实验名称：_旋转灯实验_姓名：____严旭铧_____学号：___3220101731_____

专业：电气工程及其自动化
姓名：__严旭铧_____
学号：_3220101731_____

浙江大学实验报告

课程名称：_微机原理与应用实验___指导老师：__ 吴建德___ _成绩：_____

实验名称：__旋转灯实验_____实验类型：_STM32 实验_____

实验 旋转灯实验

一、 实验目的

1. 熟悉 Cortex-M 微机系统的设计过程；
2. 掌握 Cortex-M 微机系统的调试方法；

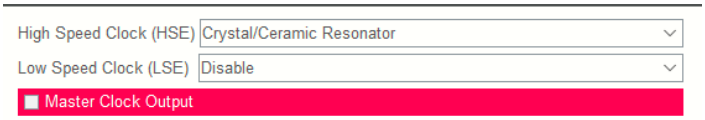
二、 实验内容

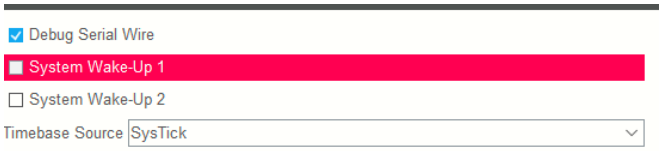
1. 正面显示时钟（3 个指针）。
2. 侧面显示姓名、学号并循环滚动。
3. 通过红外遥控器控制显示变化。（额外功能）

三、 实验过程

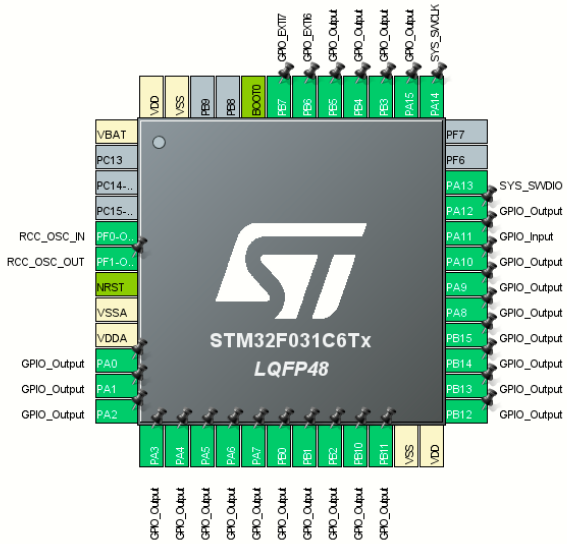
1. 配置好 Cubemx 文件。

(1) RCC 和 SYS 配置





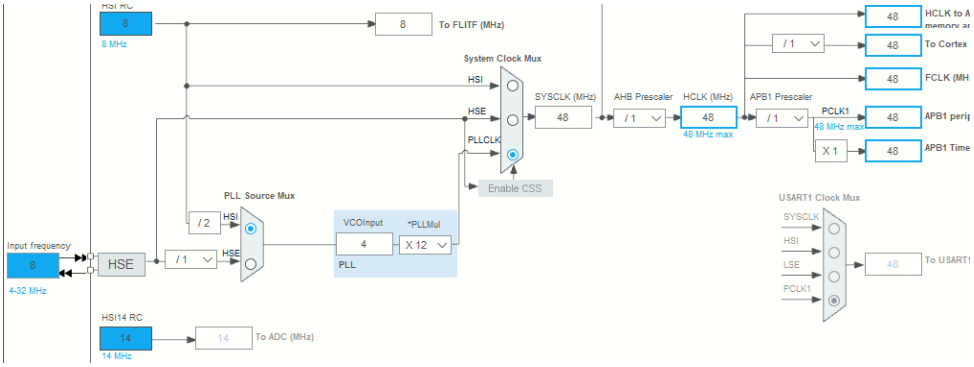
(2) GPIO 配置



其中 PB7 配置为 GPIO_EXIT7，来接受红外遥控的型号。经过测试，这个引脚需要设置为下降沿触发。

(3) CLK 配置

定时器	TIM1	TIM3	TIM14
预分频	179	0	47
Counter Period	65535	49999	65535



TIM3 采用全局中断。TIM1 将表盘分为 180 等分，TIM14 是红外遥控中断用的，TIM3 用于全局计时。

四、 模块思路与代码

1. 时间数值设定

(1) 思路

采用和 51 数字钟类似的思路，将时钟数值和显示分开。时间值存入用 ss,mm,hh 的整型变量，时间的显示值存入 ss_display 等整型变量

(2) 代码

这里分别定义时分秒变量为 hh, mm, ss，由于用 C 语言，写 n 进制就变得非常方便 ，if 语句即可。

```
195     }
196   LL_mDelay(1000);
197   ss += 1;
198   if (ss >= 60){
199       ss = 0;
200       mm += 1;
201     }
202   if (mm >= 60){
203       mm = 0;
204       hh += 1;
205     }
206   if (hh >= 12){
207       hh = 0;
208     }
```

