开 2WDMiniQ 停止；按下右转键

2WDMiniQ 右转，松开 2WDMiniQ 停止。这样大家就可以用遥控器遥控 2WDMiniQ，

让它听你的指挥。其它按键的译码方式与这几个键一样，只要大家用示

波器测出它们各自的波形，了解各自的命令码，在执行译码结果的函数

中写上对应的命令码和要执行的动作即可。

10、综合实验

1）、程序说明

将 2WDMiniQ 放到场地的非赛道部分，按下按键 key_1 按键 1 次选择左边第一个传感器，红灯亮，按下按键 key_2 按键 value[0]值加 1，直到绿灯亮，重复上述过程，换下一个传感器调整，直到五个都调整好。第六次按下按键 1，两个灯均灭，将 2WDMiniQ 放到赛道开始寻线。再一次的按下 key_1 按键，小车停止寻线。

按下 key_2 按键 1 次，小车开始执行避障功能，第二次按下 key_2 按键小车停止避障。第三次按下 key_2 按键，小车可以实现寻光功能，第四次按下 key_2

按键小车停止。

按下 key_3 按键 1 次，小车具有模拟袭击者功能，即没有物体在小车前方时小车静止不动，当有物体在小车前方，小车突然加速前进，第二次按下 key_3 按键小车停止。第三次按下 key_3 按键，小车开始模拟小鸡行为，即当前面没有物体时前进，有物体时绕行，第四次按下 key_3 按键小车停止。第五次按下 key_3 按键，小车开始执行遥控功能，按下前进键 2WDMiniQ 前进，松开 2WDMiniQ 停止；按下后退键 2WDMiniQ 后退，松开 2WDMiniQ 停止；按下左转键 2WDMiniQ 左转，松开 2WDMiniQ 停止；按下右转键 2WDMiniQ 右转，松开 2WDMiniQ 停止，第六次按下 key_3 按键小车停止。

