Motor Controller

万晨阳 3210105327

实验要求

- 做一个控制系统，输入是温度/光敏传感器，输出是步进/直流电机。
- 设定一个初始值（以当前温度/亮度为参考），如果传感器温度/亮度超过初始值，则电机正传；如果低于设定值，则电机反转
- 当前值和初始值差异越大，则电机转速越快。

功能介绍

本控制系统的功能如下：

- 利用`ds18b20`检测当前温度，并将设定的参考温度和当前温度在数码管中进行显示。数码管左边四位是参考温度，右边四位是当前温度。
- 支持通过独立按键模块中的部分按键对参考温度进行实时设定，支持的参考温度设定范围为`0~50`度。（`K1`能使得当前位置++，`K2`能使得当前位置--，`K5`能够移动操作位）.
- 根据当前温度与参考温度的相对大小关系，系统能够控制电机正转或者反转，并且转速大小与温度的差值正相关（温度差值越大，转速越快）。
- 当前温度高于参考温度时，灯`D1`会亮；反之不亮。
- 通过串口通讯，系统会每隔一段时间将当前温度、参考温度和操作位信息进行发送和打印。

基本流程

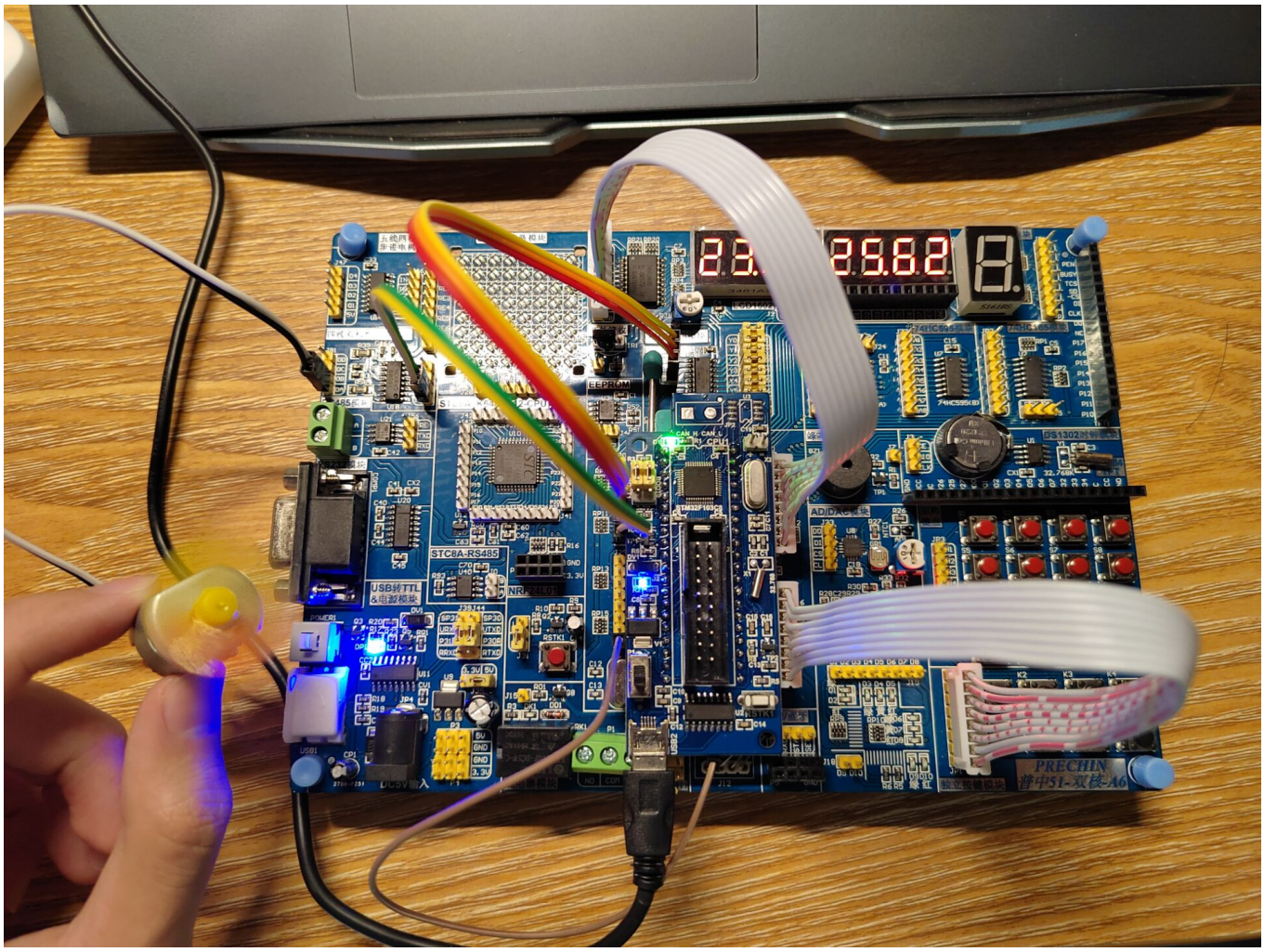
利用伪代码进行描述。

```
BEGIN:
        Init();
        while(True)
        {
                now_temperature = getTempterature();
                keyProcess();
                updateRefTemperature();
                sendSerialData(now_temperature, ref_temperature);
                driveMotor(now_temperature, ref_temperature);

                delay();
        }
END.
```

