

河北传媒学院

实验教学报告书

学 院 信息技术与文化管理学院

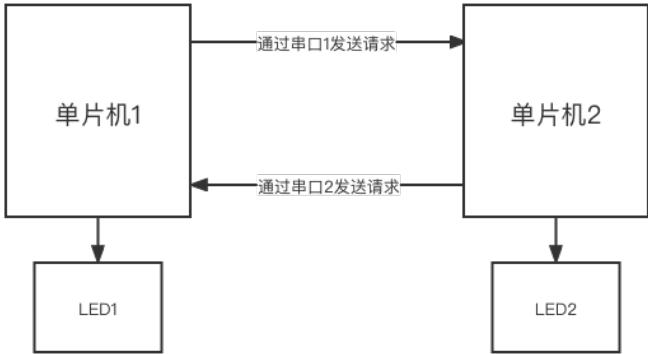
专业班级 2021 级物联网工程专接本 1 班

学生学号 210809059066

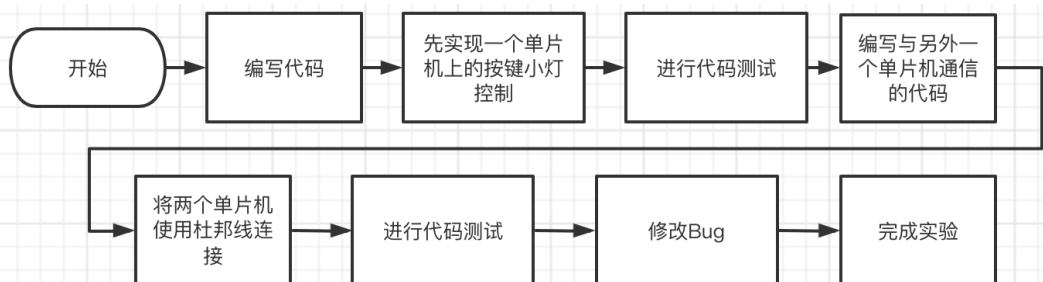
学生姓名 苑紫清

实践课程 微型计算机原理及应用

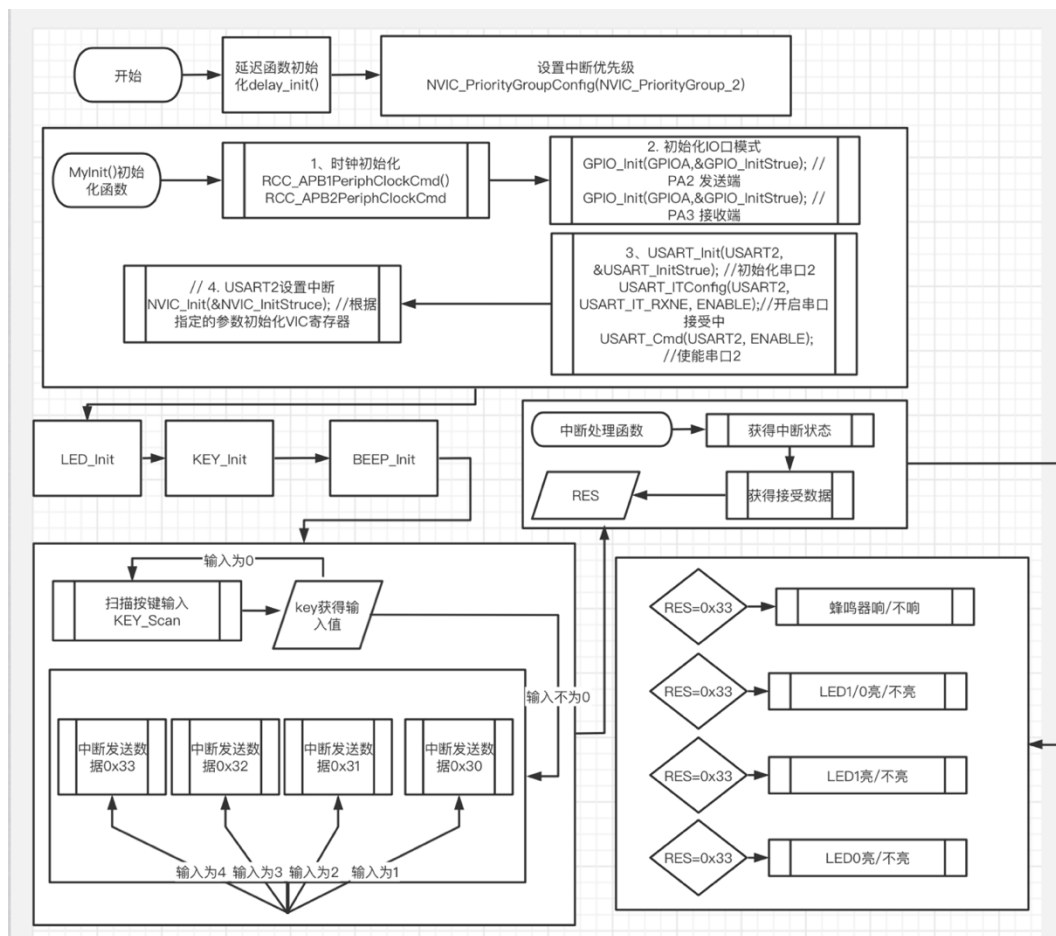
实 验 报 告

实践项目	基于 STM32 开发板的综合实训	指导教师	刘欣欣
实践时间	从 10 月 11 到 12 月 22 日共 64 学时	实践地点	兴新 429
实践目的： <ul style="list-style-type: none">一、实现双机通信二、数字时钟三、基于 STM32 的小车寻轨四、提升自己的动手能力			
实践内容： <ul style="list-style-type: none">一、实现两组 STM32 开发版 1 按按键一（Key1）；开发版 2 的 LED0 闪烁，开发版 2 按下按键 2，单片机 1 的 LED1 闪烁。二、通过 STM32 开发板，实现串口打印时钟，并在串口助手中用用定时器中断进行延时三、通过 STM32 开发板、红外对管、直流电机控制小车寻轨。			
实践过程： <p>一、双击通信</p> <p>（一）实践设备：两组 STM32 战舰版开发板，串口通信线两组，一台 PC 机。</p> <p>（二）实验思路：想要实现双方的串口通信，就要采用两组串口进行通信沟通，于是我们准备了两条串口通信线，以便于两个单片机进行沟通，通过对战舰版官方参考手册，我们发现单片机复用了 PA9 和 PA10 作为串口，我们只需要对 PA9，PA10 和串口进行使能，然后调用相关的串口库函数进行数据的发送，然后对收到的数据进行逻辑判断，使 LED 灯亮即可。</p> <p>（三）结构框图：</p> 			

(四) 实验流程



(五) 流程图



(六) 实验主要程序

```

#include "stm32f10x.h"
#include "sys.h"
#include "delay.h"
// LED 相关
#define LED0 PBout(5) // PB5
#define LED1 PEout(5) // PE5
void LED_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