2）、程序代码

```
#define EN1 6//右侧电机使能端
#define IN1 7//右侧电机方向端
#define EN2 5//左侧电机使能端
#define IN2 4//左侧电机方向端

#define FORW 1//前进
#define BACK 0//后退

#define IR_IN   8//红外接收
#define L_IR 9//左红外发送
#define R_IR 10//右红外发送

#define BUZZER 11//蜂鸣器
#define LED_RED 12//红色 LED 灯
#define LED_GREEN   13//绿色 LED 灯

#define Vr     5//参考电压值

int     count;//对返回的脉冲进行计数
float data[8]={
  0X00,0X00,0X00,0X00,0x00,0xff,0x00,0x00};//存储 8 个通道 ad 转换的值
char    value[5]={
  0x00,0x00,0x00,0x00,0x00};//五个寻线传感器的电压值
char    key_1=0x00,key_2=0x00,key_3=0x00;//按键的计数值
```

```
//计数里程
int       count_r=0,count_l=0;//对左右轮返回的脉冲进行计数
//遥控参数
int Pulse_Width=0;//存储脉宽
int    ir_code=0x00; //命令值

//控制电机转动
void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)
{
  //////////M1//////////////////////
  if(M1_DIR==FORW)//M1 方向为 FORW
       digitalWrite(IN1,FORW); //向前
  else
     digitalWrite(IN1,BACK);//向后
  if(M1_EN==0) //M1 速度为 0
       analogWrite(EN1,LOW);//置低，停止
  else
     analogWrite(EN1,M1_EN);//设置给定的数值

  //////////M2//////////////////////
  if(M2_DIR==FORW) //M2 方向为 FORW
       digitalWrite(IN2,FORW);//向前
  else
     digitalWrite(IN2,BACK);//向后
  if(M2_EN==0) //M2 速度为 0
       analogWrite(EN2,LOW);//置低，停止
  else//否则
  analogWrite(EN2,M2_EN);//设置给定的数值
}
//计数里程
void interrupt01_init(void)//外部中断初始化
{
  EICRA = 0X0F;//设置 INT0、INT1 为上升沿触发中断
  EIMSK = 0X03;//使能中断 INT0、INT1
}
ISR(INT0_vect)//INTO 中断函数
```

```
{
  if(++count_r==120)//右侧车轮返回脉冲计数
    count_r=0;//计数到 120，清零
}
ISR(INT1_vect)//INT1 中断函数
{
  if(++count_l==120)//左侧车轮返回脉冲计数
    count_l=0;//计数到 120，清零
}
//避障
void L_Send40KHZ(void)//左发射管发射频率为 40kHZ 脉冲
{
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(L_IR,LOW);//设置左发射管为高电平
    delayMicroseconds(8);//延时
    digitalWrite(L_IR,HIGH);//设置左发射管为低电平
    delayMicroseconds(8);//延时
  }
}
void R_Send40KHZ(void)//右发射管发射频率为 40kHZ 脉冲
{
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(R_IR,LOW);//设置右发射管为高电平
    delayMicroseconds(8);//延时
    digitalWrite(R_IR,HIGH);//设置右发射管为低电平
    delayMicroseconds(8);//延时
  }
}
void pcint0_init(void)//引脚变化中断初始化
{
  PCICR = 0X01;//使能第 0 组引脚变化中断
  PCMSK0 = 0X01;//使能第 0 组的第 0 个引脚变化中断
```

```
}
ISR(PCINT0_vect)//PB0 引脚变化中断
{
  count++;//计数接收到的脉冲
}
void Obstacle_Avoidance(void)//避障子函数
{
  char i;
  count=0;
  for(i=0;i<20;i++)//左发射管发射 20 个脉冲
  {
    L_Send40KHZ();
    delayMicroseconds(600);
  }
  if(count>20)//接收超过 10 个脉冲，判断有障碍物
  {
    if(key_2==1||key_3==3)//避障模式或者是小鸡模式
    {
      Motor_Control(BACK,100,BACK,100);//后退
      delay(500);//延时
      Motor_Control(BACK,100,FORW,100);//右转
      delay(500);//延时
    }
    if(key_3==1)//袭击者模式
    {
      Motor_Control(FORW,100,FORW,100);//前进
      delay(300);//延时
    }
  }
  else
  {
    if(key_2==1||key_3==3)//避障模式或者小鸡模式
      Motor_Control(FORW,100,FORW,100);//前进
    if(key_3==1)//袭击者模式
      Motor_Control(FORW,0,FORW,0);//前进
  }
```

```
  count=0;
  for(i=0;i<20;i++)//右发射管发射 20 个脉冲
  {
    R_Send40KHZ();
    delayMicroseconds(600);
  }
  if(count>20)
  {
    if(key_2==1||key_3==3)//避障模式或者小鸡行为
    {
      Motor_Control(BACK,100,BACK,100);//后退
      delay(500);//延时
      Motor_Control(FORW,100,BACK,100);//左转
      delay(500);//延时
    }
    if(key_3==1)//袭击者
    {
      Motor_Control(FORW,100,FORW,100);//前进
      delay(300);//延时
    }
  }
  else
  {
    if(key_2==1||key_3==3)//避障模式或者小鸡行为
      Motor_Control(FORW,100,FORW,100);//前进
    if(key_3==1)//袭击者行为
      Motor_Control(FORW,0,FORW,0);//前进
  }
}
//读取模拟端口电压值
void Read_Value(void)
{
  int i;
  for(i=0;i<8;i++)
```

```
  {
    data[i]=analogRead(i);//读取模拟 i 口电压值
    data[i]= ((data[i]*Vr)/1024); //转换成模拟值
  }
}
//寻线
