成　　绩	
批阅教师	
日　　期	

桂林电子科技大学

实训报告

2021 -2022 学年第 2 学期

学　院　电子信息学院

课　程　智能控制实训

姓　名　彭怿骁 黄李杰 蒋铭轩

学　号　201202060116
201202060110
201202060109

指导老师　周光祥

实训题目：水温测量仪

摘要

为满足液体的温度要求，设计了一种基于 STM32F401CCU6 单片机的小型智能水温测量仪系统。该系统利用 DS18B20 传感器检测水温，DHT11 检测环境温湿度，并通过 LCD 触摸屏显示液体温度和环境温湿度。LCD 触摸屏能实时记录液体温度和环境温湿度共 3 种数据曲线并显示。现详细介绍了水温测量仪系统的各部分硬件构成以及控制部分的程序设计。

整机演示的结果证明该设计能够达到预期效果。

关键词： STM32F401CCU6，DHT11，LCD 触摸屏，液体温度，环境温湿度

Training topic: instrument for measuring water temperature

Abstract

In order to meet the temperature requirements of liquid, a small intelligent water temperature measuring instrument system based on STM32F401CCU6 microcontroller is designed. The system uses DS18B20 sensor to detect water temperature, DHT11 to detect ambient temperature and humidity, and LCD touch screen display liquid temperature and ambient temperature and humidity. LCD touch screen can record liquid temperature and ambient temperature and humidity in real time three data curves and display. The hardware structure of each part of the water temperature measuring instrument system and the program design of the control part are introduced in detail.

The result of practical demonstration shows that the design can achieve the expected effect.

Keywords: STM32F401CCU6, DHT11, LCD touch screen, liquid temperature, ambient temperature and humidity

目 录

第一章 概述

1.1 设计要求

1.1.1 设计任务

基本要求要求： (1)用单片机实现;(2)用 c 语言编程;(3)硬件电路板布局合理;(4)用数码管显示器显示;(5)温度测量范围是: 25℃--90℃;(6)误差小于 8%。(7)每组制作的电路板上面写下组员学号。(8)温度传感器可以选择热敏电阻或者 18B20 传感器。

1.1.2 性能指标要求

（1）在 LCD 上显示温湿度小数点后两位。

（2）在 LCD 上显示液体温度和环境温湿度实时曲线。

1.2 系统实现的基本原理及框图

1.2.1 基本原理

此系统是一个智能的水温测量仪，可在 LCD 触摸屏上显示实时的环境温湿度和液体温度，并且 LCD 触摸屏能实时记录液体温度和环境温湿度共 3 种数据曲线并显示。LCD 触摸屏通过 UART 接口和主控芯片 STM32F401CCU6 连接，把曲线和温湿度信息在 LCD 触摸屏上显示出来。温湿度的采集使用的是 DHT11 传感器，DHT11 数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。水温采用的温度检测传感器为 DS18B20 温度传感器。该型号的温度传感器采用导热性高的密封胶灌封，因而具有灵敏度高和延迟度小等优点 ，同时其测量温度范围较大（-55～+125 ℃），满足系统功能需求。