```

```

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_GPIOE, ENABLE);
//使能 PB, PE 端口时钟

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;           //LED0-->PB. 5 端口配置
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;    //推挽输出
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;   //IO 口速度为 50MHz
GPIO_Init(GPIOB, &GPIO_InitStructure);             //根据设定参数初始化 GPIOB. 5
GPIO_SetBits(GPIOB, GPIO_Pin_5);                     //PB. 5 输出高

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;           //LED1-->PE. 5 端口配置, 推挽输出
GPIO_Init(GPIOE, &GPIO_InitStructure);             //推挽输出, IO 口速度为 50MHz
GPIO_SetBits(GPIOE, GPIO_Pin_5);                     //PE. 5 输出高
}

// 按键相关
#define KEY0  GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_4) //读取按键 0
#define KEY1  GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_3) //读取按键 1
#define KEY2  GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_2) //读取按键 2
#define WK_UP  GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0) //读取按键 3(WK_UP)
#define KEY0_PRES  1  //KEY0 按下
#define KEY1_PRES  2  //KEY1 按下
#define KEY2_PRES  3  //KEY2 按下
#define WKUP_PRES  4  //KEY_UP 按下(即 WK_UP/KEY_UP)
void KEY_Init(void) //IO 初始化
{
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOE, ENABLE);
    //使能 PORTA, PORTE 时钟

    GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4; //KEY0-KEY2
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; //设置成上拉输入
    GPIO_Init(GPIOE, &GPIO_InitStructure); //初始化 GPIOE2, 3, 4

    //初始化 WK_UP-->GPIOA. 0 下拉输入
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //PA0 设置成输入, 默认下拉
    GPIO_Init(GPIOA, &GPIO_InitStructure); //初始化 GPIOA. 0
}

```

```

// 按键处理函数
u8 KEY_Scan(u8 mode)
{
    static u8 key_up=1;//按键按松开标志
    if(mode)key_up=1;  //支持连接
    if(key_up&&(KEY0==0||KEY1==0||KEY2==0||WK_UP==1))
    {
        delay_ms(10);//去抖动
        key_up=0;
        if(KEY0==0)return KEY0_PRES;
        else if(KEY1==0)return KEY1_PRES;
        else if(KEY2==0)return KEY2_PRES;
        else if(WK_UP==1)return WKUP_PRES;
    }else if(KEY0==1&&KEY1==1&&KEY2==1&&WK_UP==0)key_up=1;
    return 0;// 无按键按下
}

// 蜂鸣器相关
#define BEEP PBout(8)    // BEEP, 蜂鸣器接口
void BEEP_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;//结构体成员
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);    //使能 GPIOB 端口时钟
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;                //BEEP-->PB. 8 端口配置
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;          //推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;         //速度为 50MHz
    GPIO_Init(GPIOB, &GPIO_InitStructure);    //根据参数初始化 GPIOB. 8
    GPIO_ResetBits(GPIOB, GPIO_Pin_8);        //输出 0, 关闭蜂鸣器输出
}

// 对 PA2 和 PA3 进行初始化
void My_InitIO(u32 bound){
    // 1. 时钟初始化
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    // 2. 初始化 IO 口模式
    GPIO_InitTypeDef GPIO_InitStrue;
    GPIO_InitStrue.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStrue.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStrue.GPIO_Mode = GPIO_Mode_AF_PP; // 发送端是复用推挽
    GPIO_Init(GPIOA, &GPIO_InitStrue); // PA2 发送端