void value_adjust(unsigned char num)//调整寻线传感器的值
{
  if(num==1)//调节第一个寻线传感器
  {
    if(data[0]>value[0])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==2)//调节第二个寻线传感器
  {
    if(data[1]>value[1])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==3)//调节第三个寻线传感器
  {
```

```
    if(data[2]>value[2])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==4)//调节第四个寻线传感器
  {
    if(data[3]>value[3])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
  }
  if(num==5)//调节第五个寻线传感器
  {
    if(data[4]>value[4])
    {
      digitalWrite(LED_RED,HIGH); //当前值小红灯亮
      digitalWrite(LED_GREEN,LOW);
    }
    else
    {
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);//绿灯亮
    }
```

```
    }
}
void huntline_deal(void)
{

if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]<(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))//测一下实际值
      Motor_Control(FORW,100,FORW,100);//前进
   else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]<(value[2]-1)&&data[3]<(v
alue[3]-1)&&data[7]>(value[4]-1))
      Motor_Control(BACK,20,FORW,100);//右转
   else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]<(v
alue[3]-1)&&data[7]>(value[4]-1))
      Motor_Control(BACK,100,FORW,100);//快速右转
   else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]<(v
alue[3]-1)&&data[7]<(value[4]-1))
      Motor_Control(BACK,100,FORW,100);//快速右转
   else
if(data[0]>(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]<(value[4]-1))
      Motor_Control(BACK,100,FORW,100);//快速右转
   else
if(data[0]>(value[0]-1)&&data[1]<(value[1]-1)&&data[2]<(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
      Motor_Control(FORW,100,BACK,20);//左转
   else
if(data[0]>(value[0]-1)&&data[1]<(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
      Motor_Control(FORW,100,BACK,100);//快速左转
   else
if(data[0]<(value[0]-1)&&data[1]<(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
      Motor_Control(FORW,100,BACK,100);//快速左转
```

```
  else
if(data[0]<(value[0]-1)&&data[1]>(value[1]-1)&&data[2]>(value[2]-1)&&data[3]>(v
alue[3]-1)&&data[7]>(value[4]-1))
    Motor_Control(FORW,100,BACK,100);//快速左转
}
//寻光
void hunt_light(void)//寻光子函数
{
  if (data[4] >1&&data[4]< 2)    //根据实际的实验环境进行测量
    Motor_Control(BACK,100,FORW,100);//右转
  else if (data[4] > 4 && data[4] < 5)
    Motor_Control(FORW,100,BACK,100);//左转
  else
    Motor_Control(FORW,0,FORW,0);//停止
}
//遥控功能函数
void timer1_init(void)
{
  TCCR1A = 0X00;
  TCCR1B = 0X05;//给定时器时钟源
  TCCR1C = 0X00;
  TCNT1 = 0X00;
  TIMSK1 = 0X00;  //禁止定时器溢出中断
}
void remote_deal(void)
{
  switch(ir_code)
  {
  case 0xff00:
    Motor_Control(FORW,0,FORW,0);//停止
    break;
  case 0xfe01:
    Motor_Control(FORW,100,FORW,100);//前进
    break;
  case 0xf609:
    Motor_Control(BACK,100,BACK,100);//后退
```

```
        break;
    case 0xfb04:
        Motor_Control(FORW,100,BACK,100);//左转
        break;
    case 0xf906:
        Motor_Control(BACK,100,FORW,100);//右转
        break;
    }
}
char logic_value()
{
    while(!(digitalRead(8))); //低等待
    Pulse_Width=TCNT1;
    TCNT1=0;
    if(Pulse_Width>=7&&Pulse_Width<=10)//低电平 560us
    {
        while(digitalRead(8));//是高就等待
        Pulse_Width=TCNT1;
        TCNT1=0;
        if(Pulse_Width>=7&&Pulse_Width<=10)//接着高电平 560us
            return 0;
        else if(Pulse_Width>=25&&Pulse_Width<=27) //接着高电平 1.7ms
            return 1;
    }
    return -1;
}
void pulse_deal()
{
    int i;

    //执行 8 个 0
    for(i=0; i<8; i++)
    {
        if(logic_value() != 0) //不是 0
            return;
    }
```

```
  //执行 6 个 1
  for(i=0; i<6; i++)
  {
    if(logic_value()!= 1) //不是 1
        return;
  }
  //执行 1 个 0
  if(logic_value()!= 0) //不是 0