1.2.2 总体框图

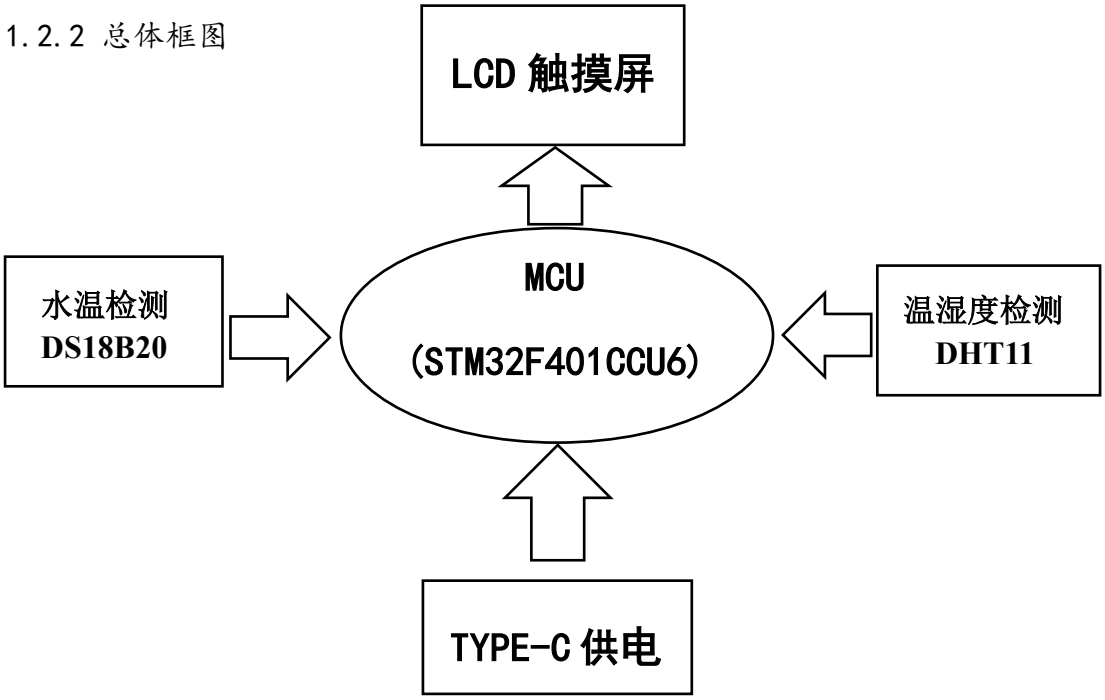


图 1 总体框图

2 系统硬件电路设计

2.1 主控芯片：STM32F401CCU6

STM32F4 采用 Cortex M4 内核，带 FPU 和 DSP 指令集，拥有多达 192KB 的片内 SRAM，带摄像头接口（DCMI）、加密处理器（CRYP）、USB 高速 OTG、真随机数发生器、OTP 存储器等。增强的外设功能，STM32F4 具有快速的模数转换、更低的 ADC/DAC 工作低压、32 位定时器、带日历功能的实时时钟（RTC）、IO 复用功能大大增强、4K 字节的电池备份 SRAM 以及更快的 USART 和 SPI 通信速度。STM32F4 最高运行频率可达 168Mhz，STM32F4 拥有 ART 自适应实时加速器，可以达到相当于 FLASH 零等待周期的性能，STM32F4 的 FSMC 采用 32 位多重 AHB 总线矩阵，拥有更快的访问速度。低功耗，STM32F40x 的功耗为：238uA/Mhz。

2.2 LCD 触摸屏

usart hmi LCD 触摸屏就是设备封装好底层功能以后，通过串口(USART 232)与用户 MCU 进行交互。 MCU 可以随时通过 USART 发指令通知设备：切换某个页面或者改变某个组件的属性。设备也可以随时通过 USART 通知用户 MCU 操作者，目前触摸了页面上的某个组件或者设备当前进入了某个页面。

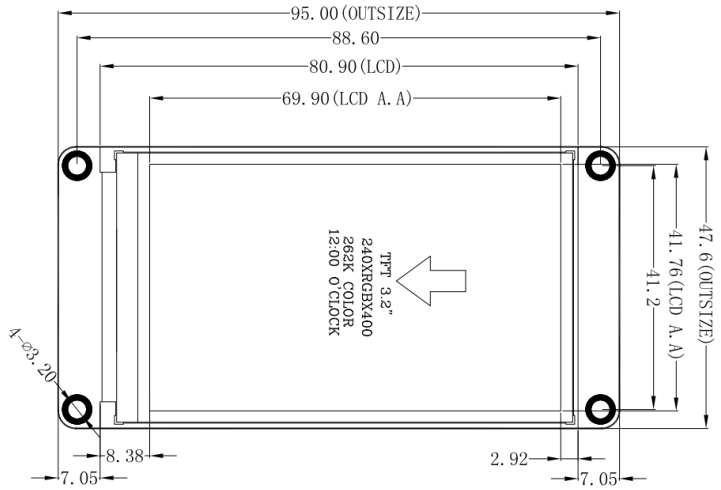


图 2 LCD 触摸屏外观图

LCD 和主控芯片的连接：LCD_VCC------5V

LCD_GND-----GND

LCD_RX----- (PA9)

LCD_TX----- (PA10)

