题目名称　　基于 STC89C52的旋转 LED时钟

一、 课程设计的内容

用 STC89C52单片机制作一测温仪旋转　LED时钟：

1. 设计并绘制硬件电路图；
2. 焊接好元器件；
3. 编写程序并将调试好的程序固化到单片机中。

二、 课程设计的要求与数据

单片机采用 STC89C52芯片，时钟芯片采用 DS1302。用电机带动驱动板旋转系统上电后， 驱动板的 LED将会在电机带动下动态扫描出时钟画面，　并且可以用红外遥控调时。

三、 课程设计应完成的工作

1. 完成软件、硬件的设计，并进行硬件的焊接制作，并将调试成功的程序固化到单片机中，最后进行硬件与软件的调试；
2. 撰写设计说明书。

四、 课程设计进程安排

序号	设计各阶段内容	地点	起止日期
1	硬件、软件设计	宿舍	12 月 26 日
2	焊接电路板	宿舍	12 月 27 日至 28 日
3	软件、硬件调试	宿舍	12 月 29 至 1 月 2 日
4	撰写说明书	宿舍	1 月 3 日至 5 日
5	答辩	工 3-317	1 月 7 日

五、 应收集的资料及主要参考文献

谭浩强 .C 语言程序设计（第二版）。北京：清华大学出版社，　1999 年 12 月

目 录

摘　　要

旋转 LED 钟，在国外一般称为“ 螺旋桨时钟 ”（propeller clock），是利用“ 视觉暂留 ”原理制作而成。将单片机控制的　　LED 流水灯设备稍作改进，让它动起来，就能神奇地显示各种字符或图案，其效果如浮在空中一般。旋转　　LED 显示是利用机械转动动态扫描代替传统逐行扫描方式，　　显示屏其实质就是与机械转动配合起来的动态扫描显示技术。本设计利用高速旋转中控制　　LED 灯的亮灭，进行字符的显示，控制器采用　　STC89C52 单片机，借助人的视觉暂留效果，通过 LED 灯的机械扫描方式来显示各种字符和图像。　　LED 旋转时钟正是基于机械转动动态扫描技术，以及人的视觉暂留效果做成的，　它主要包括单片机　STC89C52、时钟芯片　DS1302、光电耦合器件等。

我们做的这个时钟具有两种显示模式：　一种是字符式数字显示模式，　可在一个屏上显示年月日和时分秒信息；　另一种是指针式模拟显示模式，　可仿真指针式钟表显示时分秒信息。 同时还设有红外遥控功能，　可通过遥控器改变显示模式和调整时钟的时值。

关键词：视觉暂留　　旋转时钟　　动态扫描

一、课题设计的要求及目的

1.1 设计要求

（1）驱动板在电机的旋转带动下能够显示时钟画面，并能够自动计时。

（2）由于电机工作电压，环境因素的影响下，电机转速不稳定使时钟画面不稳定，所以要求程序能够自适应调整转速，使时钟画面基本稳定不变。

（3）要求能够通过红外遥控实现数字时钟和指针时钟　　2 种模式间转换

（4）要求能够通过红外遥控设置时间。

1.2 课程设计目的

（1）训练正确地应用单片微机，培养解决工业控制、工业检测等领域具体问题的初步能力。

（2）该设计熟悉单片微机应用系统开发、　研制的过程，　软硬件设计的工作方法、工作内容、工作步骤。

（3）提高学生理论与实践结合的能力，将理论知识运用到实践中来，能更好的掌握课本理论知识。

二、设计方案

(1) 供电方式选择

1、常见的供电方式

根据调查的结果，指针板的供电方式一般有以下三种：

1）自感应发电

这种方法，就是从驱动板上引出导线，接入到电机内部绕在转子上，电机旋转时该导线切割磁场产生感应电动势，经过整流后作为指针板上的电源。

A、这种方式的优点是：

设计很巧妙，无机械磨损。

更巧妙的是，由于感应出来的电动势是交流的，所以可以利用该过零信号来定位，不必另外准备定位信号了。

B、这种方式的缺点是：

提供的电流有限，只能适合　LED　较少的旋转时钟，当　LED　数量较多时，需要更多的电流，这种方式就不能满足了。

其次，这种方式要对电机本身进行改造，也有一定的难度。并不是所有的电机都适合这种改造。而且这种改造可能会给电机带来损害。

另外还有一个问题，就是这种方式，只有在电机旋转时才能发电给驱动板供电，一旦停止转动，供电也就无以为继了，这样要实现旋转时钟的不间断走时，还得另加备用电池并采用低功耗设计。

2）自备电池

这种方式，就是在驱动板上安装电池，由电池供电。一般是用两到三节 7 号电池。

A、这种方式的优点是：

Ⅰ.不用担心电压波动。

Ⅱ.也不存在机械磨损，不用担心接触不良之类问题的困扰。

B、这种方式的缺点是：

Ⅰ.很费电池，三天两头换电池，既不经济也不环保，还很麻烦！

Ⅱ.电池很重，一般的电机带不动，必须用很大很大的电机哦。这也意味了成本的上升。

3）机械传导供电

也就是采用滑环和电刷，通过机械接触传导电流。

A、这种方式的优点是：

能够提供比较大的工作电流。

B、这种方式的缺点是：

Ⅰ.有机械摩擦，会产生磨损。因此要求滑环和电刷材料要耐磨，经得起折腾。另外，还得有足够的弹性，并且要耐锈，否则会导致接触不良。

Ⅱ.有机械阻力，因此要求电机有比较大一点的功率。

Ⅲ.有机械噪音。

4）感应供电

原理和变压器原理相当，就是在 2 个相距很近的线圈中，一只线圈作为电能发送端，另一只线圈作为电能接收端，发送端接入交变电流，在相距很近的接收端就能同时感应到交变电流。

A、这种方式的优点是：

无机械噪音。

B、这种方式的缺点是：

Ⅰ.线圈耦合度低，供电效率低。

Ⅱ.制作难度大。

Ⅲ.需增加震荡电路和滤波整流电路。

综合以上三种： 第一种， 虽然优点多， 但难度很大， 并且成本很高； 第二种，没有太多的担心， 可是使用起来相当的麻烦， 可能还会因为更换电池不及时而导致其中其他的零件受损；第三种，虽然会产生些摩擦，但是能提供较大电流，而摩擦的问题可以采用其他方法来弥补。 第四种， 虽不产生机械噪音， 但是需要在驱动板上加上滤波整流电路， 增加驱动板重量。 所以我们采用的是第三种机械传导供电。

（2）过零信号产生电路选择

1 ）霍尔传感器

霍尔传感器处于工作状态时输出总是处于高电平状态， 当磁钢 N极接近传感器正面的有效距离， 输出端变为低电平。 当磁钢撤离传感器有效距离。 输出端又显示低电平，从而产生下降沿，是单片机中断口接收到下降沿，从而产生中断。

2 ）光电开关

光电开关处于工作状态时输出总是处于高电平状态，当光电开关经过挡片时，输出端变为低电平。当光电开关离开挡片时，输出端又显示低电平，从而产生下降沿，是单片机中断口接收到下降沿，从而产生中断。

光电开关利用光敏二极管对光的敏感性原理制作的，反应较灵敏，且低电平时间由挡片的宽度决定， 控制方便。而霍尔传感器利用磁场对电场的作用原理制作的，反应较迟缓， 且感应磁钢的距离远， 当转速较快时难以控制。故该设计采用光电开关。