硬件配置与实验结果

接线与正常工作状态如下：



源代码

为了实现以上所述的功能，我需要对一些冲突的管脚进行重设，所以我修改了部分头文件中的内容。这里展示的是`main.c`以及被我修改的头文件内容。

`main.c`

```
// #include "stm32f10x.h"
#include "system.h"
#include "SysTick.h"
#include "led.h"
#include "usart.h"
#include "ds18b20.h"
#include "pwm.h"
#include "smg.h"
#include "key.h"

u8 smgduan[16] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07,
                  0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71}; // 0~F
u8 NOW_TEMP[4] = {2, 8, 0, 0};
u8 REF_TEMP[4] = {2, 8, 0, 0}; // modifiable
u8 dot = 0;
float diff;
float now_temp;
float ref_temp;

void Display(void); // update the display
void Keydown(void); // scan the key and respond to the keydown event
void Drive(void);   // drive the motor

int main()
```

```c
{
    u8 cnt = 0;
    SysTick_Init(72);
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    LED_Init();
    SMG_Init();
    TIM3_CH1_PWM_Init(2000, 72 - 1);
    KEY_Init();
    USART1_Init(9600);

    while (DS18B20_Init())
    {
        printf("DS18B20 Init Failed!\r\n");
        delay_ms(500);
    }
    printf("DS18B20 Init Success!\r\n");

    while (1)
    {
        Keydown();
        now_temp = DS18B20_GetTemperture();
        ref_temp = 10 * REF_TEMP[0] + REF_TEMP[1] + 0.1 * REF_TEMP[2] + 0.01 * REF_TEMP[3];
        Display();
        Drive();
        delay_us(1000);

        if (cnt > 200)
        {
            cnt = 0;
            printf("now_temp: %f\r\n", now_temp);
            printf("ref_temp: %f\r\n", ref_temp);
            printf("dot: %d\r\n", dot);
        }
        cnt++;
    }
}

void Drive(void)
{
    if (now_temp < ref_temp)
    {
        led1 = 0;
        diff = ref_temp - now_temp;
        TIM_SetCompare1(TIM3, 0);
        TIM_SetCompare2(TIM3, diff * 50 + 250);
    }
    else if (now_temp > ref_temp)
    {
        led1 = 1;
        diff = now_temp - ref_temp;
        TIM_SetCompare1(TIM3, diff * 50 + 250);
        TIM_SetCompare2(TIM3, 0);
    }
}

void Keydown(void)
{
    u8 key;
    key = KEY_Scan(0);
    switch (key)
    {
    case KEY1_VALUE:
        REF_TEMP[dot] += 1;
        break;

    case KEY2_VALUE:
        REF_TEMP[dot] -= 1;
        break;
```

```
    case KEY5_VALUE:
        dot += 1;
        break;
    }

    if (dot == 4)
        dot = 0;

    if (REF_TEMP[3] == 10)
    {
        REF_TEMP[3] = 0;
        REF_TEMP[2]++;
    }
    if (REF_TEMP[2] == 10)
    {
        REF_TEMP[2] = 0;
        REF_TEMP[1]++;
    }
    if (REF_TEMP[1] == 10)
    {
        REF_TEMP[1] = 0;
        REF_TEMP[0]++;
    }
    if (REF_TEMP[3] == 255)
    {
        REF_TEMP[3] = 9;
        REF_TEMP[2]--;
    }
    if (REF_TEMP[2] == 255)
    {
        REF_TEMP[2] = 9;
        REF_TEMP[1]--;
    }
    if (REF_TEMP[1] == 255)
    {
        REF_TEMP[1] = 9;
        REF_TEMP[0]--;
    }
    if (REF_TEMP[0] == 255)
    {
        REF_TEMP[3] = 0;
        REF_TEMP[2] = 0;
        REF_TEMP[1] = 0;
        REF_TEMP[0] = 0;
    }
    if (10 * REF_TEMP[0] + REF_TEMP[1] + 0.1 * REF_TEMP[2] + 0.01 * REF_TEMP[3] > 50)
    {
        REF_TEMP[3] = 0;
        REF_TEMP[2] = 0;
        REF_TEMP[1] = 0;
        REF_TEMP[0] = 5;
    }
}

void Display(void)
{

    u16 now_temp_int = (u16)(now_temp * 100);

    NOW_TEMP[3] = now_temp_int % 10;         // individual digit
    NOW_TEMP[2] = now_temp_int / 10 % 10;    // ten digit