2. 显示

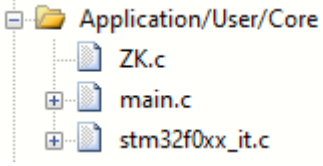
(1) 思路

i. 表盘显示

放在 main 函数中。表盘需要有时分秒针的旋转显示，同时做出刻度指示的效果。由于已经把时间 TIM1 分成 180 等分，可以用一个长度为 180 的数组 clock_disp 来存放显示的内容。这里，当时钟转起来的时候，整个表盘被分为 180 等分的小扇形组合，每个扇形上面有 32 个 LED 灯。通过写入不同的值，来控制哪个灯亮。

ii. 侧面旋转显示姓名学号

侧面 16 个 LED 灯发绿光，用 PPT 给的程序代码，用 GPIO 口控制输出。在主程序中同样写入数组，设为 words_display，数组长度为 192（经验讨论得出）。这里，由于要显示汉字和数组，需要添加字库，因此添加 ZK.c 文件于 Application/User/Core 目录下，用于存放学号、姓名和之后拓展的翻页内容的字符点阵。这里加入全局的数组 words_data(1,2,3,4…)，方便在主程序中调用。



iii. 翻页显示

这是需要红外中断控制的一个功能。设置了一个 page 变量，当检测到特定的红外遥控

信号时，可以控制 page±1，然后如果加到上、下限页可以到另一端，就是将翻页做成连续的，page0-1 就是 page4 这样。Page 就是一个状态的变量。这里翻页的显示做了表盘和侧面的显示。表盘由于是圆形，直接取字模会变形，效果不好，因此在字模网站上找到了图案的字模，虽然显示的是形变的，但是恰好有另一番风味，看上去还挺好看的。侧边 LED 用字模取了两句喜欢的歌词，翻页之后可以按顺序切换。

(2) 代码

i. 表盘显示

```
if(page== 0)
{
  for(i=0;i<180;i++)
  {
    clock_disp[179-i]=0x00+(words_data4[2*i]<<16)+(words_data4[2*i+1]<<8)+0x00;
  }
}
else
{
  sec_disp = sec*3;
  min_disp = (int)(min*3+sec/20.0);
  hour_disp = (int)(hour*15+min/20.0);
  for (k=0;k<180;k++)
  {
      if(k == hour_disp){clock_disp[k]=0xff000000;}
      if(k == min_disp){clock_disp[k]=0xffff0000;}
      if(k == sec_disp){clock_disp[k]=0xffffff00;}
      if((k!=sec_disp)&&(k!=min_disp)&&(k!=hour_disp)){clock_disp[k]=0x00000000;}
      if(k%15 == 0){clock_disp[k]=clock_disp[k]|0x00000007;}
      if(k%45 == 0){clock_disp[k]=clock_disp[k]|0x0000003f;}
      if(k == 0){clock_disp[k]=clock_disp[k]|0x0000000f;}
      if(k%3 == 0){clock_disp[k]=clock_disp[k]|0x00000003;}

      if(k==0|k==175|k==1|k==3|k==5)  {clock_disp[k]=clock_disp[k]|0x00000084;}
      if(k==179|k==176) {clock_disp[k]=clock_disp[k]|0x00000048;}
      if(k==177|k==178){clock_disp[k]=clock_disp[k]|0x00000030;}
      if(k==2|k==4){clock_disp[k]=clock_disp[k]|0x000000fc;}

      if(k==132|k==134|k==136|k==141){clock_disp[k]=clock_disp[k]|0x00000084;}
      if(k==133){clock_disp[k]=clock_disp[k]|0x000000fc;}
      if(k==137|k==140){clock_disp[k]=clock_disp[k]|0x00000048;}
      if(k==138|k==139){clock_disp[k]=clock_disp[k]|0x00000030;}
      if(k==135){clock_disp[k]=clock_disp[k]|0x00000000;}

      if(k==90|k==86){clock_disp[k]=clock_disp[k]|0x00000010;}     //????6???
      if(k==85|k==91){clock_disp[k]=clock_disp[k]|0x0000000c;}
      if(k==87|k==89){clock_disp[k]=clock_disp[k]|0x00000060;}
      if(k==88){clock_disp[k]=clock_disp[k]|0x00000080;}
      if(k==92){clock_disp[k]=clock_disp[k]|0x00000000;}
      if(k==93|k==95){clock_disp[k]=clock_disp[k]|0x00000084;}
```