      return;
  //执行 1 个 1
  if(logic_value()!= 1) //不是 1
      return;


  //解析遥控器编码中的 command 指令
  ir_code=0x00;//清零
  for(i=0; i<16;i++ )
  {
    if(logic_value() == 1)
    {
      ir_code |=(1<<i);
    }
  }
}
void remote_decode(void)
{
  TCNT1=0X00;
  while(digitalRead(8))//是高就等待
  {
    if(TCNT1>=1563)   //当高电平持续时间超过 100ms，表明此时没有按键按下
    {
      ir_code = 0xff00;
      return;
    }
  }
```

```
  //如果高电平持续时间不超过 100ms
  TCNT1=0X00;

  while(!(digitalRead(8))); //低等待
  Pulse_Width=TCNT1;
  TCNT1=0;
  if(Pulse_Width>=140&&Pulse_Width<=141)//9ms
  {

    while(digitalRead(8));//是高就等待
    Pulse_Width=TCNT1;
    TCNT1=0;
    if(Pulse_Width>=6d8&&Pulse_Width<=72)//4.5ms
    {
      pulse_deal();
      return;
    }
    else if(Pulse_Width>=34&&Pulse_Width<=36)//2.25ms
    {
      while(!(digitalRead(8)));//低等待
      Pulse_Width=TCNT1;
      TCNT1=0;
      if(Pulse_Width>=7&&Pulse_Width<=10)//560us
      {
        return;
      }
    }
  }

}

//蜂鸣器声音
void buzzer(void)//蜂鸣器发出一种声音
{
  digitalWrite(BUZZER,HIGH);//置高，蜂鸣器响
  delay(1);
```

```
  digitalWrite(BUZZER,LOW);//置低，蜂鸣器不响
  delay(10);
}
//按键扫描
void key_scan(void)
{
  if(data[5]>4.50&&data[5]<6.00)//没有按键按下
    return;//返回
  else
  {
    if(data[5]>=0.00&&data[5]<0.50)//按键 1 按下
    {
      delay(200);//延时消抖
      if(data[5]>=0.00&&data[5]<0.50)//按键 1 确实按下
      {
        buzzer();//蜂鸣器响
        key_1++;//按键 1 计数
        if(key_1>=1&&key_1<=5)
          value_adjust(key_1);//寻线传感器的值调整
      }
    }
    else if(data[5]>=0.50&&data[5]<1.5)
    {
      delay(200);//延时消抖
      if(data[5]>=0.50&&data[5]<1.5)
      {
        buzzer();//蜂鸣器响
        if(key_1>=1&&key_1<=5)//按键 1 的值在 1~5 之间
        {
          value[key_1-1]++;//传感器的分辨轨迹的界限值加加（调整）
          value_adjust(key_1);//跟实际值对比
        }
        else
        {
          key_2++;//key2 计数
        }
```

```
        }
      }
      else if(data[5]>=1.5&&data[5]<2.5)
      {
        delay(200);//延时消抖
        if(data[5]>=1.5&&data[5]<2.5)
        {
          buzzer();//蜂鸣器响
          if(key_1>=1&&key_1<=5)//按键 1 的值在 1~5 之间
          {
            value[key_1-1]--;//传感器的分辨轨迹的界限值减减（调整）
            value_adjust(key_1);//跟实际值对比
          }
          else
          {
            key_3++;//key3 计数
          }
        }
      }
    }
}
void key_deal()
{
  if(key_1==6)//寻线
  {
    digitalWrite(LED_RED,LOW);//红灯灭
    digitalWrite(LED_GREEN,LOW);//绿灯灭
    huntline_deal();//调用寻线处理子函数
  }
  if(key_2==1||key_3==1||key_3==3)//避障，小鸡行为，袭击者
  {
    digitalWrite(LED_RED,HIGH);
    pcint0_init();//引脚变化中断初始化
    interrupt01_init();//外部中断 0 和 1 中断初始化
    sei();        //全局中断使能
    Obstacle_Avoidance();
```

```
    }
    if(key_2==3)//寻光
    {
      digitalWrite(LED_RED,HIGH);
      hunt_light();//调用寻光子函数
    }
    if(key_3==5)//遥控
    {
      digitalWrite(LED_RED,HIGH);//红灯亮
      timer1_init();//定时器初始化函数
      remote_decode();   //译码
      remote_deal();     //执行译码结果
    }
    if(key_1==7||key_2==2||key_2==4||key_3==2||key_3==4||key_3==6)
    {
      //按键值清零
      key_1=0;
      if(key_2==4)
        key_2=0;
      if(key_3==6)
        key_3=0;
      digitalWrite(LED_RED,LOW);
      Motor_Control(FORW,0,FORW,0);//停止
    }
}
//低电压检测
void low_voltage_check(void)
{
    if(data[6]<2.5)//当电压低于 2.5V 时
    {
      buzzer(); //蜂鸣器响
    }
}
void setup()
{
    int i;
```

```
  for(i=4;i<=13;i++)//设置引脚模式
  {
    pinMode(i,OUTPUT);
  }
  pinMode(IR_IN,INPUT);/红外线接 收的引脚为输入
}
void loop()
{
  Motor_Control(FORW,0,FORW,0);//小车停止
  while(1)
  {
    Read_Value();//读取模拟口的值
    key_scan();//扫描按键
    key_deal();//按键处理
    low_voltage_check();//低电压检测
  }
}
```

