自动化专业——
《机器人设计与实践》实验指导书

第一版

哈尔滨工业大学（深圳）

2018 年 7 月

实验报告要求

一、实验报告

实验报告包括两大部分：预习报告和实验结果分析，其中，实验上课之前完成预习报告，实验课结束后，完成实验结果分析。

二、上机实验

要求：提交电子版文件。

三、预习报告要求

撰写内容：实验目的，实验原理，实验步骤及操作要点。

要求：用自己的理解简要的写，切记抄写指导书的全部内容。

四、实验结果分析要求

根据真实的实验结果（结论）、指导书提出的思考问题，以及实验操作过程中遇到的问题及其解决方法等撰写。

要求：简洁明了。

目　录

第一章 设备认知

1.1 SK3325J 程控直流稳定电源

可实现 3 路输出，稳压、稳流值均步进可调，每路输出均有步进可调的过压保护功能，其中：

CH1 输出：电压范围 0-32V，电流范围 0-5A，

CH2 输出：电压范围 0-32V，电流范围 0-5A，

CH3 输出：电压范围 0-6V，电流范围 0-5A。

图 1.1 前面板按键介绍

1.1.1 设置电压方法

按下【CH1】/【CH2】/【CH3】可以选择需要设定的通道。

1 选择要设定的通道
2 按下【Volt】
3 使用数字键盘输入设定值
4 按下【Enter】确认，
【Cancel键取消】

图 1.2 设置电压方法一

按下【CH1】/【CH2】/【CH3】可以选择需要设定的通道。

1 按⇩方向键
2 按下左右方向键选择设置位
3旋转旋钮改变相应参数
4按下【Enter】确认，
【Cancel键取消】

图 1.3 设置电压方法二

1.1.2 设置电流方法

按下【CH1】/【CH2】/【CH3】可以选择需要设定的通道。

1 选择要设定的通道
2 按下【Curr】
3 使用数字键盘输入设定值
4 按下【Enter】确认，
【Cancel键取消】

图 1.4 设置电流方法一

按下【CH1】/【CH2】/【CH3】可以选择需要设定的通道。

1按⇧方向键
2 按下左右方向键选择设置位
3 旋转旋钮改变相应参数
4按下【Enter】确认，
【Cancel键取消】

图 1.5 设置电流方法二

1.3 设置过压保护值

1 选择要设定的通道
2 按下【OVP】键
3 使用数字键盘输入过压保护值
4 按下【Enter】确认，
【Cancel键取消】

图 1.6 设置过压保护值

1.4 打开与关闭输出

按下红色箭头所指三个按键可打开或关闭相应通道的输出。按键指示灯会点亮或熄灭指示开关状态。在输出状态下，相应通道数码管会显示实际输出电压、电流、功率，指示灯指示该通道的输出模式（CV）、（CC）。

图 1.7 打开与关闭输出

1.2 MSOX2012A 示波器

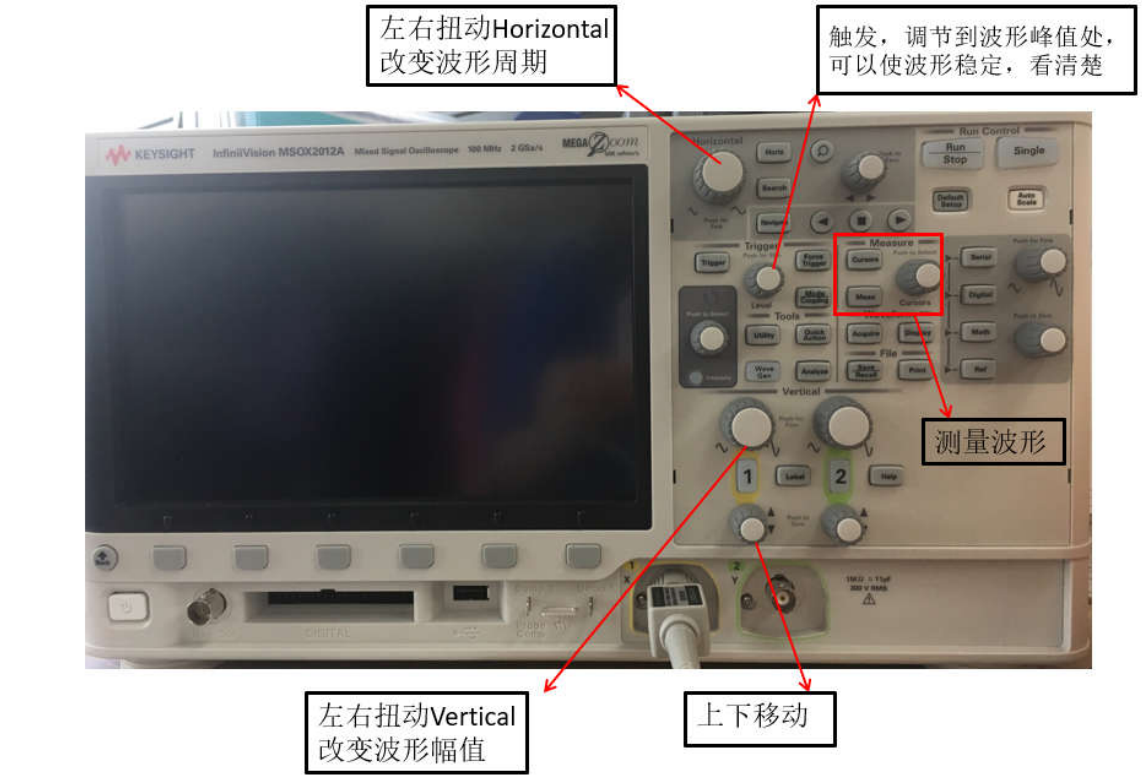
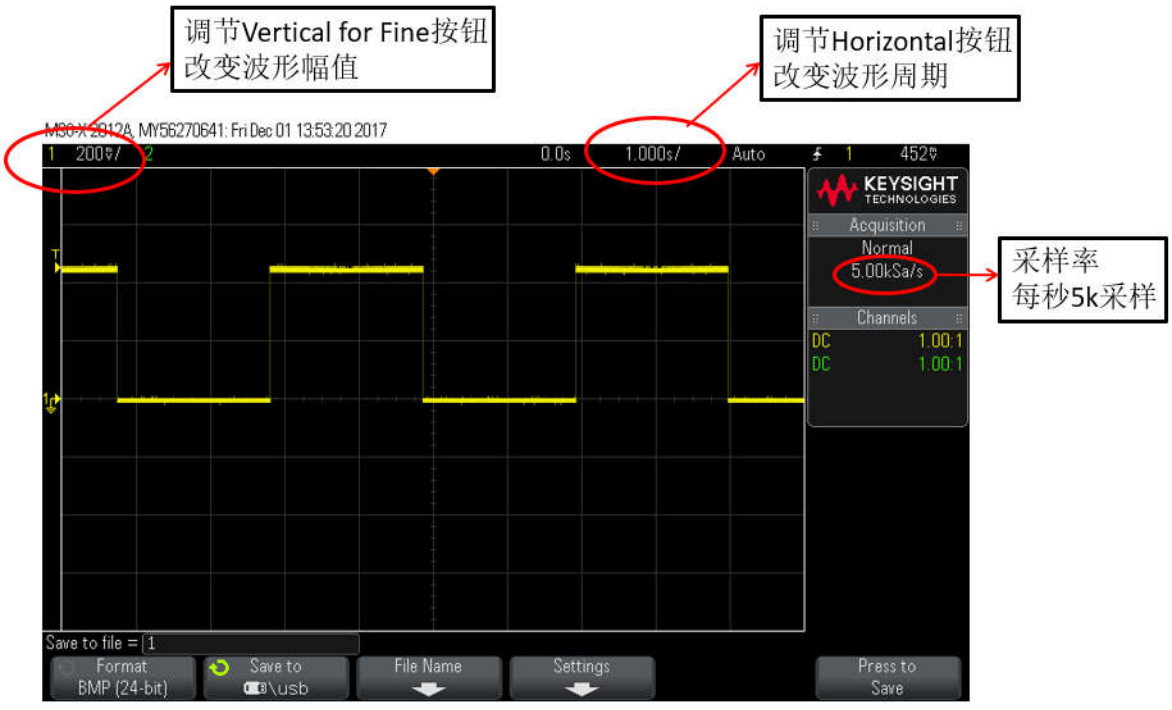


图 1.8 示波器常用按钮功能介绍



注意：探头是10:1衰减的

图 1.9 示波器测量波形说明

1.3 任意波形发生器 AFG1062

图 1.10 波形发生器按键说明

1.4 万用表 FLUKE 287C

图 1.11 万用表面板按钮说明

功能选择键：F1 F2 F3 F4分别对应屏幕最下方的4个功能方块。
比如：如图，当选择测试直流电压后，按下F1 Menu按键，则出现AC+DC，AC,DC，DC,AC，close的选择菜单，可进行直流电压的直流分量和交流分量的测试

图 1.12 万用表功能选择键说明

选择了交流电压档位测试后，按下F1 Menu按键，则出现Hz，%，ms等分量可进行选择测试

选择了二极管，电容档位测试后，按下F1 Menu按键，则出现diode，cap等分量可进行选择测试

注：“OL”是英文“OVERLOAD”的缩写,意思是过载

选择了电阻，ns档位测试后，按下F1 Menu按键，则出现Ohms，ns，通断蜂鸣等分量可进行选择测试

注：ns档是检测电导【电阻的倒数】的,1ns等于10的负九次方Ω。

图 1.13 万用表各档位测试说明

第二章 元器件认知

2.1 三极管

三极管 9012、9013、9014、9015 管脚排列都是一样的。

9012——PNP 型晶体三极管，低噪放大；

9013——NPN 型小功率三极管，低频放大；

9014——NPN 型小功率三极管；

9015—— 硅 PNP 管， 用途:低频放大。

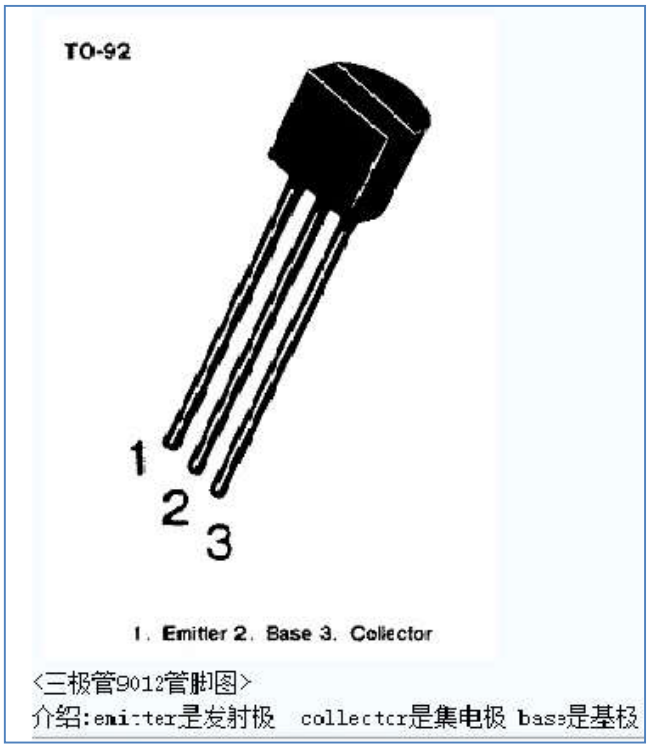


图 2.1 三极管管脚分布图

2.2 二极管

判断正负极方法：（1）有白线的一端为负极。（2）对于发光二极管，引脚长的为正极，短的为负极。（3）如果引脚被剪得一样长，管体内部金属极较小的是正极，大的片状的是负极。（4）打开万用表，将旋钮拨到通断档，将红黑表笔分别接在两个引脚。若有读数，则红表笔一端为正极；若读数为“1”，则黑表笔一端为正极。**实验中会使用二极管 IN5208。**

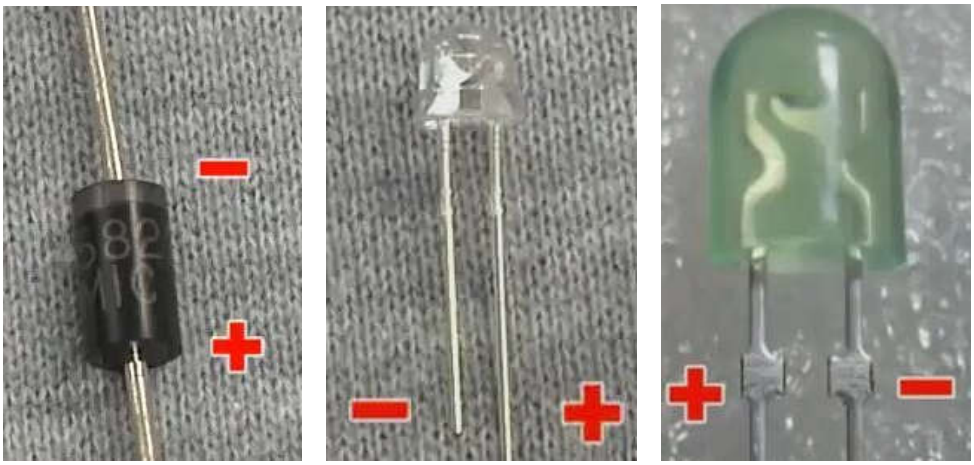


图 2.2 二极管及发光二级管管脚判断

2.3 稳压管二极管

稳压管 1N4733，稳压值为 5.1V，功率为 1W，
稳压管 1N4744，稳压值为 15V，功率为 1 W。

黑色标记的那一头是阴极，稳压管接在电路中阴极接正极（对于电源来说）、阳极接负极。

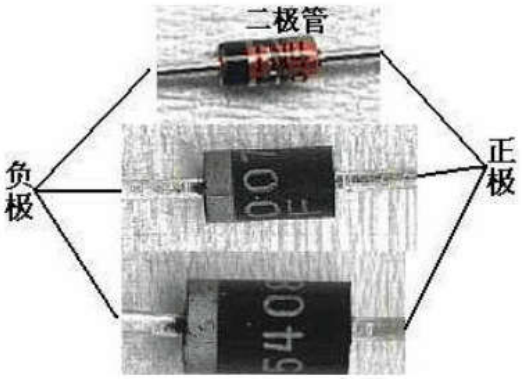


图 2.3 稳压二极管管脚判断

2.4 整流桥 KBPC3510

见图排列方向特殊的那个引脚就是"1" ，是直流正极，1 对角的 3 是负极，2、4 接交流。

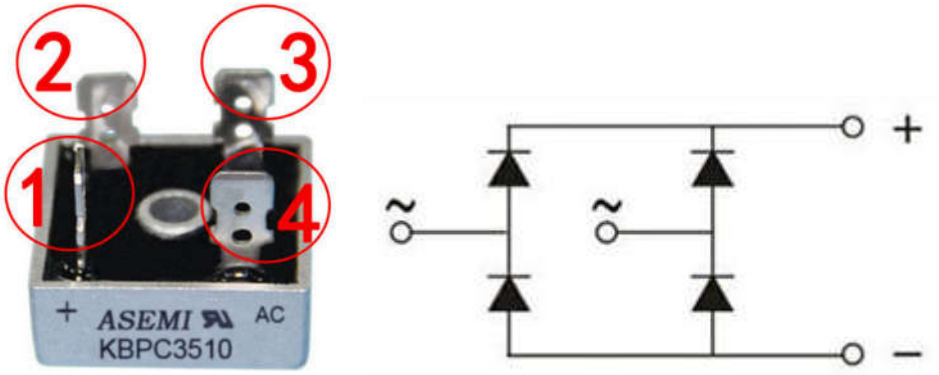


图 2.4 整流桥管脚判断

2.5 电源模块

电源模块，输入电压 4.5V~50V，输出 3V~35V。

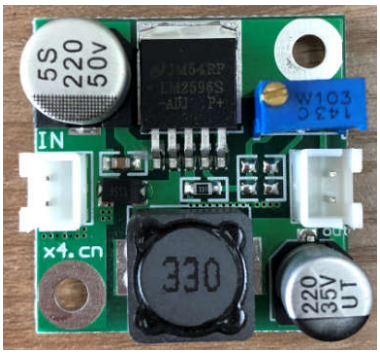


图 2.5-1 电源模块 LM2596 DC-DC 变换器

电源模块，输入电压 7~30V，输出 5V 和 3.3V。

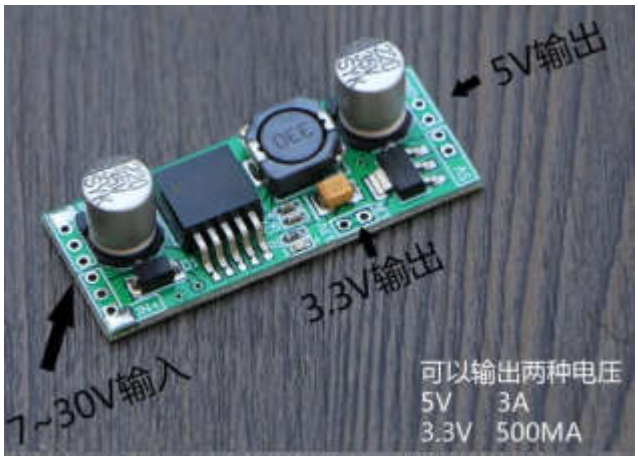
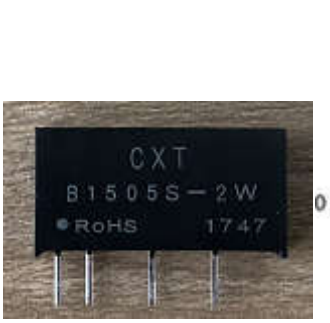
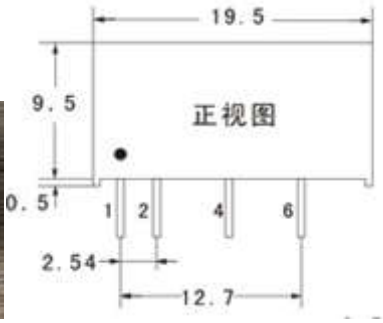


图 2.5-2 电源模块

2.6 隔离电源

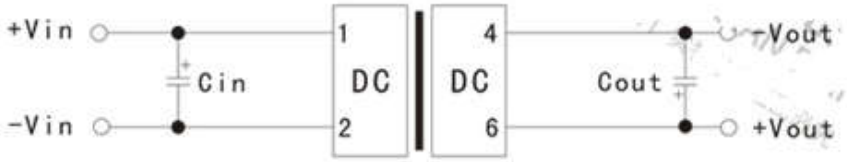




1	2	3	4	5	6
+Vin	-Vin	No Pin	-Vout	No Pin	+Vout
输入正	输入负	无引脚	输出负	无引脚	输出正

图 2.6 隔离电源 B1505S-2W

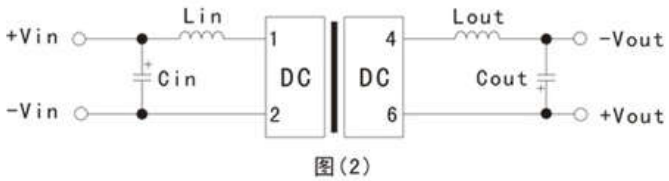
推荐电路一：对于纹波噪声要求一般的场合，可在输入端和输出端各并联一个滤波电容，如下：



Vin (VDC)	Cin	Vout (VDC)	Cout
3.3/5	4.7uF/16V	3.3/5	4.7uF/16V
9/12	2.2uF/25V	9/12	1uF/25V
15/24	1uF/50V	15/24	0.47uF/50V

表(1)

推荐电路二：对于纹波噪声要求严格的场合，参见如下：



图(2)

Vin (VDC)	3.3/5/9/12/15/24
Cin	见表(1)中Cin参数
Lin	4.7uH
Cout	见表(1)中Cout参数
Lout	4.7uH

表(2)