（3） LED选择

由于旋转 LED要求时钟的分辨率高且重量轻，长度短，故该设计选用贴片LED发光二极管

（4）单片机选择

由于旋转 LED驱动板上包括较多元器件和芯片， 空间不足，所以该设计选用 PLCC

封装的单片机，这种封装的单片机为正方形，面积小，质量轻。

三、系统框图与工作原理

3.1 单片机系统工作架构

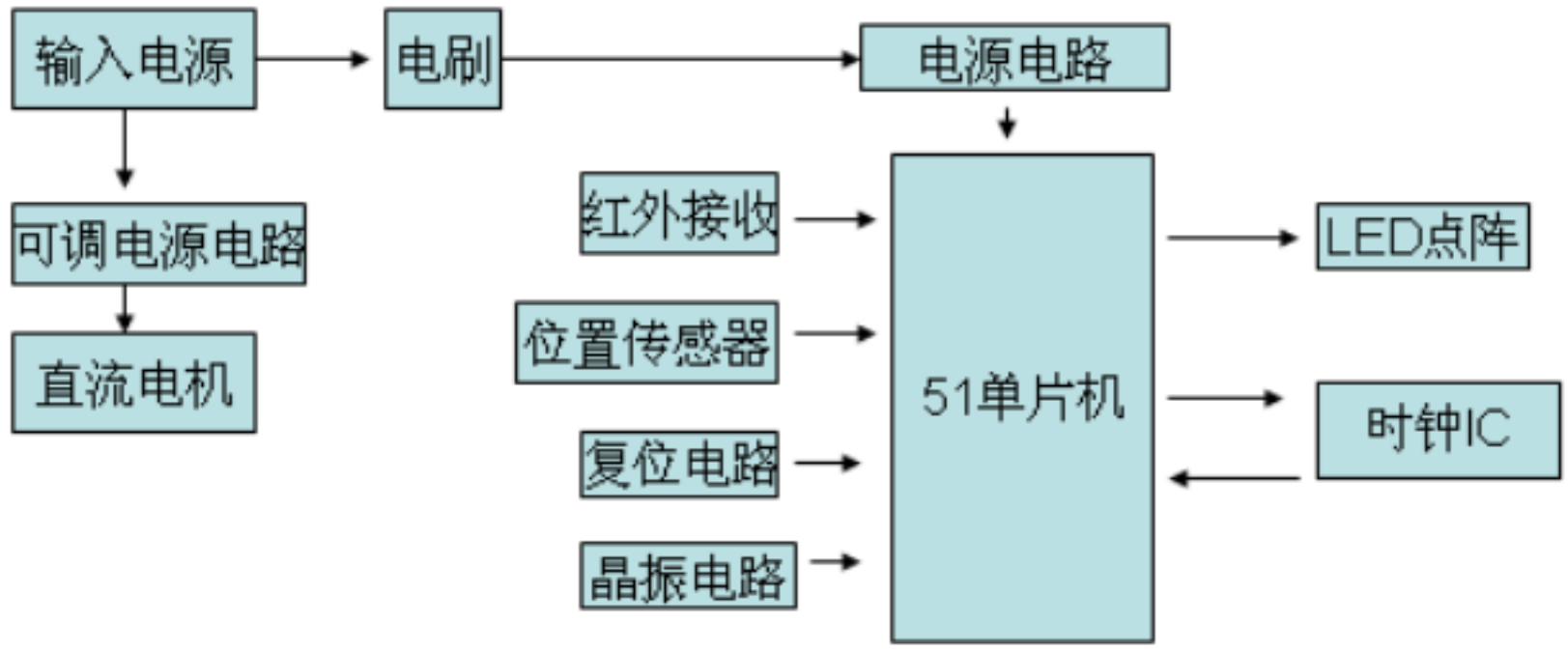


图 3.1.1　系统框图

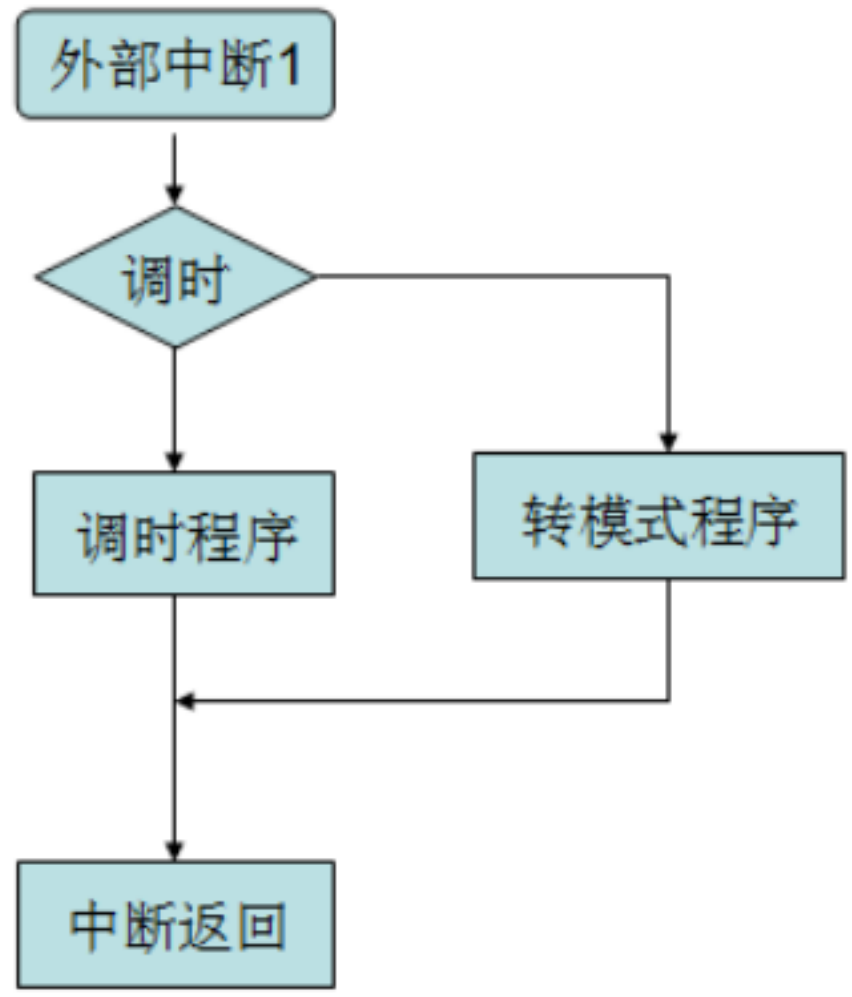


图 3.1.2 外部中断 1 流程图

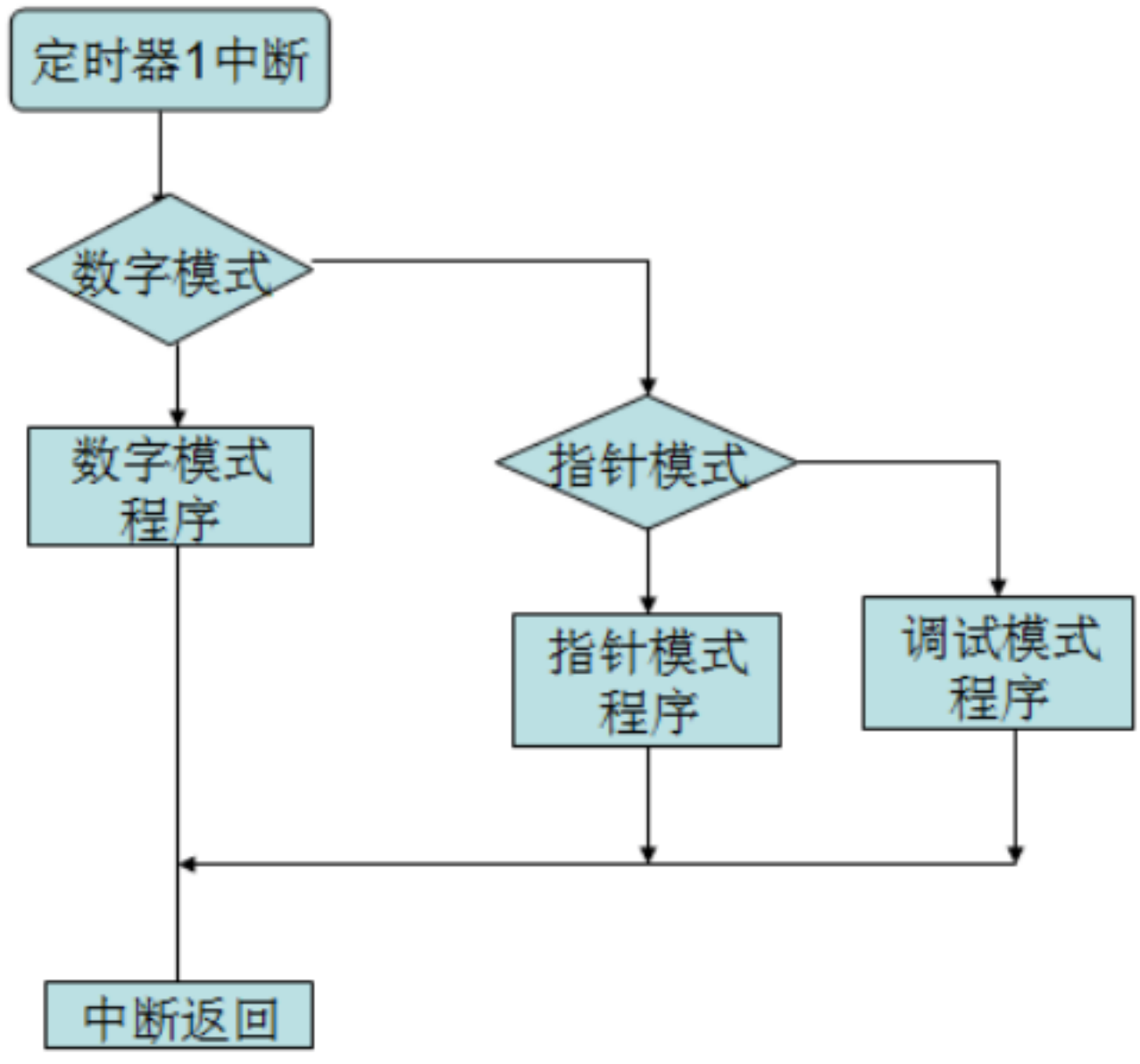


图 3.1.3　定时器 1 流程图

3.3 系统工作原理

通过光耦和外部中断控制单片机从　DS1302中读取时钟数据并在旋转　LED中的显示位置，通过红外遥控外部中断读取控制旋转　LED时钟的显示模式和时值的加减调控。

四、设计元器件说明

4.1 PLCC　STC89C52RC 简介

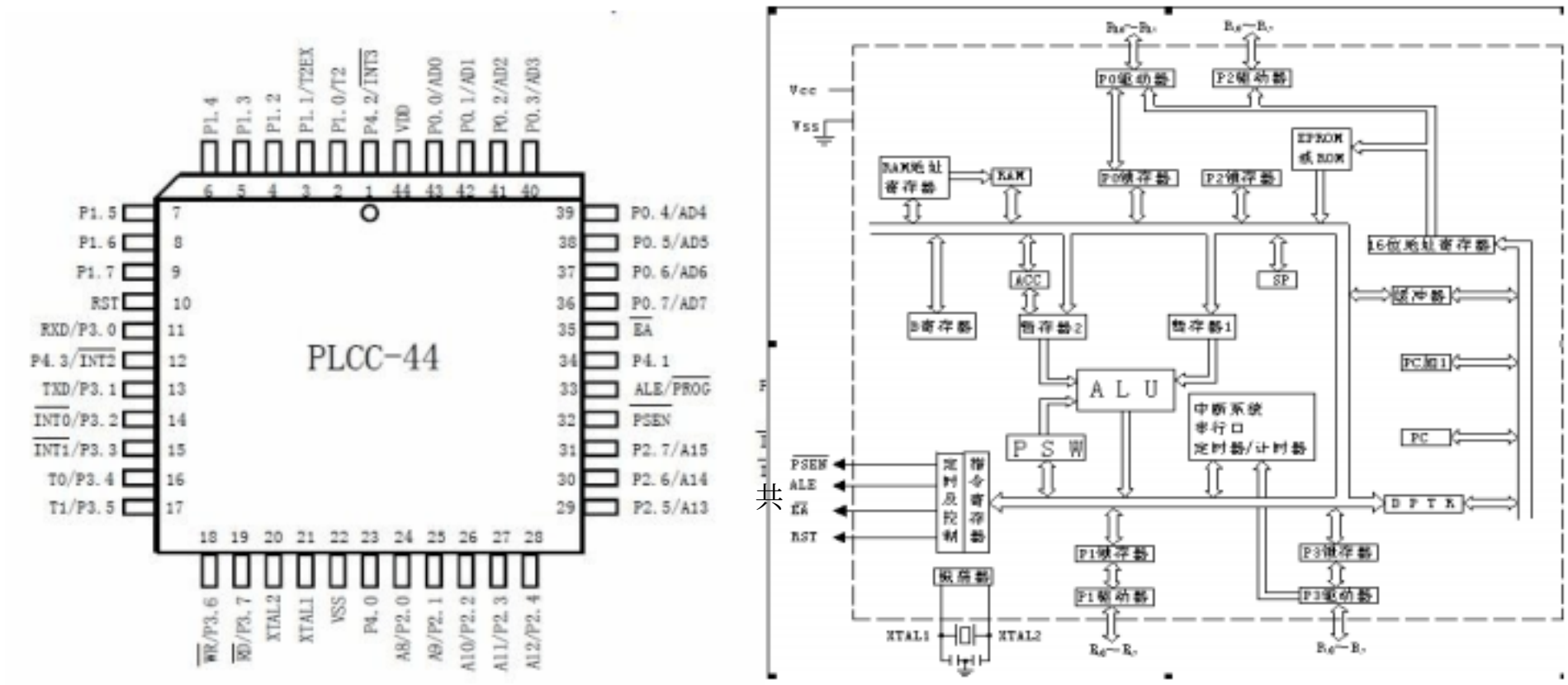


图 **4.1.1 PLCC STC89C52RC** 管脚图　　图 **4.1.2 STC89C52RC** 单片机结构

STC89C52RC是一种带 4KB可编程可擦除只读存储器的低电压，高性能微处理器，俗称单片机。单片机的可擦除只读存储器可以反复擦除 100 次。该器件采用 ATMEL高密度非易失存储器制造技术制造， 与工业标准的 MCS-51 指令集和输出管脚相兼容。由于将多功能 8 位 CPU和闪烁存储器组合在单个芯片中， STC89C52RC 是一种高效微控制器， STC89C52RC 是它的一种精简版本。 STC89C52RC 单片机为很多嵌入式控制系统提供了一种灵活性高且价廉的方案。 STC89C52RC 引脚即外观如图 4.1.1 所示，内部结构如图 4.1.2 所示。

P0 口：P0 口是一组 8 位漏极开路型双向 I/O 口，也即地址 / 数据总线复制用口，作为输入口时， 每位能吸收电流的方式驱动 8 个 TTL 逻辑门电路， 对端口写入“ 1 可作为高阻抗输入端用。在访问外部数据存储器或程序存储器时，这组口线分时转换地址（低 8 位）和数据总线复用，在访问期激活内部上拉电阻。在 Flash 编程时， PO口接收指令节，而在程序校检时，输出指令字节，校检时，要求外接上拉电阻。

P1 口：P1 口是一个带内部上拉电阻的 8 位双向 I/O 口， P1 的输出缓冲级可驱动（吸收或输出电流） 4 个 TTL 逻辑门电路。对端口写“ 1 ”，通过内部的上拉电阻把端口拉到高电平， 此时可作输入口， 作输入口时，因为内部存在上拉电阻，某个引脚被外部信号拉低时会输出一个电流 I 。 Flash 编程和程序校检期间， P1 接收低 8 位地址。

P2 口:P2 口是一个带内部上拉电阻的 8 位双向 I/O 口， P1 的输出缓冲级可驱动（吸收或输出电流） 4 个 TTL 逻辑门电路。对端口写“ 1 ”，通过内部的上拉电阻把端口拉到高电平， 此时可作输入口， 作输入口时，因为内部存在上拉电阻，某个引脚被外部信号拉低时会输出一个电流 I 。在访问外部数据存储器或 16 位地址的外部数据存储（例如执行 MOVX@DPTR 指令）时， P2 口送出高 8 位地址数据。在访问 8 位地址的外部数据存储器（如执行 MOVX@Ri 指令）时， P2 口线上的内容（也即特殊功能寄存器（ SFR）区中 R2 寄存器的内容），在整个访问期间不改变。 Flash 编程和校检时， P2 亦接收高位地址和其他控制信号。

P3口：P3口是一个带内部上拉电阻的 8位双向 I/O 口。P3口输出缓冲级可驱动（吸收或输出电流） 4个 TTL逻辑门电路。对 P3口写入“ 1”时，它们被内部上拉电阻拉高并可作输入端口，作输入端时，被外部拉低的 P3口将用上拉电阻，输出电流 I。P3口还接收一些用于 Flash 闪速存储器编程和程序校检的控制信号。

RST:复位输入，当震荡器工作时， RST引脚出现两个机器周期以上高电平将使单片机复位。

ALE/PROG 当访问外部程序存储器或数据存储器时 ,ALE(地址锁存允许）输出脉冲用于所存地址的低 8位字节。即使不访问外部存储器， ALE乃以时钟振动频率的 1/6 输出固定的正脉冲信号， 因此它可对外输出时钟或用于定时目的。 要注意的是：每当访问外部数据存储器时将跳过一个 ALE脉冲。

4.2 74HC573 芯片与光电传感器简介

4.2.1 74HC573 芯片

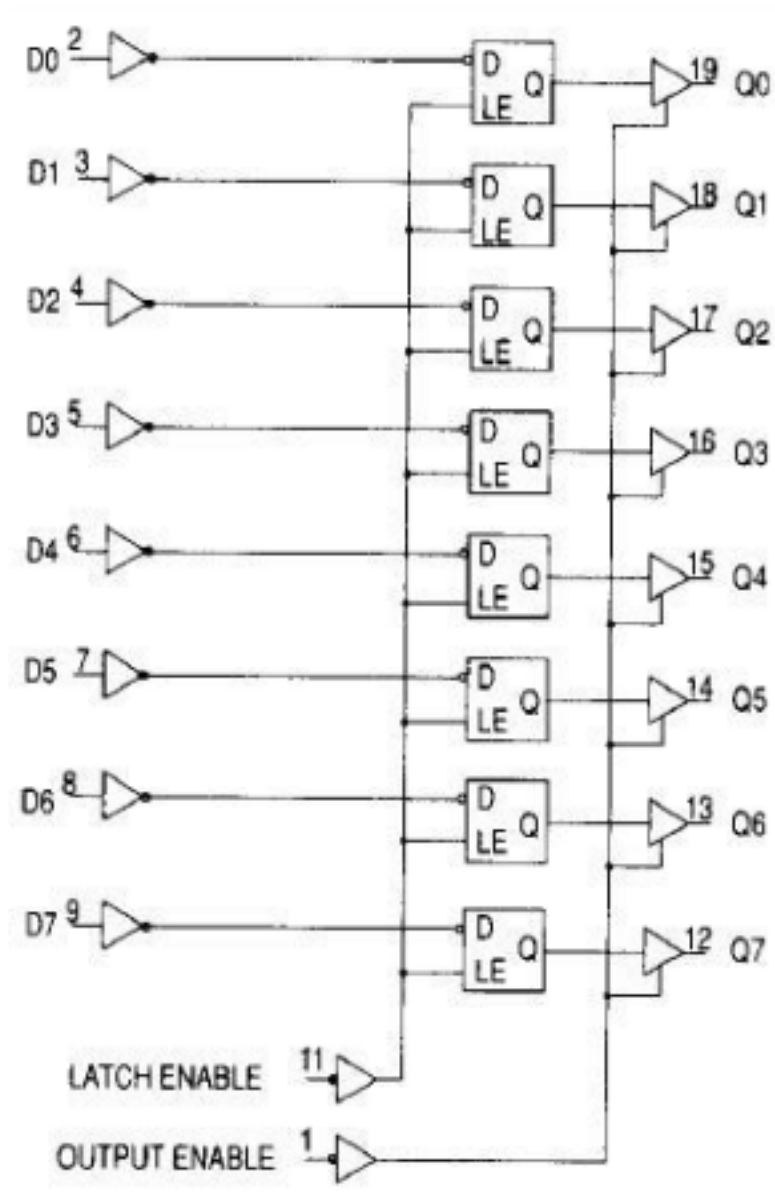


图 4.2.1 74HC573

芯片 74HC573为八进制 3 态非反转透明锁存器，是高性能硅门 cmos器件。当锁存使能端 LE为高时，这些器件的锁存对于数据是透明的（也就是说输出同步）。当锁存使能变低时，符合建立时间和保持时间的数据会被锁存。