11、无线下载接口应用

无线下载接口除了插接 wireless 模块无线下载程序外，还可以插接 xbee 无线通信模块进行无线通信。下面以 xbee Series 2 2mw 无线模块为例：

1）PC 机与小车无线通信。

将两个 xbee 模块配置成点对点模式（点对点模式的设置可参考：http://www.dfrobot.com.cn/?product-441.html）。本练习中将 xbee 模块配置为点对点模式，波特率 38400bps。将下面程序下载到小车中。

```
#define LED_RED 12//定义红色的 led 灯的引脚

void setup()
{
  pinMode(LED_RED,OUTPUT);//设置 LED 灯引脚的模式为输出
  Serial.begin(38400); //设置波特率
}
void loop()
```

```
{
    char word;
    while(1)
    {
        if (Serial.available() > 0) //判断串口缓冲器是否有数据装入
        {
            word = Serial.read();       //读取串口
            if(word=='a')                   //判断输入的字符是否为 a
            {
              digitalWrite(LED_RED,HIGH);
            }
            else
            {
              digitalWrite(LED_RED,LOW);
            }
        }
    }
}
```

下载完毕后在小车上插接 xbee 无线模块。另一个 xbee 模块插接在 xbee 适配器上，然后通过 USB 线将适配器连接在 PC 机上，打开 PC 机上的串口助手，发送字符'a'，小车接收到字符'a'后红灯亮。这样 PC 机就可以和小车进行点对点的无线通信了。

