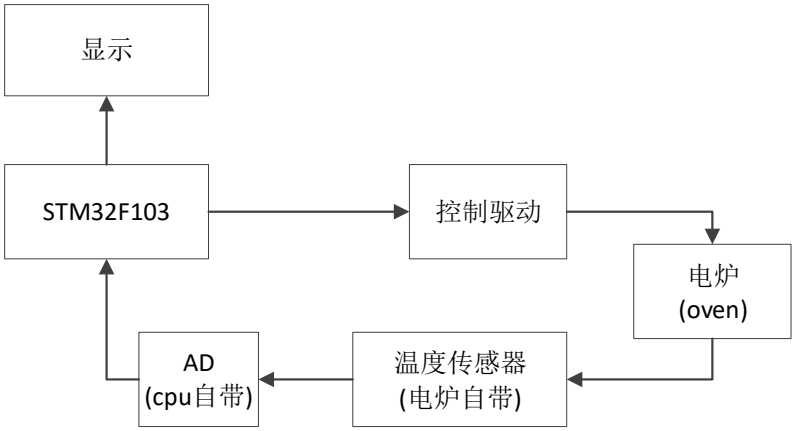
题目：基于 stm32f103 微控制器的电炉温度控制仿真系统

软件环境：推荐采用 Proteus 8.8 及以上仿真软件，Keil 软件开发系统及 STM32CubeMX，也可以采用其他软件平台实现仿真功能。

实现功能：使用 stm32f103 微控制器，搭建一个闭环电炉控制系统。

·系统框图如下：

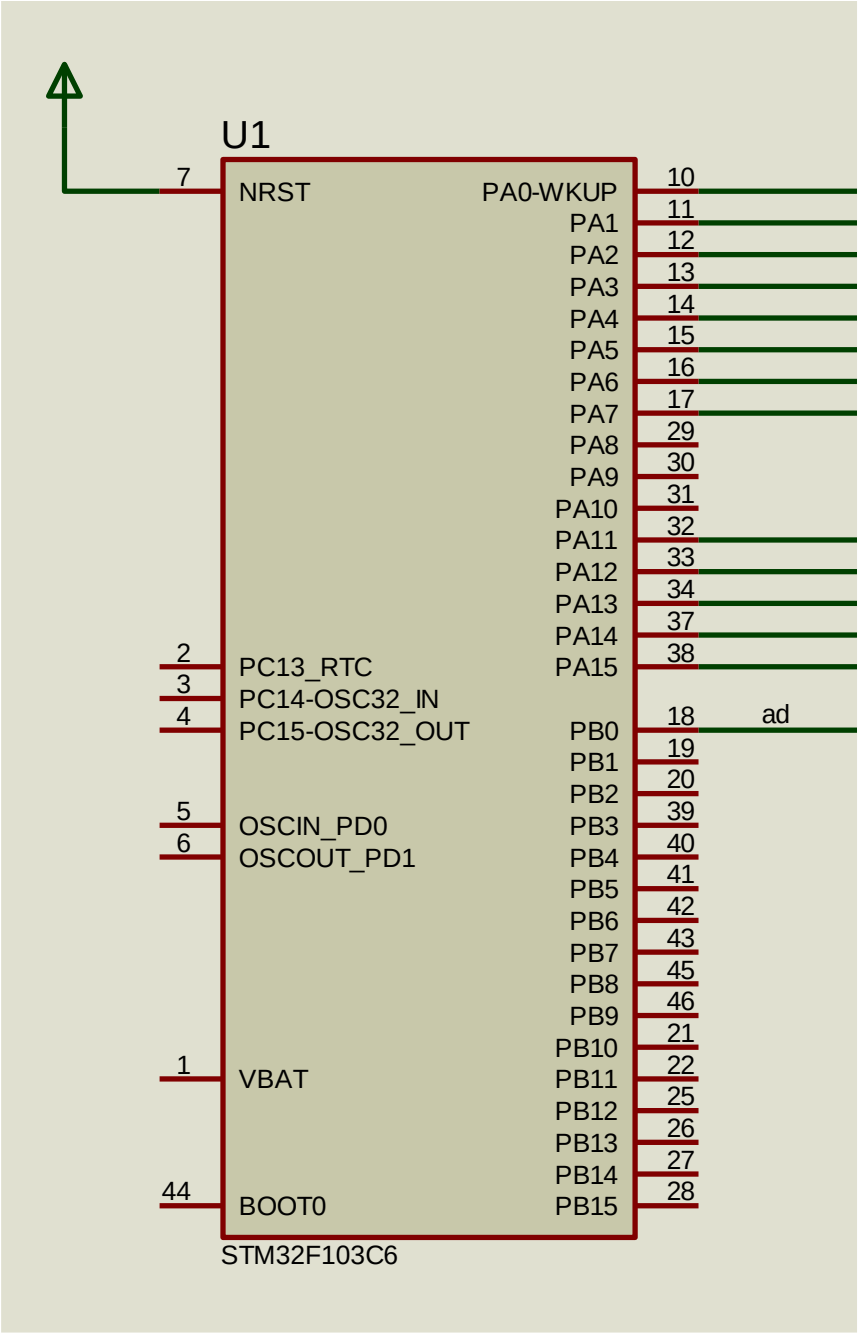


·功能：stm32f103 控制加热器从室温 25°开始加热，当到达预定的温度（摄氏 35°+（学号末两位数的和/2））时，停止加热，同时，加热器的实时温度在 LCD 显示器上显示。如:学生学号末两位数为 68,则系统加热到 42°（35+(6+8)/2=42）时停止加热。