4.2.2 光电传感器

对射式 U型槽光耦管具有，响应块，驱动简单，容易安装，易于与单片机通信等特点如上图。 上电之后光耦的光敏三极管的集电极输出低电平， 当有物体挡住了光敏三极管的红外光线时， 光敏三极管的集电极和发射极处于高阻态， 所以集电极输出高电平， 当光敏三极管再次感应到红外光源时， 集电极再次输出低电平，从而给单片机一个中断信号。

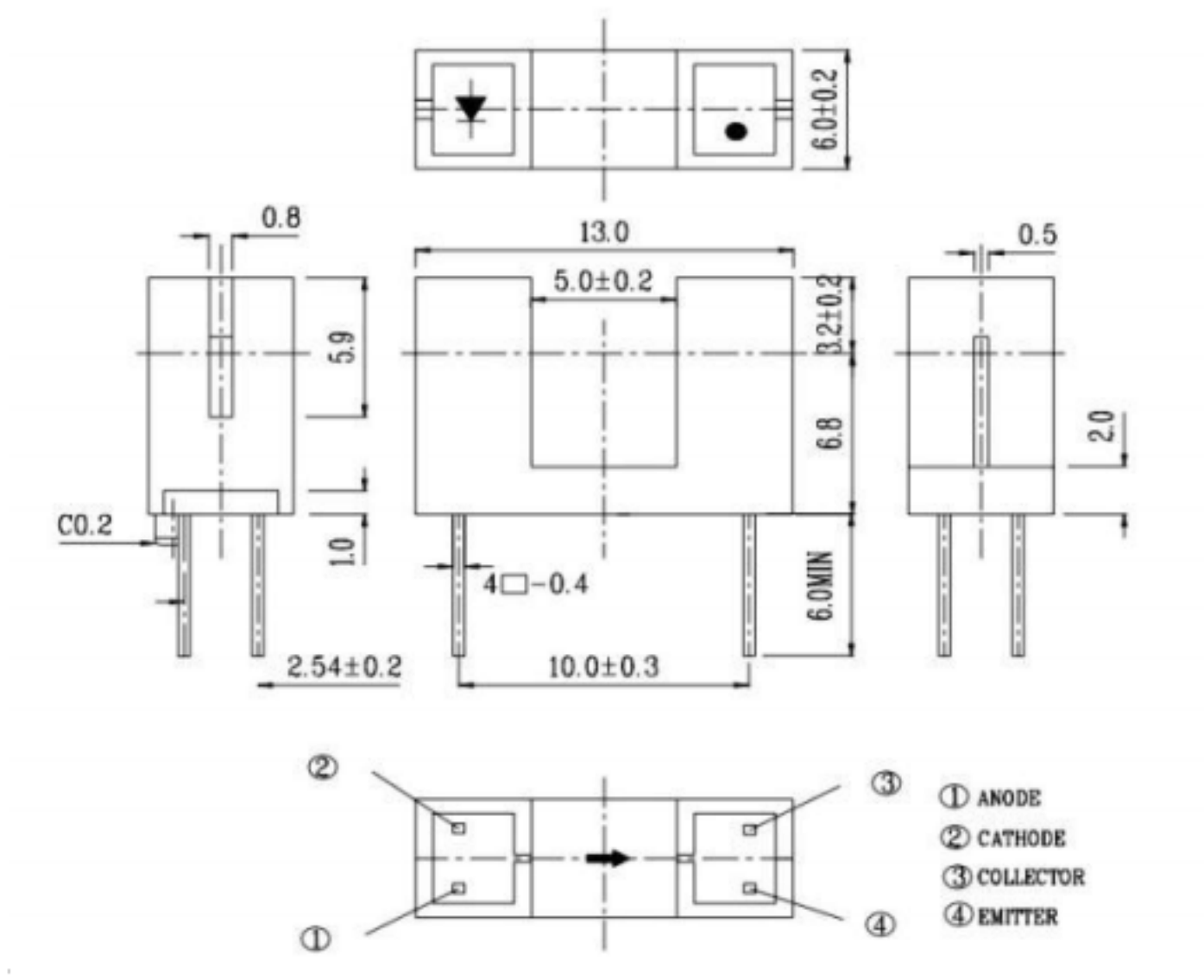


图 4.2.2 光电开关，ITR9608-F 中间宽 5MM

槽式光电开关通常是标准的 U字型结构，其发射器和接收器分别位于 U型槽的两边， 并形成一光轴， 当被检测物体经过 U型槽且阻断光轴时， 光电开关就产生了检测到的开关量信号。 槽式光电开关比较安全可靠的适合检测高速变化， 分辨透明与半透明物体。

4.3 红外简介

4.3.1　一体化红外遥控接收头

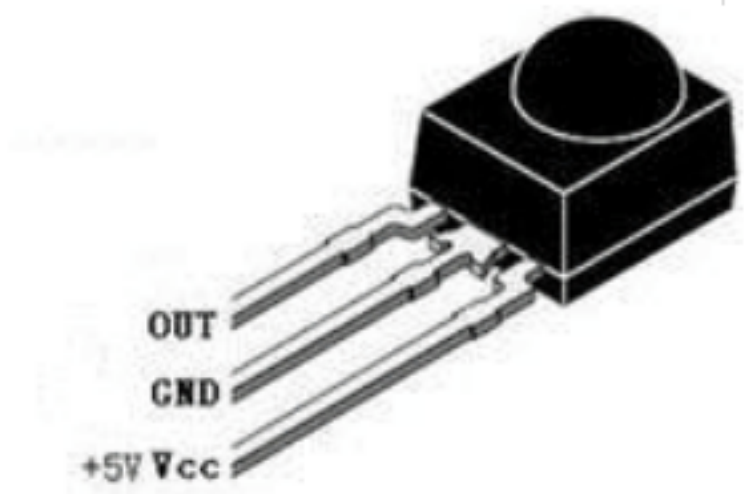


图 4.3.1　HS0038B SIP3 一体化红外遥控接收头

红外遥控信号是一连串的二进制脉冲码。　为了使其在无线传输过程中免受其他红外信号的干扰，通常都是先将其调制在特定的载波频率上　，然后再经红外发射二极管发射出去，而红外线接收装置则要滤除其他杂波　，只接收该特定频率的信号并将其还原成二进制脉冲码　，也就是解调。目前，对于这种进行了调制的红外遥控信号，　通常是采用一体化红外线接收头进行调解。　一体化红外线接收头将低噪音放大器，限幅器，带通滤波器，解调器，，以及整形驱动电路等集成在一起。一体化红外线接收头体积小　，灵敏度高，外接元件少，抗干扰能力强，使用十分方便。

4.3.2　遥控发射器

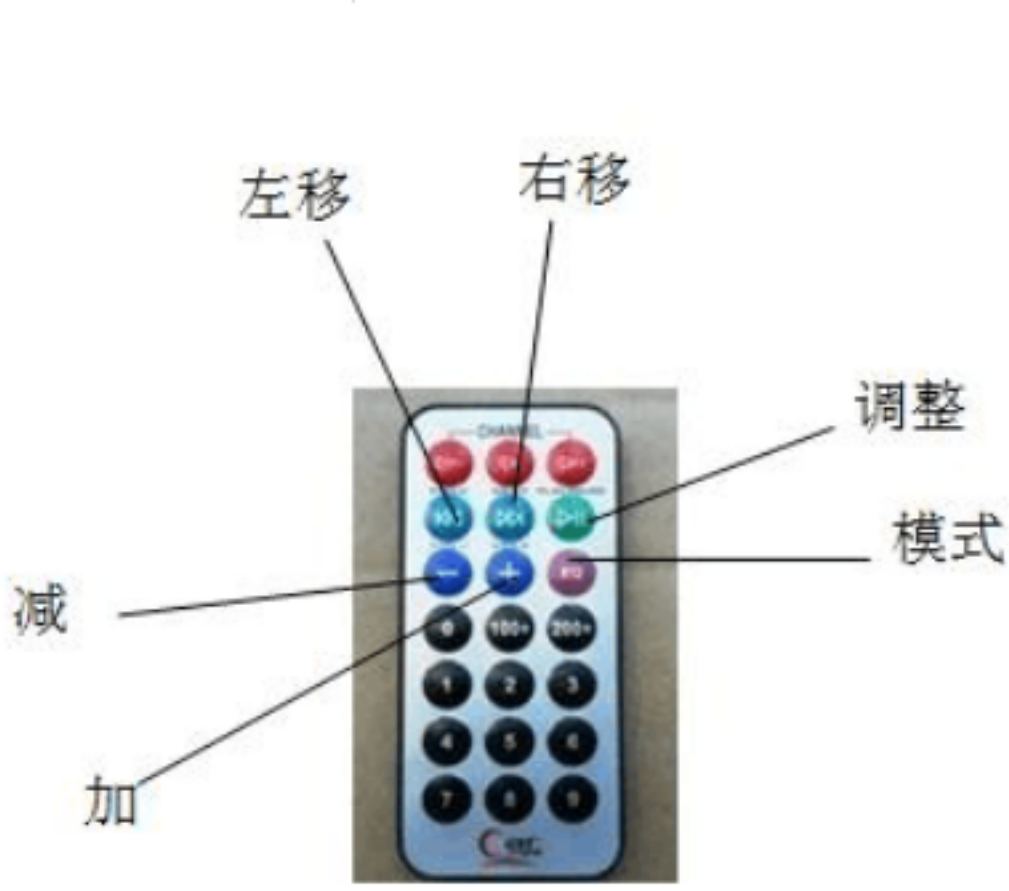


图 4.3.2　红外遥控器

遥控发射器及其编码

遥控发射器专用芯片很多，　根据编码格式可以分成两大类，　这里我们以运用比较广泛，解码比较容易的一类来加以说明，现以日本　NEC的uPD6121G组成发射电路为例说明编码原理。　当发射器按键按下后，　即有遥控码发出，　所按的键不同遥控编码也不同。这种遥控码具有以下特征：

采用脉宽调制的串行码，以脉宽为　0.565ms、间隔 0.56ms、周期为 1.125ms 的组合表示二进制的“　0 ”；以脉宽为 0.565ms、间隔

1.685ms、周期为 2.25ms的组合表示二进制的“ 1”，其波形如图 2所示。

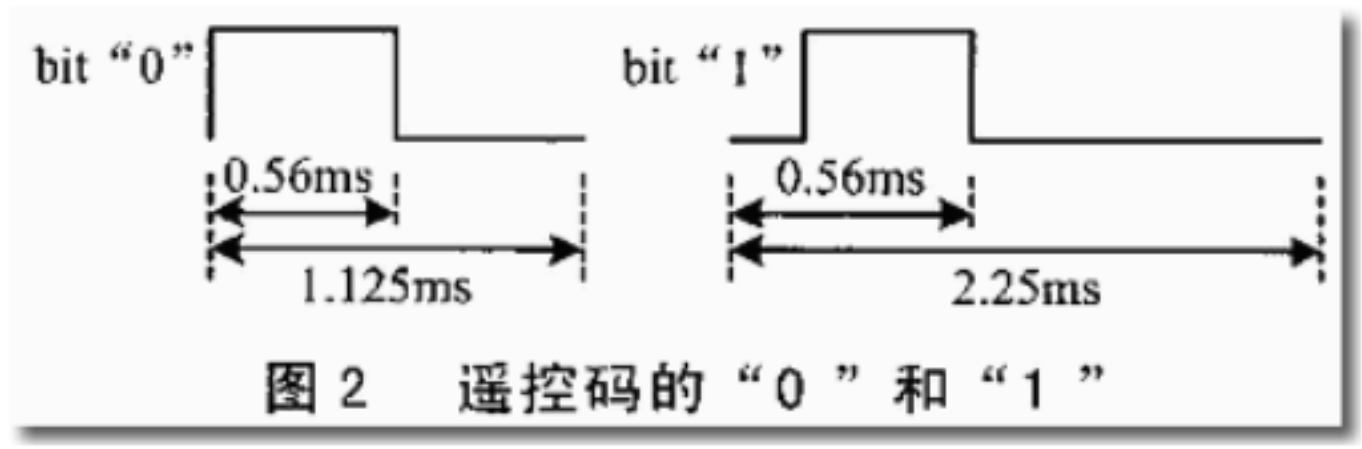


图 2 遥控码的“0”和“1”