2）小车与小车之间的点对点通信。

同上，将 xbee 模块配置成点对点模式。分别插接在两辆小车上。这样两辆小车之间就可以进行点对点通信。可以在两个小车上分别做程序，一个发送一个接收，类似上面。

3）小车进行组网通信。

小车可以利用 xbee 无线模块，进行组网通信。首先配置 xbee 模块为组网模式（组网模式的配置可参考 xbee Series 2 2mw 组网步骤.pdf）。本练习配置成组网模式，波特率 38400bps。本练习用的 xbee 模块的地址分别为 0013A200406FB799、0013A200406FB76F、0013A200406FB77F。小车用 ABC 字母任意区分为 A 车、B 车、C 车，将 A 小车下载下面程序：

```
#define EN1 6//右侧电机使能引脚
#define IN1 7//右侧电机方向引脚
#define EN2 5//左侧电机使能引脚
```

```
#define IN2 4//左侧电机方向引脚

#define FORW 1//前进
#define BACK 0//后退

#define IR_IN   8//红外接收
#define L_IR 9//左红外发送
#define R_IR 10//右红外发送

int count;//对返回的脉冲进行计数

void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)//控制电机转
动
{
  //////////M1//////////////////////
  if(M1_DIR==FORW)//M1 电机方向
    digitalWrite(IN1,FORW);//设置方向向前
  else
    digitalWrite(IN1,BACK);//设置方向向后
  if(M1_EN==0)//M1 电机速度
    analogWrite(EN1,LOW);//置低，停止
  else
    analogWrite(EN1,M1_EN);//设置给定的数值

  //////////M2//////////////////////
  if(M2_DIR==FORW)//M2 电机的方向
    digitalWrite(IN2,FORW);//设置方向向前
  else
    digitalWrite(IN2,BACK);//设置方向向后
  if(M2_EN==0) //M2 电机的速度
    analogWrite(EN2,LOW);//置低，停止
  else
    analogWrite(EN2,M2_EN); //设置给定的数值
}
void L_Send40KHZ(void)//左发射管发射频率为 40kHZ 脉冲
{
```

```
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(L_IR,LOW);//置低，导通
    delayMicroseconds(8);//延时
    digitalWrite(L_IR,HIGH);//置高，截止
    delayMicroseconds(8);//延时
  }
}
void R_Send40KHZ(void)//右发射管发射频率为 40kHZ 脉冲
{
  int i;
  for(i=0;i<24;i++)
  {
    digitalWrite(R_IR,LOW);//置低，导通
    delayMicroseconds(8);//延时
    digitalWrite(R_IR,HIGH);//置低，截止
    delayMicroseconds(8);//延时
  }
}
void pcint0_init(void)//引脚变化中断初始化
{
  PCICR = 0X01;//使能第 0 组引脚变化中断
  PCMSK0 = 0X01;//使能第 0 组的第 0 个引脚变化中断
}

ISR(PCINT0_vect)//PB0 引脚变化中断
{
  count++;//对接收到的脉冲计数
}
//查找节点
void node(void)
{
      Serial.print(126,BYTE);//7E
      Serial.print(0,BYTE);//00
      Serial.print(4,BYTE);//04
```

```
        Serial.print(8,BYTE);//08
        Serial.print(9,BYTE);//09
        Serial.print(78,BYTE);//4E
        Serial.print(68,BYTE);//44
        Serial.print(92,BYTE);//5C

}
//向模块 B 发送字符 L
void sendB_L(void)
{
        Serial.print(126,BYTE);//7E

        //有效数据长度
        Serial.print(0,BYTE);//00
        Serial.print(15,BYTE);//0F

        Serial.print(16,BYTE);//10
        Serial.print(1,BYTE);//01

        //目的地地址（即 B 模块的地址）
        Serial.print(0,BYTE);//00
        Serial.print(19,BYTE);//13
        Serial.print(162,BYTE);//A2
        Serial.print(0,BYTE);//00
        Serial.print(64,BYTE);//40
        Serial.print(111,BYTE);//6F
        Serial.print(183,BYTE);//B7
        Serial.print(111,BYTE);//6F

        Serial.print(255,BYTE);//FF
        Serial.print(254,BYTE);//FE
        Serial.print(0,BYTE);//00
        Serial.print(0,BYTE);//00

        //要发送的内容
        Serial.print(76,BYTE);//4C L
```

```
        //校验和
        Serial.print(27,BYTE);//1B
}
//向模块 B 发送字符 R
void sendB_R(void)
{
        Serial.print(126,BYTE);//7E

        //有效数据长度
        Serial.print(0,BYTE);//00
        Serial.print(15,BYTE);//0F

        Serial.print(16,BYTE);//10
        Serial.print(1,BYTE);//01

        //目的地地址（即 B 模块的地址）
        Serial.print(0,BYTE);//00
        Serial.print(19,BYTE);//13
        Serial.print(162,BYTE);//A2
        Serial.print(0,BYTE);//00
        Serial.print(64,BYTE);//40
        Serial.print(111,BYTE);//6F
        Serial.print(183,BYTE);//B7
        Serial.print(111,BYTE);//6F

        Serial.print(255,BYTE);//FF
        Serial.print(254,BYTE);//FE
        Serial.print(0,BYTE);//00
        Serial.print(0,BYTE);//00

        //要发送的内容
        Serial.print(82,BYTE);//52 R

        //校验和
        Serial.print(21,BYTE);//15
```

```
}
//向模块 C 发送字符 L
void sendC_L(void)
{
        Serial.print(126,BYTE);//7E

        //有效数据长度
        Serial.print(0,BYTE);//00
        Serial.print(15,BYTE);//0F

        Serial.print(16,BYTE);//10
        Serial.print(1,BYTE);//01

        //目的地地址（即 C 模块的地址）
        Serial.print(0,BYTE);//00
        Serial.print(19,BYTE);//13
        Serial.print(162,BYTE);//A2
        Serial.print(0,BYTE);//00
        Serial.print(64,BYTE);//40
        Serial.print(111,BYTE);//6F
        Serial.print(183,BYTE);//B7
        Serial.print(127,BYTE);//7F

        Serial.print(255,BYTE);//FF
        Serial.print(254,BYTE);//FE
        Serial.print(0,BYTE);//00
        Serial.print(0,BYTE);//00

        //要发送的内容
        Serial.print(76,BYTE);//4C L


        //校验和
        Serial.print(11,BYTE);//B
}
//向模块 C 发送字符 R
```

```
void sendC_R(void)
{
        Serial.print(126,BYTE);//7E

        //有效数据长度
        Serial.print(0,BYTE);//00
        Serial.print(15,BYTE);//0F

        Serial.print(16,BYTE);//10
        Serial.print(1,BYTE);//01

        //目的地地址（即 C 模块的地址）
        Serial.print(0,BYTE);//00
        Serial.print(19,BYTE);//13
        Serial.print(162,BYTE);//A2
        Serial.print(0,BYTE);//00
        Serial.print(64,BYTE);//40
        Serial.print(111,BYTE);//6F
        Serial.print(183,BYTE);//B7
        Serial.print(127,BYTE);//7F

        Serial.print(255,BYTE);//FF
        Serial.print(254,BYTE);//FE
        Serial.print(0,BYTE);//00
        Serial.print(0,BYTE);//00

        //要发送的内容
         Serial.print(82,BYTE);//52 R

        //校验和
        Serial.print(5,BYTE);//05
}
void Obstacle_Avoidance(void)
{
   char i;
   count=0;
```

```
  for(i=0;i<20;i++)//左发射管发射 20 个脉冲
  {
    L_Send40KHZ();
    delayMicroseconds(600);
  }
  if(count>20)//如果接收超过了 10 个脉冲，判断有障碍物
  {
    sendB_L();
    sendC_L();
    Motor_Control(BACK,100,FORW,100);//右转
    delay(50);//延时
  }
  else
  {
    count=0;
    for(i=0;i<20;i++)//右发射管发射 20 个脉冲
    {
      R_Send40KHZ();
      delayMicroseconds(600);
    }
    if(count>20)
    {
      sendB_R();
      sendC_R();
      Motor_Control(FORW,100,BACK,100);//左转
      delay(50);//延时
    }
    else
    {
      Motor_Control(FORW,0,FORW,0);//前进
    }
  }

}
void setup()
{
```

```
    char i;
    for(i=4;i<=7;i++)//设置控制两组电机的四个引脚为输出
    {
      pinMode(i,OUTPUT);
    }
    pinMode(L_IR,OUTPUT);//连接左发射管的引脚为输出
    pinMode(R_IR,OUTPUT);//连接右发射管的引脚为输出
    pinMode(IR_IN,INPUT);/连接红外 线接收的引脚为输入
    digitalWrite(R_IR,HIGH);
    digitalWrite(L_IR,HIGH);

    Serial.begin(38400); //设置波特率

    pcint0_init();//引脚变化中断初始化
    sei();        //全局中断使能
    node();
    delay(1000);
}
void loop()
{
      while(1)
     {
         Obstacle_Avoidance();
     }
}
```