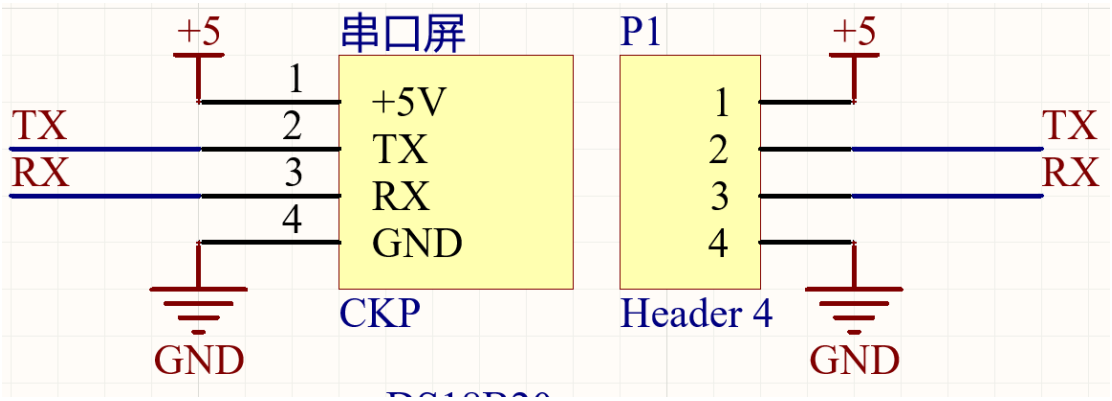


图 3 LCD 连接原理图

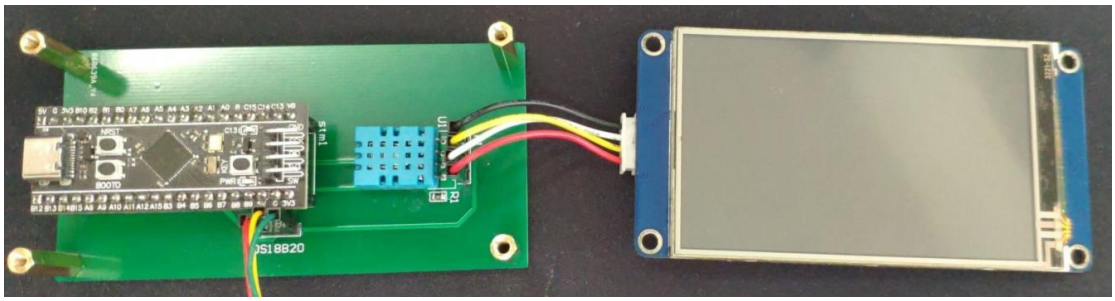


图 4 LCD 实物连接图

2.3 DHT11 电路

DHT11 数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连。测量的范围 20－90%RH 0－50℃，测湿精度±5%RH，测温精度±2℃。DHT11 和 MCU 的通讯方式是采用单总线，单总线采用单根信号线，既传输时钟又传输数据，而且数据传输是双向的，具有节省 I/O 口线、资源结构简单、成本低廉、便于总线扩展和维护等诸多优点。