上述“ 0”和“ 1”组成的 32位二进制码经 38kHz的载频进行二次调制以提高发射效率， 达到降低电源功耗的目的。 然后再通过红外发射二极管产生红外线向空间发射，如图 3所示，连发波形如图 4所示。

图 3 遥控信号编码波形图

图 4 遥控信号的周期性波形

4.4 DS1302、LM317、LM7805 芯片简介

4.4.1 DS1302

DS1302 是美国 DALLAS公司推出的一种高性能、 低功耗、带 RAM的实时时钟电路，它可以对年、月、日、周日、时、分、秒进行计时，具有闰年补偿功能，工作电压为 2.5V～5.5V。采用三线接口与 CPU进行同步通信，并可采用突发方式一次传送多个字节的时钟信号或 RAM数据。DS1302内部有一个 31×8的用于临时性存放数据的 RAM寄存器。 DS1302是DS1202的升级产品，与 DS1202兼容，但增加了主电源/后备电源双电源引脚，同时提供了对后备电源进行涓细电流充电的能力。

引脚功能及结构

DS1302的引脚排列，其中 Vcc1为后备电源， VCC2为主电源。在主电源关闭的情况下，也能保持时钟的连续运行。 DS1302由 Vcc1或Vcc2两者中的较大者供电。当Vcc2大于 Vcc1+0.2V时，Vcc2给DS1302供电。当Vcc2小于 Vcc1时，DS1302由Vcc1供电。 X1和X2是振荡源，外接 32.768kHz晶振。 RST是复位/片选线，通过把 RST输入驱动置高电平来启动所有的数据传送。 RST输入有两种功能： 首先， RST接通控制逻辑， 允许地址/命令序列送入移位寄存器； 其次， RST提供终止单字节或多字节数据的传送手段。当 RST为高电平时，所有的数据传送被初始化，允许对DS1302进行操作。如果在传送过程中 RST置为低电平，则会终止此次数据传送，I/O 引脚变为高阻态。上电运行时，在 Vcc>2.0V之前， RST必须保持低电平。只有在SCLK为低电平时，才能将 RST置为高电平。 I/O 为串行数据输入输出端 (双向)，后面有详细说明。 SCLK为时钟输入端。 下图为 DS1302的引脚功能图：

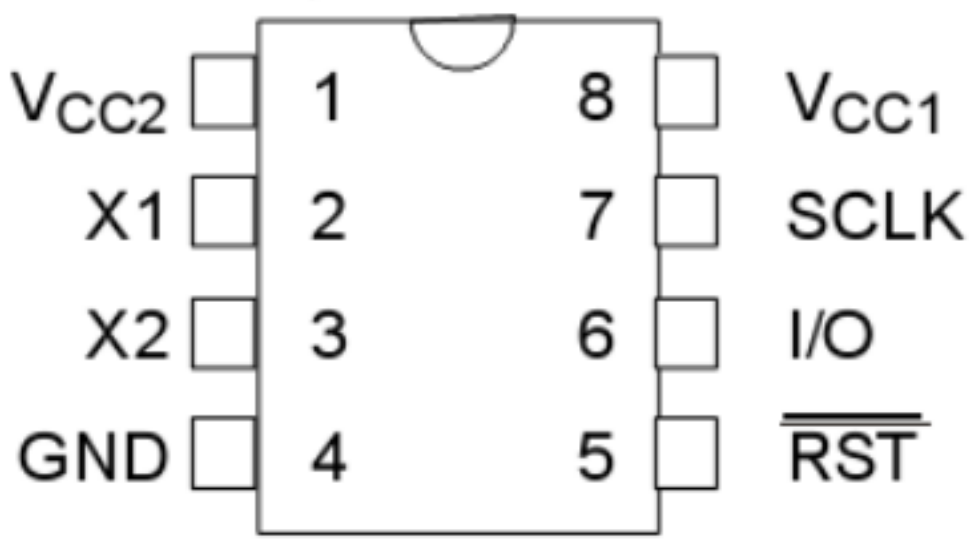
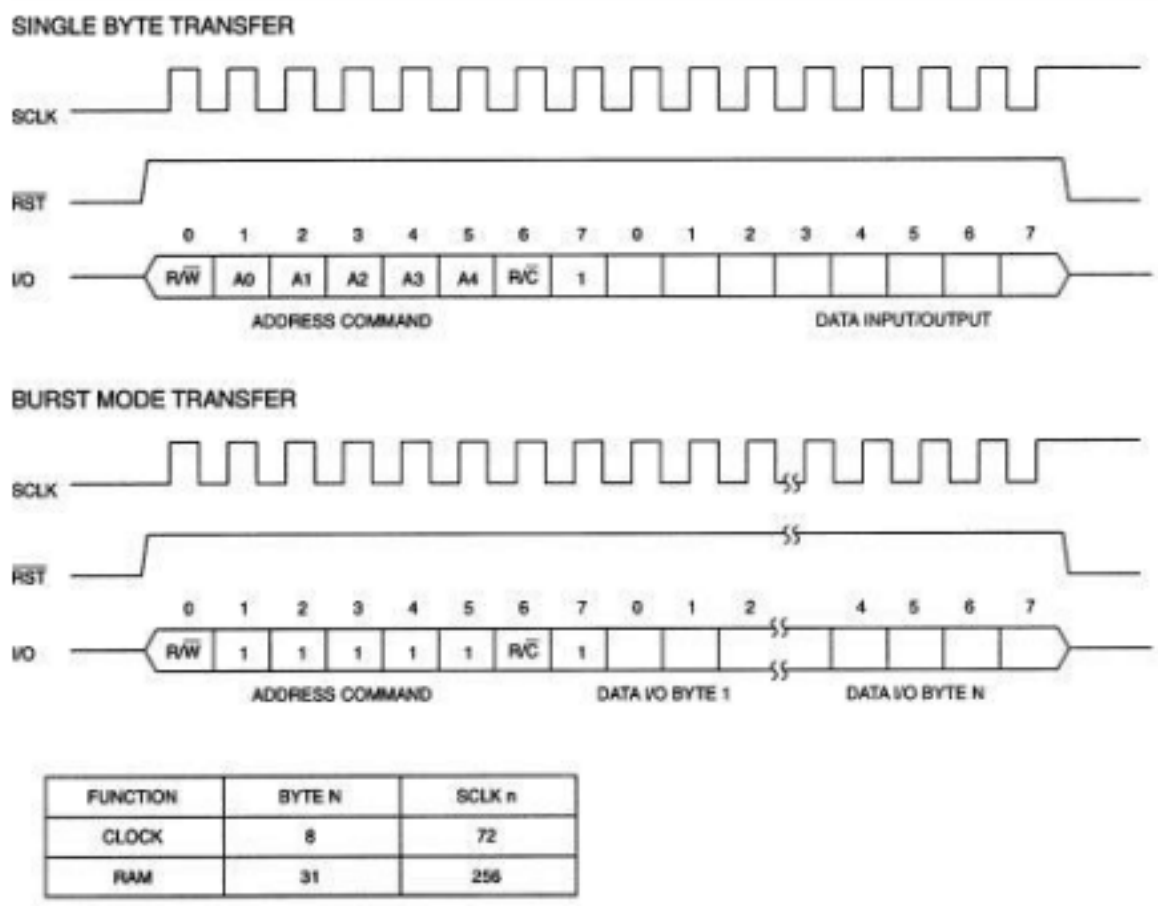


图4.4.1 DS1302 管脚定义图

数据输入输出 (I/O)

在控制指令字输入后的下一个 SCLK时钟的上升沿时，数据被写入 DS1302，数据输入从低位即位 0 开始。同样，在紧跟 8 位的控制指令字后的下一个 SCLK 脉冲的下降沿读出 DS1302的数据，读出数据时从低位 0 位到高位 7。



FUNCTION	BYTE N	SCLK n
CLOCK	8	72
RAM	31	256

图 4.4.2 数据输入输出

DS1302 的控制字节

DS1302的控制字如图 2 所示。控制字节的最高有效位 (位 7) 必须是逻辑 1，如果它为 0，则不能把数据写入 DS1302中，位 6 如果为 0，则表示存取日历时钟数据，为 1 表示存取 RAM数据; 位 5 至位 1 指示操作单元的地址 ; 最低有效位 (位 0) 如为 0 表示要进行写操作， 为 1 表示进行读操作， 控制字节总是从最低位开始输出。

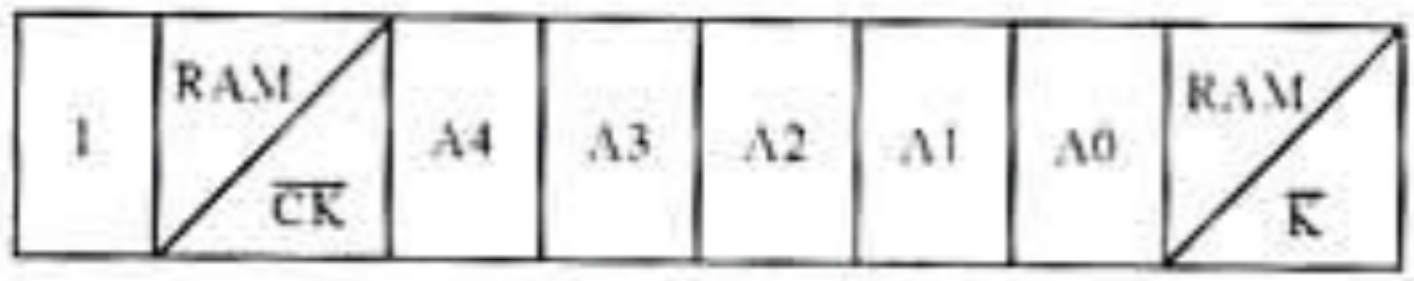
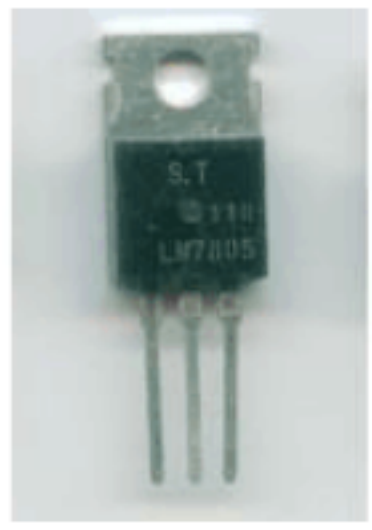


图 2　DS1302 的控制字节

4.4.2　LM7805

三端稳压集成电路 lm7805，组成稳压电源所需的外围元件极少，　电路内部还有过流、过热及调整管的保护电路，　使用起来可靠、 方便，而且价格便宜。 lm7805 输出电压为正 5V。