```

```

GPIO_InitStruc.GPIO_Pin = GPIO_Pin_3;
GPIO_InitStruc.GPIO_Mode = GPIO_Mode_IPU; // 接收端是浮空端
GPIO_Init(GPIOA, &GPIO_InitStruc); // PA3 接收端

// 3. 串口 USART2 初始化
USART_InitTypeDef USART_InitStruc;
USART_InitStruc.USART_BaudRate = bound; // 串口波特率
USART_InitStruc.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
USART_InitStruc.USART_Mode = USART_Mode_Rx | USART_Mode_Tx; // 收发模式
USART_InitStruc.USART_Parity = USART_Parity_No; // 无奇偶位校验
USART_InitStruc.USART_StopBits = USART_StopBits_1; // 一个停止位
USART_InitStruc.USART_WordLength = USART_WordLength_8b; // 字长为 8 位数据
格

USART_Init(USART2, &USART_InitStruc); //初始化串口 2
USART_ITConfig(USART2, USART_IT_RXNE, ENABLE); //开启串口接受中
USART_Cmd(USART2, ENABLE); //使能串口 2

// 4. USART2 设置中断
NVIC_InitTypeDef NVIC_InitStruc;
NVIC_InitStruc.NVIC_IRQChannel = USART2_IRQn;
NVIC_InitStruc.NVIC_IRQChannelPreemptionPriority=2 ;//抢占优先级 3
NVIC_InitStruc.NVIC_IRQChannelSubPriority = 2; //子优先级 3
NVIC_InitStruc.NVIC_IRQChannelCmd = ENABLE; //IRQ 通道使能
NVIC_Init(&NVIC_InitStruc); //根据指定的参数初始化 VIC 寄存器
}

void USART2_IRQHandler(void) {
    u8 Res;
    if(USART_GetITStatus(USART2, USART_IT_RXNE) != RESET) //接收中断(接收到的
    数据必须是 0x0d 0x0a 结尾)
    {
        Res =USART_ReceiveData(USART2); //读取接收到的数据
//        USART_SendData(USART2, Res);
//        while(USART_GetFlagStatus(USART2, USART_FLAG_TC) !=SET) {} //等待发送结束
        if (Res==0x33)
        {
            BEEP=!BEEP;
        }
        if (Res==0x32)
        {
            LED0=!LED0;
            LED1=!LED1;
        }
    }
}

```

```

    }
    if (Res==0x31)
    {
        LED1=!LED1;
    }
    if (Res==0x30)
    {
        LED0=!LED0;
    }
}
}

vu8 key=0;
int main(void)
{
    delay_init();           //延时函数初始化
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //设置 NVIC 中断分组 2:2 位
    抢占优先级, 2 位响应优先级
    My_InitIO(115200);      //串口初始化为 115200
    LED_Init();             //LED 端口初始化
    KEY_Init();             //初始化与按键连接的硬件接口
    BEEP_Init();
    while(1)
    {
        key=KEY_Scan(0); //得到键值
        if(key)
        {
            switch(key)
            {
                case WKUP_PRES:
                    delay_ms(100); //去抖动
                    USART_SendData(USART2, 0x33);
                    while(USART_GetFlagStatus(USART2, USART_FLAG_TC) != SET) {} //等待发送结束
                    break;
                case KEY2_PRES:
                    delay_ms(100); //去抖动
                    USART_SendData(USART2, 0x32);
                    while(USART_GetFlagStatus(USART2, USART_FLAG_TC) != SET) {} //等待发送结束
                    break;
                case KEY1_PRES:
                    delay_ms(100); //去抖动
                    USART_SendData(USART2, 0x31);
                    while(USART_GetFlagStatus(USART2, USART_FLAG_TC) != SET) {} //等待发送结束
                    break;
                case KEY0_PRES:

```

```

        delay_ms(100); //去抖动
        USART_SendData(USART2, 0x30);
        while(USART_GetFlagStatus(USART2, USART_FLAG_TC) != SET) {} //等待发送结束
        break;
    }
    }else delay_ms(10);
}
}