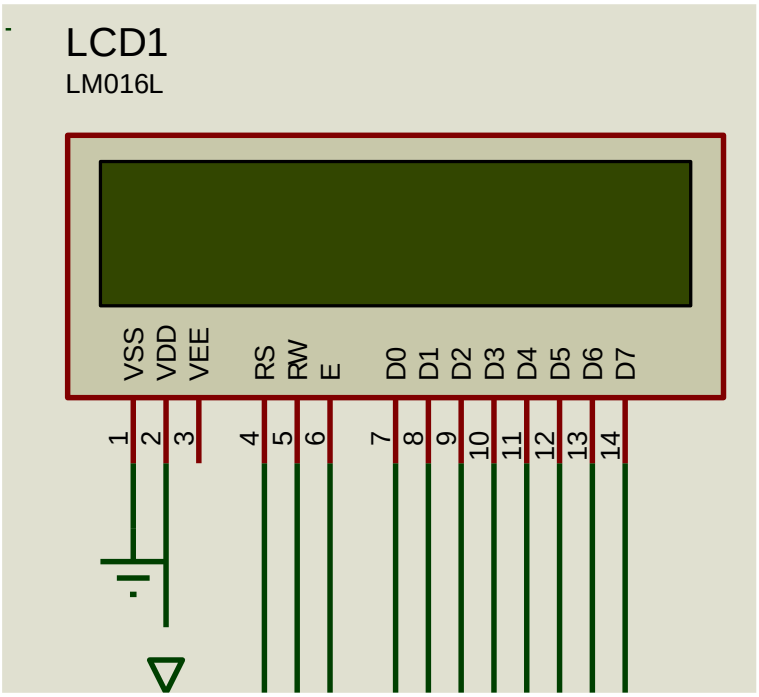
关于 Proteus，详见“Proteus 安装与使用”文档。

仿真中所用器件：

Stm32f103 微控制器，如下图所示。



LCD，见下图



建议：微控制器与 LCD 的连接方式

PA0---D0

PA1---D1

PA2---D2

PA3---D3

PA4---D4

PA5---D5

PA6---D6

PA7---D7

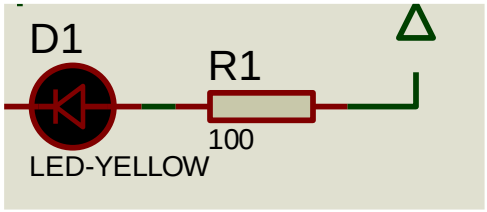
PA13---E

PA14---RW

PA15---RS

刷新时间指示灯

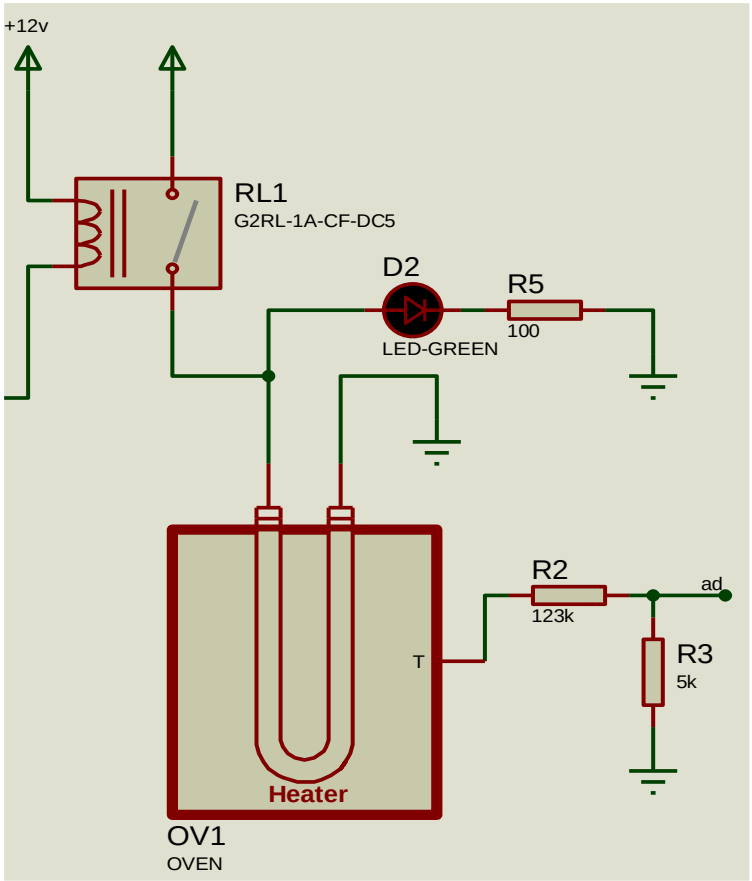
建议与微控制器的 PA11 相连



电炉控制电路

建议继电器的控制端与微控制器的 PA12 连接。

注意继电器另外一个控制端必须接 12v 电源。



Keil 软件模块：

所用 keil 模块已经由 STM32CubeMX 软件自动生成，在 ATest 文件夹内，可直接用 keil 软件打开工程，然后添加自己需要的模块。

• 添加 2 个头文件

```
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "string.h"
/* USER CODE END Includes */
```

添加 LCD 需要的变量

```
/* USER CODE BEGIN PV */
uint8_t const table1[]="Temperature:";
char buff[10];
/* USER CODE END PV */
```

添加 LCD 显示控制模块

```
/* USER CODE BEGIN 0 */
void printFloat(float value)
{
    int tmp,tmp1;
    tmp = (int)value;
    tmp1=(int)((value-tmp)*10)%10;
    sprintf(&buff[0],"%d.%d\r\n",tmp,tmp1);
}

void Delay_us(uint16_t us)
{
    uint16_t differ=0xffff-us-5;

    __HAL_TIM_SET_COUNTER(&htim3,differ);

    HAL_TIM_Base_Start(&htim3);

    while(differ<0xffff-6)
        differ=__HAL_TIM_GET_COUNTER(&htim3);

    HAL_TIM_Base_Stop(&htim3);
}
```

```c
void LcdWriteCom(uint8_t com)
{
    Delay_us(20);
    GPIOA->BSRR = 0x00ff0000;
    GPIOA->BSRR = (com);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_15,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_14,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,GPIO_PIN_RESET);
    Delay_us(10);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,GPIO_PIN_SET);
    Delay_us(10);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,GPIO_PIN_RESET);
    Delay_us(10);
}

void LcdWriteDate(uint8_t date)
{
    Delay_us(20);
    GPIOA->BSRR = 0x00ff0000;