2.7 电源芯片

7815，7812，7805 型号的前两位 78 代表正稳压器、输出正电压，后面的 05 或 12，代表他的输出稳压值是 5V 或 12V。

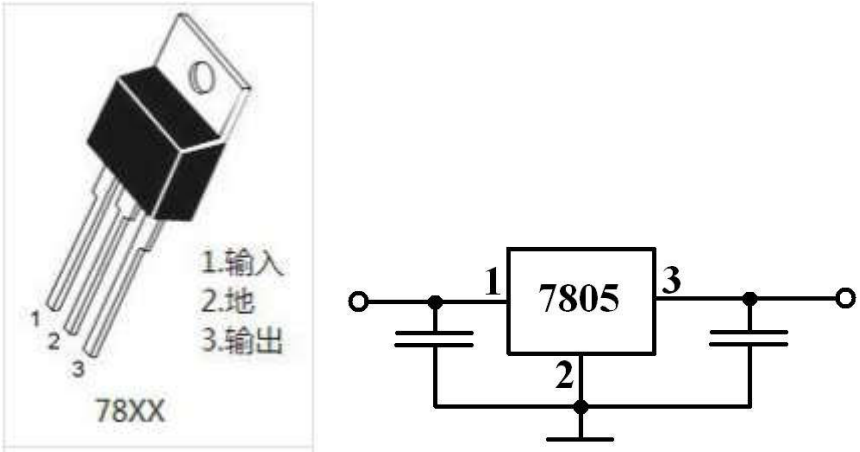


图 2.7 电源芯片引脚及接线

2.8 Mosfet

N 沟道 Mosfet，IRF750A。

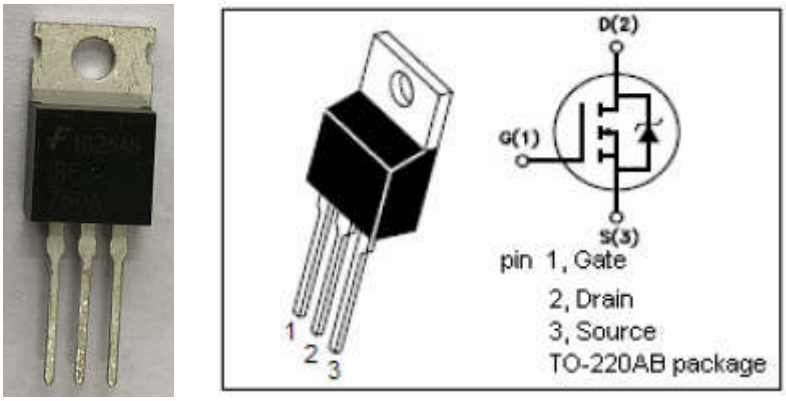


图 2.8 IRF750A 引脚分布图

2.9 半桥驱动

半桥驱动芯片 IR2104（V_{CC} 在 10~20V 之间）。

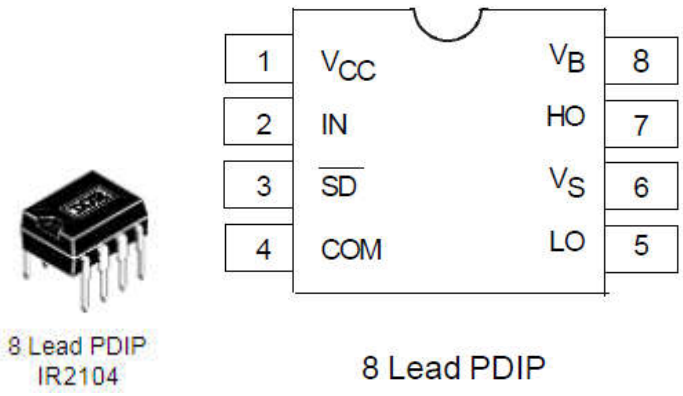
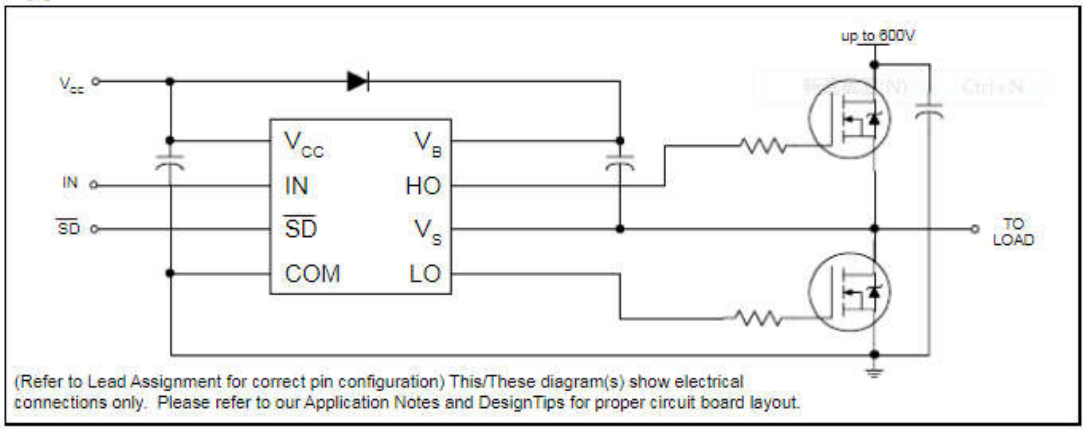


图 2.9 IR2104 芯片图

Lead Definitions

Symbol	Description
IN	Logic input for high and low side gate driver outputs (HO and LO), in phase with HO
$\overline{SD}$	Logic input for shutdown
V_B	High side floating supply
HO	High side gate drive output
V_S	High side floating supply return
V_{CC}	Low side and logic fixed supply
LO	Low side gate drive output
COM	Low side return

Typical Connection



2.10 比较器

比较器 **LM393** 供电电压 2~36V。

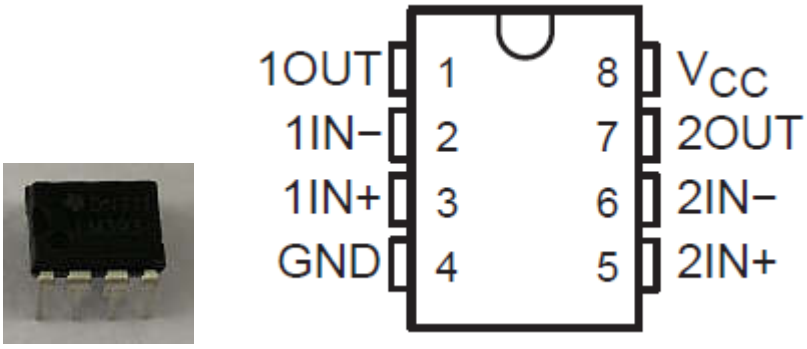


图 2.10 比较器芯片图

2.11 光电隔离芯片

光电隔离芯片 6N137 一般供电电压为 5V。

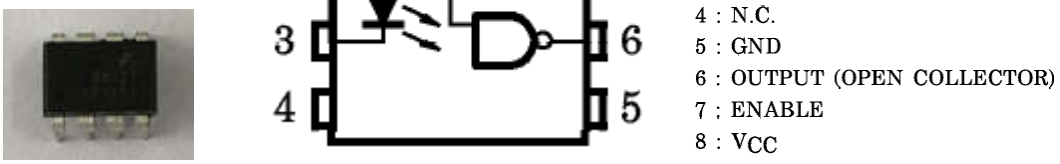


图 2.11 光电隔离芯片 6N137

第三章　电机驱动电路设计实验

一、实验目的

1、了解直流电机控制所需的硬件模块组成；

2、了解直流电机控制电路中各元器件的作用；

3、掌握电路的设计、搭建以及调试方法。

二、实验设备与材料

波形发生器，示波器，万用表，表 1 给出的电子元器件等。

表 1 电子元器件等材料清单

实验器件	功能描述
单片机	产生 PWM 对电机进行调速
变压器、整流桥、电源芯片等	220V 交流电压转换成 12V 等直流电压
12V 直流电机（2 个）	实验电路负载
6N137（2 个）	单片机 PWM 隔离
12V 转 5V 芯片 7805（1 个）	给单片机、芯片等供电
LM393 电压比较器（1 个）	产生互补的 PWM
IR2104 半桥驱动芯片（4 个）	对一桥臂 MOSFET 进行驱动
N 沟道 MOSFET 管 IRF750A（8 个）	MOSFET 开关
二极管 IN5208（4 个）	基本器件
12 个 1K 电阻，8 个 10Ω 电阻	基本器件
8 个 0.1μF 电容，4 个 1μF 电容	基本器件
印刷电路板（1 块）	用于焊接电子元器件
排针，排座，导线，杜邦线等	基本耗材
烙铁，锡，剥线钳等工具	焊接工具

三、实验原理

1、直流电机控制总体方案

通过单片机产生 PWM 波形，经过光电隔离模块，将单片机信号与控制驱动信号进行隔离，避免对单片机产生干扰。由于单片机产生的 PWM 信号无法直接驱动主回路中的 MOSFET，需要通过半桥驱动芯片 IRF2104，提高 PWM 波形的驱动能力，实现 MOSFET 的开通与关断。通过主回路 H 桥电路，实现对直流电机的正反转控制。图 3.1 给出直流电机控制整体方案框图。

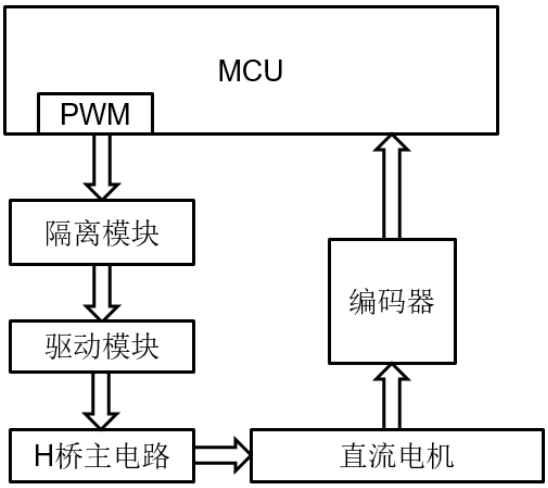


图 3.1 直流电机控制总体方案框图

2、H 桥直流电机调速

（1）H 桥基本介绍

电动机 M 两端的极性随开关器件驱动电压变化而变化，其控制方式有双极性、单极性、受限制单极性等多种，本试验采用的是双极性控制。首先 Q1 和 Q4 导通，Q2 和 Q3 关闭，Uab=Uvccp；然后 Q1 和 Q4 关闭，Q2 和 Q3 导通，Uab=-Uvccp；因此在电机两端产生双极性 PWM，实现对电机的正反转控制。

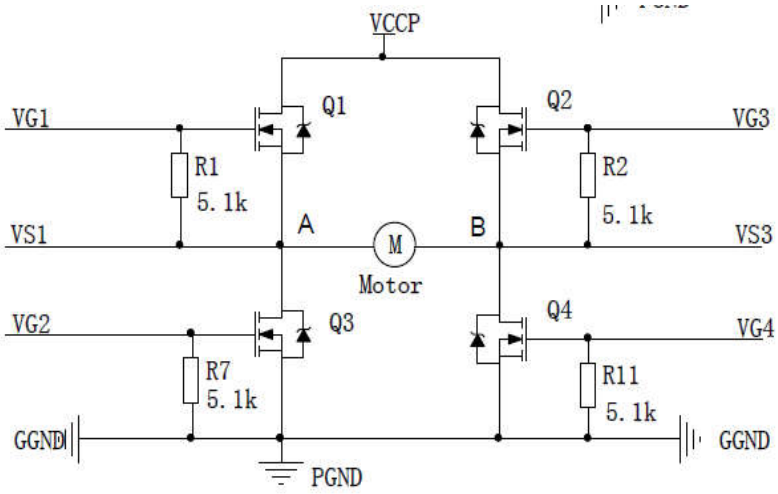


图 3.2　H 桥 PWM 直流电机控制主回路

（2）H 桥工作原理

工作波形：

（a）正向运行

第一阶段，$0 \le t \le t_{on}$期间，U_{g1}、U_{g4}为正，VT_1、VT_4导通，U_{g2}、U_{g3}为负，VT_2、VT_3截止，电流i_d沿回路 1 流通，电动机 M 两端电压$U_{AB} = +U_S$;

第二阶段，$t_{on} \le t \le T$期间，U_{g1}、U_{g4}为负，VT_1、VT_4截止，VD_2、VD_3续流，并钳位使VT_2、VT_3保持截止，电流i_d沿回路 2 流通，电动机 M 两端电压$U_{AB} = -U_S$。

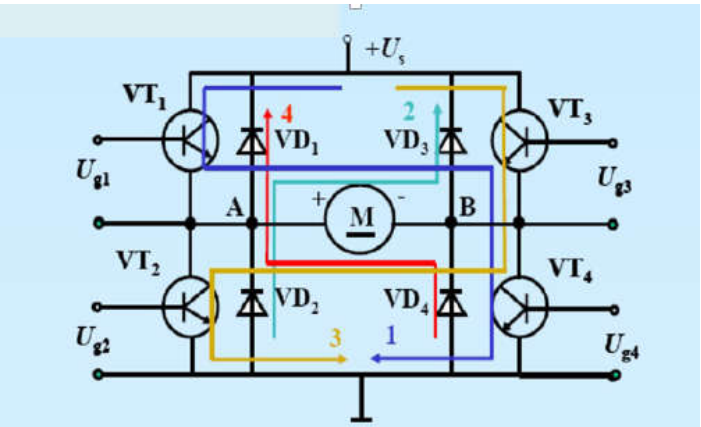


图 3.3 双极式 H 型可逆 PWM 变换电路

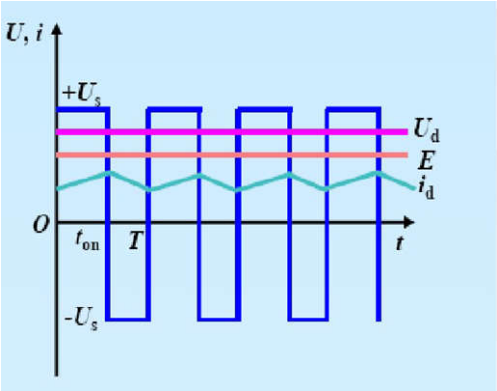


图 3.4 正向电动运行时电压 u, i 波形

（b）反向运行

第一阶段，$0 \le t \le t_{on}$期间，U_{g2}、U_{g3}为负，VT_2、VT_3截止，VD_1、VD_4续流，并钳位使VT_1、VT_4保持截止，电流$-i_d$沿回路 4 流通，电动机 M 两端电压$U_{AB} = +U_S$

第二阶段，$t_{on} \le t \le T$期间，U_{g2}、U_{g3}为正，VT_2、VT_3导通，U_{g1}、U_{g4}为负，使VT_1、VT_4保持截止，电流$-i_d$沿回路 3 流通，电动机 M 两端电压$U_{AB} = -U_S$