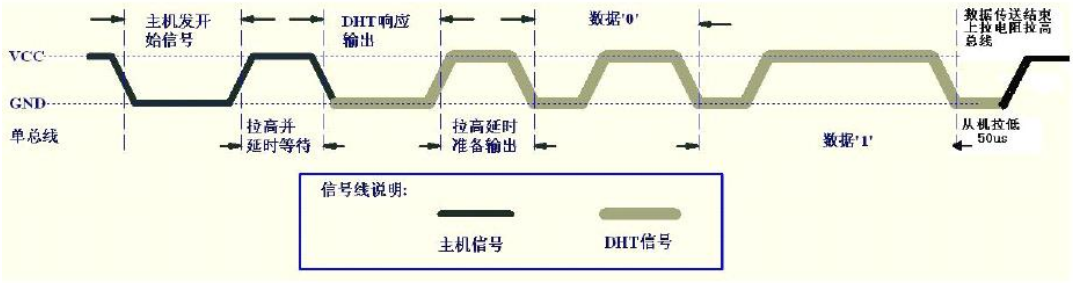


图 5 DHT11 通讯时序（部分）

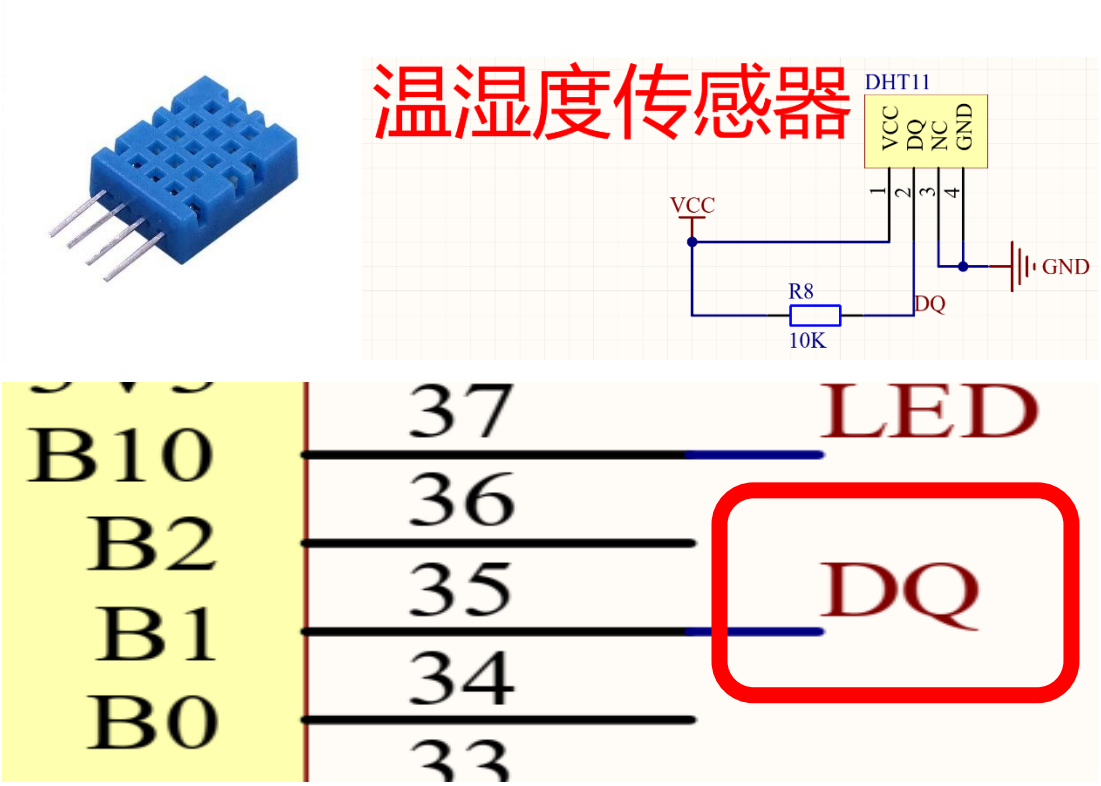


图 6 DHT11 原理图连接

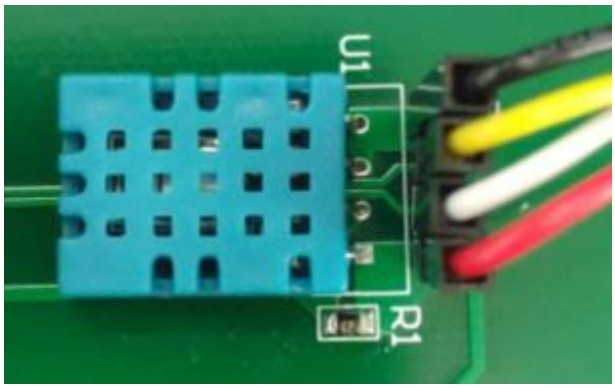


图 7 DHT11 实物图

2.4 DS18B20 电路

2.4.1 DS18B20 的特性及引脚

DS18B20 的特性及引脚 DS18B20 数字式温度传感器的外部形状如图 8 所示。DS18B20 的电路很简单,由一片 DS18B20 和一只 4.7kΩ 的上拉电阻构成。DS18B20 内集成了一个温度传感器、64 位 ROM、9 字节 RAM、3 字节 EERAM (掉电可保存),可将温度信号转换为数字信号直接输出。DS18B20 与外部的接口为单总线方式,即数据的输入、输出及同步均由同一根线完成。其温度测量范围为 55℃~125℃,在-10C~80C 范围内精度为士 0.5C,输出的温度值可编程为 9~12 位。VD 接电源,3V5V; GND 为地; DQ 为数据的输入输出。DQ 作为输出时为漏极开路,必须加 47k 的上拉电阻。