    NOW_TEMP[1] = now_temp_int / 100 % 10;   // hundred digit
    NOW_TEMP[0] = now_temp_int / 1000 % 10; // thousand digit

    // 000
    GPIO_ResetBits(GPIOA, GPIO_Pin_3);
    GPIO_ResetBits(GPIOA, GPIO_Pin_4);
    GPIO_ResetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[REF_TEMP[0]]) << 8);
```

```
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);

    // 001
    GPIO_SetBits(GPIOA, GPIO_Pin_3);
    GPIO_ResetBits(GPIOA, GPIO_Pin_4);
    GPIO_ResetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[REF_TEMP[1]] + 0x80) << 8);
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);

    // 010
    GPIO_ResetBits(GPIOA, GPIO_Pin_3);
    GPIO_SetBits(GPIOA, GPIO_Pin_4);
    GPIO_ResetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[REF_TEMP[2]]) << 8);
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);

    // 011
    GPIO_SetBits(GPIOA, GPIO_Pin_3);
    GPIO_SetBits(GPIOA, GPIO_Pin_4);
    GPIO_ResetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[REF_TEMP[3]]) << 8);
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);

    // 100
    GPIO_ResetBits(GPIOA, GPIO_Pin_3);
    GPIO_ResetBits(GPIOA, GPIO_Pin_4);
    GPIO_SetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[NOW_TEMP[0]]) << 8);
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);

    // 101
    GPIO_SetBits(GPIOA, GPIO_Pin_3);
    GPIO_ResetBits(GPIOA, GPIO_Pin_4);
    GPIO_SetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[NOW_TEMP[1]] + 0x80) << 8);
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);

    // 110
    GPIO_ResetBits(GPIOA, GPIO_Pin_3);
    GPIO_SetBits(GPIOA, GPIO_Pin_4);
    GPIO_SetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[NOW_TEMP[2]]) << 8);
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);

    // 111
    GPIO_SetBits(GPIOA, GPIO_Pin_3);
    GPIO_SetBits(GPIOA, GPIO_Pin_4);
    GPIO_SetBits(GPIOA, GPIO_Pin_5);
    GPIO_Write(SMG_PORT, (u16)(smgduan[NOW_TEMP[3]]) << 8);
    delay_us(1000);
    GPIO_Write(SMG_PORT, 0x0000);
}
```

pwm.c

因为要使得电机正转与反转，所以我选择同时使用`TIM3`的两个比较通道。这里修改的内容是添加了对比较通道2的初始化。

```
#include "pwm.h"

void TIM3_CH1_PWM_Init(u16 per, u16 psc)
{
```

```
    TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
    TIM_OCInitTypeDef TIM_OCInitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    /* 开启时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

    /*  配置GPIO的模式和IO口 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; // 复用推挽输出
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    TIM_TimeBaseInitStructure.TIM_Period = per;    // 自动装载值
    TIM_TimeBaseInitStructure.TIM_Prescaler = psc; // 分频系数
    TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up; // 设置向上计数模式
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStructure);

    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OC1Init(TIM3, &TIM_OCInitStructure);          // 输出比较通道1初始化
    TIM_OC1PreloadConfig(TIM3, TIM_OCPreload_Enable); // 使能TIMx在 CCR1 上的预装载寄存器
    TIM_ARRPreloadConfig(TIM3, ENABLE);               // 使能预装载寄存器

    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OC2Init(TIM3, &TIM_OCInitStructure);          // 输出比较通道2初始化
    TIM_OC2PreloadConfig(TIM3, TIM_OCPreload_Enable); // 使能TIMx在 CCR1 上的预装载寄存器
    TIM_ARRPreloadConfig(TIM3, ENABLE);               // 使能预装载寄存器


    TIM_Cmd(TIM3, ENABLE); // 使能定时器
}
```