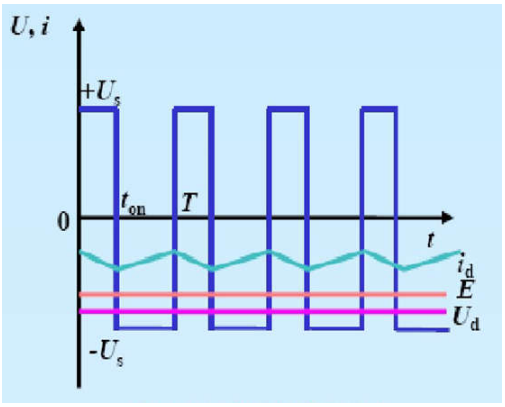


图 3.5 反向电动运行波形

四、实验方法与步骤

1）系统电源的获得方法之——使用变压器和整流桥等

首先得到整个电路所用到的各种电源模块，电机电源 15V，单片机隔离 5V，光耦 5V，驱动 12V

注意：整流滤波电解电容 C1 是有正负极性，不能接反，否则会爆炸！

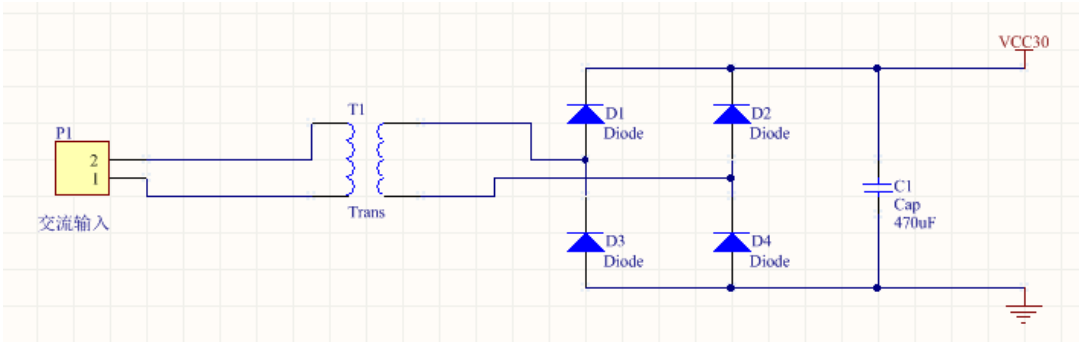


图 3.6　220 转 30V 交流电整流电路

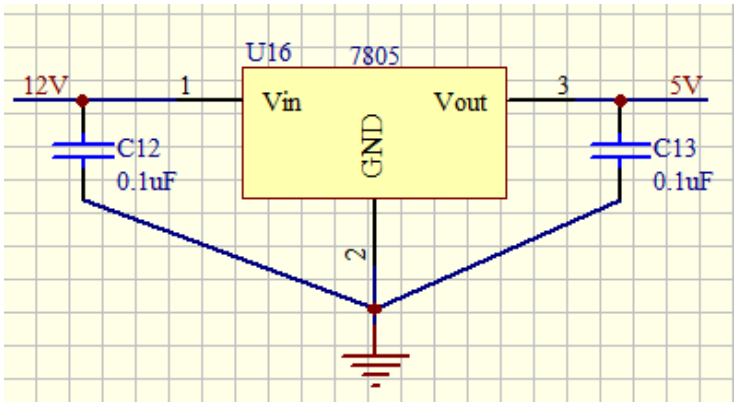


图 3.7　12V 转 5V 电路（使用 7805）

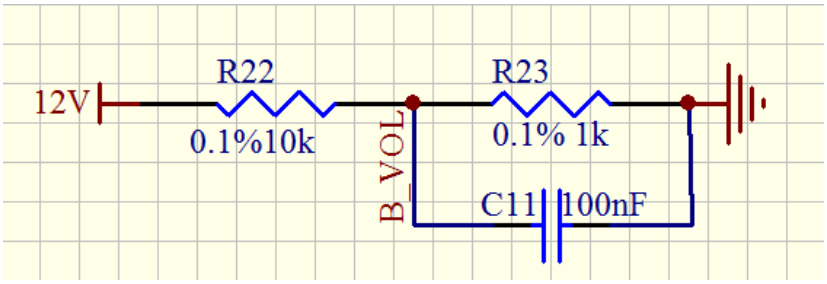


图 3.8 锂电池电压检测电路（低于 9.6V 需要及时充电）

2）得到电源后，对单片机输出的 PWM 进行光耦隔离，使用 6N137 芯片。

提示：这里使用的是反向光耦隔离的连接方法。

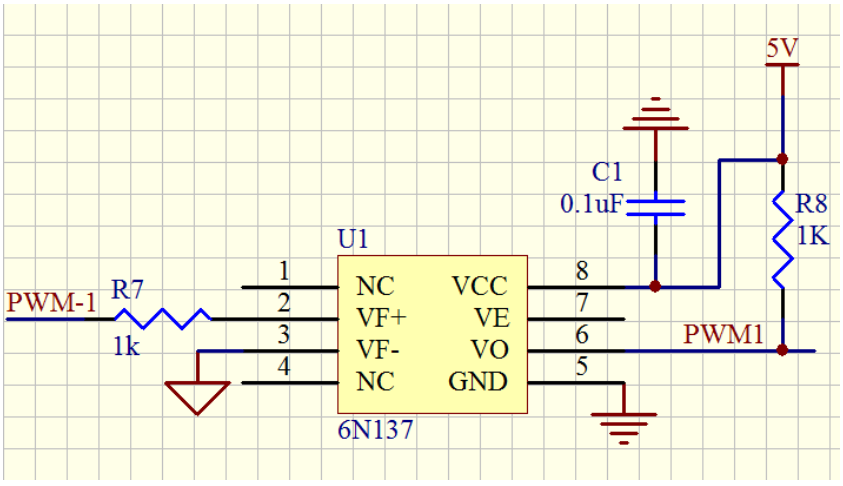


图 3.9 光耦隔离模块原理图

6N137 电路调试方法：

A、2 引脚输入 5V 电压，测量 6 引脚输出电压为 0V 或零点几伏，即输入高电平，输出低电平；相反测试，相反结果。

B、将波形发生器输出的方波接入引脚 2，用示波器查看引脚 6 输出波形，应该得到与输入相反的波形。

3）因为 H 桥是双桥臂，因此还需要一路与原来互补的 PWM 控制另一路桥臂，本实验通过电压比较器 LM393 实现。

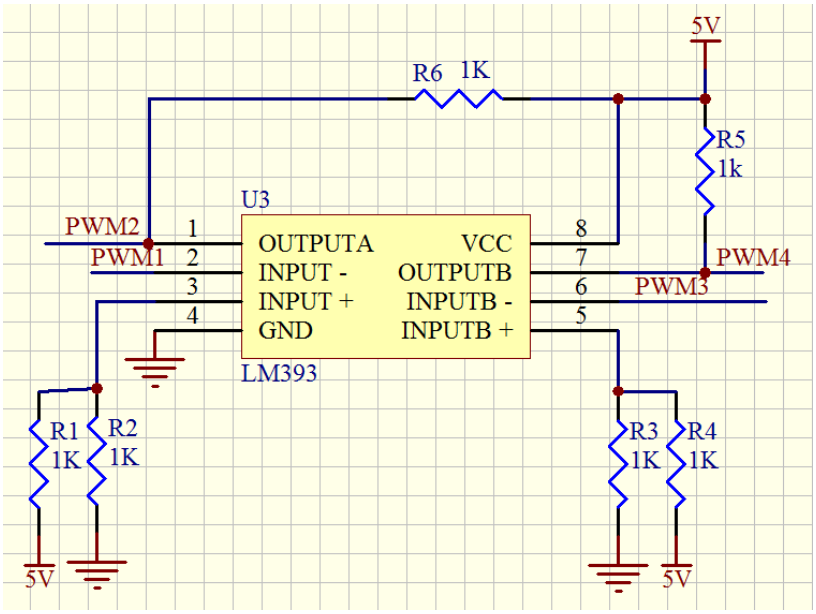


图 3.10　PWM 互补波形获得电路

LM393 电路调试方法：

这里使用 LM393 的反向作为输入，即引脚 2 和引脚 6；

3 和 5 引脚为参考电压，输入的信号与此参考电压相比较，输出反向的电平；

例如：2 引脚输入 5V，即高电平，输出端 1 引脚输出为 0V，即为低电平；反之相反结果。

4)得到两路互补波形后，通过两个 IR2104 半桥驱动芯片实现对 H 桥的驱动控制。注意：2104 自带互补死区，自举二极管需要有较高耐压值。

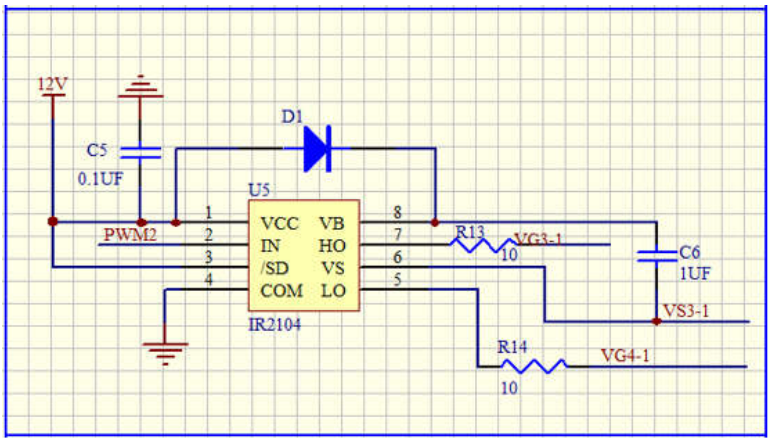


图 3.11　H 桥双桥臂驱动电路-驱动一个直流电机（半桥驱动 IR2104）

IR2104 电路调试方法：

A、输入 PWM 波，占空比为 90%，此时 MOS 管 2、3 导通，

测量第一个 2104，HO 与 VS 间电压为 1V 左右，LO 与 GND 电压为 10V 左右；

测量第二个 2104，HO 与 VS 间电压为 6.8V 左右，LO 与 GND 电压为 1V 左右；

B、输入 PWM 波，占空比为 10%，此时 MOS 管 1、4 导通，

测量第一个 2104，HO 与 VS 间电压为 9V 左右，LO 与 GND 电压为 1.1V 左右；

测量第二个 2104，HO 与 VS 间电压为 0.8V 左右，LO 与 GND 电压为 10V 左右；

5）最后是 H 桥电路

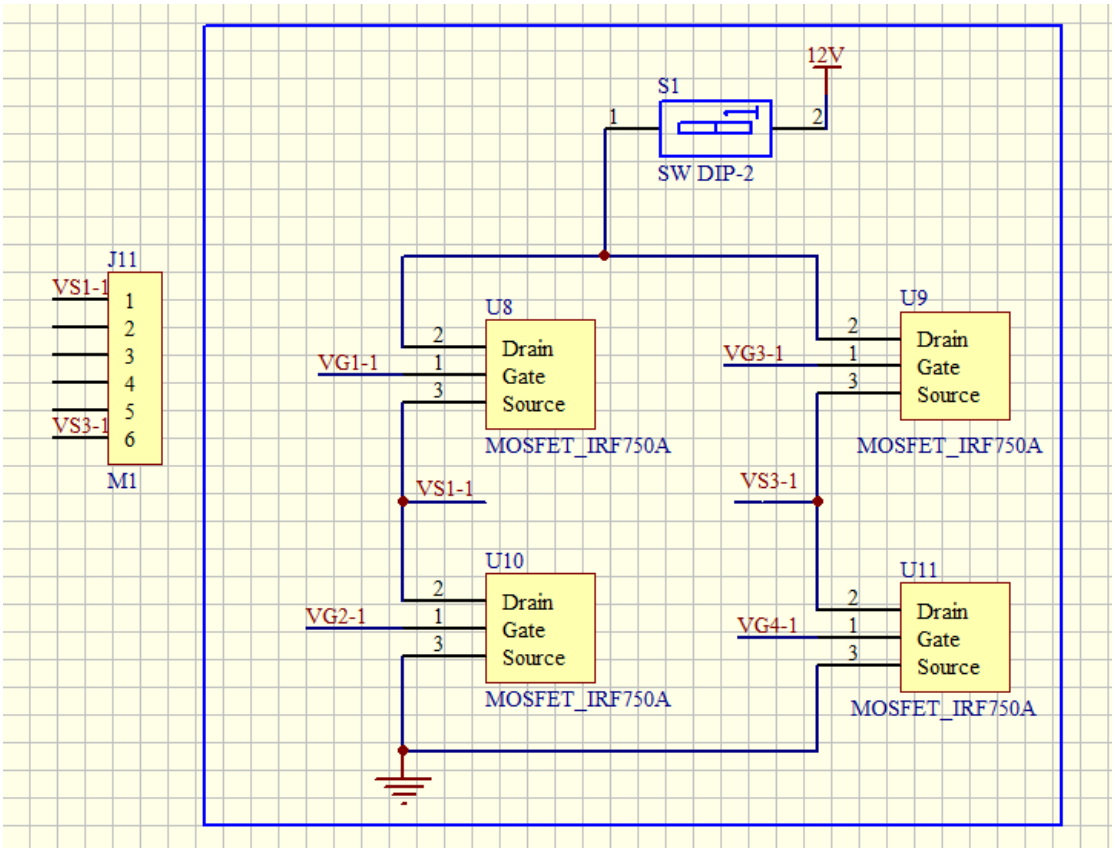


图 3.12　H 桥主回路电路-电机 1(NMOS 管为 IRF750A)

IRF750A 电路调试方法：

使用波形发生器，给出 PWM 占空比可调的波形，高电平为 5V，低电平为 0V；

将 PWM 接入到一路电机控制电路中：

占空比小于 50%，电机顺时针转动；

占空比大于 50%，电机逆时针转动；

占空比等于 50%，电机停止转动。

（注：电机正反转会因电机正负端接线不同而不同）

6）整个电路完整原理图

见下一页图 20，完整直流电机控制电路原理图。

五、思考问题

1）不加光耦会产生什么样的问题？

2）MOSFET 管的工作原理。

2）H 桥电路在一个周期的工作原理。

六、实验报告要求

用自己的理解写出电机驱动电路设计的原理与设计方法，附电路图说明。

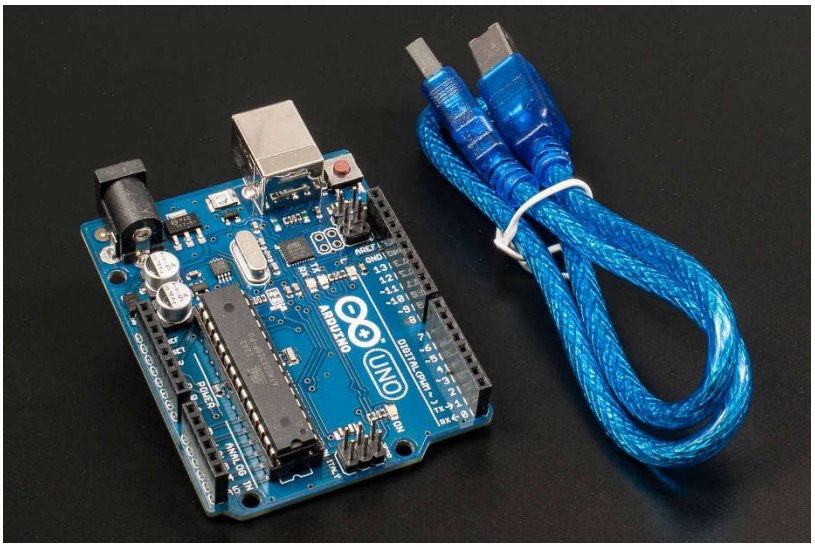
第四章 Arduino 入门电路设计实验

这部分实验包括三个实验项目：1、LED 灯的亮灭控制；2、使用 PS2 遥杆模块控制舵机旋转；3、测量温湿度及 OLED 屏幕显示。

预准备

1、准备 Arduino/Genuino 开发板和 USB 连接线

在这个教程里，我们默认你用的是一块 Genuino Uno. 如果你使用的是其他电路板，请在开始介绍手册里面阅读对应的内容。你也可以用一条标准的 USB 线（A 接口对 B 接口），比如说你连接到打印机 USB 接口用的线。（对于 Genuino Micro，你需要的是一条 A 接口对 MiniB 接口的线）



2、连接开发板

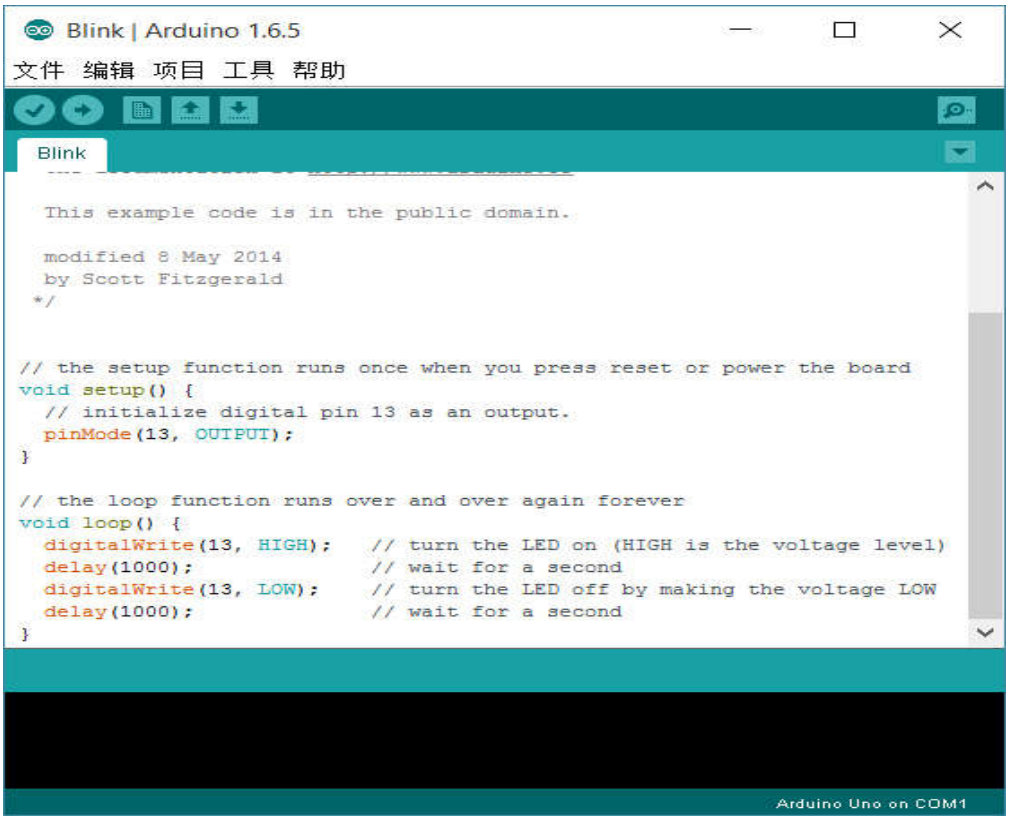
Genuino UNO 能自动通过连接到电脑的 USB 线或者外部的供电来获取电源。你可以用 USB 电源给板子供电。当你用 USB 线把 UNO 板子和电脑连接起来后，你会看到电路板上面绿色的 LED（标注着“PWR”）被点亮。

3、启动 Arduino 软件（IDE）

双击电脑 Arduino 应用程序。(注：若 Arduino 软件加载所用语言不适合，您可以通过系统配置对话框更改设置)。

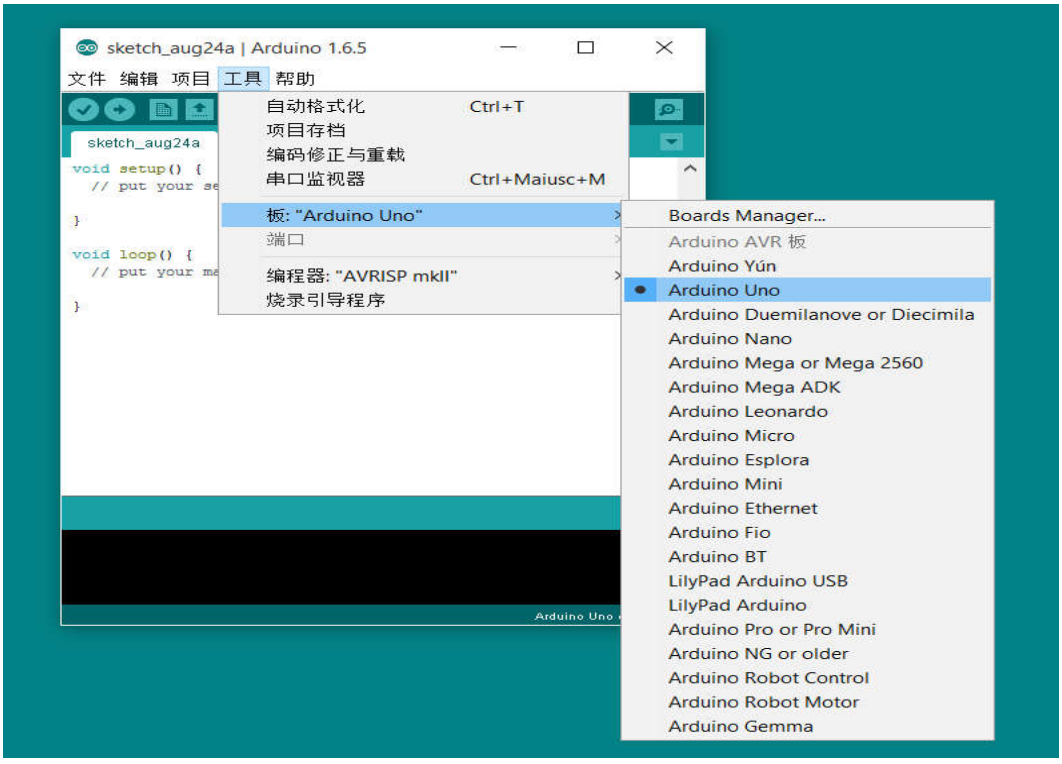
4、 打开闪烁示例

打开 LED Blink 样例代码：文件>示例>01.Basics > Blink。



5、选择开发板

你将需要在“工具〉开发板”菜单选择对应的 UNO 板子，上面会列着 Arduino/Genuino 以及板子的类型（Uno, Mega 2560 or Micro).不同版本 IDE，名称可能会变，选择 Arduino/Genuino Uno 板子。



6、选择串行端口

从“工具|端口”菜单中选择 Arduino/ Genuino 开发板的串口设备，可能是 COM3 或更高版本设备((COM1 和 COM2 通常用于硬件串行端口)。为找到串口设备，可以先断开 Arduino/Genuino 开发板，并重新打开菜单；消失不见的条目即为 Arduino/开发板对应的条目。重新连接开发板，并选择上述刚才消失的串行端口。

7、上传程序

现在，你只需要在 Arduino 软件（IDE)点击“上传”按钮. 稍等片刻，你就会看到板子上面标有 RX 和 TX 的 LED 灯在闪。 如果上传成功,状态栏会出现“上传成功”的字样。

上传完之后，你会很快看到板子 13 号针脚（L）的 LED 开始在闪。这证明上传成功了，你已经把 Genuino 和 Arduino 软件（IDE)跑起来了！

实验模块一 LED 灯的亮灭控制

一、实验目的

了解 LED 的控制方法。

二、实验设备

Arduino 开发板，示波器，三色的 LED 模块，面包板，杜邦线

三、实验原理

本实验用 Arduino 的 GPIO 管脚来控制发光二极管（LED）亮度。由于 LED 的亮度与施加的电压有关，因此可以通过控制电压来达到调节亮度的目的。如图 4.1 所示，给 LED1_P 引脚输出低电平或者输出 PWM 信号来点亮 LED。

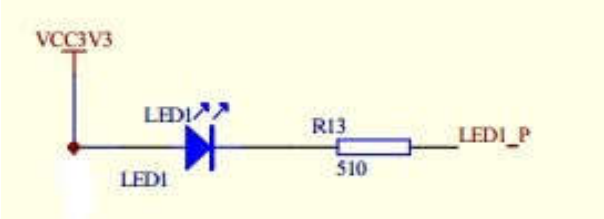


图 4.1 LED 电路

Arduino 是一个开源硬件组织，其开发的系列开发板是带有 USB 和丰富 I/O 接口的嵌入式开源硬件开发板，具有类似于 Java、C 语言的集成开发环境，既可以外接一些电子元器件，例如开关、传感器、LED、直流马达或其他输入、输出装置，也可以独立运行，成为一个可以跟上位机软件沟通的接口装置，例如：Flash、Processing 等互动软件。Arduino 开发环境 IDE 全部开放源代码，可以免费下载，还可以开发出更多互动作品。如图 4.2 和图 4.3 所示，分别为 Arduino 硬件平台的实物图和电路布局图。

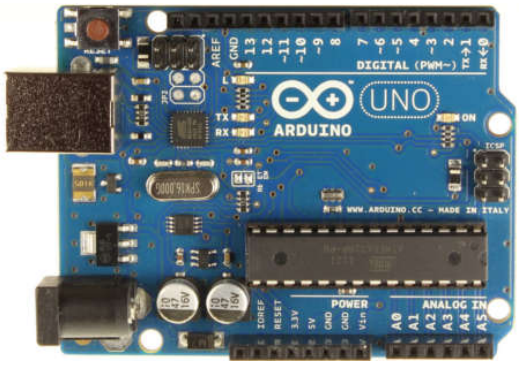


图 4.2 Arduino 实物图

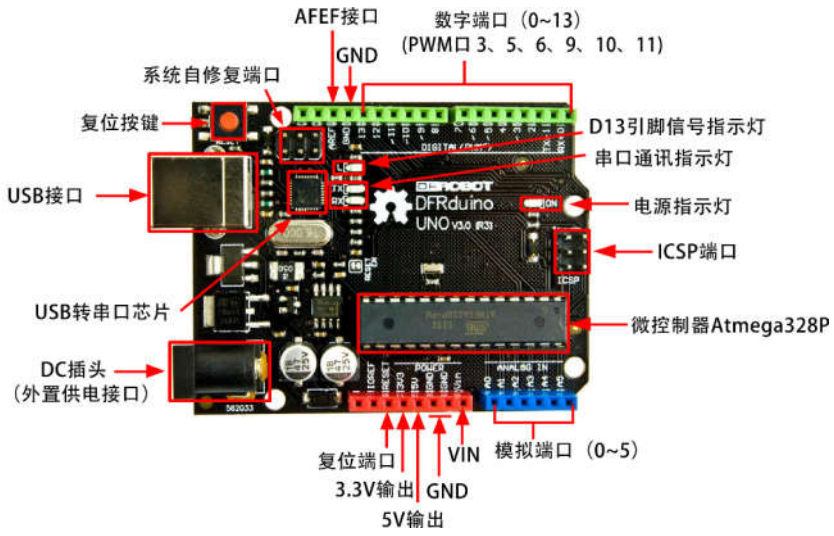


图 4.3　Arduino 硬件平台电路布局图

LED（Light Emitting Diode）是发光二极管的简称。这种半导体元件一般是作为指示灯、显示板，它不但能够高效率地直接将电能转化为光能，而且拥有最长达数万小时的使用寿命，同时具备不像传统灯泡易碎并省电等优点。LED 可分为普通单色 LED（红、绿、黄、蓝、白）、高亮度 LED、超高亮度 LED、变色 LED、闪烁 LED、电压控制型 LED、红外 LED 和负阻 LED 等。

LED 特点有：

- 工作电压低，工作电流很小；
- 适合于易变的环境，对环境无污染；
- 稳定性好，可靠性高，寿命长；
- 价格比较低廉，性价比高。

四、实验内容

由于 Arduino UNO 板卡的 I/O 口只能产生数字电压，即高电压（5V）或者低电压（0V），而不能产生连续变化的模拟电压，因此必须采用脉宽度调制技术（PWM，Pulse Width Modulation）来模仿模拟电压。

在 Arduino 数字 I/O 端子 3、5、6、9、10 和 11（板卡上引脚标号带有~符号）上，我们可以通过 analogWrite()函数来产生模拟输出。该函数有两个参数，其中第一个参数是要产生模拟信号的端子（3、5、6、9、10 或者 11）；第二个参数是用于产生模拟信号的脉冲宽度，取值范围是 0 到 255。脉冲宽度的值取 0 可以产生 0V 的模拟电压，取 255 则可以产生 5V 的模拟电压。不难看出，脉冲宽度的取值变化 1，产生的模拟电压将变化 0.0196V（5/255 = 0.0196）。实验电路原理图如图 4.4 所示。从图中可见，LED 连接的是 D11 端子。

注意：接好线后，一定要仔细检查正负电压有没有接反，然后再上电；若是接反，有可能导致电路板烧毁。

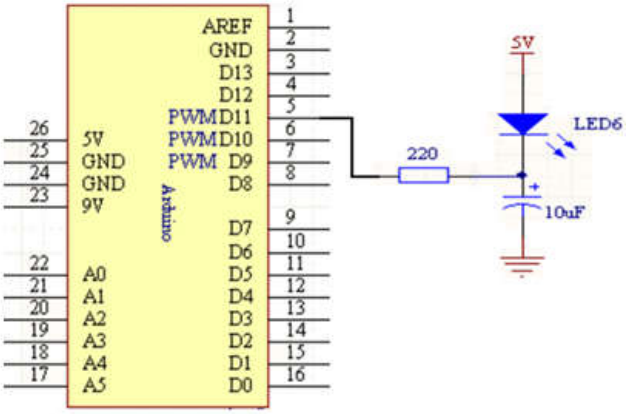


图 4.4 电路原理图

本次实验直接使用一种三色的 LED 模块，其使用原理与上图类似，电路实物图如图 4.5 所示。

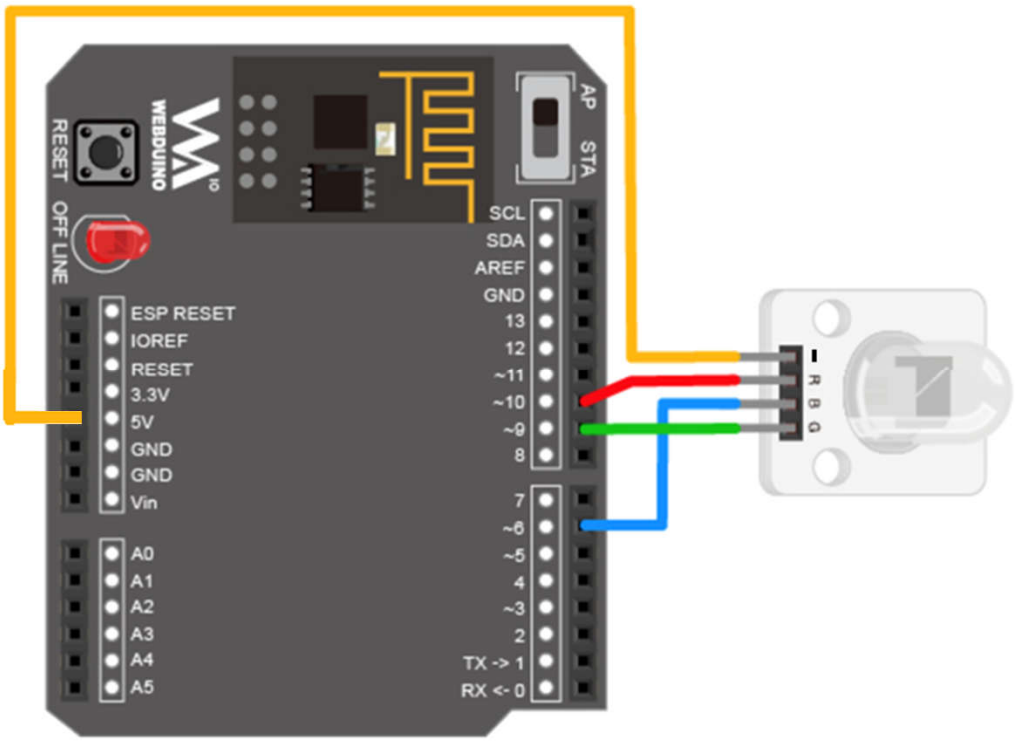


图 4.5 电路实物图

呼吸灯实验参考程序：

```
1.  int ledPin = 10;        //设定控制 LED 的数字 I/O 脚，对应 LED 的红灯
2.  void setup () {
3.      pinMode(ledPin, OUTPUT);
4.  }
5.  void loop() {
6.      for (int a=0; a<=255; a++) { //循环语句，控制 PWM 亮度的增加
7.          analogWrite(ledPin,a);
8.          delay(8); //当前亮度级别维持的时间,单位毫秒
```

```
    }
    for (int a=255; a>=0; a--) { //循环语句，控制 PWM 亮度减小
        analogWrite(ledPin,a);
        delay(8); //当前亮度的维持的时间,单位毫秒
    }
    delay(300); //完成一个循环后等待的时间,单位毫秒
}
```