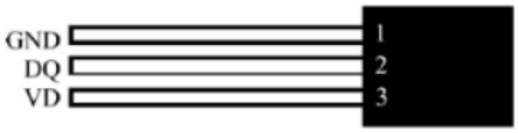


图 8

2.4.2 工作方式

ＤＳ18Ｂ20 传感器进行的功能操作是在发送命令的基础上完成的　上电后传感器处于空闲状态　需要控制器发送命令才能完成温度转换。对传感器的功能操作的次序是首先完成对芯片内部的ＲＯＭ操作　有 5 条操作ＲＯＭ的指令可用于器件识别　它们分别是：ＲｅａｄＲＯＭ（33Ｈ）、ＭａｔｃｈＲＯＭ（55Ｈ）、ＳｋｉｐＲＯＭ（ＣＣＨ）、ＳｅａｒｃｈＲＯＭ（Ｆ0Ｈ）、ＡｌａｒｍＳｅａｒｃｈ（ＥＣＨ）。

ＲｅａｄＲＯＭ：用于读出 64 位ＲＯＭ数据　适用于仅有 1 个ＤＳ18Ｂ20 的场合。

ＭａｔｃｈＲＯＭ：查找与给定 64 位ＲＯＭ数据相匹配的ＤＳ18Ｂ20。

ＳｋｉｐＲＯＭ：适用于仅有 1 个ＤＳ18Ｂ20 的场合需给出 64 位码就能快速选定器件。

ＳｅａｒｃｈＲＯＭ:适用于多个ＤＳ18Ｂ20 的场合　该指令可识别出每个器件的ＩＤ号。

ＡｌａｒｍＳｅａｒｃｈ：用于温度报警查询。

9 字节ＲＡＭ中　字节 1、2 用来存放当前测量的温度值　1 为低 8 位　2 为高 8 位；字节 3、4 用来存放预设报警温度的上下限值　3 为上限　4 为下限；字节 5 用于配置寄存器　确定温度数据的位数　相关位为Ｄ5、Ｄ6　和温度位数的对应关系如图 9 所示　字节 5 的其余位均为无关位；字节 6、7、8 均为保留字节　一般不用；字节 9 存放前 8 个字节循环冗余校验码（ＣＲＣ码）。

D6	D5	温度位数	最大转换时间
0	0	9位	93.75ms
0	1	10位	187.5ms
1	0	11位	375ms
1	1	12位	750ms

图 9　D5 D6 和温度位数的对应关系

3 字节的ＥＥＲＡＭ分别对应于ＲＡＭ区的字节 3、4、5　用于备份系统设置。

对ＲＡＭ的操作指令有 6 条　分别为：Ｗｒｉｔｅ（4ＥＨ）、Ｒｅａｄ（ＢＥＨ）、Ｃｏｐｙ（48Ｈ）、Ｃｏｎｖｅｒｔ（44Ｈ）、Ｒｅｃａｌｌ（B8Ｈ）、ＲｅａｄＰｏｗｅｒ（B4Ｈ）。

Ｗｒｉｔｅ：写ＲＡＭ存储器　随后的 3 个字节分别写入ＲＡＭ字节 3、4、5　该指令必须在复位操作前完成。

Ｒｅａｄ：读出ＲＡＭ中所有 9 个字节的数据　该指令可随时被复位操作所终止。

Ｃｏｐｙ：将ＲＡＭ区的 3、4、5 字节备份至ＥＥＲＡＭ。

Ｒｅｃａｌｌ：将ＥＥＲＡＭ中的数据装入ＲＡＭ。

Ｃｏｎｖｅｒｔ：温度转换开始指令。

ＲｅａｄＰｏｗｅｒ：读电源指令　此处不作介绍。

对ＤＳ18Ｂ20 的每一次操作均由 4 个步骤组成：（1 ）初始化（复位操作）；（2 ）对ＲＯＭ操作指令（识别器件）；（3 ）对ＲＡＭ操作指令（读、写、转换）；（4）收发数据　时序图如图 2 所示。