```

（七）效果

单片机 1 按 key1 键，单片机 2 上的 LED0 闪烁；单片机 2 按 key2 键，单片机 1 上的 LED1 闪烁。

（八）遇到的问题及解决方案

问题一：不知道杜邦线如何连接，连接到哪一个接口。

解决方案：通过查阅 STM32 文档 WarShip STM32F1_V3.4_SCH。

问题二：写完了程序写入后没有效果。

解决方案：将工程程序一点点的剥离，然后分别测试每一个模块是否能够正常运行，最后找到出问题的模块。

问题三：板子不够用

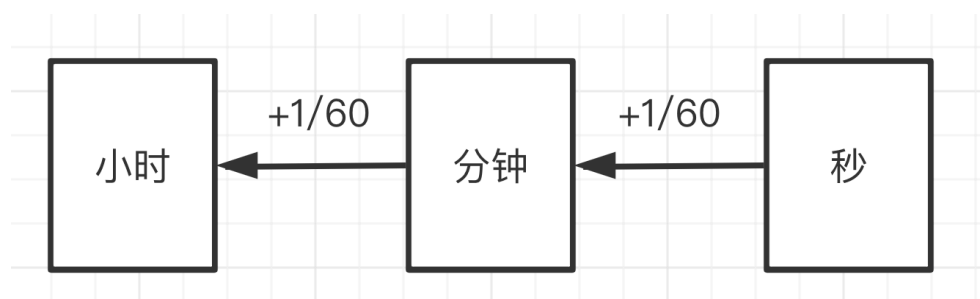
解决方案：先让组里面写的快的同学先进行测试，写得快的后用板子。

二、数字时钟

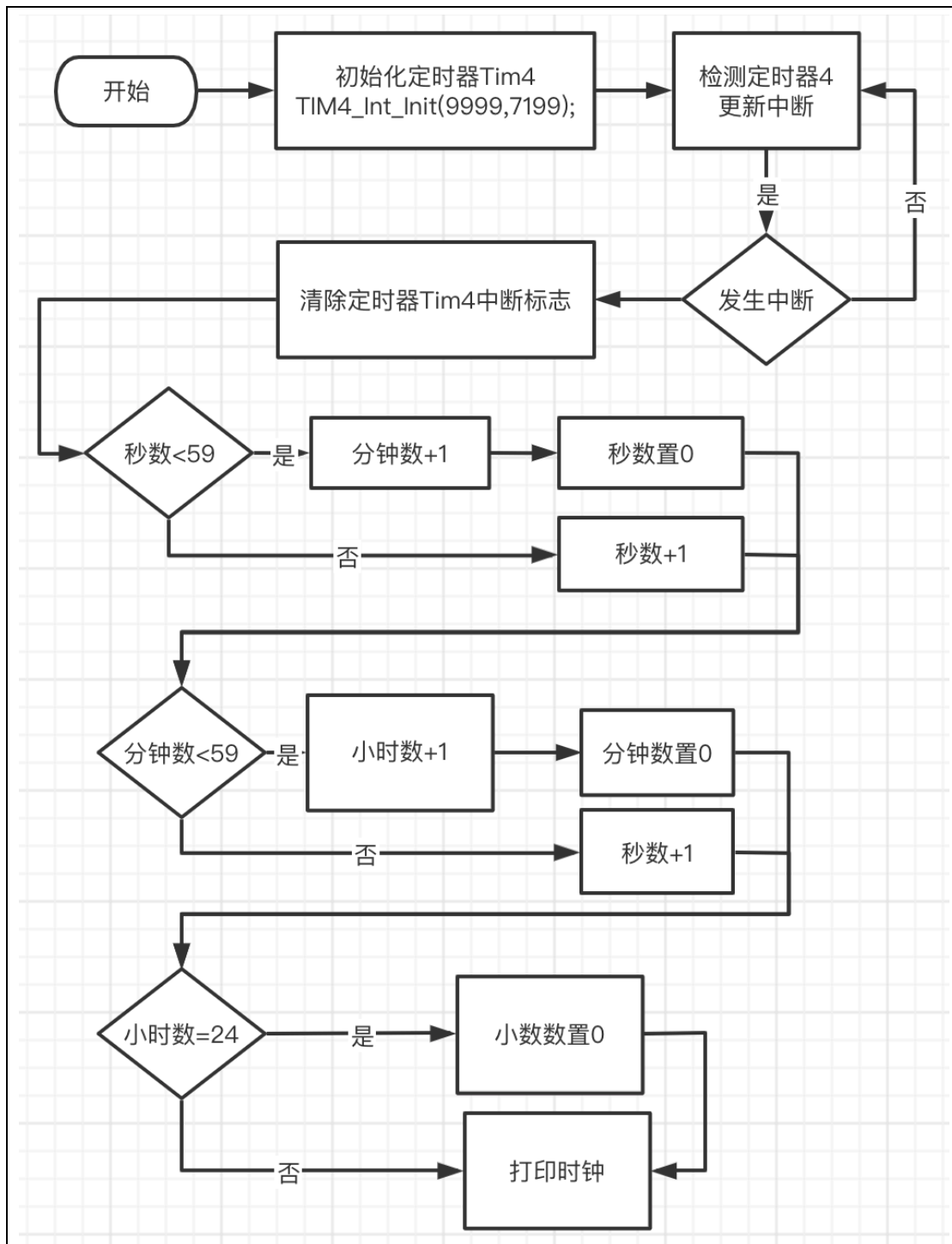
（一）实践设备：STM32 开发板、串口助手、PC 机、数据线

（二）实验思路：定义三个变量，分别代表秒、分钟和小时，三个按照时间的客观规律进行条件判断，并进行增加和置 0 操作，最后在中断处理函数中对时间字符串进行打印。

（三）结构框图



（四）流程图



(五) 实验主要程序

```