通过调用 analogWrite()函数，在 Arduino UNO 的 10 号数字端口上模仿输出一个模拟电压给 LED，每产生一次输出后都设置了相应的延时。实验中可以观察到发光二极管逐渐变亮后再逐渐变暗的效果。

注意：如果出现上传失败的提示：查看菜单栏“工具-端口”是否选择了正确的端口。

五、实验要求与思考问题

1、请使用示波器，看看管脚所输出的波形；

2、上述代码只使用了命令 analogWrite()，请使用 digitalWrite() 控制 LED 灯；

3、上述代码只控制了管脚 10，请尝试下其他管脚，如果 RGB 三种颜色的管脚都使用了，会有什么效果？

实验模块二 使用 PS2 遥杆模块控制舵机旋转

一、实验目的

1、掌握 PS2 遥杆原理；

2、掌握 Arduino 对 PS2 的数据读取；

3、掌握舵机的控制原理；

4、掌握 Arduino 对舵机的控制。

二、实验设备

Arduino 开发板，舵机，PS2 游戏遥杆，杜邦线，面包板

三、实验内容与实验原理

使用 Arduino Uno 开发板，实时读取 PS2 遥杆模块的当前的通道数值，并据此来控制舵机旋转。以下针对 PS2 和舵机这两个模块的工作原理分别进行介绍。

1、PS2 游戏遥杆工作原理

PS2 游戏双轴摇杆传感器模块，采用金属 PS2 摇杆电位器制作，具有(X,Y)2 轴模拟输出，对应 URx，URy 引脚，随着遥杆方向不同，对应端口会输出不同的电压值。(Z)1 路按钮数字输出，对应 SW（switch 即按钮）引脚，SW 引脚按下去时输出低电平，反之输出高电平。配合 Arduino 可制作遥控器等互动作品。

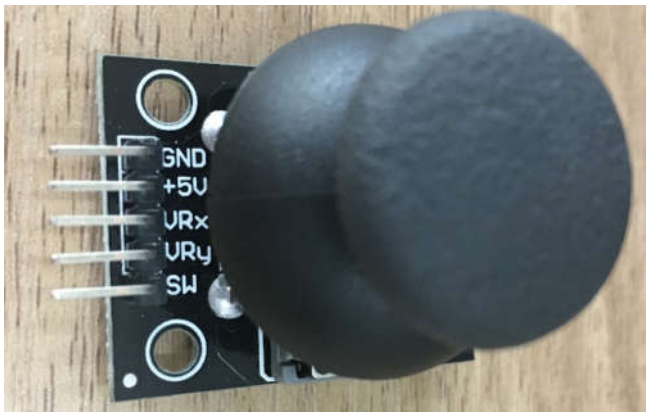


图 4.6　PS2 摇杆图

根据不同功能的需求 X、Y 轴还有复位和不带复位之分。按照阻值不同，摇杆电位器常用的有 10K 和 5K 的。根据材料不同分类，如：

碳膜电位器：工作原理是靠一个金属片摩擦碳膜，通过金属片在碳膜上不同的位置来判断你的移动位置；

光电电位器：光电电位器是通过发光器的光透过光栅论，由光珊论的移动变化来判断你移动的位置。光电电位器的好处是无磨损，除非芯片自己烧了，可以说它是没有坏的可能性的，而且相当精准也很细腻，抗干扰能力强。但光电电位器也有缺点，那就是曲线过于直，而且由于采用光电的电位器的基准经常会以你移动的范围为中心点自动进行校准，从而会导致使用时不方便。

霍尔磁感应电位器：电位器垂直在磁场中时，两侧由此会产生电位差，成为霍尔效应。磁感的好处是温度稳定性好，无磨损，精准细腻，曲线表现好于光电。

2、舵机 SG90 工作原理

舵机也叫伺服电机，最早用于船舶上实现其转向功能，由于可以通过程序连续控制器转角，因而被广泛应用智能小车以实现转向以及机器人各类关节运动中。舵机是小车转向的控制机构，具有体积小、力矩大、外部机械简单、稳定性高等特点。图 4.7 为舵机的外形图。

图 4.7 舵机外形图

一般来讲，舵机主要由以下几个部分组成，舵盘、减速齿轮组、位置反馈电位计、直流电机、控制电路等，如图 4.8 所示。

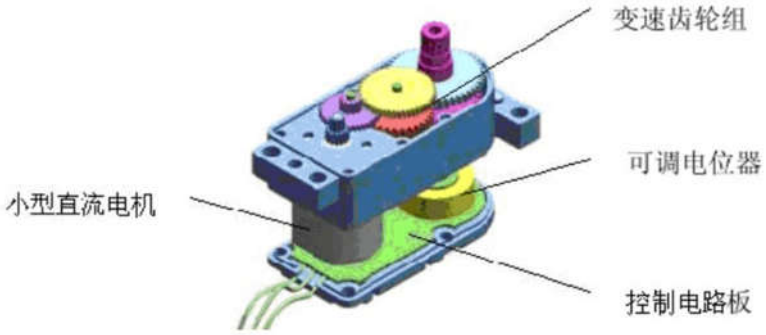


图 4.8 舵机组成示意图

舵机的输入线共有三条，如图 4.9 所示，红色中间，是电源线，一边黑色的是地线，这两根线给舵机提供最基本的能源保证，主要是电机的转动消耗。电源有两种规格，一是 4.8V，一是 6.0V，分别对应不同的转矩标准，即输出力矩不同，6.0V 对应的要大一些，具体看应用条件；另外一根线是控制信号线，Futaba 牌子的一般为白色，JR 牌子的一般为桔黄色。另外要注意一点，SANWA 牌子的某些型号舵机引线电源线在边上而不是中间，需要辨认。但记住红色为电源，黑色为地线，一般不会搞错。本实验以 SG90 为例舵机 **SG90** 有 3 根线，棕色为地，红色为电源正，橙色为信号线。图 4.12 为舵机实物图。

图 4.9 舵机的输出线

图 4.12 舵机实物图

控制电路板接受来自信号线的控制信号，控制电机转动，电机带动一系列齿轮组，减速后传动至输出舵盘。舵机的输出轴和位置反馈电位计是相连的，舵盘转动的同时，带动位置反馈电位计，电位计将输出一个电压信号到控制电路板，进行反馈，然后控制电路板根据所在位置决定电机转动的方向和速度，从而达到目标停止。其工作流程为：控制信号→控制电路板→电机转动→齿轮组减速→舵盘转动→位置反馈电位计→控制电路板反馈。舵机的控制信号周期为 20ms 的脉宽调制（PWM）信号，其中脉冲宽度从 0.5~2.5ms，相对应的舵盘位置为 0~180 度，呈线性变化。也就是说，给它提供一定的脉宽，它的输出轴就会保持一定的对应角度上，无论外界转矩怎么改变，直到给它提供一个另外宽度的脉冲信号，它才会改变输出角度到新的对应位置上。舵机是一种位置伺服驱动器，转动范围不能超过 180 度，适用于那些需要不断变化并可以保持的驱动器中，比如说机器人的关节、飞机的舵面等。

由图 4.11 可以计算出舵机角度控制与 PWM 脉冲宽度的关系为：

$$\frac{Angle - 0°}{90°} = \frac{Value - 1.5}{1}$$

其中：Angle 为舵机转角，Value 为 PWM 脉冲的宽度。

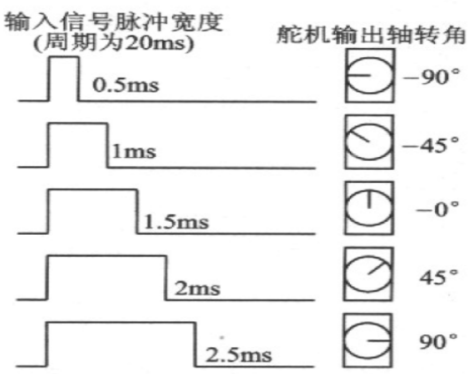


图 4.11 舵机输出转角与输入脉冲的关系

注意：由于舵机牌子不同，其控制器解析出的脉冲宽度也不同，所以对于同一信号，不同牌子的舵机旋转的角度也不同。

3、相关函数介绍

Arduino 自带的舵机控制和 PS2 模块相关的函数及其语句。

Serial 函数（串口用来调试）

```
Serial.begin();      //开启串行通信接口并设置通信波特率
Serial.print();      //写入字符串数据到串口
Serial.println();    //写入字符串数据+换行到串口
```

延时函数

```
delay(ms);              //延时 x ms
```

数字端口配置函数

```
pinMode(pin,mode);      //配置引脚为输出(OUTPUT)或者输入(INPUT)模式
digitalWrite(pin,value); //设置引脚为高电平或者低电平
```

模拟端口配置函数

```
analogRead(pin);        //读取引脚的模拟量电压值，范围 0~1023
```

表 1 Servo 函数表

函数	说明
attach()	设定舵机的接口，只有9或10接口可利用。
write()	用于设定舵机旋转角度的语句，可设定的角度范围是0°到180°。
writeMicroseconds()	用于设定舵机旋转角度的语句，直接用微秒作为参数。
read()	用于读取舵机角度的语句，可理解为读取最后一条write()命令中的值。
attached()	判断舵机参数是否已发送到舵机所在接口。
detach()	使舵机与其接口分离，该接口（9或10）可继续被用作PWM接口。

注：以上语句的书写格式均为“舵机变量名.具体语句（）”例如：

Servo myservo;　　//创建一个舵机控制对象

myservo.attach(9)。

四、实验步骤

步骤一：硬件电路连接

首先，根据实验内容和实验原理，按照下图所示将 PS2 模块和舵机模块与 Arduino 开发板连接。如图 4.14 所示。注：舵机 5V 和 GND 一定要接在开发板对应的电平上。

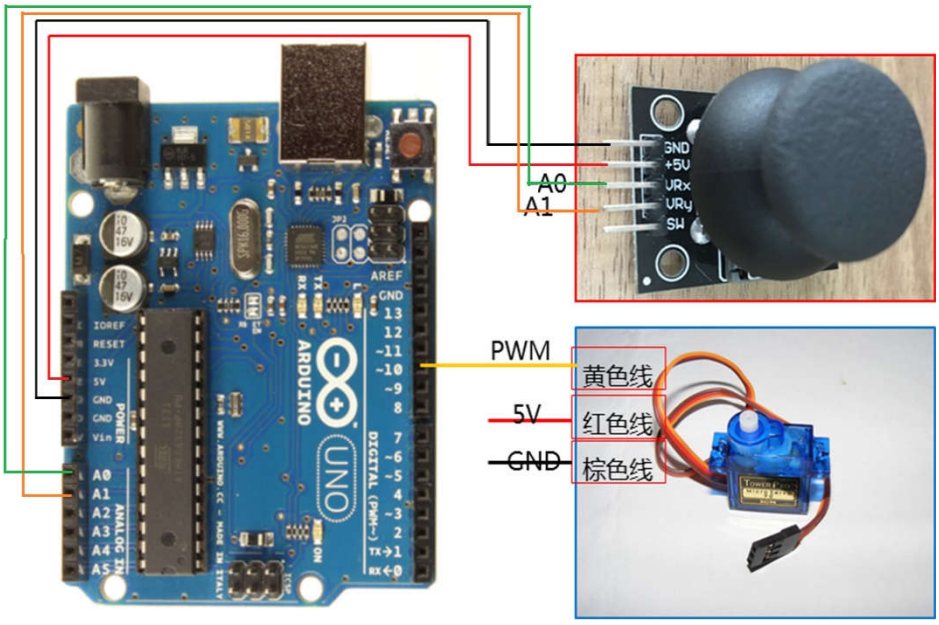


图 4.14　Arduino+PS2+舵机接线图

其中 PS2 模块的 URx 端口连接 Analog 0 (Pin A0)，URy 端口连接 Analog 1 (Pin A1)，VCC 和 GND 连接开发板的+5V 和 GND 端口。舵机 SG90 的 PWM（黄色线为信号线）端口连接 Arduino 的 Digital 10 (Pin 10)引脚，VCC（红色线）和 GND（棕色线）连接开发板的+5V 和 GND 端口，**在本实验中，舵机 5v 电源就暂时使用 Arduino 上的 VCC 接口，但不可长时间使用，因为舵机转动时电流比较大**，Arduino 上的电源芯片可能会因过热而保护甚至损坏，所以正常使用时需要使用外部电源给舵机供电。整个 Arduino 开发板采用 USB 线从 PC 口取电。

步骤二：代码实现

在 setup 函数中，需要先配置 PS2 模块和舵机模块的相关引脚。舵机的 PWM 引脚设置为 PWM 输出。同时，开启了串口功能，有利于代码调试。

在主循环 loop 函数中，通过 URx 端口不断读取 PS2 模块 x 轴（这里只用了 x 轴，作为实例，同学们可以在此基础上继续扩展）的电压对应模数转换值（0-1023），并将其等比例转化成舵机的控制角度（0-180°），最后通过 PWM 实现对舵机的控制。参考代码如下：

```
1.  //定义头文件，这里有一点要注意，可以直接在 Arduino 软件
```

```
//菜单栏点击 Sketch ->Importlibrary -> Servo,调用 Servo 函数，也可以
//直接#include <Servo.h>，但是在输入时要注意在#include 与<Servo.h>之间要
//有空格，否则编译时会报错
#include <Servo.h>
#define PIN_SERVO 10   //定义舵机控制端口
Servo myservo;         //创建一个舵机控制对象

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);   //设置串口波特率，目的是将数据传到电脑串口监控器
  myservo.attach(PIN_SERVO);   //定义舵机接口 9/10，这里我们以端口 10 为例
}

void loop() {
  // put your main code here, to run repeatedly:
  int x, y, angle;
  x = analogRead(A0);
  y = analogRead(A1);
  angle = x / 1023.0 * 180.0;
  myservo.write(angle);         //设置舵机旋转角度
  Serial.print("X: ");
  Serial.println(x);
  Serial.print("Y: ");
  Serial.println(y);
  delay(100);
}

```

步骤三：代码下载与调试

最后，选择正确的板卡型号和端口，将上述程序上传到控制器中。打开串口监视器，查看数据。

五、实验现象

程序运行后，用手沿着 PS2x 轴方向改变遥杆的方向，可以发现舵机会跟着一起旋转。

实验模块三 测量温湿度及 OLED 屏幕显示

一、实验目的

1、掌握 Arduino 对 DHT11 型温湿度传感器的数据读取；

2、掌握 Arduino 的 OLED 屏幕文字显示；

3、进一步熟悉和掌握 Arduino 的使用。

二、实验设备

Arduino 开发板，DHT11 温湿度传感器，OLED 显示屏，面包板，杜邦线

三、实验内容与实验原理

使用 Arduino 开发板，实时读取 DHT11 温湿度传感器的当前数值，并在 OLED 屏幕上显示出来。

1、DHT11 温度模块湿度模块

DHT11 数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。该产品可应用于：暖通空调，测试及检测设备，汽车，数据记录器，自动控制，气象检测，医疗设备，除湿器等。

工作参数：

尺寸：长 28mmX 宽 12mmX 高 7.2mm

工作电压：直流 5V

湿度测量范围：20—90%RH

湿度测量精度：±5%RH

温度测量范围：0—50℃

温度测量精度：±2℃

数字信号输出，数据端口带上拉电阻

2、OLED 显示屏

128*64 的 OLED，是一块功能完整屏幕，利用这块屏幕可以设计一些像贪吃蛇等简单游戏。与液晶相比，OLED 屏幕方式更具竞争力，其中有许多优点，如亮度高，自发射、高对比度、薄细轮廓、宽视角、宽温度范围和低功耗。

3、相关函数介绍

由于温湿度传感器和 OLED 屏幕的使用较为复杂，这里将直接使用 DHT11 和 OLED 的函数库。

DHT 库函数

函数	
begin()	**初始化 DHT 传感器**
readHumidity()	**读取湿度值**
readTemperature()	**读取温度值**

Adafruit_SSD1306 库函数：

函数	
begin()	**初始化 OLED 屏幕**
clearDisplay()	**清除屏幕**
setTextSize()	**设置屏幕文字大小**
setTextColor()	**设置屏幕文字颜色**
setCursor()	**设置文字显示坐标**
println()	**显示文字**
display()	**执行显示**

注：以上语句的书写格式均为“对象名.具体语句（）”例如：

Adafruit_SSD1306 display;

display.begin(SSD1306_SWITCHCAPVCC, 0x3C)。

四、实验步骤

步骤一：硬件电路连接

首先，根据实验内容和实验原理，按照下图所示将 DHT11 模块和 OLED 模块与 Arduino 开发板连接。

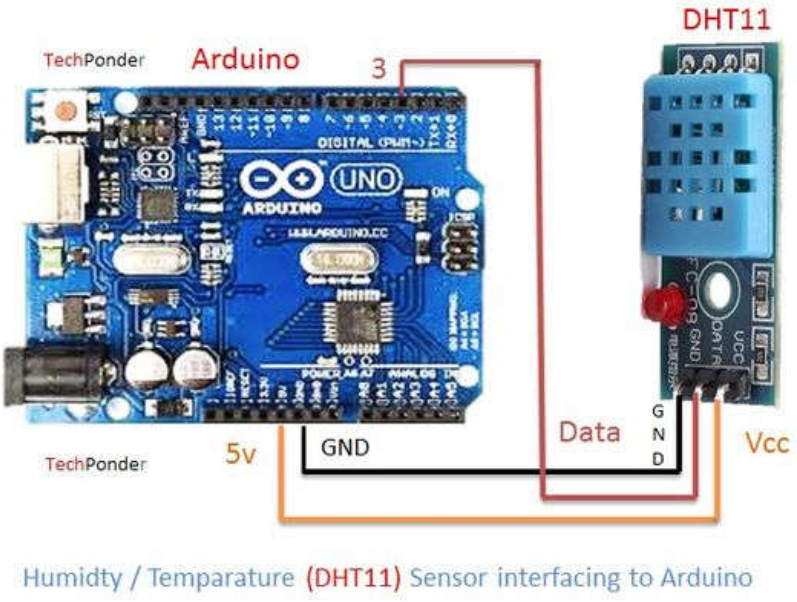


图 4.15 温湿度传感器与 Arduino 硬件连接图

其中，温湿度模块的 DATA 端口连接 Digital8 (Pin 8)，VCC 和 GND 连接开发板的+5V 和 GND 端口。OLED 模块的 SCL 端口连接 Arduino 的 A5 引脚，SDA 端口连接 Arduino

的 A4 引脚，VCC（红色线）和 GND（黑色线）连接开发板的+3.3V 和 GND 端口。整个 Arduino 开发板采用 USB 线从 PC 口取电。（注意：不同的 OLED 模块，有可能 VCC 和 GND 的排布与图 4.16 相反，需要对照模块上的文字正确连接）

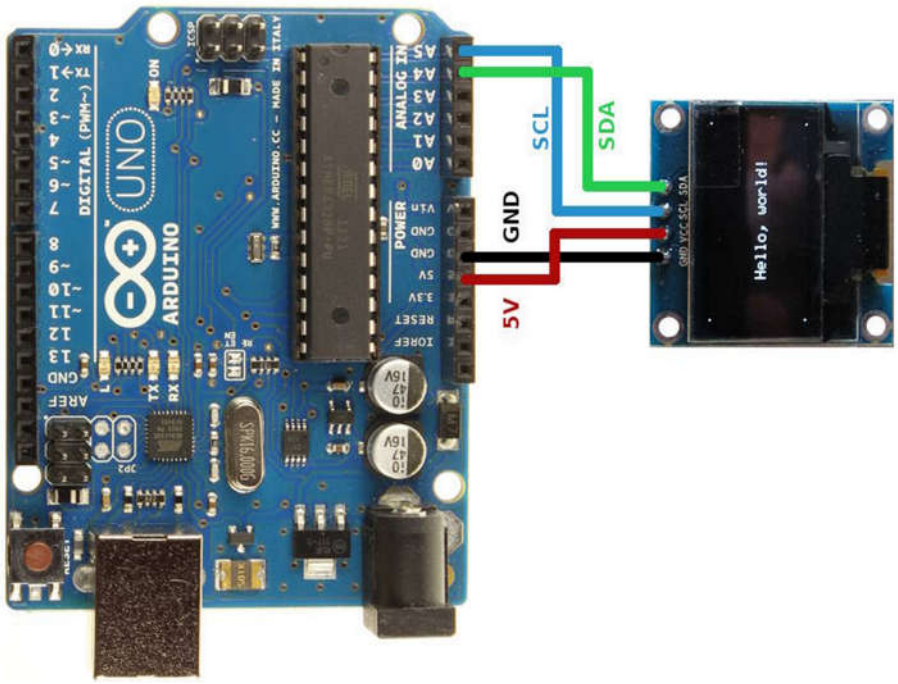


图 4.16 OLED 屏幕与 Arduino 硬件连接图

步骤二：代码实现

1. 首先安装库：点击菜单“项目->加载库->管理库”,搜索 Adafruit SSD1306 库和 DHT。找到“DHT sensor library”并点击安装 1.2.1 版本；或者输入“SSD1306”，找到“Adafruit SSD1306” 进行安装。如下图所示。**注：DHT sensor library 要按照 1.2.1 版本，选择其他版本可能会有编译错误。**