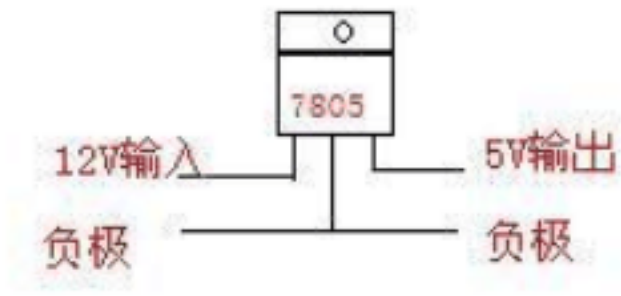


图1　LM7805封装图　　图2　LM7805引脚图

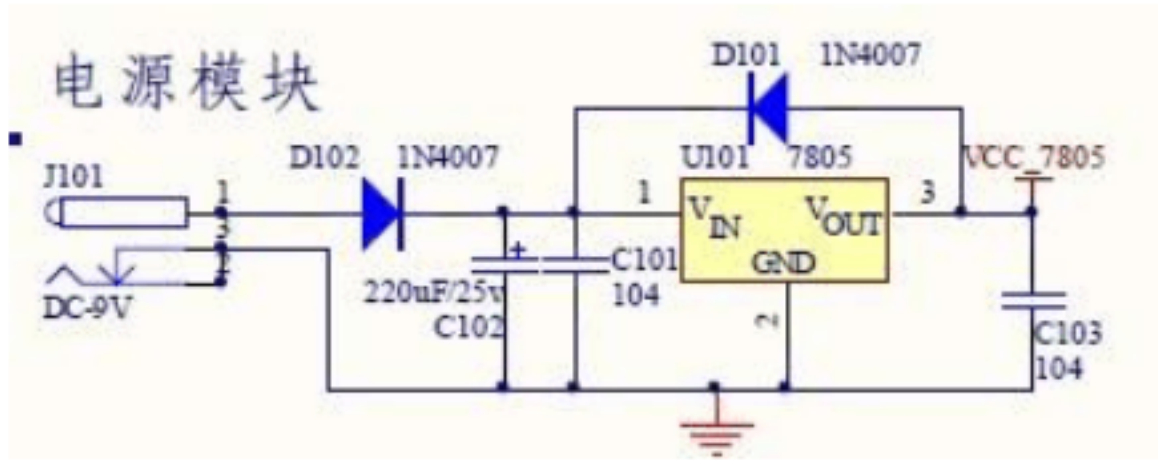


图3　LM7805典型应用电路

4.5 LED 动态显示原理

LED 具有低功耗，接口控制方便等优点，而且与模块的接口信号和操作指令具有广泛的兼容性，能直接与单片机接口，方便实现各种不同的操作。

旋转 LED是一种通过同步控制发光二极管的位置和点亮状态来实现图文显

示，可视角能达 360 度，本设计采用 32 个发光二级管，利用人眼的“ 视觉暂留效应 ”显示时间和温度。

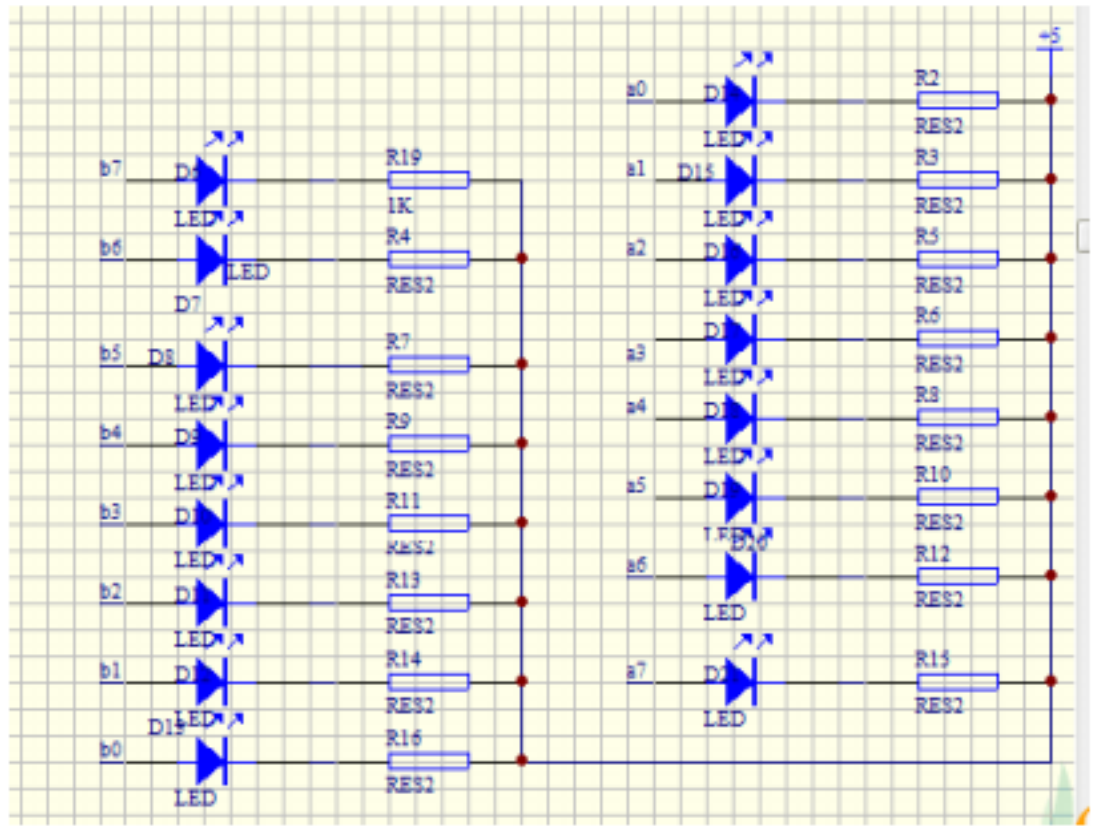
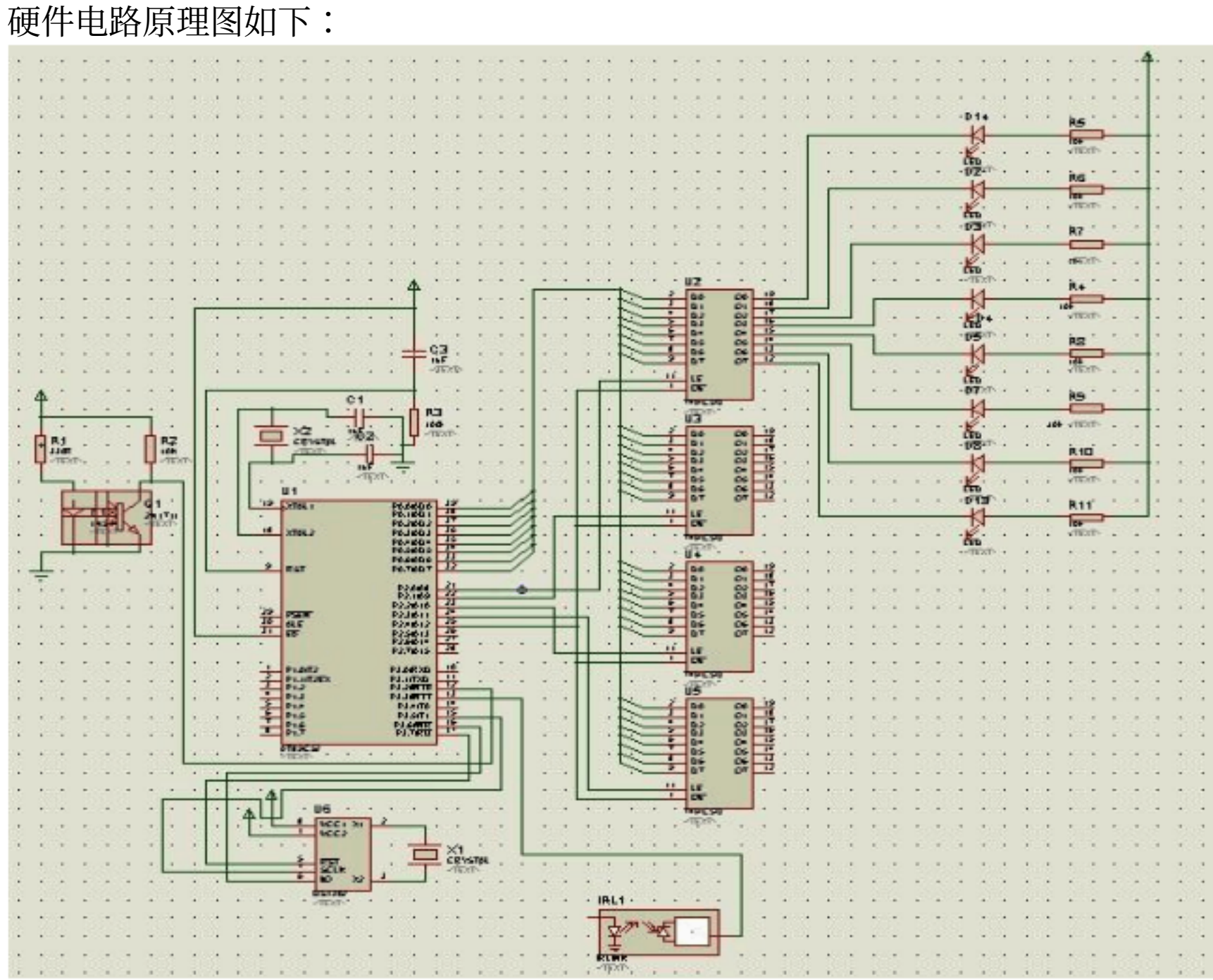


图4.5 Led显示模块

五、系统硬件电路设计

硬件电路原理图如下：



LED灯为8*4组共32颗

六、系统软件设计

6.1 单片机解码红外信号程序

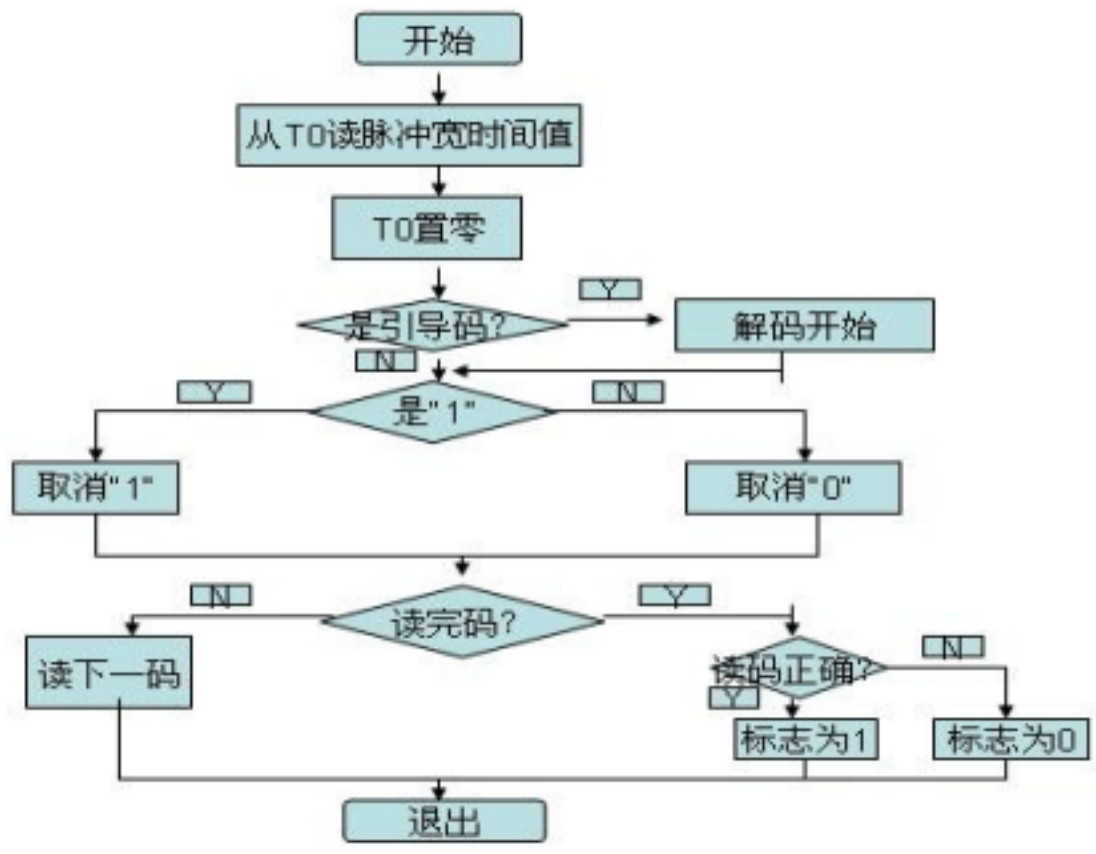


图6.1.1 红外解码流程图

以下为单片机解码红外的程序：

```
#include <reg52.h>
#define uchar unsigned char
uchar code table[]={
0xc0,0xf9,0xa4,0xb0,
0x99,0x92,0x82,0xf8,
0x80,0x90,0x88,0x83,
0xc6,0xa1,0x86,0x8e};

#define Imax 14000                  //          此处为晶振为  11.0592 时的取值 ，
#define Imin 8000                  //          如用其它频率的晶振时    ，
#define Inum1 1450                  //          要改变相应的取值。
#define Inum2 700
#define Inum3 3000

uchar Im[4]={0x00,0x00,0x00,0x00};          //      存放 4个字节 32位编码
uchar show[2]={0,0};                        //      存储数据码
unsigned long m,Tc;                 //              两脉冲间隔
uchar flag,IrOK;


void delay(uchar i)
{
  uchar j,k;
  for(j=i;j>0;j--)
```

```
    for(k=125;k>0;k--);
}

void display()
{
   P1=6;
   P0=table[show[0]];
   delay(5);

   P1=5;
   P0=table[show[1]];
   delay(5);
}
// 外部中断解码程序
void INT_1(void) interrupt 2 using 1
{
 Tc=TH0*256+TL0;                    //                    提取中断时间间隔时长
 TH0=0;
 TL0=0;                             //                    定时中断重新置零
 if((Tc>Imin)&&(Tc<Imax))            //9.12ms+4.5ms
     {
       m=0;
       flag=1;
       return;
     }      //            找到启始码
if(flag==1)
    {
      if(Tc>Inum1&&Tc<Inum3)          //2.25ms
       {
         Im[m/8]=Im[m/8]>>1|0x80; m++;     //1                    取码
       }
      if(Tc>Inum2&&Tc<Inum1)          //1.125ms
       {
         Im[m/8]=Im[m/8]>>1; m++;          //0                          取码
       }

if(m==32)
  {
    m=0;
    flag=0;
    if(Im[2]==~Im[3])
     {
       IrOK=1;
     }
```

```
            else IrOK=0;                    //                  取码完成后判断读码是否正确
          }                                 //                  准备读下一码
       }
}


/* 演示主程序  */
void main(void)
{
unsigned int  a;
m=0;
flag=0;
EA=1;
IT1=1;   //        下降沿触发
EX1=1;   //        外部中断 1允许
TMOD=0x01;
TH0=0;
TL0=0;
TR0=1;//   开启定时器  0
  while(1)
  {
      if(IrOK==1)
{
  show[1]=Im[2] & 0x0F;            //                  取键码的低四位
  show[0]=Im[2] >> 4;
  IrOK=0;
}
          if(Im[2]==69)
          {
           P0=0x55;
          }
          if(Im[2]==70)
          {
           P0=0xaa;
          }
          if(Im[2]==7)
          {
           P0=0xff;
          }
          if(Im[2]==9)
          {
           P0=0x00;
          }
for(a=100;a>0;a--)
```

```
        {
            display();
        }
    }
}
```