    GPIOA->BSRR = (date);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_15,GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_14,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,GPIO_PIN_RESET);
    Delay_us(10);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,GPIO_PIN_SET);
    Delay_us(10);
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,GPIO_PIN_RESET);
    Delay_us(10);
}
void LCD1602Init(void)
{
    uint8_t index=0;
    HAL_Delay(100);
    LcdWriteCom(0x38);  //设置16*2显示，8位数据接口
    LcdWriteCom(0x0c); //开显示，显示光标且闪烁
    LcdWriteCom(0x06);//写一个指针自动加一
    LcdWriteCom(0x01);//清屏
    HAL_Delay(100);//延时一段时间时间，等待LCD1602稳定

    LcdWriteCom(0x80);//设置第一行 数据地址指针
    for(index=0;index<13;index++)
        LcdWriteDate(table1[index]);  //写入数据
}
```

```
void LCD1602WriteCommand(uint8_t comm)
{
    LcdWriteCom(0xc0 + 14);
    LcdWriteDate(comm);  //写入数据
}
```

在 main 函数中添加

```
/* USER CODE BEGIN 1 */
  uint16_t adc_v;
  uint8_t i;
/* USER CODE END 1 */

  /* USER CODE BEGIN 2 */
    HAL_ADCEx_Calibration_Start(&hadc1);
    LCD1602Init();
    HAL_Delay(10);
  /* USER CODE END 2 */

    /* USER CODE BEGIN 3 */
    HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_11);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 50);
    if(HAL_IS_BIT_SET(HAL_ADC_GetState(&hadc1), HAL_ADC_STATE_REG_EOC))
    {
        adc_v = HAL_ADC_GetValue(&hadc1);
        printFloat((float)adc_v*128/4096);//%0.2f 5.0 3.3

        LcdWriteCom(0xc0);//设置第2行 数据地址指针
        for(i=0;i<strlen(buff);i++)
            LcdWriteDate(buff[i]);  //写入数据
    }
    if(((float)adc_v * 128 / 4096) > 30.0)
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_SET);
    else
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_RESET);

    HAL_Delay(500);
  }
  /* USER CODE END 3 */
```

有关cpu 自带的TIMER3 及AD 的参数设置，在STM32CubeMX 系统中已经设置好，也可以在 main.c 模块中自行修改参数。