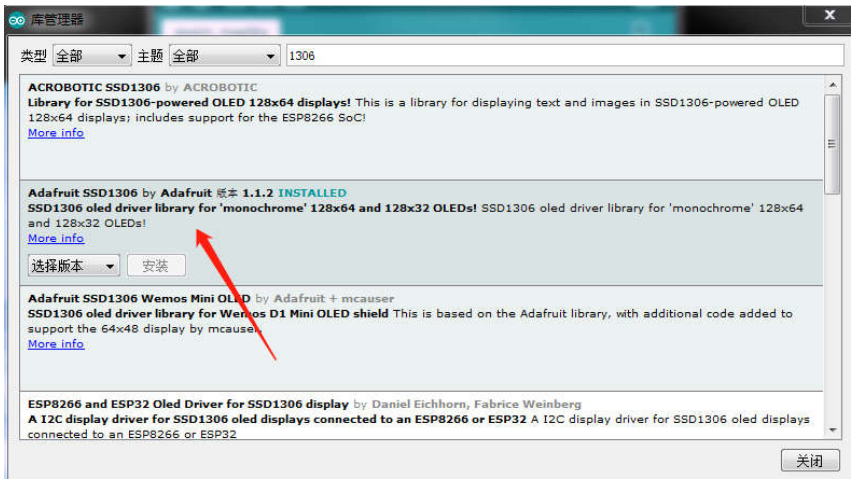


图 4.17 库安装

2. 打开 OLED 屏幕的示例代码（OLED 的主控芯片是 SSD1306）。

图 4.18 OLED 的示例

编译，并看看提示什么错误。

注释掉产生错误的一行，

“//#error("Height incorrect, please fix Adafruit_SSD1306.h!");”

并且将 61 行的代码”display.begin(SSD1306_SWITCHCAPVCC, 0x3D);”

改为：

display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //0x3C 是我们所使用 OLEDI2C 接口地址

观察 OLED 屏幕的现象。

3. 运行如下参考代码：

```
#include "DHT.h"          // 温湿度传感器的头文件，加载库要选择 1.2.3 版本
#include <Adafruit_SSD1306.h>     // OLED 屏幕的头文件
//温湿度传感器
#define DHTPIN 8         // 连接温湿度传感器的管脚为 Arduino 的管脚 8
#define DHTTYPE DHT11    // 温湿度传感器为 DHT11
DHT dht(DHTPIN, DHTTYPE);  //建立温湿度传感器的对象
//OLED
Adafruit_SSD1306 oled;
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  dht.begin();                                  //初始化温湿度传感器
  oled.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // OLED 屏幕初始化，其接口 I2C 的地址为 0x3C (for the 128x64)
  oled.clearDisplay();                          // 清除屏幕
  oled.display();                               // 执行显示,这一步非常重要，每一次显示，都需要运行这一函数
}
void loop() {
  delay(800);   // 两次测量间隔一段时间
  // 读取温湿度值需要 250 ms 左右!
  float h = dht.readHumidity();
  // 读取温度单位为摄氏度 (默认)
  float t = dht.readTemperature();
  // 读取温度单位为 Fahrenheit
  float f = dht.readTemperature(true);
  // 判断所获取的数字是否错误
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.println(" *F\t");
  //OLED 屏幕显示温湿度
  oled.setTextSize(1);        //设置屏幕文字大小
  oled.setTextColor(WHITE);  //设置屏幕文字颜色
  oled.setCursor(0, 0);       //设置文字显示坐标
  oled.clearDisplay();        //清除屏幕
```

```
43.   oled.print("Humidity: ");  //显示文字
44.   oled.print(h);
45.   oled.println(" %");
46.   oled.print("Temperature: ");
47.   oled.print(t);
48.   oled.println(" *C");
49.   oled.print(f);
50.   oled.println(" *F");
51.   oled.display();
52. }
```

如果发现 Adafruit_GFX 的头文件找不到，那么可以通过下面方法解决：

首先点击“管理库”进行相应第三方库的安装；

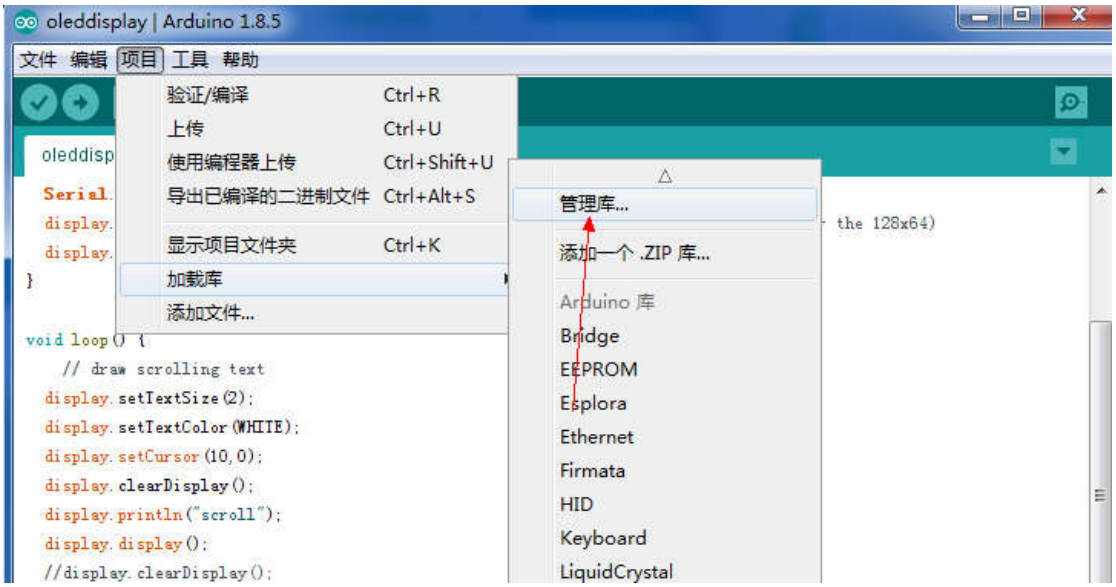


图 4.19 管理库

然后在搜索栏输入“GFX”，找到“DHT sensor library”并点击安装 1.2.1 版本；或者输入“SSD1306”，找到“Adafruit SSD1306” 进行安装。

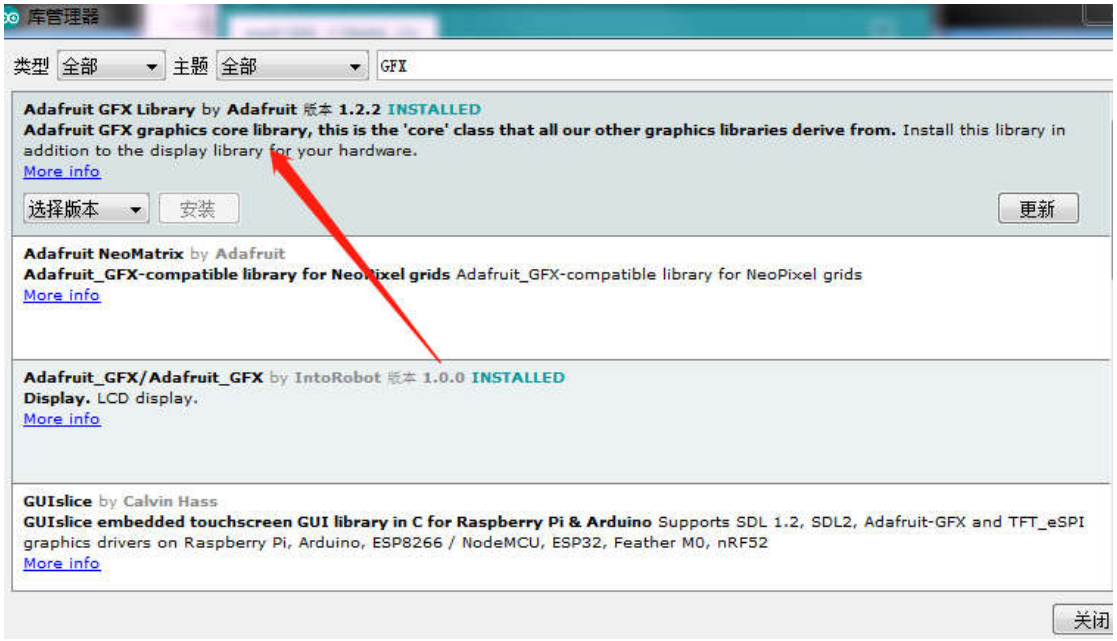


图 4.20 第三方库的搜索及安装

步骤三：代码下载与调试

最后，选择正确的板卡型号和端口，将上述程序上传到控制器中。

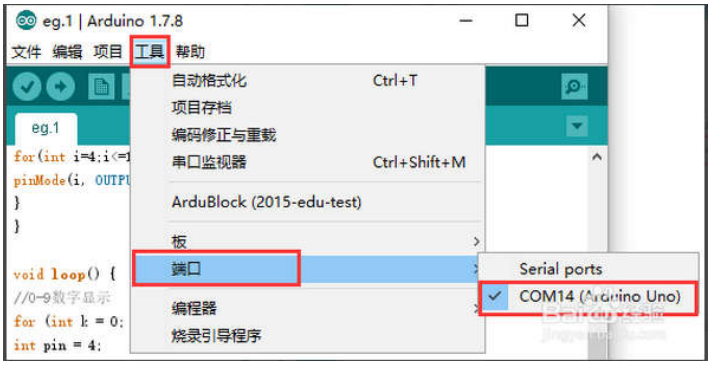


图 4.21 设置设置端口号

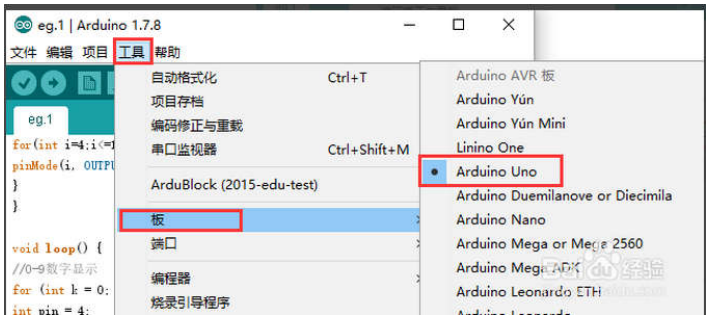


图 4.22 选择板卡类型

六、实验现象

程序运行后，用手捂住温湿度传感器，观察现象。

第五章 轮式机器人各模块设计与编程实验

该实验共有四个模块实验，分别是直流电机转动测试、编码器数据读取、线性 CCD 数据的读取。拓展实验内容：使用 Arduino 读取超声波测距传感器数据。

实验模块 1 直流电机转动测试实验

一、实验目的

1、了解电机的结构、组成及控制原理

2、了解电机控制的要点及控制方法

二、实验器材

Arduino 开发板，电机驱动模块 L298N 或自制驱动模块，锂电池，直流电机，电源模块，面包板，杜邦线。

三、实验原理与实验方法

1、接线

如图 5.1 所示，直流电机的控制很简单，在电机的正负端（即图中 1 和 6 端口）接上一定的电压就可以运转。图 5.1 另外四根线是用于测量电机转速的编码器接口。提到控制电机，则有两个关键因素：一是速度，二是方向。本实验选用的电机工作电压 7~13V，电流 540mA。一般选择单片机控制器来控制电机，Arduino 的电压为 5V，I/O 口输出的电流最大为 40mA，不足以驱动电机运转，因此需要一个中间介质，那就是驱动电路，保证给电机提供一定功率的输出。图 5.2 左图给出驱动直流电机的一种驱动模块 L298N，它的内部结构是由 MOS 管搭建的 H 桥电路。

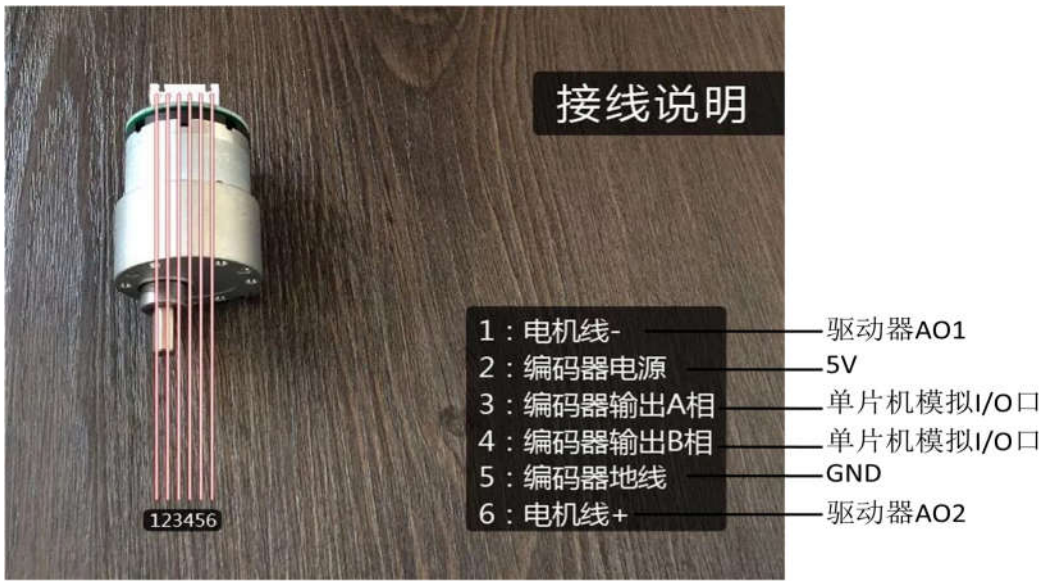


图 5.1 电机与驱动模块接线

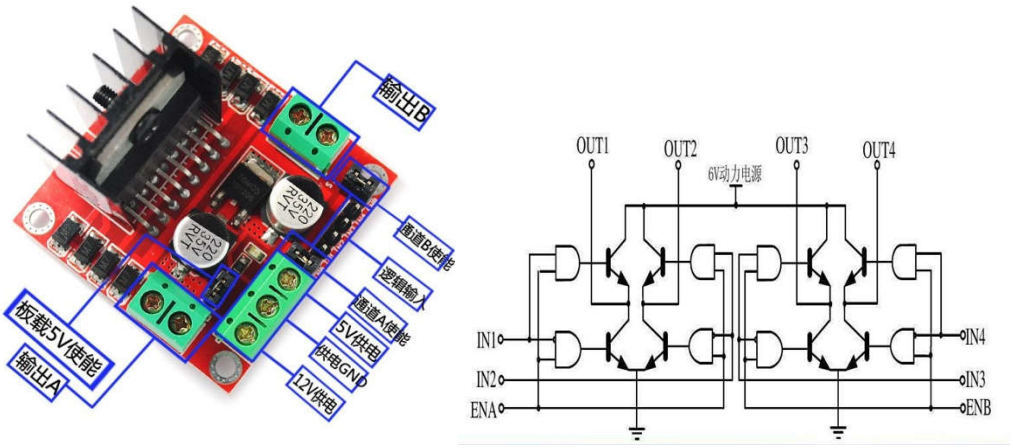


图 5.2 L298N 模块结构图与内部原理图

由于 L298N 是 2 路的 H 桥驱动，所以可以同时驱动两个电机，分别从调速端 A（即通道 A 使能 ENA）输入 PWM 信号，IN1 和 IN2 输入高低电平驱动电机 1 的转速和方向，分别从从调速端 B（即通道 B 使能 ENB）输入 PWM 信号，IN3 和 IN4 输入高低电平驱动电机 2 的转速和方向。L298N 驱动模块原理图和逻辑时序图分别如图 5.2 和 5.3 所示。

Arduino+电机+驱动模块 L298N 参考接线：（非常重要，线路接法参照下面）

（1）Arduino 数字引脚 2→电机 3 引脚编码器输出 A 相

（2）Arduino 数字引脚 3→电机 4 引脚编码器输出 B 相

（3）Arduino 数字引脚 10→驱动器 L298N ENA

（4）Arduino 数字引脚 11→驱动器 L298N IN2

（5）Arduino 数字引脚 12→驱动器 L298N IN1

（6）驱动器 L298N 输出 AO1→电机 6 引脚电机+（或电机 1 引脚电机-）

（7）驱动器 L298N 输出 AO2→电机 1 引脚电机-（或电机 6 引脚电机+）

（8）L298N 模块：12V 供电接 12V 电池；5V 供电接 5V；GND 接 Arduino GND

（9）电机上 2 引脚接 5V，5 引脚接 Arduino GND。

注意：绝对不可以把电池 12V 电压直接接到 Arduino Uno 板卡任何一个引脚上。

注意：绝对不可以把电池 12V 电压直接接到 Arduino Uno 板卡任何一个引脚上。

注意：绝对不可以把电池 12V 电压直接接到 Arduino Uno 板卡任何一个引脚上。

直流电机	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

图 5.3 L298N 模块逻辑时序图

2、PWM 调速

PWM 调速是通过改变输出方波的占空比使负载上的平均电流功率从 0~100%变化，从而改变电机速度。利用 PWM 调制的优点是电源的能量功率能得到充分的利用、电路的效率最高。例如：当输出为 50%的方波时，PWM 电路输出能量功率也为 50%，即几乎所有的能量都转换给负载。而采用常见的电阻降压调速时，要使负载获得电源最大 50%的功率，电源必须提供 71%以上的输出功率，这其中 21%消耗在电阻的压降及热耗上。

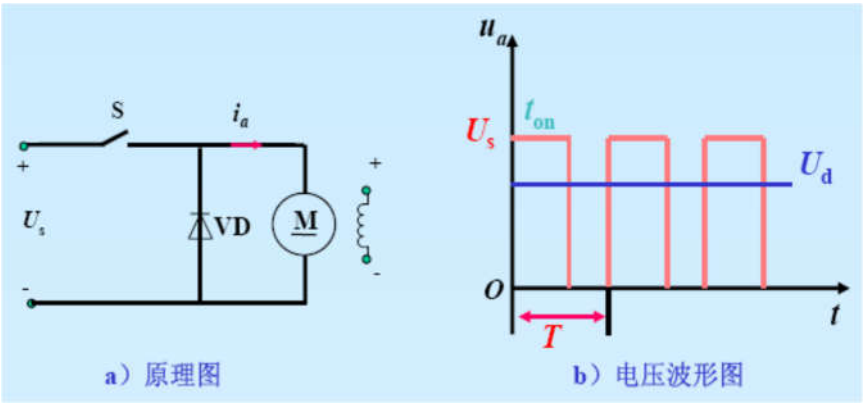


图 5.4 直流斩波器——电动机系统原理图和电压波形

在图 5.4 中，VT 表示电力电子开关器件，VD 表示续流二极管。当 VT 导通时，直流电源电压 Us 加到电动机上；当 VT 关断时，直流电源与电机脱开，电动机电枢电流经 VD 续流，两端电压接近于零。如此反复，电枢端电压波形如图，好像是电源电压 Us 在 t_{on} 时间内被接上，又在 $T-t_{on}$ 时间内被斩断，故称“斩波”。电动机得到的平均电压为：

$$U_d = \frac{t_{on}}{T} U_s = \rho U_s$$

式中：

T -电力电子开关器件的开关周期；

t_{on}-开通时间；

ρ-占空比，$\rho = \frac{t_{on}}{T} = t_{on} f$ 其中：f-开关频率。

3、单个电机全速转动测试（参考程序）

```
#define PIN_EN 10//定义 PWM 输出引脚
#define PIN_AIN2 11//定义正反转控制引脚
#define PIN_AIN1 12//定义正反转控制引脚
void setup() {
  pinMode(PIN_EN, OUTPUT); //定义引脚为输出
  pinMode(PIN_AIN2, OUTPUT);
  pinMode(PIN_AIN1, OUTPUT);
}
void loop() {
  digitalWrite(PIN_EN, HIGH);
  digitalWrite(PIN_AIN2, HIGH);
  digitalWrite(PIN_AIN1, LOW);
}
```

4、使用 PWM 波调整电机转速

```
#define PIN_PWM 10//定义 PWM 输出引脚
#define PIN_AIN2 11//定义正反转控制引脚
#define PIN_AIN1 12//定义正反转控制引脚
void setup() {
  pinMode(PIN_PWM, OUTPUT); //定义引脚为输出
  pinMode(PIN_AIN2, OUTPUT);
  pinMode(PIN_AIN1, OUTPUT);
}
void loop() {
  analogWrite(PIN_PWM, 100);
  digitalWrite(PIN_AIN2, HIGH);
  digitalWrite(PIN_AIN1, LOW);
  delay(3000);

  analogWrite(PIN_PWM, 100);
  digitalWrite(PIN_AIN2, LOW);
  digitalWrite(PIN_AIN1, HIGH);
  delay(3000);

  analogWrite(PIN_PWM, 200);
  digitalWrite(PIN_AIN2, HIGH);
  digitalWrite(PIN_AIN1, LOW);
  delay(3000);

  analogWrite(PIN_PWM, 200);
  digitalWrite(PIN_AIN2, LOW);
  digitalWrite(PIN_AIN1, HIGH);
  delay(3000);
}
```