程序功能：实时避障，如 A 车左边有障碍物，立即向 B、C 车发送字符串 L，A 车右转避障。程序中将地址为 0013A200406FB76F、0013A200406FB77F 的两个模块作为了目的模块 B 和目的模块 C。其中函数名为红色的函数中的有效数据长度、目的地地址、要发送的内容、校验和。需要根据具体情况而定。可参考 API 模式举例说明.pdf。

程序下载完毕后将没有被程序作为目的模块的 xbee 模块，插接在 A 小车上。

将下面程序分别下载到 B 车、C 车中。程序如下：

```
#define EN1 6//右侧电机使能引脚
#define IN1 7//右侧电机方向引脚
```

```
#define EN2 5//左侧电机使能引脚
#define IN2 4//左侧电机方向引脚

#define FORW 1//前进
#define BACK 0//后退

#define IR_IN   8//红外接收
#define L_IR 9//左红外发送
#define R_IR 10//右红外发送

void Motor_Control(int M1_DIR,int M1_EN,int M2_DIR,int M2_EN)//控制电机转
动
{
  //////////M1//////////////////////
  if(M1_DIR==FORW)//M1 电机方向
    digitalWrite(IN1,FORW);//设置方向向前
  else
    digitalWrite(IN1,BACK);//设置方向向后
  if(M1_EN==0)//M1 电机速度
    analogWrite(EN1,LOW);//置低，停止
  else
    analogWrite(EN1,M1_EN);//设置给定的数值

  //////////M2//////////////////////
  if(M2_DIR==FORW)//M2 电机的方向
    digitalWrite(IN2,FORW);//设置方向向前
  else
    digitalWrite(IN2,BACK);//设置方向向后
  if(M2_EN==0) //M2 电机的速度
    analogWrite(EN2,LOW);//置低，停止
  else
    analogWrite(EN2,M2_EN); //设置给定的数值
}

void node(void)
{
```

```
        Serial.print(126,BYTE);//7E
        Serial.print(0,BYTE);//00
        Serial.print(4,BYTE);//04
        Serial.print(8,BYTE);//08
        Serial.print(9,BYTE);//09
        Serial.print(78,BYTE);//4E
        Serial.print(68,BYTE);//44
        Serial.print(92,BYTE);//5C
}
void setup()
{
   char i;
   for(i=4;i<=7;i++)//设置控制两组电机的四个引脚为输出
   {
      pinMode(i,OUTPUT);
   }
    pinMode(12,OUTPUT);//红灯
    digitalWrite(12,LOW);
    Serial.begin(38400); //设置波特率

    node();
    delay(1000);
}
void loop()
{
        int temp_char;
        int start_flag = 0;
        int data_index=0;
        int com_index=0;
        int data[17];
        int dataLen = 0;
        int SUM;
        int i;
        int state;

        while(1)
```