6.2 单片机读写 DS1302程序

以下为典型的 DS1302读写程序：

```
/*****************************************************************
****
*
*  名称 :    RTInputByte
*  说明 :
*  功能 :  往DS1302写入1Byte数据
*  调用 :
*  输入 : ucDa  写入的数据
*  返回值 :  无
*****************************************************************
******/
void    RTInputByte(unsigned char ucDa)
{
unsigned char i;
ACC = ucDa;
for(i=8; i>0; i--)
{
T_IO = ACC0; /* 相当于汇编中的    RRC */
T_CLK = 1;
T_CLK = 0;
ACC = ACC >> 1;
}
}

/*****************************************************************
****
*
*  名称 : unsigned char uc_RTOutputByte
*  说明 :
*  功能 :  从DS1302读取1Byte数据
*  调用 :
*  输入 :
*  返回值 : ACC
*****************************************************************
```

```
******/
        unsigned char uc_RTOutputByte(void)
        {
        unsigned char i;
        for(i=8; i>0; i--)
        {
        ACC = ACC >>1; /* 相当于汇编中的    RRC */
        ACC7 = T_IO;
        T_CLK = 1;
        T_CLK = 0;
        }
        return(ACC);
        }

        /*************************************************************
****
        *
        *  名称 :    W1302
        *  说明 : 先写地址，后写命令  /数据
        *  功能 : 往DS1302写入数据
        *  调用 :    RTInputByte()
        *  输入 : ucAddr: DS1302地址 , ucDa: 要写的数据
        *  返回值 : 无
        *************************************************************
******/
        void    W1302(unsigned char ucAddr, unsigned char ucDa)
        {
        T_RST = 0;
        T_CLK = 0;
        T_RST = 1;
        RTInputByte(ucAddr); /*   地址，命令   */
        RTInputByte(ucDa); /*   写1Byte数据 */
        T_CLK = 1;
        T_RST =0;
        }

        /*************************************************************
****
        *
        *  名称 : uc_R1302
        *  说明 : 先写地址，后读命令  /数据
        *  功能 : 读取 DS1302某地址的数据
        *  调用 :    RTInputByte() , uc_RTOutputByte()
        *  输入 : ucAddr: DS1302地址
```

```
*  返回值 : ucDa :读取的数据
*********************************************************************
******/
unsigned char uc_R1302(unsigned char ucAddr)
{
unsigned char ucDa;
T_RST = 0;
T_CLK = 0;
T_RST = 1;
RTInputByte(ucAddr); /*  地址，命令  */
ucDa = uc_RTOutputByte(); /*  读1Byte数据  */
T_CLK = 1;
T_RST =0;
return(ucDa);
}

void    Set1302(void)
{
W1302(0x8e,0x00); /*  控制命令 ,WP=0,写操作 ?*/
W1302(0x8c,NUM2BCD(Time[5]));
W1302(0x8a,NUM2BCD(Time[6]));
W1302(0x88,NUM2BCD(Time[4]));
W1302(0x86,NUM2BCD(Time[3]));
W1302(0x84,NUM2BCD(Time[2]));
W1302(0x82,NUM2BCD(Time[1]));
W1302(0x80,NUM2BCD(Time[0]));
W1302(0x8e,0x80); /*  控制命令 ,WP=1,写保护 ?*/
}

void    Get1302(void)
{
Time[5] = BCD2NUM(uc_R1302(0x8d));
Time[6] = BCD2NUM(uc_R1302(0x8b));
Time[4] = BCD2NUM(uc_R1302(0x89));
Time[3] = BCD2NUM(uc_R1302(0x87));
Time[2] = BCD2NUM(uc_R1302(0x85));
Time[1] = BCD2NUM(uc_R1302(0x83));
Time[0] = BCD2NUM(uc_R1302(0x81));
}
```

6.3 自适应转速

系统开机后，程序先进入测试转速阶段，测试 2 次中断之间（旋转一圈）定时器中断数，与所需的中断数对比，通过对比调整定时器初设值，达到改变旋

转 1 圈定时器中断数的目的。

```
/* 外部中断  0 处理函数 */
        void intersvr0(void) interrupt 0 using 1
    {
                        D=D+(S-N)*2;            //  修正值
                        Ti0=600+D;              //  得到定时器  T0 的初设值
                        S=0;                    //  计数器清零，将重新计数
                }
              /*  定时中断  0 处理函数 */
             void timer0(void) interrupt 1 using 1
{
  TH1=-Ti0/256;TL1=-Ti0%256; //  设置定时器  T0 的初设值
                        S++;                    //  计数
 }
```

其中 S 为旋转一圈定时器 1 实际中断数（既实际显示的列数），D 为调整值，N为旋转一圈定时器所需的中断数（既所需显示的列数） 。程序分析如下：

0、在外部中断的处理程序里，先给定时器一个合适的初设值 Pt。

程序开始， D=0， Ti0 即为 600（按需设定），并得到定时器初设值。

1、定时器开始计数，每溢出一次 S 自加一次。

2、完成一圈后，处理外部中断函数。

当 S>N时，修正值 D 增大，使定时器 T1 的时间值增大，随之 S 值减小。

当 S<N时，修正值 D 减小，使定时器 T1 的时间值减小，随之 S 值增大。

当 S=N时，修正值不产生变化。

函数中 N 值是按需设定的常数。

6.4 数字显示模式

时钟的上半部分和下半部分的显示是相反的，故需要对数字显示进行调整。故分 2 步显示，第一步显示上半部分，正常显示。程序如下：

```
if(ii<16)
        {
```

```
                P2=0xf1;P1=~nAsciiDot1[jj*2+v[ii]*16];          //          显示的上半圆部
分（正显）
                P2=0xf2;P1=~nAsciiDot1[1+jj*2+v[ii]*16];
        }
```

16 表示显示 16 个字

第二步显示下半部分，因需和上半部现实相反，故下半部反向显示，程序如下：

```
    else if(ii<32)
            {
                P2=0xf2;P1=~nAsciiDot2[14-jj*2+v[ii]*16];          //          显示的下半圆部
分（反显）
                P2=0xf1;P1=~nAsciiDot2[15-jj*2+v[ii]*16];
            }
```

6.5 指针显示模式

显示时，由于要将 12 点的位置和数字模式统一显示到正上方，而不是中断发生处，故需进行转换：

```
            P2=0xf1;P1=(0xfe<<3*(ii%5==4))&(0xff>>2*(ii==Tme0))&(0xfe<<5*(ii
%15==14));
            P2=0xf2;P1=(0xff*(ii!=Tme0)*(ii!=Tme1))&(0xff>>4*(ii==Tme2));
            P2=0xf4;P1=(0xff*(ii!=Tme0)*(ii!=Tme1)*(ii!=Tme2));
            P2=0xf8;P1=(0xff*(ii!=Tme0)*(ii!=Tme1)*(ii!=Tme2));
```

七、总结与体会

回顾起此次单片机课程设计， 至今我仍感慨颇多， 从选题到定稿， 从理论到实践，在整整两星期的日子里， 可以说得是苦多于甜， 但是可以学到很多很多的的东西，同时不仅可以巩固了以前所学过的知识， 而且学到了很多在书本上所没有学到过的知识。通过这次课程设计使我懂得了理论与实际相结合是很重要的，只有理论知识是远远不够的， 只有把所学的理论知识与实践相结合起来， 从理论中得出结论， 才能真正为社会服务， 从而提高自己的实际动手能力和独立思考的

能力。在设计的过程中遇到问题，可以说得是困难重重，这毕竟第一次做的，难免会遇到过各种各样的问题， 同时在设计的过程中发现了自己的不足之处， 对以前所学过的知识理解得不够深刻，掌握得不够牢固。 DS1302 和单片机解码红外这一部份花了一天多才弄明白，深感自己对所学过的知识的理解之浅。

这次的课设硬件弄的时间比较长， 因为是用万用版焊制， 出现比较多的问题，例如芯片的管脚就因大意弄错了序号好几次， 经过几次的仔细查找才找出了问题的所在。程序方面倒没有太大的错误发生，不过到最好作品完成的时候 DS1302 突然坏了， 但未能及时发现， 直到花了一天的时间排查了所有硬件才最终确定了 DS1302 的损坏导致显示出现乱码，这也直接导致了我们的作品未能在第一天被验收。

这次单片机课程设计。 我们发挥团队精神。 相互去学习，去解决问题的所在，以及一起探讨。 希望我们在实训结束后同样能够去更进一步的去学习单片机， 巩固和加深对单片机的学习。

八、参考文献

[1] ． 李朝青 . 单片机原理及接口技术（第 3 版）. 北京航天航空大学出版社, 2005 年 10 月

[2]. 谭浩强 .C 语言程序设计（第二版） · 北京：清华大学出版社, 1999 年 12 月

[3]. 彭伟． 单片机 C语言程序设计实训 100 例 :基于 8051+Proteus 仿真 . 电子工业出版社 .2009 年

[4]. 网上资料

附录 A 完整源程序

```
#include <reg52.h>
#include <intrins.h>
#include <absacc.h>
#define uchar unsigned char
unsigned char code maxnum[]={59,23,31,12,99}; //              调整值最大限量
unsigned char code minnum[]={0,0,1,1,0};      //              调整值最小限量
unsigned char code nAsciiDot1[] =             // ASCII
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // - -
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
```

```
0x00,0x0C,0x00,0x06,0x00,0x03,0x80,0x01,  // -/-
0xC0,0x00,0x60,0x00,0x30,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x06,  // -:-
0x30,0x06,0x00,0x00,0x00,0x00,0x00,0x00,

0xF8,0x07,0xFC,0x0F,0x04,0x09,0xC4,0x08,  // -0-
0x24,0x08,0xFC,0x0F,0xF8,0x07,0x00,0x00,

0x00,0x00,0x10,0x08,0x18,0x08,0xFC,0x0F,  // -1-
0xFC,0x0F,0x00,0x08,0x00,0x08,0x00,0x00,

0x08,0x0E,0x0C,0x0F,0x84,0x09,0xC4,0x08,  // -2-
0x64,0x08,0x3C,0x0C,0x18,0x0C,0x00,0x00,

0x08,0x04,0x0C,0x0C,0x44,0x08,0x44,0x08,  // -3-
0x44,0x08,0xFC,0x0F,0xB8,0x07,0x00,0x00,

0xC0,0x00,0xE0,0x00,0xB0,0x00,0x98,0x08,  // -4-
0xFC,0x0F,0xFC,0x0F,0x80,0x08,0x00,0x00,

0x7C,0x04,0x7C,0x0C,0x44,0x08,0x44,0x08,  // -5-
0xC4,0x08,0xC4,0x0F,0x84,0x07,0x00,0x00,

0xF0,0x07,0xF8,0x0F,0x4C,0x08,0x44,0x08,  // -6-
0x44,0x08,0xC0,0x0F,0x80,0x07,0x00,0x00,

0x0C,0x00,0x0C,0x00,0x04,0x0F,0x84,0x0F,  // -7-
0xC4,0x00,0x7C,0x00,0x3C,0x00,0x00,0x00,

0xB8,0x07,0xFC,0x0F,0x44,0x08,0x44,0x08,  // -8-
0x44,0x08,0xFC,0x0F,0xB8,0x07,0x00,0x00,

0x38,0x00,0x7C,0x08,0x44,0x08,0x44,0x08,  // -9-
0x44,0x0C,0xFC,0x07,0xF8,0x03,0x00,0x00,
};

unsigned char code nAsciiDot2[] =              // ASCII
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // - -
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

    0x00,0x30,0x00,0x60,0x00,0xC0,0x01,0x80,  // -/-
```

0x03,0x00,0x06,0x00,0x0C,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x0C,0x60, // -:-
0x0C,0x60,0x00,0x00,0x00,0x00,0x00,0x00,

0x1F,0xE0,0x3F,0xF0,0x20,0x90,0x23,0x10, // -0-
0x24,0x10,0x3F,0xF0,0x1F,0xE0,0x00,0x00,

0x00,0x00,0x08,0x10,0x18,0x10,0x3F,0xF0, // -1-
0x3F,0xF0,0x00,0x10,0x00,0x10,0x00,0x00,

0x10,0x70,0x30,0xF0,0x21,0x90,0x23,0x10, // -2-
0x26,0x10,0x3C,0x30,0x18,0x30,0x00,0x00,

0x10,0x20,0x30,0x30,0x22,0x10,0x22,0x10, // -3-
0x22,0x10,0x3F,0xF0,0x1D,0xE0,0x00,0x00,

0x03,0x00,0x07,0x00,0x0D,0x00,0x19,0x10, // -4-
0x3F,0xF0,0x3F,0xF0,0x01,0x10,0x00,0x00,

0x3E,0x20,0x3E,0x30,0x22,0x10,0x22,0x10, // -5-
0x23,0x10,0x23,0xF0,0x21,0xE0,0x00,0x00,

0x0F,0xE0,0x1F,0xF0,0x32,0x10,0x22,0x10, // -6-
0x22,0x10,0x03,0xF0,0x01,0xE0,0x00,0x00,

0x30,0x00,0x30,0x00,0x20,0xF0,0x21,0xF0, // -7-
0x23,0x00,0x3E,0x00,0x3C,0x00,0x00,0x00,

0x1D,0xE0,0x3F,0xF0,0x22,0x10,0x22,0x10, // -8-
0x22,0x10,0x3F,0xF0,0x1D,0xE0,0x00,0x00,

0x1C,0x00,0x3E,0x10,0x22,0x10,0x22,0x10, // -9-
0x22,0x30,0x3F,0xE0,0x1F,0xC0,0x00,0x00,
};
unsigned char code HZ_12[] =
{

0x80,0x00,0x88,0x1F,0x30,0x48,0x00,0x24, //"　　调"
0xF0,0x1F,0x10,0x01,0x50,0x1D,0xF0,0x15,
0x50,0x1D,0x10,0x41,0xF8,0x7F,0x10,0x00,