四、思考问题

1、直流电机调速方法。
2、两个直流电机的同步控制方法。

实验模块二 编码器数据读取

一、实验目的

了解编码器原理

二、实验设备

Arduino 控制板，电机，锂电池，电源模块，面包板，杜邦线。

三、实验原理

M 法是测量单位时间内的脉数换算成频率，因存在测量时间内首尾的半个脉冲问题，可能会有 2 个脉冲的误差。速度较低时，因测量时间内的脉冲数变少，误差所占的比例会变大，所以 M 法宜测量高速。如要降低测量的速度下限，可以提高编码器线数或加大测量的单位时间，使一次采集的脉冲数尽可能多。

使用 M 法测速的时候，会通过测量单位时间内 A 相（或 B 相）的上升沿或者下降沿，也就是图 5.5 中对应的数字 1234 中的某一个，这样图中三个脉冲就只能计数 3 次。四倍频的方法是测量 A 相和 B 相编码器的上升沿和下降沿，这样在同样的时间内，可以计数 12 次（3 个 1234 的循环）。（相当于对编码器的细分）

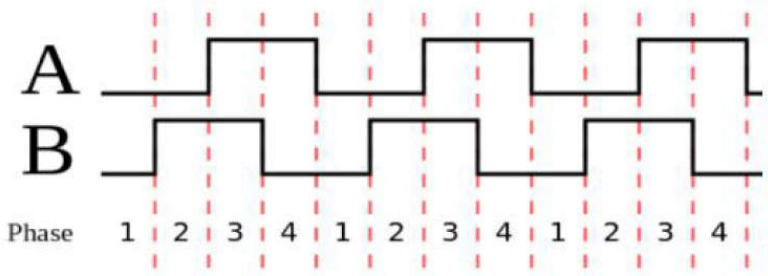


图 5.5 编码器输出波形图

因为编码器输出的是标准的方波，所以可以使用 Arduino UNO 等单片机工具直接读取。在软件中的处理方法分两种，自带双路编码器接口的单片机如 STM32，可以直接使用硬件计数。而没有编码器接口的单片机如 Arduino UNO，可以通过外部中断读取，比如把编码器 A 相输出接到单片机的外部中断入口，这样就可以通过跳变沿触发中断，然后在对应的外部中断服务函数里面，通过 B 相的电平来确定正反转。如当 A 相来一个跳变沿的时候，如果 B 相是高电平就认为是正转，低电平就认为是反转。

本实验采取边沿触发计数方式，编码器 A、B 相只要有跳变，就计数；将电机控制的周期定位 20ms 一个周期，那么同时编码器计数也是每 20ms 为一个计数周期，之后清零，再重新计数。以下给出计数步骤和计数值与电机转速的计算方法：

1、边沿触发计数

2、20ms 定时器计时计数

3、设 20ms 编码器计数 X 个脉冲，则转速为：

X/20ms=X*50/s=X*5 转/s ，这里假设电机每转一圈，编码器发出 10 个脉冲。

注：本章节暂时不使用电机转速计算，后面实验第六章节会介绍。

四、编码器数据读取参考程序

```
#define ENCODER_A    2   //编码器读取采用中断方式，该引脚为第 0 个中断接口
#define ENCODER_B    3
long encoderVal; //左编码器值
//函数功能：外部中断读取编码器数据，具有二倍频功能注意外部中断是跳变沿触发
void getEncoder(void) {
  if (digitalRead(ENCODER_A) == LOW)  {  //如果是下降沿触发的中断
    if (digitalRead(ENCODER_B) == LOW) {
      encoderVal--;  //根据另外一相电平判定方向
    }
    else {
      encoderVal++;
    }
  }
  else { //如果是上升沿触发的中断
    if (digitalRead(ENCODER_B) == LOW) {
      encoderVal++; //根据另外一相电平判定方向
    }
    else {
      encoderVal--;
    }
  }
}
void setup() {
  //编码器引脚都设置输入
  pinMode(ENCODER_A, INPUT);
  pinMode(ENCODER_B, OUTPUT); //此处有错误，请找出并纠正
  //下面这一条语句此处有错误，请找出并纠正
  attachInterrupt(2, getEncoder, CHANGE);  //使能编码器引脚外部中断 0，对应 Arduino 引脚
2
  Serial.begin(9600);
}
void loop() {
  delay(100);
  Serial.print("encoder val = ");
  Serial.print(encoderVal);
  Serial.print("\r\n");
}
```

注意：上面程序中有两处错误，请找出来并更正，程序才能正常运行。写入程序后，转动你的电机轮子，看看串口观察器里的数值变化。

五、思考问题

1、如何通过编码器获得轮子的转速。（提示:我们所使用的电机输出轴，即连轮子的轴每转一圈，编码器输出 490 个脉冲）

2、编码器测速方法。

实验模块三 线性 CCD 数据采集实验

一、实验目的

了解线性 CCD 数据的采集方法与数据处理

二、实验设备

Arduino 控制板，线性 CCD，面包板，杜邦线

三、实验原理

线性 CCD 原理：使用的线性 CCD 为 TSL1401，该 CCD 为线性输出，输出数据为 128*1 线性矩阵，详见 TSL1401 的数据手册。

TERMINAL NAME	TERMINAL NO.	DESCRIPTION
AO	3	Analog output.
CLK	2	Clock. The clock controls charge transfer, pixel output, and reset.
GND	6, 7	Ground (substrate). All voltages are referenced to the substrate.
NC	5, 8	No internal connection.
SI	1	Serial input. SI defines the start of the data-out sequence.
V_{DD}	4	Supply voltage. Supply voltage for both analog and digital circuits.

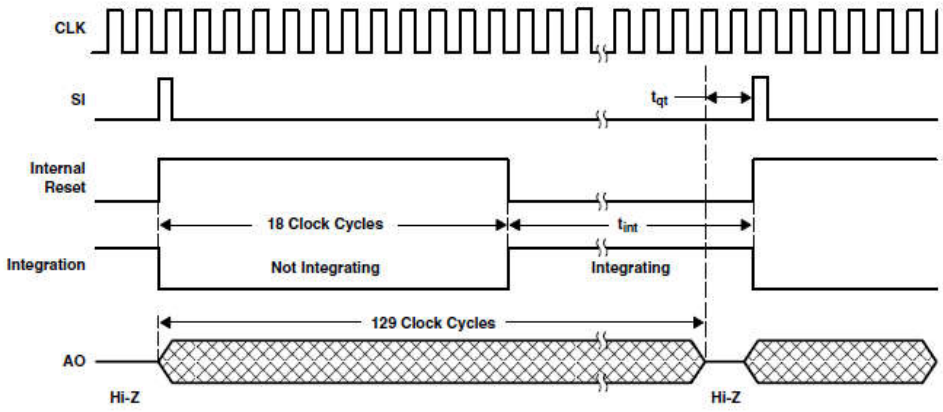


图 5.6 CCD 引脚和时序图（TSL1401）

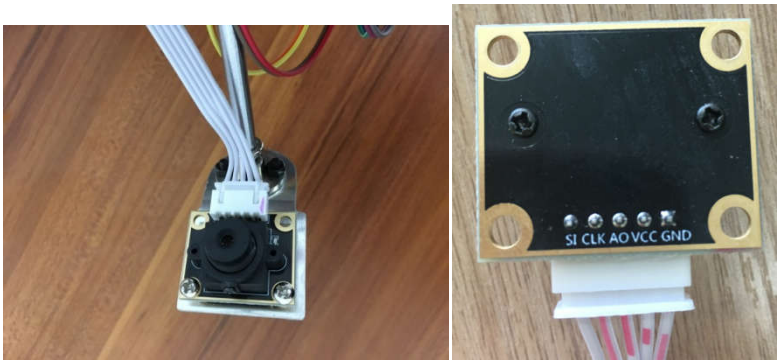


图 5.7 线性 CCD 实物及引脚

线性 CCD 与 Arduino 接线：

CCD_DATA →Arduino A0

CCD_CLK →Arduino A1

CCD_SI　→Arduino A2

CCD_VCC →Arduino 3.3V

CCD_GND→Arduino GND

CCD 的使用需要考虑到的因素有：室内光线的亮暗程度，CCD 安装的位置，在程序编写时，首先要在使用的环境下测量数据，寻找 CCD 中值，作为标准值的参考，当采集的数据为该中值时，说明小车在赛道的中间。如果环境光线极强，CCD 将输出很大的值，那么不适合测试巡线。

四、参考程序

（注意：在此实验之前，要准备一张白纸，然后将黑胶布贴在纸张的中间）

```
//定义 CCD 引脚
#define AO_PIN     14 //A0
#define CLK_PIN    15 //A1
#define SI_PIN      16 //A2
uint8_t _adVal[128];    //存储 CCD 采样的 AD 值
uint8_t _ccdThreshold; //CCD 阈值
//初始化 ccd 引脚
void ccdInit(void) {
  pinMode(CLK_PIN, OUTPUT);
  pinMode(SI_PIN, OUTPUT);
  pinMode(AO_PIN, INPUT);
  digitalWrite(CLK_PIN, LOW);
  digitalWrite(SI_PIN, LOW);
}
//ccd 的 AD 采样
void readTsl1401(void) {
  uint8_t i = 0;
  //曝光，看不懂程序的，参考 CCD 的图 1 逻辑，或者百度 TSL1401 心得，有很多资料
  digitalWrite(SI_PIN, HIGH);
  digitalWrite(CLK_PIN, HIGH);
  delayMicroseconds(1);
  digitalWrite(SI_PIN, LOW);
  digitalWrite(CLK_PIN, LOW);
  for (i = 0; i < 129; i++) {
    digitalWrite(CLK_PIN, HIGH);
    digitalWrite(CLK_PIN, LOW);
  }
  delay(30);  //这里 30ms 是调整曝光时间，不同的值会获得不同的采样值
  //读取像素值
  digitalWrite(SI_PIN, HIGH);
  digitalWrite(CLK_PIN, HIGH);
  digitalWrite(SI_PIN, LOW);
  delayMicroseconds(1);
```

```
  for (i = 0; i < 128; i++) {
    _adVal[i] = analogRead(AO_PIN) >> 2; //Arduino UNO 的模拟数字采样是 10 位的，最大值为 1023
    digitalWrite(CLK_PIN, LOW);
    digitalWrite(CLK_PIN, HIGH);
    delayMicroseconds(2);
  }
  digitalWrite(CLK_PIN, HIGH);
  digitalWrite(CLK_PIN, LOW);
}
uint8_t ccdFindMidVal(void)//CCD 寻找中值函数返回值为中值
{
  uint8_t i, j, left, right, _ccdMidVal;
  float maxVal, minVal;
  readTsl1401();
  maxVal = _adVal[0]; //动态阈值算法，读取最大和最小值
  for (i = 5; i < 123; i++) {   //两边各去掉 5 个点
    if (maxVal <= _adVal[i]) {
      maxVal = _adVal[i];
    }
  }
  minVal = _adVal[0]; //最小值
  for (i = 5; i < 123; i++) {
    if (minVal >= _adVal[i])
      minVal = _adVal[i];
  }
  _ccdThreshold = (maxVal + minVal) / 2.0; //计算出本次中线提取的阈值
  left = 0;
  for (i = 5; i < 118; i++) { //寻找左边跳变沿
    if (_adVal[i] > _ccdThreshold &&
        _adVal[i + 1] > _ccdThreshold &&
        _adVal[i + 2] > _ccdThreshold &&
        _adVal[i + 3] < _ccdThreshold &&
        _adVal[i + 4] < _ccdThreshold &&
        _adVal[i + 5] < _ccdThreshold) {
      left = i;
      break;
    }
  }
  right = 128;
  for (j = 118; j > 5; j--)//寻找右边跳变沿
  {
    if (_adVal[j]   < _ccdThreshold &&
        _adVal[j + 1] < _ccdThreshold &&
        _adVal[j + 2] < _ccdThreshold &&
        _adVal[j + 3] > _ccdThreshold &&
```

```
        _adVal[j + 4] > _ccdThreshold &&
        _adVal[j + 5] > _ccdThreshold)
    {
      right = j;
      break;
    }
  }
  if (left == 0 || right == 128) {
    //未检测到边沿
    _ccdMidVal = 255;
  }
  else
    _ccdMidVal = (uint8_t)((right + left) / 2); //计算中线位置
  return _ccdMidVal;
}
void setup() {
  ccdInit();
  Serial.begin(9600);
}
void loop() {
  uint8_t midVal = ccdFindMidVal();
  //Serial.print("CCD Middle Value = ");
  //Serial.println(midVal);
  for (int i = 5; i < 123; i++) {
    Serial.println(_adVal[i]);
  }
  delay(100);
}
```

注意：使用串口绘图器查看结果

五、实验结果分析

通过串口工具，分析 CCD 获取的数据；使用手机上的 LED 灯模仿不同的光照条件，分析数据的变化。

六、思考问题

1、对 CCD 数据的处理方法，全部像素都使用还是取部分值，如何根据不同的光照条件，调整曝光时间？

2、Arduino 控制器对其它传感器数据的读取方法，如 8 路寻迹传感器。

3、分析你所使用的传感器在实际系统中的作用。

拓展实验：使用 Arduino 读取超声波测距

一、 实验目的

掌握 Arduino 读取各类传感器数据的方法；

掌握传感器数据处理的方法。

二、实验设备和材料

Arduino UNO，超声波测距传感器，8 路寻迹传感器，面包板，杜邦线。

三、 传感器原理及使用方法

1、超声波传感器

超声波传感器主要技术参数：

使用电压：DC5V；

静态电流：小于 2mA；

电平输出：高 5V；

电平输出：低 0V；

感应角度：不大于 15 度；

探测距离：50px-11250px，高精度可达 5px；

接线方式端口：VCC（电源）、trig（控制端）、echo（接收端）、GND（地）。

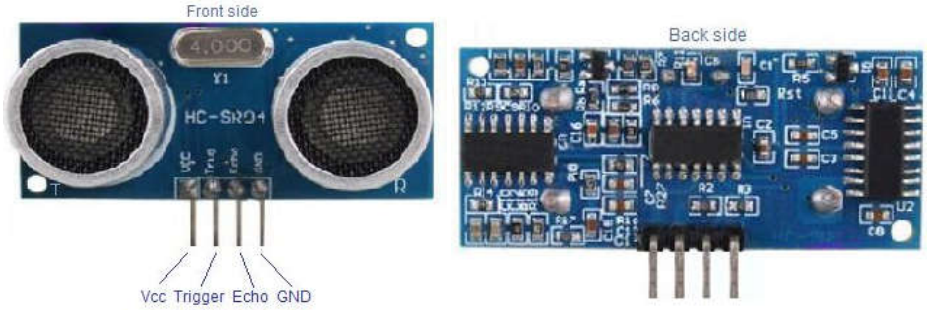


图 5.8 超声波测距传感器实物图

超声波测距传感器工作原理：

（1）采用 I/O 触发测距，给至少 10us 的高电平信号；

（2）模块自动发送 8 个 40kHZ 的方波，自动检测是否有信号返回；

（3）有信号返回，通过 I/O 输出一高电平，高电平持续的时间就是超声波从发射到返回的时间。

测试距离=(高电平时间*声速(340M/S))/2。

超声波传感器主要利用声波传播时间测距原理，通过晶振向外发射超过人体能感知的高频超声波；一般选用 25～40kHz 波，然后控制模块检测反射回来波，模块计算声波传输时间来进行测距。

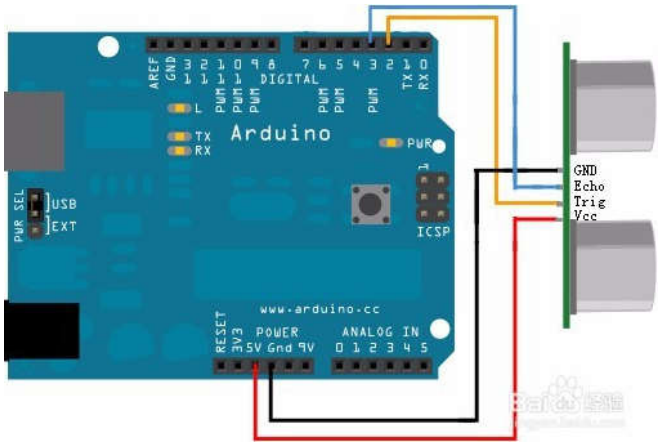


图 5.9　Arduino 与超声波测距传感器接线

参考程序：

```
1.  const int TrigPin = 2;
2.  const int EchoPin = 3;
3.  float cm;
4.  void setup()
5.  {
6.    Serial.begin(9600);
7.    pinMode(TrigPin, OUTPUT);
8.    pinMode(EchoPin, INPUT);
9.  }
10. void loop()
11. {
12.   digitalWrite(TrigPin, LOW); //低高低电平发一个短时间脉冲去 TrigPin
13.   delayMicroseconds(2);
14.   digitalWrite(TrigPin, HIGH);
15.   delayMicroseconds(10);
16.   digitalWrite(TrigPin, LOW);
17.   cm = pulseIn(EchoPin, HIGH) / 58; //将回波时间换算成 cm
18.   cm = (int(cm * 100)) / 100; //保留两位小数
19.   Serial.print(cm);
20.   Serial.print("cm");
21.   Serial.println();
22.   delay(1000);
23. }
```

第六章 直流电机调速实验

直流电机调速实验分为三种方案（任选其一做）：基于 L298N 驱动模块的直流电机调速、基于自己焊接的电机驱动板的直流电机调速和基于 TB6612 驱动模块的电机调速。

一、实验目的

1、掌握直流电机调速的方法；

2、理解 PID 调速原理及方法。

二、实验器材

Arduino，直流电机（轮式小车车体），驱动模块，电源模块，锂电池等。

三、实验方法与步骤

1、电机与驱动模块接线；

2、编码器测速，通过对编码器 A、B 相的计数来计算电机的速度，转速以弧度/s 计算；（参考编码器读取实验，**在中断里面使用的全局变量，需要定义成 volatile 类型， 例如：volatile long encoderVal;**）

采取边沿触发计数方式，编码器 A、B 相只要有跳变，就计数；将电机控制的周期定时 PERIOD 为一个周期，那么同时编码器计数也是每 PERIOD ms 为一个计数周期，之后清零，再重新计数。以下给出计数步骤和计数值与电机转速的计算方法：

1） 边沿触发计数

2） 定时器定时计数

3） 大家的电机是轮子每转动一圈输出 390 个脉冲，实验三编码器读取使用边缘触发中断，则可以实现脉冲的上升和下降计数。所以在计算速度的时候应采用 390*2=780 来计算速度。以下 PERIOD 为测速周期，ms 为单位，encoderVal 为记录到的边缘触发次数（为脉冲数的 2 倍），则速度为

velocity = (encoderVal / 780.0)* 3.1415 * 2.0 * (1000 / PERIOD);

注意：可以使用 millis()或者 MsTimer2 来实现周期定时，建议使用 MsTimer2 来实现定时，millis()精度较差。MsTimer2 的使用，需要通过 Arduino IDE 的库管理进行安装（参考前面温湿度传感器的库安装方法）。以下定时器代码供参考：

```
#include <MsTimer2.h>
//定时器库的头文件，除了使用 MsTimer2，也可以使用 millis(),但很容易受到干扰，定时不准
void control()
{
  //这里面可以放置编码器测速算法和电机转速 PID 控制算法
}
void setup()
{
  MsTimer2::set(PERIOD, control); //设置每隔 PERIOD 时间，执行 control 函数一次
```

```
10.   MsTimer2::start();
11. }
12. void loop()
13. {
14. }
```

3、P（比例）或 PI（比例积分）控制器算法设计；

增量式 PID 算法，由以下公式计算获得 PID 控制器的输出值：

$$\Delta u(k) = u(k) - u(k-1) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2)$$

其中 $\Delta u(k)$ 为增量式 PID 的输出，$e(k)$，$e(k-1)$，$e(k-2)$ 分别为当前时刻偏差，前一时刻偏差，更前一时刻偏差，K_P 为比例增益，T_i 为积分时间，T_d 为微分时间。

$$q_0 = K_p(1 + \frac{T}{T_i} + \frac{T_d}{T})$$

$$q_1 = -K_p(1 + \frac{2T_d}{T})$$

$$q_2 = K_p \frac{T_d}{T}$$

或者写作：

$$\Delta u(k) = K_p\{e(k) - e(k-1)\} + K_i e(k) + K_d\{e(k) - 2e(k-1) + e(k-2)\}$$

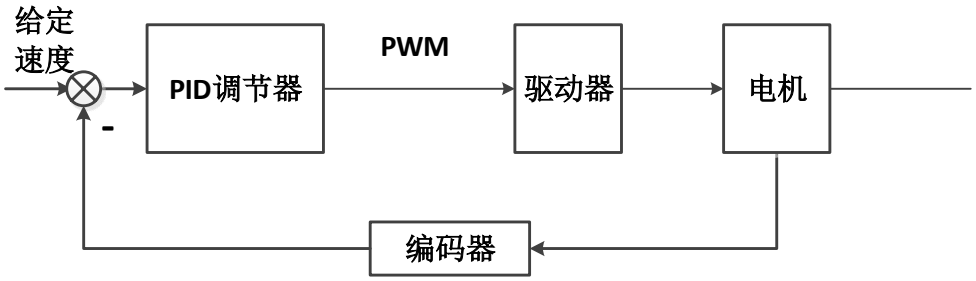
其中：

$e(k)$为当前时刻偏差	K_p为比例系数
$e(k-1)$为前一时刻偏差	K_i为积分系数
$e(k-2)$为更前一时刻偏差	K_d为微分系数

注意：大家可以把上述算法编写成一个 pidController 的函数，如

```
int pidController(float targetVelocity, float currentVelocity)
{
    //targetVelocity 为目标速度，currentVelocity 为当前速度
    //这里实现算法，注意上述算法求出来的是Δu(k)，所以要和前一次
    //的输出 u 相加才能得到当前输出值。所以需要使用全局变量保存上
    //一时刻u(k)，上一时刻的误差e(k-1)，上上时刻的误差e(k-2)。
    return 当前时刻计算得到的控制输出；
}
```