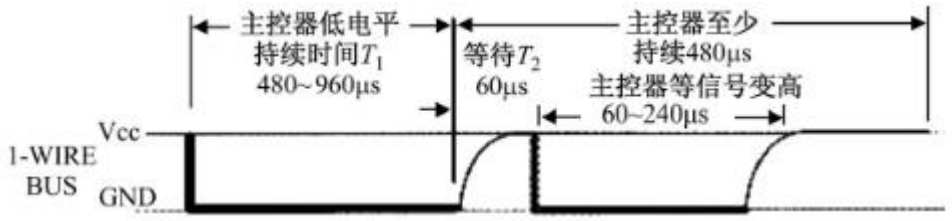


图 10 初始化时序图

图 10　初始化时序图初始化过程如下：

主控器将信号线拉低　持续时间Ｔ1（480μｓ＜ｔ1＜960μｓ）；

主控器等待时间Ｔ2（15μｓ＜ｔ2＜60μｓ）；

主控器检测信号线　若为低　进行下一步　否则重新初始化；

主控器在 240μｓ内等待信号线变高　如变高表示初始化完成　否则重新初始化；

主控器延时至少 480μs 确保应答正确；

在收发数据过程中 每一位的读写时间应至少持续 60μs 以确保读写正确。

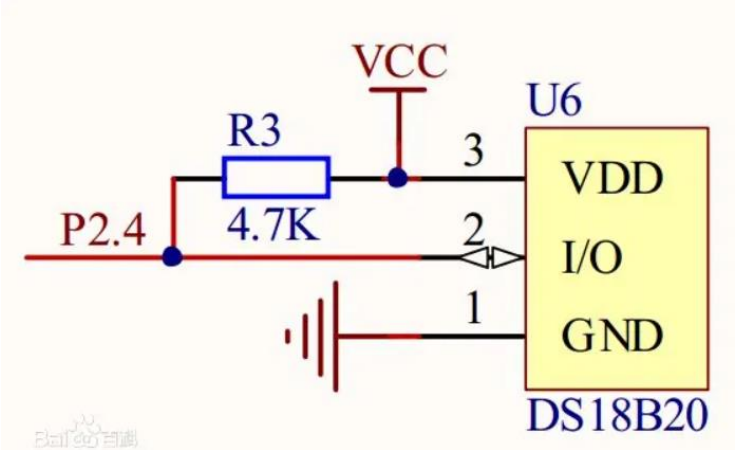


图 11 DS18B20 实物电路

3 系统程序设计

3.1 系统主程序流程设计

3.1.1main 函数执行流程框图及代码

```
int main(void)
{
  /* USER CODE BEGIN 1 */
      uint8_t i, DS18B20ID[8];
       float temperature;
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  DHT11_Init();
  /* USER CODE BEGIN 2 */

  printf("DS18B20 Read\n");
    while(DS18B20_Init())
    {
        printf("DS18B20 not\n");
      HAL_Delay(1000);
    }
    printf("find DS18B20 success\n");
    DS18B20_ReadId(DS18B20ID);
    printf("DS18B20 number: 0x");
     for ( i = 0; i < 8; i ++ )
```

```
      printf ( "%.2X", DS18B20ID[i]);
    printf("\n");

    DS18B20_GetTemp_MatchRom(DS18B20ID);
    HAL_Delay(100);
    setvbuf(stdin,NULL,_IONBF,0);
    setvbuf(stdout,NULL,_IONBF,0);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      temperature=DS18B20_GetTemp_MatchRom(DS18B20ID);
          /* 打印通过 DS18B20 序列号获取的温度*/

          /* 1s 读取温度*/
          printf("t3.txt=\"%.2f\"\xff\xff\xff",temperature);
          printf("add 10,0,%d\xff\xff\xff",(int)temperature);
          if(DHT11_Read_TempAndHumidity(&DHT11_Data)==SUCCESS)
                    {
                      printf("t2.txt=\"%.2f \"\xff\xff\xff",DHT11_Data.humidity);
                      printf("t1.txt=\" %.2f\"\xff\xff\xff",DHT11_Data.temperature);
                     printf("add 9,0,%d\xff\xff\xff",(int)DHT11_Data.humidity);
                     printf("add 8,0,%d\xff\xff\xff",(int)DHT11_Data.temperature);
                    }
                    else
                    {
                      printf("real DHT11 fail\n");
                    }
                HAL_Delay(1000);
  }
}
```