这里用了非常多的 if。为了使表针转动，时分秒分别对应到_disp 变量，由于是 180 格，秒针一圈 60，分针一圈 60，但是为了显示确切，作了秒针的修正；时针同理，一圈是 12 大格，因此数值乘 15。表盘上剩余的刻度部分，作了小格和大格的划分，长度有所不同，这个只要对 k 取值枚举即可。这里小端模式下，最低位的对应最外围的 LED 灯，置 1 亮灯。除此之外，表盘上 3、6、9、12 处用了罗马数字显示，表盘上还手搓了一点

小的装饰。

值得一提的是，之后做到的红外调整时间的功能，在表盘显示的框架下，只要直接改 hour,min,sec 等变量的值即可做到表针转动相应格数，非常方便。

ii. 侧面显示

```
for(i=0;i<192;i++)
{
  words_disp[191-i]=(words_data[2*i]<<8)+(words_data[2*i+1]);
}
for(i=0;i<180;i++)
{
  clock_disp[i]=0;
}
```

```
]const unsigned char words_data[408] = {

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0(
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0(

/*-- ID:7,□□?□□?:"1",ASCII□□□□??:A3B1,??□□|□□?:?□□x??=8x16,?-2?:?□□W=8 ??
0x00,0x00,0x10,0x04,0x1F,0xFC,0x3F,0xFC,0x00,0x04,0x00,0x00,
/*-- ID:C,□□?□□?:"3",ASCII□□□□??:A3B3,??□□|□□?:?□□x??=8x16,?-2?:?□□W=8 ??
0x00,0x00,0x08,0x10,0x18,0x18,0x30,0x0C,0x21,0x04,0x21,0x04,0x33,0x8C,0x1
/*-- ID:B,□□?□□?:"7",ASCII□□□□??:A3B7,??□□|□□?:?□□x??=8x16,?-2?:?□□W=8 ??
0x00,0x00,0x38,0x00,0x30,0x00,0x30,0x3C,0x30,0xFC,0x33,0xC0,0x37,0x00,0x3(
```

```
void DisplayLED(void)
  {
    //gLEDs1,gLEDs2,gLEDs3
    uint32_t temp,i,j;

    temp=gLEDs2&0xff;
    LL_GPIO_ResetOutputPin(GPIOA,temp);
    temp=temp^0xff;
    LL_GPIO_SetOutputPin(GPIOA,temp);

    temp=((gLEDs2>>8)&0x7)|((gLEDs2>>1)&0x7c00);

    LL_GPIO_ResetOutputPin(GPIOB,temp);
    temp=temp^0x7c07;
    LL_GPIO_SetOutputPin(GPIOB,temp);
    //595
    LL_GPIO_ResetOutputPin (GPIOB,LL_GPIO_PIN_3);       //CE
    LL_GPIO_ResetOutputPin (GPIOB,LL_GPIO_PIN_4);       //RCK
    LL_GPIO_ResetOutputPin (GPIOB,LL_GPIO_PIN_5);       //LE
    LL_GPIO_ResetOutputPin (GPIOA,LL_GPIO_PIN_15);      //DATA
    //  temp=(gLEDs1|(gLEDs3<<16));
```