void TIM4_IRQHandler(void) {
    if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET) {
        TIM_ClearITPendingBit(TIM4, TIM_IT_Update );
        if(s<=59) {m++;s=0;}
        else s++;
        if(m<=59) {h++;m=0;}
    }
}
  
```

```

        if(h==24) {h=0;}

        printf("2021 年 11 月 17 日%d 点%d 分%d 秒\r\n", h, m, s);

    }

}

```

（六）效果

在串口助手上每隔一秒打印一次数据显示一个小的时钟在不断增加数字
每 60 自动从低位向高位加一

（七）遇到的问题及解决方案

问题：程序不能延时

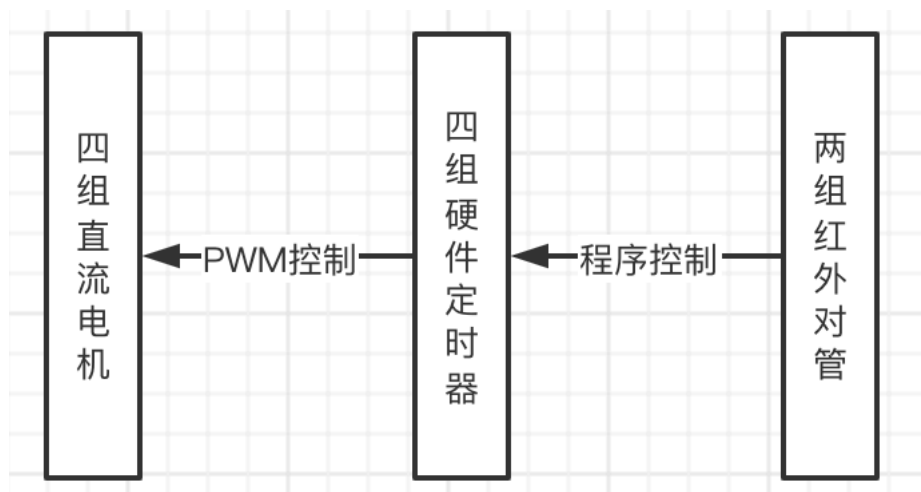
解决方案：放在中断程序内问题在中断程序中使用 PRINTF 会有警告解决
加上#include "usart.h"头文件