3.2 系统各功能模块子程序流程设计

3.2.1 DHT11 程序

```
//复位 DHT11
void DHT11_Rst(void)
{
    DHT11_IO_OUT();    //SET OUTPUT
  DHT11_DQ_OUT=0;  //拉低 DQ
  delay_ms(20);         //拉低至少 18ms
  DHT11_DQ_OUT=1;  //DQ=1
    delay_us(30);      //主机拉高 20~40us
}
```

```
//等待 DHT11 的回应
//返回 1:未检测到 DHT11 的存在
//返回 0:存在
u8 DHT11_Check(void)
{
    u8 retry=0;
    u8 fsd;
    DHT11_IO_IN();//SET INPUT
    while (DHT11_DQ_IN&&retry<100)//DHT11 会拉低 40~80us
    {
        retry++;
        delay_us(1);
    };
    if(retry>=100)return 1;
    else retry=0;
    while (!DHT11_DQ_IN&&retry<100)//DHT11 拉低后会再次拉高 40~80us
    {
        retry++;
        delay_us(1);
    };
    if(retry>=100)return 1;
    return 0;
}
//从 DHT11 读取一个位
//返回值：1/0
u8 DHT11_Read_Bit(void)
{
    u8 retry=0;
    while(DHT11_DQ_IN&&retry<100)//等待变为低电平
    {
        retry++;
        delay_us(1);
    }
    retry=0;
    while(!DHT11_DQ_IN&&retry<100)//等待变高电平
    {
        retry++;
        delay_us(1);
    }
    delay_us(40);//等待 40us
    if(DHT11_DQ_IN)return 1;
    else return 0;
}
//从 DHT11 读取一个字节
//返回值：读到的数据
u8 DHT11_Read_Byte(void)
{
    u8 i,dat;
    dat=0;
    for (i=0;i<8;i++)
    {
        dat<<=1;
        dat|=DHT11_Read_Bit();
    }
    return dat;
}
//从 DHT11 读取一次数据
```

```
//temp:温度值(范围:0~50° )
//humi:湿度值(范围:20%~90%)
//返回值：0,正常;1,读取失败
u8 DHT11_Read_Data(u8 *temp,u8 *humi)
{
    u8 buf[5];
    u8 i;
    DHT11_Rst();
    if(DHT11_Check()==0)
    {
        for(i=0;i<5;i++)//读取 40 位数据
        {
            buf[i]=DHT11_Read_Byte();
        }
        if((buf[0]+buf[1]+buf[2]+buf[3])==buf[4])
        {
            *humi=buf[0];
            *temp=buf[2];
          fsd=buf[4];
        }
    }else return 1;
    return 0;
}


//初始化 DHT11 的 IO 口 DQ 同时检测 DHT11 的存在
//返回 1:不存在
//返回 0:存在
u8 DHT11_Init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;

  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
//使能 GPIOG 时钟

  //GPIOF9,F10 初始化设置
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;//普通输出模式
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;//推挽输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//50MHz
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;//上拉
  GPIO_Init(GPIOB, &GPIO_InitStructure);//初始化
    DHT11_Rst();
    return DHT11_Check();
}
```