4、将系统构成闭环，给定电机或小车初始速度，让其运动；



5、通过调节 K_p（比例）、T_i（积分时间）和 T_d（微分时间）参数，使电机或小车达到给定速度控制（使用课上讲的秘诀）；

PID 参数调节过程比较繁琐、枯燥，请同学结合课堂理论知识进行调节和尝试。在实际工程中，PID 参数调节需要丰富的实际经验，以下工程中总结出来的 PID 参数调试口诀，供各位学生参考：

参数整定找最佳，从小到大顺序查；
先是比例后积分，最后再把微分加；
曲线振荡很频繁，比例度盘要放大；
曲线漂浮绕大湾，比例度盘往小扳；
曲线偏离回复慢，积分时间往下降；
曲线波动周期长，积分时间再加长；
曲线振荡频率快，先把微分降下来；
动差大来波动慢，微分时间应加长；
理想曲线两个波，前高后低 4 比 1；
一看二调多分析，调节质量不会低。

四．实验报告

(1) 根据实验不同参数的数据，画出闭环控制特性曲线。

(2) 总结闭环系统的 PID 参数调节心得。

附录：程序框架供参考（自己编写程序的同学，可以以此为框架编写程序）

```
#include <MsTimer2.h>        //定时器库的头文件
//定义电机编码器 A B 相引脚（自行补充引脚号）
#define ENCODER_A          //注意这里需要补充接入中断引脚
#define ENCODER_B
//定义电机驱动的控制信号（自行补充引脚号）
#define PWML
#define INL1
#define INL2
//全局变量
```

```
volatile long encoderVal; //编码器值
float velocity;
//自行定义其他全局变量

//获取编码器值
void getEncoder(void)
{
  //自行补充代码
}
//PID 控制器，初始值供参考，Kp=1，Ti=10，Td=0，T=PERIOD
int pidController(float targetVelocity, float currentVelocity)
{
  float output;
  //自行补充代码
  return (int)output;
}
//直流电机控制
void control(void)
{
  //注意，大家的电机是轮子每转动一圈输出 390 个脉冲，实验室三编码器读取使用边缘触发中断，则可以实现脉冲的上升和下降计数。所以在计算速度的时候应采用 390*2=780 来计算速度。以下 PERIOD 为测速周期，ms 为单位， encoderVal 与实验三的编码器读取代码相同变量
  velocity = (encoderVal / 780.0) * 3.1415 * 2.0 * (1000 / PERIOD);
  //思路：根据电机当前转速
  encoderVal = 0;
  int output = pidController(TARGET, velocity);
  //TARGET 为目标速度值，目前大家的电机基本上最快为每秒转 50 弧度，所以不要设置超过这个值
  //以下代码很重要，大家可能需要根据电机实际转动，调整 INL1 和 INL2 的值
  if (output > 0)
  {
    digitalWrite(INL1, LOW);
    digitalWrite(INL2, HIGH);
    analogWrite(PWML, output);
  }
  else
  {
    digitalWrite(INL1, HIGH);
    digitalWrite(INL2, LOW);
    analogWrite(PWML, abs(output));
  }
}
void setup()
{
  //9,10 两个管脚的 PWM 由定时器 TIMER1 产生，这句程序改变 PWM 的频率，勿删
  TCCR1B = TCCR1B & B11111000 | B00000001;
  //这里自行添加代码，初始化各管脚，各全局变量，中断，串口通讯等
```

```
  MsTimer2::set(PERIOD, control);
  MsTimer2::start();
}
void loop()
{
  //撰写代码，使其能够通过“串口绘图器”观察速度波形，以下代码仅供参考
  Serial.print(velocity);
  Serial.print("\r\n");
}
```

附录：

参考程序（需要读懂程序，根据运行效果，调节其中的参数或修改某些语句）

```
#include <MsTimer2.h>       //定时器库的 头文件
//定义电机编码器 A B 相引脚
#define ENCODER_A 2
#define ENCODER_B 4
//定义电机驱动的控制信号
#define PWML 10
#define INL1  12
#define INL2  11
#define PERIOD 20
#define TARGET -20
//全局变量
volatile long encoderVal; //编码器值
float velocity;
float u = 0;
float eI;
float eII;
float eIII;

//获取编码器值
void getEncoder(void)
{
  if (digitalRead(ENCODER_A) == LOW)
  {
    if (digitalRead(ENCODER_B) == LOW)
    {
      encoderVal--;
    }
    else
    {
      encoderVal++;
    }
  }
```

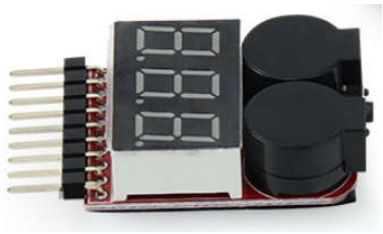
```
  else
  {
    if (digitalRead(ENCODER_B) == LOW)
    {
      encoderVal++;
    }
    else
    {
      encoderVal--;
    }
  }
}
//PID 控制器，初始值供参考，Kp=500，Ti=10，Td=400，T=PERIOD
int pidController(float targetVelocity, float currentVelocity)
{
  float output;
  eI = targetVelocity - currentVelocity;
  float Kp = 5, Ti = 140, Td = 80, T = PERIOD;
  float q0 = Kp * (1 + T / Ti + Td / T);
  float q1 = -Kp * (1 + 2 * Td / T);
  float q2 = Kp * Td / T;
  u = u + q0 * eI + q1 * eII + q2 * eIII;
  eIII = eII;
  eII = eI;
  if (u >= 255)
  {
    u = 255;
  }
  if (u <= -255)
  {
    u = -255;
  }
  output = u;
  return (int)output;
}
//直流电机控制
void control(void)
{

  encoderVal = 0;
  int output = pidController(TARGET, velocity);
  //TARGET 为目标速度值，目前大家的电机基本上最快为每秒转 50 弧度，所以不要设置超过这个值
  //以下代码很重要，大家可能需要根据电机实际转动，调整 INL1 和 INL2 的值
  if (output > 0)
  {
    digitalWrite(INL1, LOW);
```

```
    digitalWrite(INL2, HIGH);
    analogWrite(PWML, output);
  }
  else
  {
    digitalWrite(INL1, HIGH);
    digitalWrite(INL2, LOW);
    analogWrite(PWML, abs(output));
  }
}
void setup()
{
  //9,10 两个管脚的 PWM 由定时器 TIMER1 产生，这句程序改变 PWM 的频率，勿删
  TCCR1B = TCCR1B & B11111000 | B00000001;
  //初始化各管脚，各全局变量，中断，串口通讯等
  pinMode(PWML, OUTPUT);
  pinMode(INL2, OUTPUT);
  pinMode(INL1, OUTPUT);
  pinMode(ENCODER_A, INPUT);
  pinMode(ENCODER_B, INPUT);
  attachInterrupt(0, getEncoder, CHANGE);
   Serial.begin(9600);
   MsTimer2::set(PERIOD, control);
   MsTimer2::start();
}
void loop()
{
  //撰写代码，使其能够通过“串口绘图器”观察速度波形
  Serial.print(velocity);
  Serial.print("\r\n");
}
```

第七章 轮式机器人自主巡线控制实验

重要：绝对不可以对锂电池过放电，当电池过放时，会永久损坏电池，当电压低于一定的压值时，必须停止使用电池，进行充电：2S 航模锂电池最低检测电压为 7.4V，3S 航模锂电池最低检测电压为 9.6V。使用过程，应时不时测量电池电压，或者全程接入电池报警器。



一、实验目的

1、掌握直流电机调速的方法；

2、理解 PID 调速原理及方法；

3、掌握电机位置闭环控制的实现原理；

4、了解轮式机器人巡线控制的硬件系统构成和软件实现方法；

5、掌握轮式机器人控制中电机速度环和位置环实现原理。

二、实验设备

智创-HIT-I 型实验箱，示波器，直流电源，电机转速测速计。

三、实验原理

1、基本原理

图 7.1 给出了循迹小车控制原理框图，该图由一个位置反馈闭环和两个速度反馈闭环构成，位置反馈体现在小车巡线运动的轨迹跟踪，位置反馈环节由多路寻迹光电传感器或者线性 CCD 摄像头完成；速度闭环体现在小车转弯时，左右车轮的速度改变，当小车运动到直线轨迹时，需要调节回原来的给定速度上。调节器采用 P、PI、PD 或 PID 算法，由 Arduino 控制器实现。

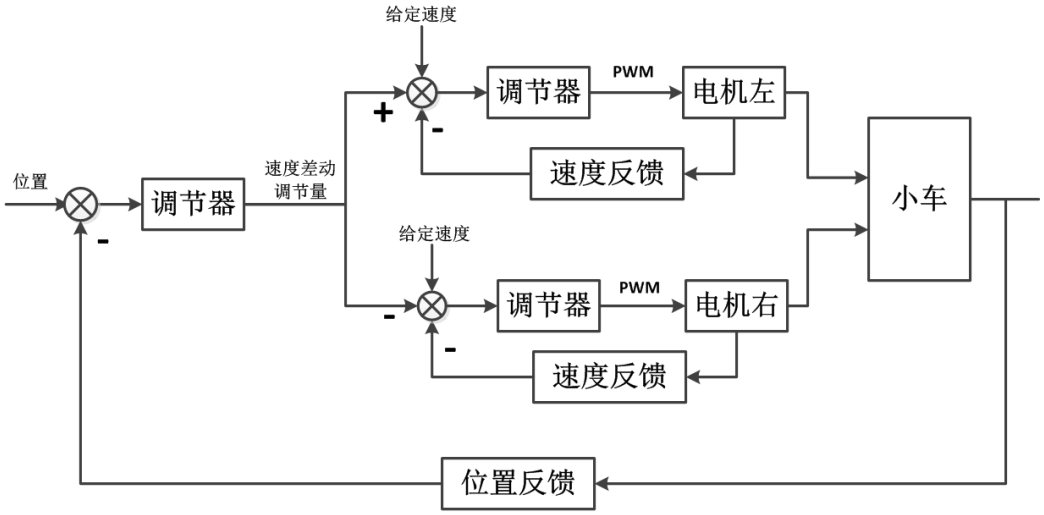


图 7.1 循迹小车控制原理框图

速度环使用增量式 PI 算法，给定的速度为人为设定，测量的速度由编码器测量获得，编码器有 A、B 两项，使用 M 测速法测量电机的速度（详见实验三）。位置环使用增量式 PD 算法，给定的位置信息设定为多路寻迹光电传感器或者 CCD 的中间阈值，测量的位置信息由多路寻迹光电传感器或者 CCD 实际测量获得，测量值与给定值的差为 PID 算法中的偏差 $e(k)$ 。

当使用 CCD 时，对循迹小车运行速度不能太快，因为速度太快，CCD 摄像头还没来得及处理下一位置的信息，小车将会跑出运行轨道，不可控制，图 7.2 给出小车运动轨道与运动速度简易分析图。如果使用多路寻迹光电传感器，则运行速度会比 CCD 更快。

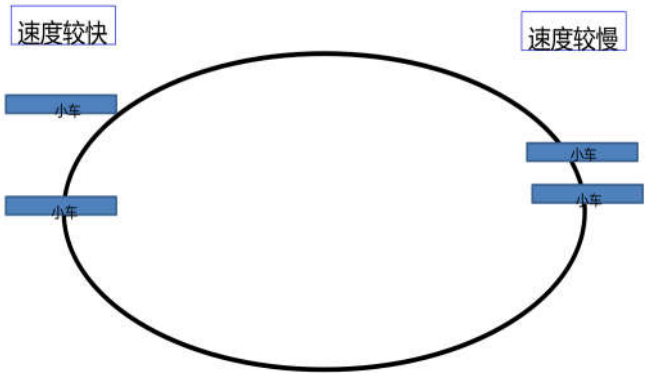


图 7.2 小车运行轨道与速度分析

2、光电寻迹模块的使用

在此我们使用寻迹模块 CTRT5000，该模块体积小，灵敏度较高，还可以通过转动上面的电位器来调节检测范围。

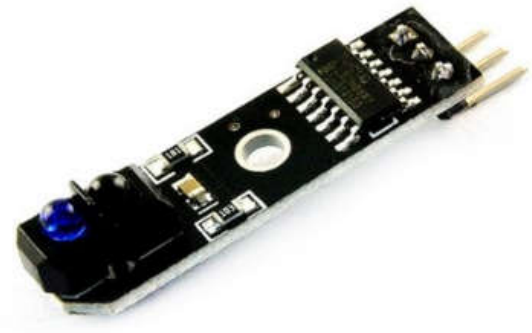


图 7.3 光电寻迹模块

1)、采用 TCRT5000 红外反射传感器

2)、检测距离：1mm~8mm 适用，焦点距离为 2.5mm

3)、比较器输出，信号干净，波形好，驱动能力强，超过 15mA。

4)、工作电压 3.3V-5V

5)、输出形式 OUT ：数字开关量输出（0 和 1）

6)、设有固定螺栓孔，方便安装

TCRT5000 传感器的红外发射二极管不断发射红外线，当发射出的红外线没有被反射回来或被反射回来但强度不够大时，光敏三极管一直处于关断状态，此时模块的输出端为低电平，指示二极管一直处于熄灭状态；被检测物体出现在检测范围内时，红外线被反射回来且强度足够大，光敏三极管饱和，此时模块的输出端为高电平，指示二极管被点亮。由于黑色具有较强的吸收能力，当循迹模块发射的红外线照射到黑线时，红外线将会被黑线吸收，导致循迹模块上光敏三极管处于关闭状态，此时模块上一个 LED 熄灭。在没有检测到黑线时，模块上 LED 常亮。模块在检测到黑线输出低电平，白线输出高电平。

循迹模块的工作一般要求距离待检测的黑线距离 1-2cm，因此我建议大家可以将循迹模块向下延伸。我自己是在硬纸板上面打了几个孔，固定循迹模块，每个模块之间可以留 1cm 左右的距离。安装时可以按照图 7.4 进行安装，装的越多，角度分辨率越高。实际上，寻线控制不需要太高的角度分辨率就可以运行很好。

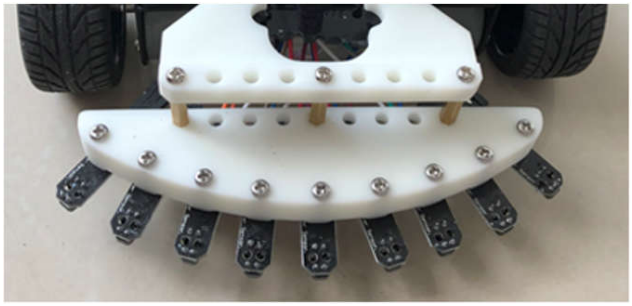


图 7.4 光电寻迹模块安装

四、实验步骤

（一）硬件系统构建步骤

1、系统电源供电：航模锂电池（最高 12.6V），5V 3.3V 电压模块（提供 5V 和 3.3V 电压），注意：系统中需要的电压，不能从 Arduino 控制器提供，必须由电压模块提供，因为 Arduino 控制器没有带负载能力，其输出功率有限。

2、电机位置环的搭建

线性 CCD 接线参考第五章实验模块三。

光电寻线传感器只需接上电源并将输出接入到 Arduino 的 I/O 口上即可。

3、电机速度环的搭建

使用电机驱动模块驱动左右轮直流电机，驱动模块可选用 L298N，TB6612 或自己设计的驱动电路。

4、电池电量检测电路

小车使用航模锂电池供电，当电池过放时，会永久损坏电池，当电压低于一定的压值时，必须停止使用电池，关闭电机：2S 航模锂电池最低检测电压为 7.4V，3S 航模锂电池最低检测电压为 9.6V，可以在程序里实现此功能。航模锂电池及充电器、接头等如图 7.3 所示。

注意：充电的时候不能使用电池，锂电池过放将导致电池永久损坏。3S 航模锂电池满电时为 12.6V，锂电池不能满电长时间（超过一周）存放，否则容易导致电池寿命下降，长期存放的最佳电压是 11.5V。

图 7.3 航模锂电池及充电器

图 7.4 给出了精密电阻电压测量电路图，使用两个 1K 和 10K 的精密电阻对电池电压进行分压，OUT 接 Arduino 的 A0 引脚，A0 引脚可以作为 A/D 转换功能来使用，注意：Arduino 的 A/D 精度是 10 位，输入的最大电压为 5V，所以对输入的电压一定要限制在 0~5V 范围内。

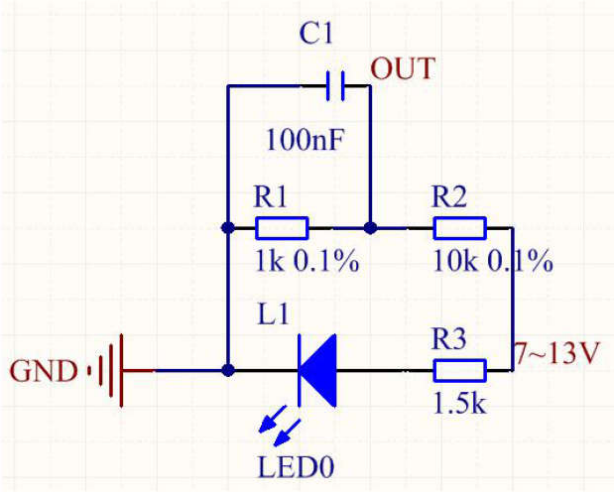


图 7.4 精密电阻电压测量电路图

A/D 计算方法（程序中的实现方法）：

采集出来的 AD 值和是 200 次的和 那么平均值：AD =sum/200

AD/1024 = V/5

电压 V =（AD*5）/1024

电压比是 10:1

真实电压是 batt = V*11 = （AD*5*11）/1024

转化为以 mv 单位就是 batt=AD*5*11*1000/1024 = AD*53.71 因为 AD=sum/200

所以真实电压就是 batt=sum*53.71/200 = sum*0.5371/2

注意程序中写的是 0.05371 是缩小了 10 倍。

图 7.5 为组装完成后的小车。

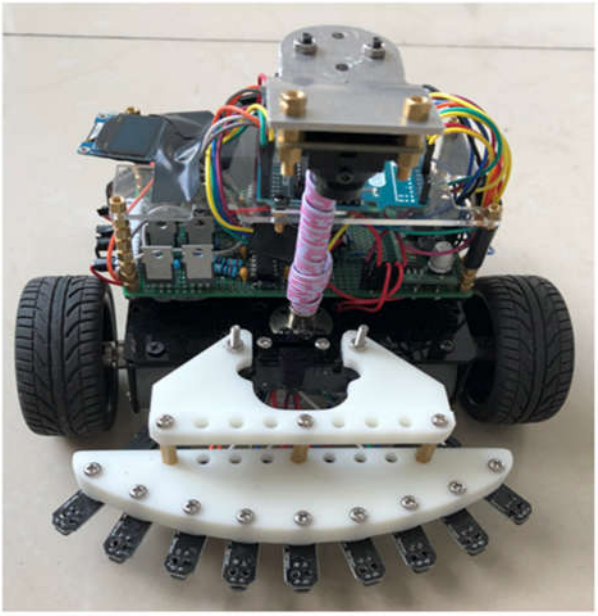


图 7.5 组装完成后的小车

（二）软件需要完成以下功能：

①打开小车电源开关，按下按键，则小车按照实际轨迹运行；

②电池电量检测，当电池电压低于 9.6V，则需停止小车运动，立即充电；

③ CCD 信号检测（A/D），并对获取的信号进行处理；或者检测光电寻迹传感器阵列

④电机控制（参考程序使用 298 驱动）：

A、速度环由编码器反馈构成（体现在转弯调速，左右轮速度改变，到直线运行时，需要将小车速度调回到原来给定的速度上，参考程序中控制方法采用 PI 控制实现速度环控制）；

B、位置环由传感器（CCD 或光电寻迹模块）反馈构成（体现在传感器采集数据即小车实际运行轨迹与标准值之间的偏差调节，参考程序中控制方法采用 PD 控制）。

（三）软件流程框图与涉及到的重要函数模块（图 7.6）

1、检测电池电量

BatteryVoltage()

2、获取 ccd ad 值

void CCD_ReadTsl1401()

计算 ccd 中值

uint8_t CCD_FindMiddleVal()

ccd 采样和计算转弯角度

void CCD_Run(void)

3、电机引脚初始化

void MotorInit(void)

4、小车运动数学模型，给定速度和转角

void KinematicAnalysis()

5、增量 PI 控制器

Incremental_PI_A ()