三、小车寻轨

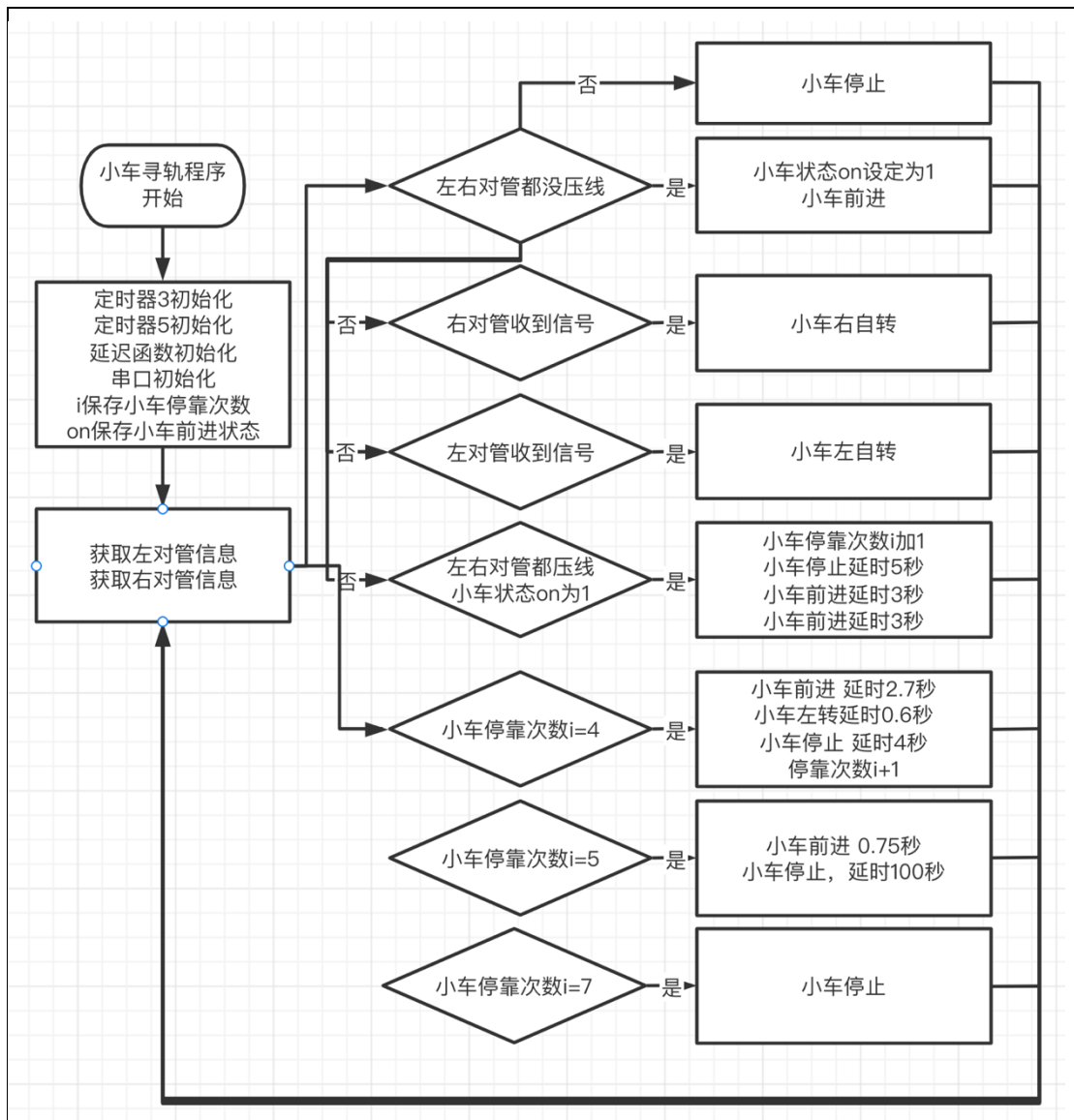
（一）实验设备：

（二）实验思路：将 STM32 开发版与四直流电机进行连接，使用 STM32 定时器控制直流电机的 PWM 实现小车的前进、后退左传、右转、左自转和右自转，安装两个红外对管，分别安装在左和右，当左对管检测到黑线，小车就向左拐，当右对管检测到黑线，小车就向右拐，再根据具体的赛道制定小车行进策略，最后使用 Keil5 和 C 语言开发相关逻辑代码。