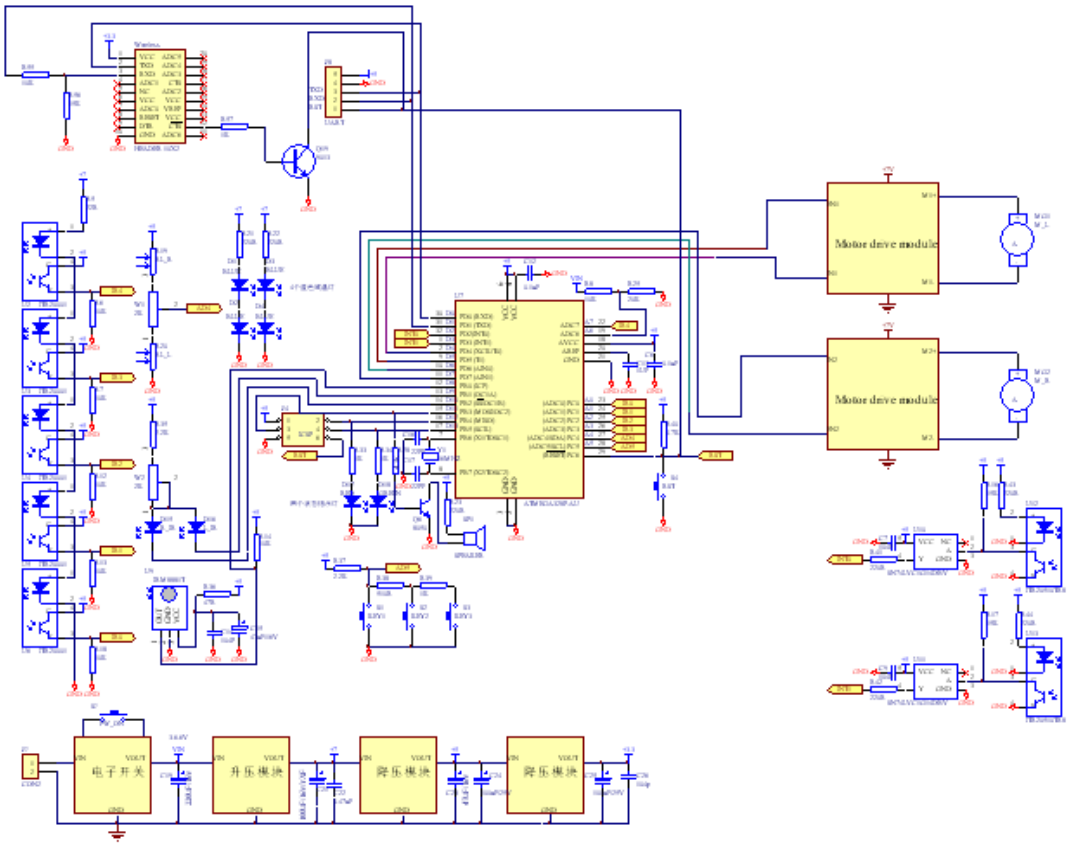
```
{
    temp_char = Serial.read();
    if(temp_char!=-1)
    {
        if(start_flag==0)
        {
            if(temp_char == 126)
            {
                start_flag=1;
                data[0]=126;
                data_index = 0;
               com_index = 0;
            }
        }
        else if(com_index<2)
        {
            switch(com_index)
            {

                case 0: data[1]=temp_char;
                        break;
                case 1: data[2]=temp_char;
                        dataLen=(data[1]*256)+data[2];
                        break;
            }
            com_index++;
        }
        else if(data_index < dataLen)
        {
            data[data_index+3]=temp_char;
            data_index++;
        }
        else
        {
            data[16]=temp_char;
            state=1;     //一帧接收完毕
```

```
                start_flag = 0;//清标志
            }
        }
        if(state==1)
        {
            state=0;
            SUM=0;
            for(i=3;i<16;i++)SUM+=data[i];
            SUM%=256;
            SUM=255-SUM;
            if(SUM==data[16])
            {
                if(data[15]=='L')//接收到字符 L 说明左边有障碍物
                {
                    Motor_Control(BACK,100,FORW,100);//右转
                    delay(60);//延时
                    Motor_Control(BACK,0,FORW,0);//停止
                    delay(60);//延时
                }
                else if(data[15]=='R')//接收到字符 R 说明右边有障碍物
                {
                    Motor_Control(FORW,100,BACK,100);//左转
                    delay(60);//延时
                    Motor_Control(BACK,0,FORW,0);//停止
                    delay(60);//延时
                }
            }
        }
    }
}
```

程序功能：接收 A 车发送的字符，如接收到字符 L，说明左边有障碍物，小车右转。将做为目的模块的 xbee 模块分别插在 B、C 小车上。

完成上述后，将三个小车的开关打开。在主控车 A 左边放障碍物，可以看到，A 车右转避障，同时 BC 车也跟着右转避障。

2WDMiniQ 电路原理图



发布日期	版本号	备注
年月日	**V1.0**	建文档
年月日	**V1.1**	
年月日	**V1.2**	