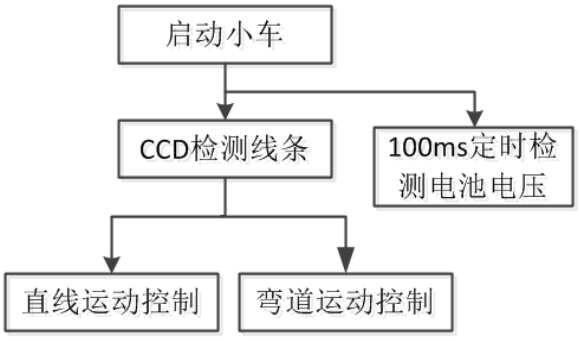


图 7.6 软件流程框图

四、 实验参考程序设计

4.1 使用线性 CCD 循迹小车程序

（注：需要读懂程序，调节参数或部分语句）

```
1.  #include <PinChangeInt.h>     //外部中断
```

```
#include <MsTimer2.h>      //定时中断

//速度 PWM 设置的 KP 参数
#define VELOCITY_KP    0.5
//速度 PWM 设置的 PID KI 参数
#define VELOCITY_KI    0.5

/////////L298N 电机驱动引脚////
#define AIN1   11
#define AIN2   5
#define BIN1   6
#define BIN2   3

//编码器采集引脚
#define ENCODER_LEFT_A    8  //编码器采集引脚每路 2 个共 4 个
#define ENCODER_LEFT_B    4
#define ENCODER_RIGHT_A   7
#define ENCODER_RIGHT_B   2

//CCD 引脚定义
#define CCD_DATA_PIN  A1 //15 //AO
#define CCD_CLK_PIN   A2 //16 //CLK
#define CCD_SI_PIN    A3 //17 //SI

//按键引脚
#define KEY    A4 //18

//电池电压检测引脚
#define BATT_VOL_PIN    A0 //13

//左右编码器值(中间值获取后清零)
uint32_t encoderLeftVal;
uint32_t encoderRightVal;

//小车速度
int velocityLeftEncoder; //后续做运算要用的左编码器值
int velocityRightEncoder;//后续做运算要用的右编码器值
int targetVelocity; //给定速度值
int turnAngle;   //转弯角度

float leftVelocity; //给定速度和转弯角度计算得出的速度(左)
float rightVelocity;//给定速度和转弯角度计算得出的速度(右)

//ccd 相关变量
uint8_t _adVal[128] = {0}; //保存 ccd 采样 ad 值
uint8_t _ccdThreshold; //ccd 阈值
```

```
uint8_t _ccdMiddleVal; //ccd 中值

uint8_t carRunFlag = 0; //小车是否运行 1 运行 0 关闭电池电压低于 9V 时需关闭

//电池电压
int batteryVoltage;

//ccd 延时
void CCD_Delay(uint32_t tick)
{
  for (uint8_t i = 0; i < tick; i++);
}

//ccd 初始化 IO 口
void CCD_Init(void)
{
  pinMode(CCD_CLK_PIN, OUTPUT);
  pinMode(CCD_SI_PIN, OUTPUT);
}

//获取 ccd ad 值
void CCD_ReadTsl1401(void)
{
  unsigned char i = 0;
  digitalWrite(CCD_CLK_PIN, HIGH);
  digitalWrite(CCD_SI_PIN, LOW);
  CCD_Delay(10);
  digitalWrite(CCD_SI_PIN, HIGH);
  digitalWrite(CCD_CLK_PIN, LOW);
  CCD_Delay(10);
  digitalWrite(CCD_CLK_PIN, HIGH);
  digitalWrite(CCD_SI_PIN, LOW);
  CCD_Delay(10);

  for (i = 0; i < 128; i++)  {
    digitalWrite(CCD_CLK_PIN, LOW);
    CCD_Delay(10);  //调节曝光时间
    CCD_Delay(10);  //调节曝光时间
    _adVal[i] = analogRead(CCD_DATA_PIN) >> 2;
    digitalWrite(CCD_CLK_PIN, HIGH);
    CCD_Delay(10);
  }
}

//计算 ccd 中值
uint8_t CCD_FindMiddleVal(void)
```

```
{
  unsigned int i, j, left, right, maxVal, minVal, middleVal;

  CCD_ReadTsl1401();

  maxVal = _adVal[0];  //动态阈值算法，读取最大和最小值

   for (i = 5; i < 123; i++)  //两边各去掉 5 个点
   {
     if (maxVal <= _adVal[i]) {
       maxVal = _adVal[i];
     }
   }
   minVal = _adVal[0];  //最小值
   for (i = 5; i < 123; i++)
   {
     if (minVal >= _adVal[i]) {
       minVal = _adVal[i];
     }
   }

   _ccdThreshold = (maxVal + minVal) / 2; //计算出本次中线提取的阈值

   for (i = 5; i < 118; i++)  //寻找左边跳变沿
   {
     if (_adVal[i]   > _ccdThreshold &&
         _adVal[i + 1] > _ccdThreshold &&
         _adVal[i + 2] > _ccdThreshold &&
         _adVal[i + 3] < _ccdThreshold &&
         _adVal[i + 4] < _ccdThreshold &&
         _adVal[i + 5] < _ccdThreshold)
     {
       left = i;
       break;
     }
   }

   for (j = 118; j > 5; j--) //寻找右边跳变沿
   {
     if (_adVal[j]   < _ccdThreshold &&
         _adVal[j + 1] < _ccdThreshold &&
         _adVal[j + 2] < _ccdThreshold &&
         _adVal[j + 3] > _ccdThreshold &&
         _adVal[j + 4] > _ccdThreshold &&
         _adVal[j + 5] > _ccdThreshold)
     {
```

```
      right = j;
      break;
    }
  }

  middleVal = (right + left) / 2; //计算中线位置
  return middleVal;
}

//左编码器 A 相计数
void EncoderGetLeftA(void)
{
  encoderLeftVal++;
}

//左编码器 B 相计数
void EncoderGetLeftB(void)
{
  encoderLeftVal++;
}

//右编码器 A 相计数
void EncoderGetRightA(void)
{
  encoderRightVal++;
}

//右编码器 B 相计数
void EncoderGetRightB(void)
{
  encoderRightVal++;
}

//编码器引脚初始化
void EncoderInit(void)
{
  pinMode(ENCODER_LEFT_A, INPUT);
  pinMode(ENCODER_LEFT_B, INPUT);
  pinMode(ENCODER_RIGHT_A, INPUT);
  pinMode(ENCODER_RIGHT_B, INPUT);
  attachPinChangeInterrupt(ENCODER_LEFT_A, EncoderGetLeftA, CHANGE); //外部中断库函数
  attachPinChangeInterrupt(ENCODER_LEFT_B, EncoderGetLeftB, CHANGE);
  attachPinChangeInterrupt(ENCODER_RIGHT_A, EncoderGetRightA, CHANGE);
  attachPinChangeInterrupt(ENCODER_RIGHT_B, EncoderGetRightB, CHANGE);
}
```

```
185.
186.
187. /**************************************************************************
188.   函数功能：小车运动数学模型
189.   入口参数：给定速度和转角
190. **************************************************************************/
191. void KinematicAnalysis(float velocity, float turn) {
192.   leftVelocity = velocity + turn;
193.   rightVelocity = velocity - turn;       //后轮差速
194. }
195.
196. /**************************************************************************
197.   函数功能：增量 PI 控制器
198.   入口参数：编码器测量值，目标速度
199.   返回值：电机 PWM
200.   根据增量式离散 PID 公式
201.   pwm+=Kp[e（k）-e(k-1)]+Ki*e(k)+Kd[e(k)-2e(k-1)+e(k-2)]
202.   e(k)代表本次偏差
203.   e(k-1)代表上一次的偏差以此类推
204.   pwm 代表增量输出
205.   在我们的速度控制闭环系统里面，只使用 PI 控制
206.   pwm+=Kp[e（k）-e(k-1)]+Ki*e(k)
207. **************************************************************************/
208. int Incremental_PI_A (int encoder, int target)
209. {
210.   static float bias, pwm, last_bias;
211.   bias = encoder - target;                                  //计算偏差
212.   pwm += VELOCITY_KP * (bias - last_bias) + VELOCITY_KI * bias; //增量式 PI 控制器
213.   if (pwm > 255) pwm = 255;                                 //限幅
214.   if (pwm < -255) pwm = -255;                               //限幅
215.   last_bias = bias;       //保存上一次偏差
216.   return pwm;                                           //增量输出
217. }
218.
219. int Incremental_PI_B (int encoder, int target)
220. {
221.   static float bias, pwm, last_bias;
222.   bias = encoder - target;                                  //计算偏差
223.   pwm += VELOCITY_KP * (bias - last_bias) + VELOCITY_KI * bias; //增量式 PI 控制器
224.   if (pwm > 255) pwm = 255;                                 //限幅
225.   if (pwm < -255) pwm = -255;                               //限幅
226.   last_bias = bias;                                       //保存上一次偏差
227.   return pwm;                                           //增量输出
228. }
229.
230. /**************************************************************************
```

```
231.   函数功能：赋值给 PWM 寄存器
232.   入口参数：PWM
233. ***************************************************************************/
234. void SetMotorPwm(int motorLeft, int motorRight) {
235.   if (motorLeft > 0)       analogWrite(AIN2, motorLeft), digitalWrite(AIN1, LOW); //赋值给 PWM 寄存器
236.   else                  digitalWrite(AIN1, HIGH), analogWrite(AIN2, 255 + motorLeft); //赋值给 PWM 寄存器
237.
238.   if (motorRight > 0)        digitalWrite(BIN2, LOW), analogWrite(BIN1, motorRight); //赋值给 PWM 寄存器
239.   else                   analogWrite(BIN1, 255 + motorRight), digitalWrite(BIN2, HIGH); //赋值给 PWM 寄存器
240. }
241.
242. //小车控制 10ms 运行一次(定时器定时 10ms)
243. void CarControl(void)
244. {
245.   int motorLeftPwm, motorRightPwm;
246.
247.   //读取左右轮编码器数据，并清零，这就是通过 M 法测速（单位时间内的脉冲数）得到速度。
248.   velocityLeftEncoder = encoderLeftVal;
249.   velocityRightEncoder = encoderRightVal;
250.   encoderLeftVal = 0;
251.   encoderRightVal = 0;
252.   CCD_Run();
253.
254.   KinematicAnalysis(targetVelocity, turnAngle);
255.   motorLeftPwm = Incremental_PI_A(leftVelocity, velocityLeftEncoder);
256.   motorRightPwm = Incremental_PI_B(rightVelocity, velocityRightEncoder);
257.   if (carRunFlag)
258.   {
259.     SetMotorPwm(motorLeftPwm, motorRightPwm);
260.   }
261.
262.   // BatteryVoltage(); //检测电池电量
263.
264.   if (ButtonClick()) //按键开关运行
265.   {
266.     if (carRunFlag == 0)
267.     {
268.       carRunFlag = 1;
269.     }
270.     else
271.     {
272.       carRunFlag = 0;
```

```
      TurnOffMotor();
    }
  }
}

//电机引脚初始化
void MotorInit(void)
{
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
}

//ccd 采样和计算转弯角度
void CCD_Run(void)
{
  static float ccd_last_bias;
  float bias;

  _ccdMiddleVal = CCD_FindMiddleVal();
  bias = _ccdMiddleVal - 64;   //提取偏差
  turnAngle = bias * 0.12 + (bias - ccd_last_bias) * 3; //PD 控制
  ccd_last_bias = bias;   //保存上一次的偏差
  if (turnAngle < -35)turnAngle = -35;//舵机角度限制
  if (turnAngle > 35)turnAngle = 35;//舵机角度限制
}

//检测电池电压
void BatteryVoltage(void)
{
  static uint8_t count;
  static uint32_t sum;
  sum += analogRead(BATT_VOL_PIN); //累加电池 ad 值
  if (++count >= 200) //累计 200 次
  {
    batteryVoltage = sum * 0.05371 / 2;//将电池电压降为 1/11 之后读取
    count = 0;
    sum = 0;
    Serial.println(batteryVoltage);
    if (batteryVoltage < 700)
    {
      carRunFlag = 0;
      TurnOffMotor();
    }
  }
```

```
}

/**************************************************************************
  函数功能：按键扫描
  入口参数：无
  返回值：按键状态 0：无动作 1：单击
**************************************************************************/
unsigned char ButtonClick (void)
{
  static byte flag_key = 1; //按键按松开标志
  if (flag_key && (digitalRead(KEY) == 0))
  { //如果发生单击事件
    flag_key = 0;
    if (digitalRead(KEY) == 0)
      return 1;
  }
  else if (digitalRead(KEY) == 1)
  {
    flag_key = 1;
  }
  return 0;//无按键按下
}

//关闭电机
unsigned char TurnOffMotor(void)
{
  digitalWrite(AIN1, LOW);  //电机驱动的电平控制
  digitalWrite(AIN2, LOW);  //电机驱动的电平控制
  digitalWrite(BIN1, LOW);  //电机驱动的电平控制
  digitalWrite(BIN2, LOW);  //电机驱动的电平控制
}

void setup()
{
  targetVelocity = 20; //设定目标速度
  pinMode(KEY, INPUT);
  CCD_Init();
  EncoderInit();
  MotorInit();
  MsTimer2::set(10, CarControl);   //使用 Timer2 设置 10ms 定时中断
  MsTimer2::start();               //定时器使能
  Serial.begin(9600);
}

void loop()
{
```

```
365.   BatteryVoltage(); //检测电池电量
366.   delay(100);
367. }
```

4.2 使用光电寻迹模块的循迹小车程序

```
1.  #include<MsTimer2.h>//定时器库的头文件
2.  //-----------------------------------定义管脚-----------------------------------
3.  #define ENCODER_A1 2  //电机 1
4.  #define ENCODER_B1 5
5.  #define ENCODER_A2 3  //电机 2
6.  #define ENCODER_B2 4
7.  #define PWM1 9
8.  #define PWM2 10
9.  #define L1 14    //左红外
10. #define L2 15
11. #define L3 16
12. #define R1 17    //右红外
13. #define R2 18
14. #define R3 19
15. #define M  13   //中红外
16.
17. //-----------------------------------定义常值-----------------------------------
18. #define PERIOD 20
19. #define Kp 7.0
20. #define Ti 50.0
21. #define Td 10.0
22.
23. //-----------------------------------全局变量-----------------------------------
24. float target1 = 2.0, t1;    //左 保守
25. float target2 = 2.0, t2;    //右 速度
26. volatile long encoderVal1;//编码器 1 值
27. float velocity1;          //转速 1
28. volatile long encoderVal2;//编码器 2 值
29. float velocity2;          //转速 2
30. float T = PERIOD;
31. float q0 = Kp * (1 + T / Ti + Td / T);
32. float q1 = -Kp * (1 + 2 * Td / T);
33. float q2 = Kp * Td / T;
34. float u1, ek11, ek12;
35. float u2, ek21, ek22;
36.
37. //-----------------------------------测速与转向
    -----------------------------------
38. #define V 6.0   //基础速度
```

```
void control(void)
{
  if (digitalRead(M) == LOW)
  {
    target1 = V;
    target2 = V;
  }
  if (digitalRead(L1) == LOW)  //低右转
  {
    target1 = V * 0.9;
    target2 = V * 0.5;
  }
  if (digitalRead(R1) == LOW)  //低左转
  {
    target1 = V * 0.5;
    target2 = V * 0.9;
  }
  if (digitalRead(L2) == LOW)  //中右转
  {
    target1 = V * 0.75;
    target2 = V * 0.15;
  }
  if (digitalRead(R2) == LOW)  //中左转
  {
    target1 = V * 0.15;
    target2 = V * 0.75;
  }
  if (digitalRead(L3) == LOW)  //直角右转
  {
    target1 = 0.65 * V;
    target2 = 0;
  }
  if (digitalRead(R3) == LOW)  //直角左转
  {
    target1 = 0;
    target2 = 0.65 * V;
  }
  target1 = target1 * 0.6 + t1 * 0.4;
  target2 = target2 * 0.6 + t2 * 0.4;

  velocity1 = (encoderVal1 / 780.0) * 3.1415 * 2.0 * (1000 / PERIOD);
  encoderVal1 = 0;
  velocity2 = (encoderVal2 / 780.0) * 3.1415 * 2.0 * (1000 / PERIOD);
  encoderVal2 = 0;

  int output1 = pidController1(target1, velocity1);
```

```
  analogWrite(PWM1, abs(output1));
  int output2 = pidController2(-target2, velocity2);
  if (output2 < 0)
    analogWrite(PWM2, abs(output2));
  t1 = target1;
  t2 = target2;
}

//--------------------------------------主函数
-------------------------------------
void setup() {
  //9,10 两个管脚的 PWM 由定时器 TIMER1 产生，这句程序改变 PWM 的频率，勿删
  TCCR1B = TCCR1B & B11111000 | B00000001;
  MsTimer2::set(PERIOD, control);
  MsTimer2::start();
  pinMode(ENCODER_A1, INPUT);
  pinMode(ENCODER_B1, INPUT);
  pinMode(ENCODER_A2, INPUT);
  pinMode(ENCODER_B2, INPUT);
  attachInterrupt(0, getEncoder1, CHANGE);
  attachInterrupt(1, getEncoder2, CHANGE);
  Serial.begin(9600);
  pinMode(PWM1, OUTPUT);
  pinMode(PWM2, OUTPUT);
  pinMode(R1, INPUT);
  pinMode(R2, INPUT);
  pinMode(R3, INPUT);
  pinMode(L1, INPUT);
  pinMode(L2, INPUT);
  pinMode(L3, INPUT);
  pinMode(M, INPUT);
}
void loop() {

  Serial.print(velocity1);  Serial.print("\t");
  Serial.println(velocity2);
  //Serial.print("\n");
  //Serial.println(u2);
  //Serial.println(u2);
}
//----------------------------------编码器中断函数
-------------------------------------
void getEncoder1(void)
{
  if (digitalRead(ENCODER_A1) == LOW)
  {
```

```
    if (digitalRead(ENCODER_B1) == LOW)
    {
      encoderVal1--;
    }
    else
    {
      encoderVal1++;
    }
  }
  else
  {
    if (digitalRead(ENCODER_B1) == LOW)
    {
      encoderVal1++;
    }
    else
    {
      encoderVal1--;
    }
  }
}
void getEncoder2(void)
{
  if (digitalRead(ENCODER_A2) == LOW)
  {
    if (digitalRead(ENCODER_B2) == LOW)
    {
      encoderVal2--;
    }
    else
    {
      encoderVal2++;
    }
  }
  else
  {
    if (digitalRead(ENCODER_B2) == LOW)
    {
      encoderVal2++;
    }
    else
    {
      encoderVal2--;
    }
  }
}
```

```
175. //-------------------------------------PID 控制器
     ------------------------------------
176. int pidController1(float targetVelocity, float currentVelocity)
177. {
178.   float ek10;
179.   ek10 = targetVelocity - currentVelocity;
180.   u1 = u1 + q0 * ek10 + q1 * ek11 + q2 * ek12;
181.   if (u1 > 255)
182.   {
183.     u1 = 255;
184.   }
185.   if (u1 < -255)
186.   {
187.     u1 = -255;
188.   }
189.   ek12 = ek11;
190.   ek11 = ek10;
191.   return (int)u1;
192. }
193. int pidController2(float targetVelocity, float currentVelocity)
194. {
195.   float ek20;
196.   ek20 = targetVelocity - currentVelocity;
197.   u2 = u2 + q0 * ek20 + q1 * ek21 + q2 * ek22;
198.   if (u2 > 255)
199.   {
200.     u2 = 255;
201.   }
202.   if (u2 < -255)
203.   {
204.     u2 = -255;
205.   }
206.   ek22 = ek21;
207.   ek21 = ek20;
208.   return (int)u2;
209. }
```

五、实验要求

1、构建巡线小车控制硬件系统；

2、使用 Arduino 编程实现线性 CCD 位置数据的采集，根据室内光线亮、暗变化，用串口调试助手分析数据，获取适合的阈值和曝光时间；

3、测量电池电压检测电路的 OUT 输出，分析对应的模拟电压 A 与转换后的数字量 D 之间的关系，如有偏差，请计算补偿数值；

4、使用 Arduino 编程实现小车巡线控制，改变小车速度，分析不同速度下，小车跟踪轨迹运行的效果。

六、拓展实验

1、手持遥感控制小车运动；

2、不规则轨道下小车巡线控制；

3、两辆小车同步运行控制，保持一定的车距。

七、实验报告要求

1、按照自己的理解写出小车巡线控制实验实现过程，采用的方法，得到的结论；

2、分析 CCD 数据，确定阈值和曝光时间，可以给出数据表；

3、电池电压检测的 A、D 数值计算；

4、对实验过程中出现的问题进行总结，写出解决方法与结论。

附录：

第一届机器人竞速赛最佳技术奖欧阳俊源、刘浩文

调试经验总结

1、直角弯的实现：定义左转右转的标志 bool 变量。

A、红外：对左数 1-4 个红外的黑线值进行计数，有 3 个左右的黑色即为左转，左转标志赋为 true，否则为 false,右转同理为 4-7。

B、CCD: CCD 的读取本来就很慢，需要进行优化，否则使速度有瓶颈。并且只只需要在 0-63 的几个等分附近几个连续的数进行判断就可以了(如对 11-15，31-35，51-55)如果这几个值都是小于阈值即为左转。

C、在周期控制函数中对标记变量进行判断是否为 true，是左转则左轮-15 速度，右轮 15 速度，然后延时 300 毫秒。

2、Arduino 的 IDE 有毒，很难 Debug，也不好调试。只能分模块调试好再整合以免从头写到尾后全是 bug 不好找。

3、两个电机调速时候需要查看绘图器，如何在绘图器中绘制两条曲线？

Serial.print(a);Serial.print(“,”);Serial.println(b);

4、线性 ccd 读不出中线或读出的中线不稳定，移动时绘图器中不连续。

可能问题及解决方法：

黑胶布贴着的白纸大一点，清空桌面以免移动 CCD 时看到纸张边缘导致以为 CCD 读出的时错的。

光线干扰（ccd 对光线要求较高）比如阴影什么的。

调整曝光时间。

CCD 读出的 128 个灰度值中其实在两端会慢慢衰减(即看同一张白纸但绘图器上两边的值会低于中间的值，且越来越低)，这会使得有时候两边会被视为黑线处。可以用一张白纸在光线较好处试验，用串口绘图器看，把读出来的数据分别乘上某个系数使得绘图器中的图形是一条直线，然后再进行中线提取效果更好。

也可以调整动态阈值系数。

去除两边的点，或者去除个别长度低于(或许是 1-3)的极低值。

5、CCD 提取中线函数放在周期控制函数中。

现象：CCD 单独实验时能提取到中线，但整合在调速程序中时 CCD 的中线提取出来有问题

解决方法：放在 loop 中(但红外模块不需要)。

6、电量不足（格外注意），电池损坏。

现象：供电不足时小车一般会抖动,行动突然变得和程序不同或反常。

解决方法，可以一直将电压检测模块一直与电池相连，或者每隔一会儿检测一次，

或者设置一个检测电量的函数。

7、程序设计流程（使用光电传感器循迹）