0x00,0x02,0x00,0x41,0x80,0x41,0x60,0x31, //"　　分"
0x18,0x0F,0x00,0x01,0x00,0x21,0x38,0x41,

```
0x40,0x3F,0x80,0x00,0x00,0x01,0x00,0x01,

0xE0,0x1F,0x20,0x09,0x20,0x09,0x20,0x09,  //"          时"
0xE0,0x1F,0x40,0x00,0x40,0x01,0x40,0x26,
0x40,0x40,0xF8,0x7F,0x40,0x00,0x40,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0xF0,0x3F,  //"          日"
0x10,0x11,0x10,0x11,0x10,0x11,0x10,0x11,
0x10,0x11,0xF8,0x3F,0x10,0x00,0x00,0x00,

    0x00,0x40,0x00,0x20,0x00,0x10,0xF8,0x0F,  //"          月"
    0x48,0x02,0x48,0x02,0x48,0x22,0x48,0x42,
    0x48,0x42,0xF8,0x3F,0x00,0x00,0x00,0x00,

    0x80,0x04,0x40,0x04,0x20,0x04,0x98,0x07,  //"          年"
    0x90,0x04,0x90,0x04,0xF0,0x7F,0x90,0x04,
    0x90,0x04,0x98,0x04,0x90,0x04,0x10,0x04
};

// 延时函数
#define Delay10Us
{_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();}
#define Delay100Us {Delay10Us;Delay10Us;Delay10Us;Delay10Us;Delay10Us;\
                    Delay10Us;Delay10Us;Delay10Us;Delay10Us;Delay10Us;}

#define NUM2BCD(x) ((((x)/10)<<4)|(x%10))
#define BCD2NUM(x) (((x)>>4)*10+((x)&0x0f))

sbit T_CLK = P3^5; /*       实时时钟时钟线引脚     */
sbit T_IO  = P3^6; /*       实时时钟数据线引脚     */
sbit T_RST = P3^7; /*       实时时钟复位线引脚     */

sbit ACC0=ACC^0;
sbit ACC7=ACC^7;

unsigned char
Time[]={0x00,0x15,0x11,0x04,0x07,0x09,0x02};//Second,Minute,Hour,Day,Month,Yea
r,Week
#define Imax 28000    //       此处为晶振为  11.0592 时的取值 ,
#define Imin 25000    //       如用其它频率的晶振时    ,
#define Inum1 2300    //       要改变相应的取值。
#define Inum2 2000
#define Inum3 4600
```

```
uchar Im[4]={0x00,0x00,0x00,0x00};          //    存放 4个字节 32位编码
unsigned long m,Tc;          //    两脉冲间隔
uchar flag,IrOK;

unsigned char ii,jj;               //              循环变量，用于显示   LED
unsigned char Tme0,Tme1,Tme2;          //          时间中间变量
unsigned int Ti0;               //              定时器 T1预设值
unsigned char v[32];              //              显示缓冲区
unsigned char TZ;               //              调整项
unsigned char ST;               //              调整标志
unsigned char DM;
unsigned int S;
unsigned long D;                //              显示模式
// 外部中断解码程序
void INT_1(void) interrupt 2 using 1
{//ET1=0; P1=0xff;P2=0xff;
 Tc=TH0*256+TL0;               //                   提取中断时
间间隔时长
 TH0=0;//              09oik
 TL0=0;        //         定时中断重新置零
 if((Tc>Imin)&&(Tc<Imax))    //9.12ms+4.5ms
    {
     m=0;
     flag=1;
     return;
    }    //       找到启始码
 if(flag==1)
    {
     if(Tc>Inum1&&Tc<Inum3)    //2.25ms
      {
       Im[m/8]=Im[m/8]>>1|0x80; m++; //1
      }
     if(Tc>Inum2&&Tc<Inum1)    //1.125ms
      {
       Im[m/8]=Im[m/8]>>1; m++; //0                  取码
      }


     if(m==32)
       {
        m=0;
        flag=0;
        if(Im[2]==~Im[3])
```

```
            {
               IrOK=1;
            }
           else IrOK=0;   //                  取码完成后判断读码是否正确
         }
            //              准备读下一码
      }
}

/*********************************************************************
*
*   名称 : RTInputByte
*   说明 :
*   功能 :  往DS1302写入 1Byte 数据
*   调用 :
*   输入 : ucDa    写入的数据
*   返回值 :  无
*********************************************************************/
void  RTInputByte(unsigned char ucDa)
{
unsigned char i;
ACC = ucDa;
for(i=8; i>0; i--)
{
T_IO = ACC0; /*    相当于汇编中的    RRC */
T_CLK = 1;
T_CLK = 0;
ACC = ACC >> 1;
}
}

/*********************************************************************
*
*   名称 : unsigned char uc_RTOutputByte
*   说明 :
*   功能 :  从DS1302读取 1Byte 数据
*   调用 :
*   输入 :
*   返回值 : ACC
*********************************************************************/
unsigned char uc_RTOutputByte(void)
{
```

```
unsigned char i;
for(i=8; i>0; i--)
{
ACC = ACC >>1; /*    相当于汇编中的     RRC */
ACC7 = T_IO;
T_CLK = 1;
T_CLK = 0;
}
return(ACC);
}

/*************************************************************************
*
*   名称 :  W1302
*   说明 :   先写地址，后写命令    / 数据
*   功能 :   往DS1302写入数据
*   调用 :  RTInputByte()
*   输入 : ucAddr: DS1302    地址 , ucDa:     要写的数据
*   返回值 :   无
*************************************************************************/
void  W1302(unsigned char ucAddr, unsigned char ucDa)
{
T_RST = 0;
T_CLK = 0;
T_RST = 1;
RTInputByte(ucAddr); /*          地址，命令    */
RTInputByte(ucDa); /*         写1Byte 数据 */
T_CLK = 1;
T_RST =0;
}

/*************************************************************************
*
*   名称 : uc_R1302
*   说明 :   先写地址，后读命令    / 数据
*   功能 :   读取 DS1302某地址的数据
*   调用 :  RTInputByte() , uc_RTOutputByte()
*   输入 : ucAddr: DS1302    地址
*   返回值 : ucDa :    读取的数据
*************************************************************************/
unsigned char uc_R1302(unsigned char ucAddr)
{
unsigned char ucDa;
T_RST = 0;
```

```
T_CLK = 0;
T_RST = 1;
RTInputByte(ucAddr); /*          地址，命令    */
ucDa = uc_RTOutputByte(); /*          读1Byte 数据  */
T_CLK = 1;
T_RST =0;
return(ucDa);
}


void  Set1302(void)
{
W1302(0x8e,0x00); /*        控制命令  ,WP=0, 写操作 ?*/
W1302(0x8c,NUM2BCD(Time[5]));
W1302(0x8a,NUM2BCD(Time[6]));
W1302(0x88,NUM2BCD(Time[4]));
W1302(0x86,NUM2BCD(Time[3]));
W1302(0x84,NUM2BCD(Time[2]));
W1302(0x82,NUM2BCD(Time[1]));
W1302(0x80,NUM2BCD(Time[0]));
W1302(0x8e,0x80); /*        控制命令  ,WP=1, 写保护 ?*/
}


void  Get1302(void)
{
Time[5] = BCD2NUM(uc_R1302(0x8d));
Time[6] = BCD2NUM(uc_R1302(0x8b));
Time[4] = BCD2NUM(uc_R1302(0x89));
Time[3] = BCD2NUM(uc_R1302(0x87));
Time[2] = BCD2NUM(uc_R1302(0x85));
Time[1] = BCD2NUM(uc_R1302(0x83));
Time[0] = BCD2NUM(uc_R1302(0x81));
}


/*  外部中断  0处理程序  */
void intersvr0(void)  interrupt 0 using 1
{
   unsigned int a;
   if(DM==1) a=60;
   else a=256;

  //    通过定时器  T0测出定时器  T1初设值
```

```
  D=D+(S-a)*2;
  Ti0=600+D;
  TH1=-1;TL1=-1;        //              让定时器  T1处理程序与外部中断处理      0程序错开
  ii=0;jj=0;S=0;
}

void timer1(void) interrupt 3 using 1
  {
    S++;
    TH1=-Ti0/256; TL1=-Ti0%256;     //              设置定时器  T1初设值

    if(DM==0)                                //                                显示模式一
    {
      if(ii<16)
      {
        P2=0xf1;P1=~nAsciiDot1[jj*2+v[ii]*16];          //                              显示的上半圆部分（正显）
        P2=0xf2;P1=~nAsciiDot1[1+jj*2+v[ii]*16];
      }
      else if(ii<32)
      {
        P2=0xf2;P1=~nAsciiDot2[14-jj*2+v[ii]*16];       //                              显示的下半圆部分（反显）
        P2=0xf1;P1=~nAsciiDot2[15-jj*2+v[ii]*16];
      }
     else
     {
       P2=0xf1;P1=0xff;
       P2=0xf2;P1=0xff;
     }
     //      显示外圆
     P2=0xf8;P1=0xff;
     P2=0xf4;P1=0xfe;
     P2=0x00;
      jj++; if(jj>7) {ii++; jj=0;}
      Delay100Us;
      P2=0xf1;P1=0xfe;
      P2=0xf2;P1=0xff;
      P2=0xf4;P1=0xfe;
      P2=0xf8;P1=0xff;
      P2=0x00;
}
```

```
    if(DM==1)                          //                          显示模式二
    {
      Tme0=(Time[0]+14)%60;
      Tme1=(Time[1]+14)%60;
      Tme2=((Time[2]%12)*5+Time[1]/12+14)%60;

P2=0xf1;P1=(0xfe<<3*(ii%5==4))&(0xff>>2*(ii==Tme0))&(0xfe<<5*(ii%15==14));
      P2=0xf2;P1=(0xff*(ii!=Tme0)*(ii!=Tme1))&(0xff>>4*(ii==Tme2));
     P2=0xf4;P1=(0xff*(ii!=Tme0)*(ii!=Tme1)*(ii!=Tme2));
     P2=0xf8;P1=(0xff*(ii!=Tme0)*(ii!=Tme1)*(ii!=Tme2));
     P2=0x00;
     Delay100Us;Delay100Us;Delay100Us;
     ii++;//if(ii>59) ET1=0;
     P2=0xf8;P1=0x3f;
     P2=0xf4;P1=0xff;
     P2=0xf2;P1=0xff;
     P2=0xf1;P1=0xfa;
     P2=0x00;
    }

   if(DM==2)                           //                          显示模式三
   {
     if(ii<44)
     {
       ii++;
       P2=0xf2;P1=0xff;
       P2=0xf1;P1=0xfe;
       P2=0x00;
     }
     else
     {
       if(jj<24)
       {
         P2=0xf1;P1=~HZ_12[TZ*24*(jj>11)+jj*2];
         P2=0xf2;P1=~HZ_12[TZ*24*(jj>11)+1+jj*2];
       P2=0x00;
   }

   if((jj>23)&&(jj<32))
   {
     P2=0xf1;P1=~nAsciiDot1[(jj-8)*2];
     P2=0xf2;P1=~nAsciiDot1[1+(jj-8)*2];
     P2=0x00;
```

```
}

if((jj>31)&&(jj<40))
{
  P2=0xf1;P1=~nAsciiDot1[(jj-8)*2+(Time[TZ+1]/10)*16];
  P2=0xf2;P1=~nAsciiDot1[1+(jj-8)*2+(Time[TZ+1]/10)*16];
  P2=0x00;
}

  if((jj>39)&&(jj<48))
  {
    P2=0xf1;P1=~nAsciiDot1[(jj-16)*2+(Time[TZ+1]%10)*16];
    P2=0xf2;P1=~nAsciiDot1[1+(jj-16)*2+(Time[TZ+1]%10)*16];
    P2=0x00;
  }

  if(jj<48) jj++;
  else

{

  P2=0xf2;P1=0xff;
      P2=0xf1;P1=0xfe;
      P2=0x00;
    }
      Delay100Us;
      P2=0xf1;P1=0xfe;
      P2=0xf2;P1=0xff;
      P2=0xf4;P1=0xfe;
      P2=0xf8;P1=0xff;

      P2=0x00;
    }
  }
}

void main(void)
{
  Ti0=0;
  TZ=0;
  ST=0;
  IT1=1;EX1=1;
```

```
TMOD=0x11;
PX1=1;
TH0=0;TL0=0;
TR0=1;ET0=1;

TH1=0;TL1=0;
TR1=1;ET1=1;

IT0=1;EX0=1;
EA=1;

//Set1302();

for(;;){
                          //4.22                        及 0.94 为校正值，由试验
确定
  Get1302();              //                    读时钟

  if(DM==0){              //                  数字显示模式时，将刷新显示缓冲区
    v[4]=Time[2]/10+3;
    v[5]=Time[2]%10+3;

    v[7]=Time[1]/10+3;
    v[8]=Time[1]%10+3;

    v[10]=Time[0]/10+3;
    v[11]=Time[0]%10+3;

    v[21]=Time[3]/10+3;
    v[20]=Time[3]%10+3;

    v[24]=Time[4]/10+3;
    v[23]=Time[4]%10+3;

    v[27]=Time[5]/10+3;
    v[26]=Time[5]%10+3;

    v[0]=0;
    v[1]=0;
    v[2]=0;
    v[3]=0;
    v[6]=2;
    v[9]=2;
    v[12]=0;
```

```
v[13]=0;
v[14]=0;
v[15]=0;
v[16]=0;
v[17]=0;
v[18]=0;
v[19]=0;
v[22]=1;
v[25]=1;
 v[28]=0;
 v[29]=0;
 v[30]=0;
 v[31]=0;
}

Delay100Us;Delay100Us;Delay100Us;          //                  延时
Delay100Us;Delay100Us;Delay100Us;
Delay100Us;Delay100Us;Delay100Us;
Delay100Us;Delay100Us;Delay100Us;

if(IrOK==1)                                //                  红外接收有效
{

 if(Im[2]==9)
{
  ST=!ST;                                  //                  进入或退出设置状态
     if(ST==1)                             //                  如在设置状态下
     {
       DM=2;                               //                  进入显示模式三
  }
     else
     {
       Set1302();                          //                  保存时间调整值
       DM=0;                               //                  回到显示模式一
     }
}

if(ST==0)                                  //                  非设置状态下，改变显示模式才有效
{
  if(Im[2]==67) DM=!DM;                    //                  时钟显示模式
}

if(ST==1)                                  //                  设置状态下对调整键处理
{
```

```
if(Im[2]==64)
    {
      if(TZ<4) TZ++; else TZ=0;                  //                          增量调整项
    }
if(Im[2]==68)
{
      if(TZ>0) TZ--; else TZ=4;                  //                          减量调整项
    }
if(Im[2]==21)
              {

                if(Time[TZ+1]<maxnum[TZ]) Time[TZ+1]++; else Time[TZ+1]=minnum[TZ];
Im[2]=0;IrOK=0; //        增大调整值
                Set1302();
              }

          if(Im[2]==7)
              {

                if(Time[TZ+1]>minnum[TZ]) Time[TZ+1]--; else Time[TZ+1]=maxnum[TZ];
Im[2]=0;IrOK=0; //        减小调整值
                Set1302();
              }
      }
  Im[2]=0;IrOK=0;                //                          退出遥控处理程序时，初始化遥控程序相关变量
            }
      }
}
```