侧面显示的驱动就是老师给的程序，在主程序中把字库里的数据导入即可。

iii. 翻页显示

显示的内容由上面一串 if 和字库文件给出，page 在红外遥控部分改变。

3. 红外模块

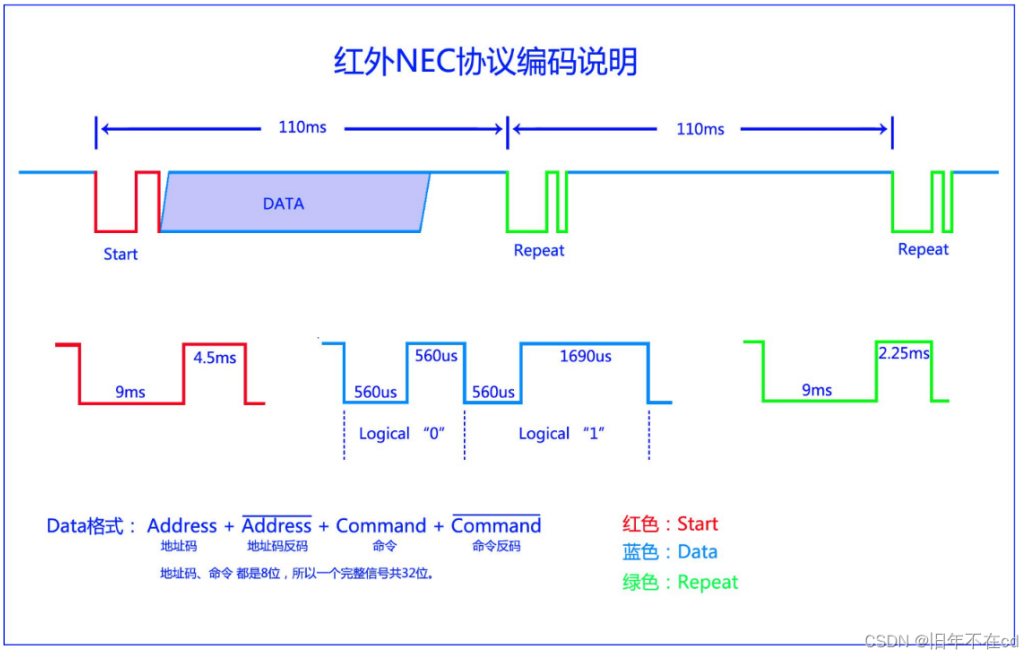
(1) 思路

i. 转速控制

硬件上，旋转灯有两个红外的模块，其中有一黑一白一对红外对管，上方黑色的是红外接收管，下方透明的是红外发射管。这一对对管用于硬件控制转速与定时器时钟匹配，以达到稳定显示的效果。

ii. 红外遥控

扇叶部分有一个红外遥控信号接收器，这个接收器可以接受遥控器发出的型号。本次实验中的遥控器型号为 CarMp3，采用 NEC 红外编码协议，该部分红外驱动代码主要来自网络资料搜索。



我们主要关心前导码和数据码部分，前导码采集可以识别是否接受到了遥控信号，数据码可以辨别是哪个信号，按下了哪个按键。根据资料显示，有以下对应关系：

```
#define IRR_CH_Neg    0x00FFA25D
#define IRR_CH        0x00FF629D
#define IRR_1         0x00FF30CF
#define IRR_2         0x00FF18E7
#define IRR_3         0x00FF7A85
#define IRR_4         0x00FF10EF
#define IRR_5         0x00FF38C7
#define IRR_6         0x00FF5AA5
#define IRR_7         0x00FF42BD
#define IRR_8         0x00FF4AB5
#define IRR_9         0x00FF52AD
```

另有：NEXT——0x00FF0002
PREA——0x00FF0022
PALY——0x00FF00C2
VOL+——0x00FF00E0
VOL-——0x00FF00A8

(2) 代码

实验名称：_旋转灯实验_姓名：____严旭铧_____学号：___3220101731_____

---

i. 转速控制

```
void EXTI4_15_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI4_15_IRQn 0 */
static uint16_t LastTime;
  uint16_t temp,temp1;

  /* USER CODE END EXTI4_15_IRQn 0 */
  if (LL_EXTI_IsActiveFlag_0_31(LL_EXTI_LINE_6) != RESET)
  {
    LL_EXTI_ClearFlag_0_31(LL_EXTI_LINE_6);
    /* USER CODE BEGIN LL_EXTI_LINE_6 */

    temp = TIM1 -> CNT;                 // ??????1?1??????temp
    temp1 = temp - LastTime;
    TIM3 -> ARR = temp1;                // ??temp1??????????3
    LastTime = temp;
    /* USER CODE END LL_EXTI_LINE_6 */
  }
```