key.h

默认的按键占用管脚与数码管冲突，所以我把独立按键模块的8个按键重新分配到`PB0`到`PB7`的范围内。这里修改的内容是调整了按键对应的管脚并增加了按键。

```
#ifndef _key_H
#define _key_H

#include "system.h"

#define KEY1_Pin GPIO_Pin_0
#define KEY2_Pin GPIO_Pin_1
#define KEY3_Pin GPIO_Pin_2
#define KEY4_Pin GPIO_Pin_3
#define KEY5_Pin GPIO_Pin_4
#define KEY6_Pin GPIO_Pin_5
#define KEY7_Pin GPIO_Pin_6
#define KEY8_Pin GPIO_Pin_7

#define KEY_Port (GPIOB)

#define KEY1 PBin(0)
#define KEY2 PBin(1)
#define KEY3 PBin(2)
#define KEY4 PBin(3)
#define KEY5 PBin(4)
#define KEY6 PBin(5)
#define KEY7 PBin(6)
#define KEY8 PBin(7)
```

```
#define KEY1_VALUE 1
#define KEY2_VALUE 2
#define KEY3_VALUE 3
#define KEY4_VALUE 4
#define KEY5_VALUE 5
#define KEY6_VALUE 6
#define KEY7_VALUE 7
#define KEY8_VALUE 8

void KEY_Init(void);
u8 KEY_Scan(u8 mode);
#endif
```

key.c

前面按键的地方改了，所以这里在按键扫描函数里也添加了`K5`。

```
#include "key.h"
#include "SysTick.h"

void KEY_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure; // 定义结构体变量
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

    GPIO_InitStructure.GPIO_Pin = (KEY1_Pin | KEY2_Pin | KEY5_Pin);
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; // 上拉输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(KEY_Port, &GPIO_InitStructure);
}

u8 KEY_Scan(u8 mode)
{
    static u8 key = 1;
    if (key == 1 && (KEY1 == 0 || KEY2 == 0 || KEY5 == 0)) // 任意一个按键按下
    {
        delay_ms(10); // 消抖
        key = 0;
        if (KEY1 == 0)
        {
            return KEY1_VALUE;
        }
        else if (KEY2 == 0)
        {
            return KEY2_VALUE;
        }
        else if (KEY5 == 0)
        {
            return KEY5_VALUE;
        }
    }
    else if (KEY1 == 1 && KEY2 == 1 && KEY5 == 1) // 无按键按下
    {
        key = 1;
    }
    if (mode == 1) // 连续按键按下
    {
        key = 1;
    }
    return 0;
}
```

smg.c

这里引入`PA3 PA4 PA5`通过74HC138对数码管进行位选。

```h
#ifndef _smg_H
#define _smg_H

#include "system.h"

#define SMG_PORT GPIOB
#define SMG_PIN (GPIO_Pin_8 | GPIO_Pin_9 | GPIO_Pin_10 | GPIO_Pin_11 | GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 |
GPIO_Pin_15)
#define SMG_PORT_RCC RCC_APB2Periph_GPIOB
#define LS_PORT GPIOA
#define LS_PIN (GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5)
#define LS_PORT_RCC RCC_APB2Periph_GPIOA

void SMG_Init(void);

#endif
```

修改

老师是的修改要求是把温度设定到30℃±2℃的范围内。修改的思路很简单，只要对于main.c中的Drive函数中，把pwm波的占空比与温度差之间的函数关系式修改一下，使得其满足要求即可。（使用线性函数，只要把系数调大一些即可）。

修改之后默认参考温度为30℃，能够保证在28℃和32℃的时候输出的pwm波达到占空比约为100%。