附录 B 实物图

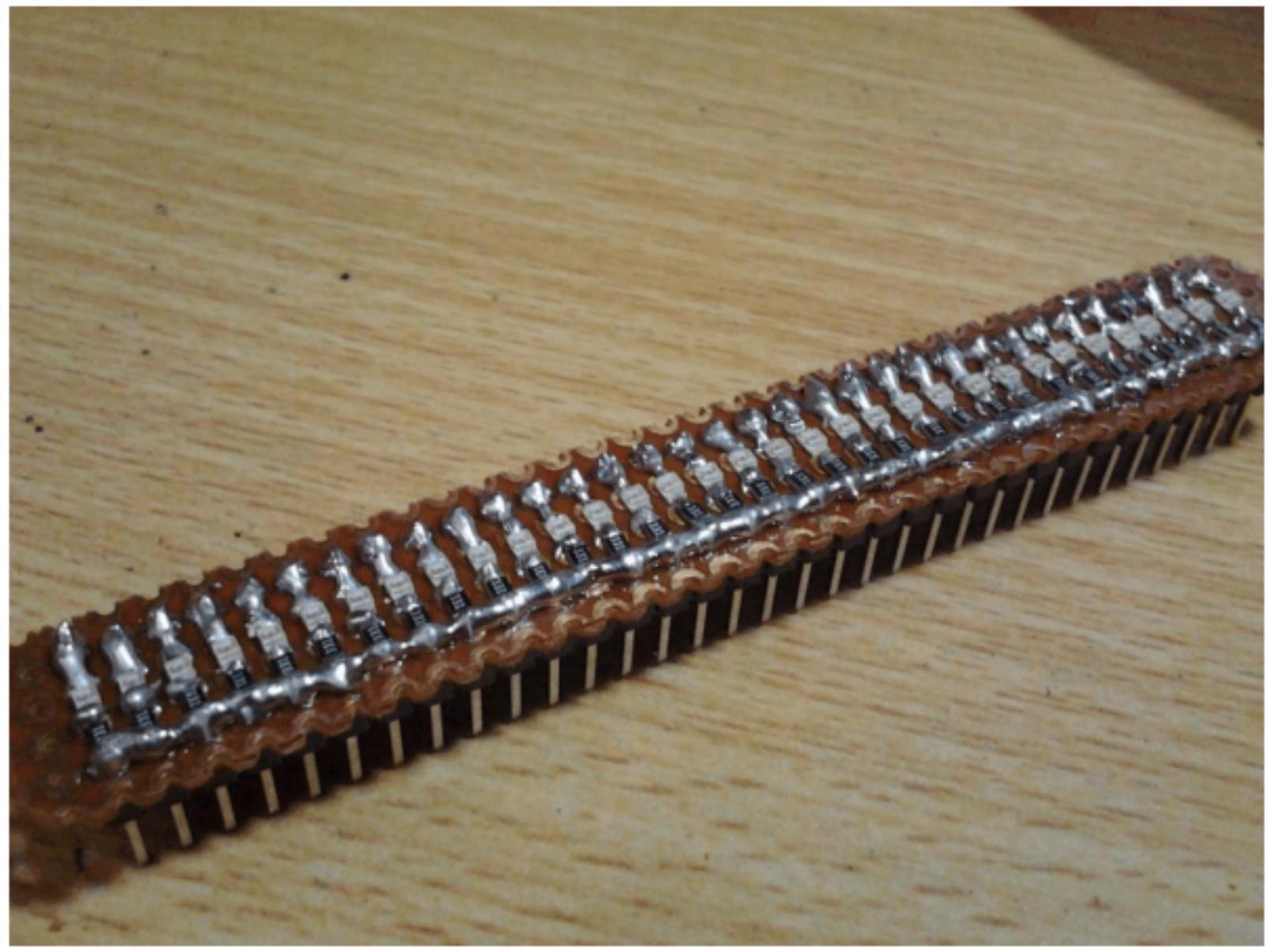


图5.1.1　　贴片LED灯

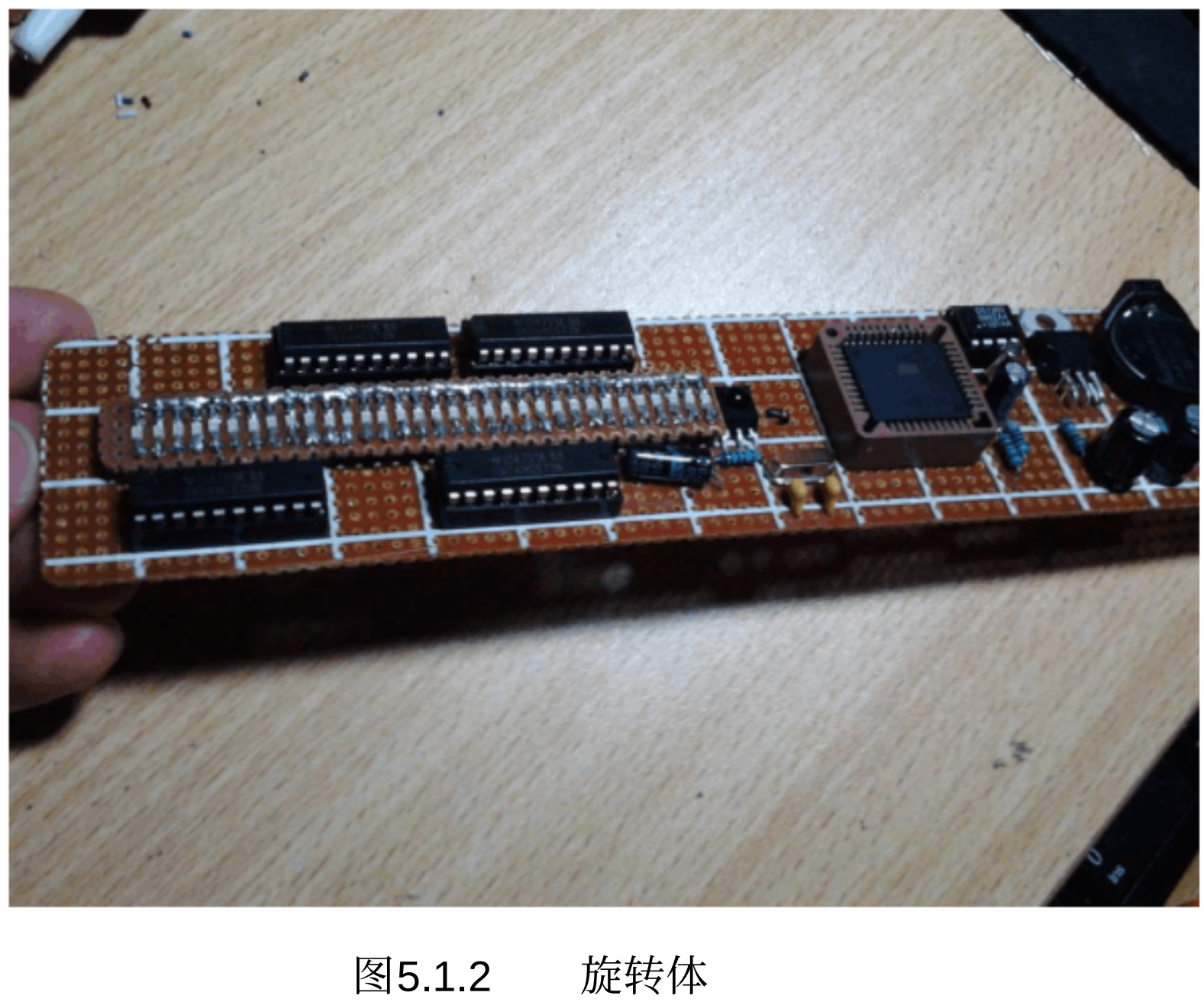


图5.1.2　　旋转体

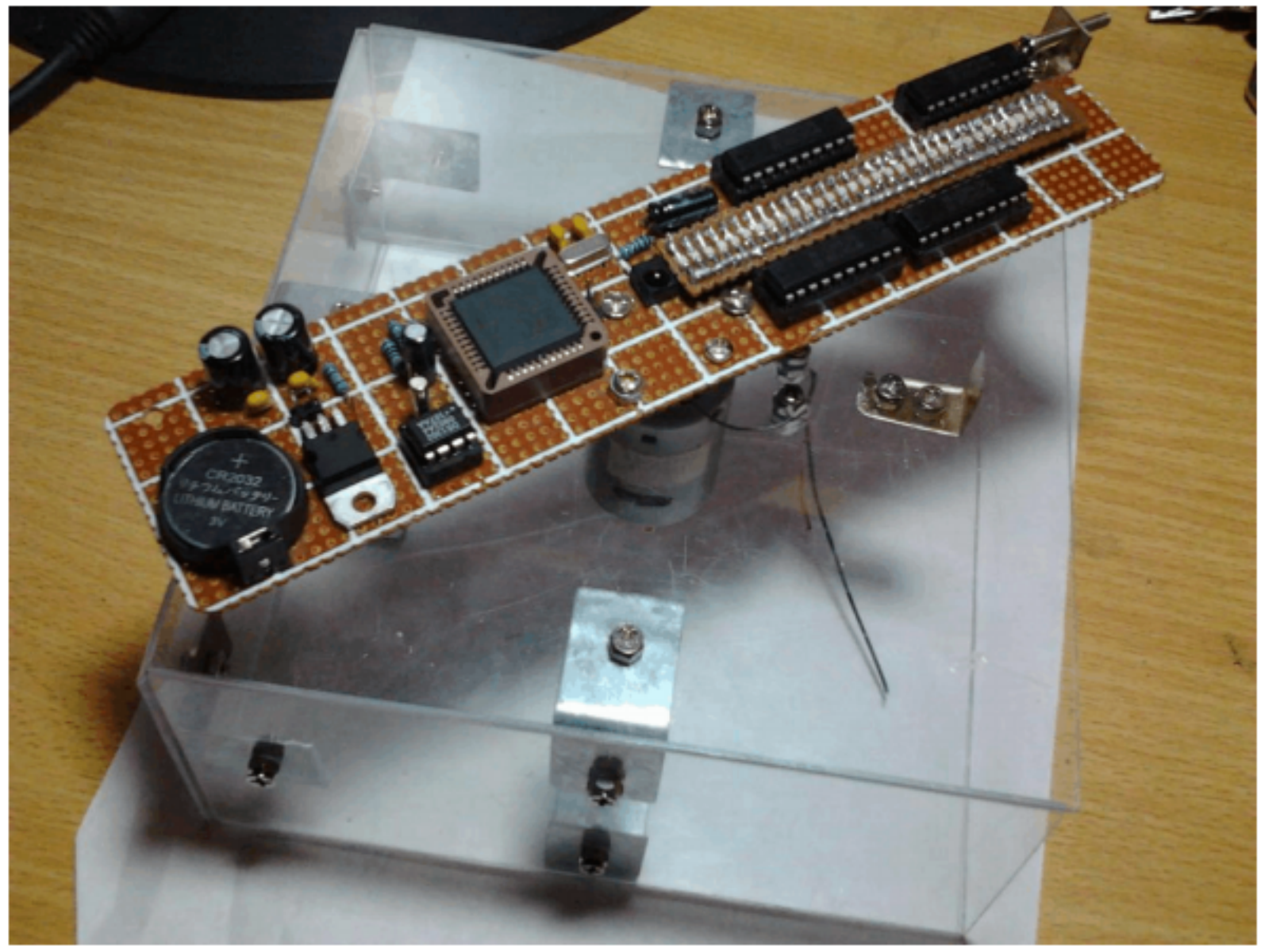


图5.1.3　　旋转LED时钟整体图