上面的 if 做了将 TIM1 的值赋给 TIM3 的操作。我的电脑上 keil 注释无法调成 GB2312，重开之后注释就会乱码，很讨厌。

ii. 红外遥控

代码的实现分为两个部分，一个部分是采集函数 IR_NEC_Read_Decode，另一个部分是根据采集到的红外信号判读数据并做出响应。这里设计成 NEXT 键增加秒数，PREA 键增加分钟数，PLAY 键增加小时数，VOL+键增加页码数，VOL-键减少页码数。代码编写时均考虑了进位。

```
/* USER CODE BEGIN LL_EXTI_LINE_7 */
IR_NEC_Read_Decode();

 if( IR_READ_DATA2[2] == 0x02)    // □□□□NEXT□□□□
 {
   sec+=1;
    if (sec >= 60){
      sec = 0;
      min += 1;
   }
  if (min >= 60){
      min = 0;
      hour += 1;
   }
  if (hour >= 12){
      hour = 0;
   }

     IR_READ_DATA2[2]=0;
 }
 else if( IR_READ_DATA2[2] == 0x22)    // □□□□PREA□□□□
```

```
void IR_NEC_Read_Decode(void)  // (*val)(void)
{
        TIM14 -> ARR = 60000;
    if (IR_STATE == 0) //???????
    {//??0
        IR_STATE = 1;
        LL_TIM_EnableCounter(TIM14);
            //LL_TIM_EnableIT_UPDATE(TIM14);
        LL_TIM_SetCounter(TIM14,0);   //????
    }
    else if (IR_STATE == 1) //???????????????
    {
        IR_READ_TIME = TIM14->CNT;
        if ((IR_READ_TIME > (13500  - 500)) && (IR_READ_TIME < (13500 +
500))) // 13.5ms?? ???
        {//??1.1
            IR_STATE = 2;
            LL_TIM_SetCounter(TIM14,0);
            IR_READ_OK = 0;
            for (int i = 0; i < 4; i++)
                IR_READ_DATA[i] = 0;
            IR_READ_BYTE = 0;  //??
                    IR_READ_DECODE_i = 0;
            IR_READ_DECODE_j = 0;
        }

        else if ((IR_READ_TIME > (11250      -  1000)) && (IR_READ_TIME <
(11250 + 1000))) // 11.25ms?? ???
        {//??1.2
            IR_STATE = 0;
            IR_READ_OK = 2;   //???
            //for (int i = 0; i < 4; i++)
            //    IR_READ_DATA[i] = IR_READ_DATA2[i];
            LL_TIM_SetCounter(TIM14,0);
        }

        else //??????? ??
        {
            IR_STATE = 0;
                    //LL_TIM_SetCounter(TIM14,0);
        }
    }
    else if (IR_STATE == 2) //????
    {//??2
```

实验名称：_旋转灯实验_ 姓名：____严旭铧_____学号：___3220101731_____

五、 呈现结果

1. 一共设计了 4 页，page 0-3。
2. 通电后，首先进入 page0，表盘显示图案，侧面滚动显示姓名和学号
3. 按 VOL+键翻到 page1，显示时间表盘，侧面依旧滚动显示姓名学号
4. Page1-3 时均显示有时间表盘，此时按 NEXT,PREA,PLAY 键分别显示
5. 继续翻页到 page2，表盘不变，侧面滚动显示“我迎着风向前狂奔”
6. 继续翻页到 page3，侧面滚动显示“这速度能不能抛开忧伤”
7. 继续翻页回到 page0.

六、 硬件部分调试心得

1. 根据我们在实验室的尝试，发现如果不安装亚克力支架，那么红外遥控在静态调试的时候虽然正常，但是一旦转起来之后，遥控不灵敏，并且会干扰到同步用的红外管，导致表盘整个旋转或晃动，显示效果不好。装上支架后会好一些，但是仍很有可能受到干扰，尤其是遥控器在转子高度进行遥控时。
2. 红外对管焊接时将管脚尽量留长，让两根管子头对头，距离小一点，这样可以减少干扰。尤其是上方的接收管，尽可能伸到亚克力板下，然后红外遥控在亚克力板上方进行。
3. 接收管和遥控接收器可以在侧面包上电工胶带，减少干扰。遥控接收器可以往上翻。
4. 本次实验亚克力板有点问题，有一部分板子开的孔是不对的，可以拿标准板对照，然后用焊枪戳孔即可。
5. **复位：**
   在最开始的时候，配置 cubemx 时 debug 选项忘记勾选，然后 IO 口配置的时候 PB13 配置错了，导致烧录了一遍程序后，芯片锁死，无法再写入。此时需要进行手动复位。从 RST 引脚和电路图上 D1 正极（好像是这个，记不清了）引出两根导线短接，然后在 keil 钟点击加载程序的时候，马上松手断开，多试几次，等到下面进度条能读了说明复位成功。此时把配置改回正确的即可。当然也可能会有其他类型的错误，这里我不懂了。
6. 本次给 C 口供电的供电线好像只有 A to C 的线可以，C-C 的线就不能上电。

七、 实验体会、思考和建议

1. 集思广益的一次实验，大家一起调试分享思路和技巧。
2. 实验的理论教学感觉太少，前置铺垫太少，就像是只交了打螺丝就要造火箭，只有极个别大佬能完全靠自己手搓出来。
3. Cubemx 配置和硬件焊接的时候注意事项最好能够在 PPT 里呈现，不然把硬件搞坏了会很麻烦。
4. 旋转灯的效果还是很不错的，真正做成之后还是很有成就感。