（三）结构框图



（四）流程图



(五) 实验主要程序

```

int main(void)
{
    u8 left;
    u8 right;
    char on=0;

    int i=0;

    delay_init();          //延时函数初始化

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);  //设置 NVIC 中
断分组 2:2 位抢占优先级, 2 位响应优先级

    uart_init(115200);     //串口初始化为 115200

```

```
LED_Init();           //LED 端口初始化
TIM3_PWM_Init(699, 0); //不分频。PWM 频率=72000000/900=80Khz
TIM5_PWM_Init(699, 0);

while(1) {
    right = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12);
    left = GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_0);
    if(i==4) { //停四次
        car_go();
        //delay_ms(500);
        delay_ms(2700);
        car_left();
        //delay_ms(1800);
        delay_ms(600);
        car_stop();
        delay_ms(4000);
        i++;
    }
    else if(i==6) { //停五次
        car_go();
        delay_ms(75);
        car_stop();
        delay_ms(10000);
        car_stop();
        delay_ms(10000);
        car_stop();
        delay_ms(10000);
        car_stop();
        delay_ms(10000);
        car_stop();
    }
}
```

```
        delay_ms(10000);
        car_stop();
        delay_ms(10000);
        car_stop();
    }

    else if(i==7) { //停五次
        car_stop();
    }

    if(!left&&!right) { //都收到信号 反0
        on=1;
        car_go();

    }

    }else if(right==0) { //右收到信号
        car_right();
    }

    else if(left==0) { //左收到信号
        car_left();
    }

    else if(left&&right&&on) {
        i++;
        car_stop();
        delay_ms(5000);
        car_go();
        delay_ms(3000);
        car_go();
    }

    else{
        car_stop();
    }
}
```

```
        delay_ms(10);  
    }  
}
```

（六）效果

小车通过红外对管控制自动在既定的赛道上完成了寻轨任务。

（七）遇到的问题及解决方案

问题一：小车遇到赛道上不平整的坑洼后直接停止。

解决方案：将 PWM 的值增加，增加小车电机的转速，从而让小车获得更大的动力，从而越过小坡。

问题二：小车在最后那个倒车入库的路口转向过多，从而无法入库。

解决方案：将程序中小车左自转前面那个延时函数的值从 1800 改为 600，让其从旋转 270 度变为旋转 45 度。

实践总结与体会：

通过本次的综合实训，使得我对 STM32 软硬件系统都有了一个比较深刻的认识和体会。

在双击通信实验中，使用了两组 STM32 开发版并使用了两组杜邦线将双方的串口进行连接，既学习了串口复用和串口初始化的流程，也对 STM32 的原理图的运用更加的熟悉了，在程序中使用了串口的发送和接收流程，对串口的通信有了一个较为深刻的认识。

在定时器实验中，通过对 32 时钟定时器完成了小车时钟的计算，既对时钟初始化进行了学习，也对时钟回调函数进行了进一步的实践。

在小车寻轨实验中，需要将电机、板子、驱动版进行连接，在期间需要焊接大量的线材，并且需要将线按照元器件的引脚说明进行连接，这对动手能力提高是有很大大益处的，在小车程序设计中，对寻轨地区的观察与思考，是对逻辑思维的一次考验，在小车寻轨程序的调试中小车的延时函数的数值和 PWM 设定的值稍有不慎小车就会冲出赛道，这也是对观察和程序调试能力的一个考验。

在这三次实验中，我充分的体会到了程序设计的不容易，对任何事物都要保持一颗敬畏之心，在实验中要保持一颗好奇心，在整个学习生涯中更是要有一颗恒心，三次实验无论结果如何都是一次生动有趣的学习经历，对日后的上岗实习都有十足的帮助。

实践指导教师评语：

实践成绩：优秀 ☐ 良好 ☐ 一般 ☐ 及格 ☐ 不及格 ☐

签名：

年 月 日

附：