3.2.2 DS18B20 程序

```
#include "bsp_DS18B20.h"

/* 私有类型定义 --------------------------------------------------------------*/
/* 私有宏定义 ----------------------------------------------------------------*/
#define Delay_ms(x)   HAL_Delay(x)
/* 私有变量 ------------------------------------------------------------------*/
/* 扩展变量 ------------------------------------------------------------------*/
/* 私有函数原形 --------------------------------------------------------------*/
static void DS18B20_Mode_IPU(void);
static void DS18B20_Mode_Out_PP(void);
static void DS18B20_Rst(void);
```

```
static uint8_t DS18B20_Presence(void);
static uint8_t DS18B20_ReadBit(void);
static uint8_t DS18B20_ReadByte(void);
static void DS18B20_WriteByte(uint8_t dat);
static void DS18B20_SkipRom(void);
static void DS18B20_MatchRom(void);

/* 函数体 ---------------------------------------------------------------------*/
/**
  * 函数功能:
  * 输入参数: 无
  * 返 回 值: 无
  * 说    明: 无
  */
static void DS18B20_Delay(uint32_t udelay)
{
  uint32_t startval,tickn,delays,wait;

  startval = SysTick->VAL;
  tickn = HAL_GetTick();
  //sysc = 72000;  //SystemCoreClock / (1000U / uwTickFreq);
  delays =udelay * 84; //sysc / 1000 * udelay;
  if(delays > startval)
    {
      while(HAL_GetTick() == tickn)
        {
        }
      wait = 84000 + startval - delays;
      while(wait < SysTick->VAL)
        {
        }
    }
  else
    {
      wait = startval - delays;
      while(wait < SysTick->VAL && HAL_GetTick() == tickn)
        {
        }
    }
}
/**
  * 函数功能: DS18B20 初始化函数
  * 输入参数: 无
  * 返 回 值: 无
  * 说    明: 无
  */
uint8_t DS18B20_Init(void)
{
  DS18B20_Dout_GPIO_CLK_ENABLE();
  DS18B20_Mode_Out_PP();
     DS18B20_Dout_HIGH();
     DS18B20_Rst();
  return DS18B20_Presence ();
}
/**
  * 函数功能: 使DS18B20-DATA引脚变为上拉输入模式
  * 输入参数: 无
```

```
  * 返 回 值: 无
  * 说    明: 无
  */
static void DS18B20_Mode_IPU(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;

  /* 串口外设功能GPIO配置 */
  GPIO_InitStruct.Pin   = DS18B20_Dout_PIN;
  GPIO_InitStruct.Mode  = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull  = GPIO_PULLUP;
  HAL_GPIO_Init(DS18B20_Dout_PORT, &GPIO_InitStruct);

}
/**
  * 函数功能: 使DS18B20-DATA引脚变为推挽输出模式
  * 输入参数: 无
  * 返 回 值: 无
  * 说    明: 无
  */
static void DS18B20_Mode_Out_PP(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;

  /* 串口外设功能GPIO配置 */
  GPIO_InitStruct.Pin = DS18B20_Dout_PIN;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(DS18B20_Dout_PORT, &GPIO_InitStruct);
}
/**
  * 函数功能: 主机给从机发送复位脉冲
  * 输入参数: 无
  * 返 回 值: 无
  * 说    明: 无
  */
static void DS18B20_Rst(void)
{
    /* 主机设置为推挽输出 */
    DS18B20_Mode_Out_PP();
    DS18B20_Dout_LOW();
    /* 主机至少产生480us的低电平复位信号 */
    DS18B20_Delay(750);
    /* 主机在产生复位信号后，需将总线拉高 */
    DS18B20_Dout_HIGH();
    /*从机接收到主机的复位信号后，会在15~60us后给主机发一个存在脉冲*/
    DS18B20_Delay(15);
}
/**
  * 函数功能: 检测从机给主机返回的存在脉冲
  * 输入参数: 无
  * 返 回 值: 0：成功，1：失败
  * 说    明: 无
  */
static uint8_t DS18B20_Presence(void)
{
    uint8_t pulse_time = 0;
```

```
    /* 主机设置为上拉输入 */
    DS18B20_Mode_IPU();
    /* 等待存在脉冲的到来，存在脉冲为一个60~240us的低电平信号
     * 如果存在脉冲没有来则做超时处理，从机接收到主机的复位信号后，会在15~60us后给主机发一个存在脉冲
     */
    while( DS18B20_Data_IN() && pulse_time<100 )
    {
        pulse_time++;
        DS18B20_Delay(1);
    }
    /* 经过100us后，存在脉冲都还没有到来*/
    if( pulse_time >=100 )
        return 1;
    else
        pulse_time = 0;

    /* 存在脉冲到来，且存在的时间不能超过240us */
    while( !DS18B20_Data_IN() && pulse_time<240 )
    {
        pulse_time++;
        DS18B20_Delay(1);
    }
    if( pulse_time >=240 )
        return 1;
    else
        return 0;
}

/**
  * 函数功能: 从DS18B20读取一个bit
  * 输入参数: 无
  * 返 回 值: 读取到的数据
  * 说    明: 无
  */
static uint8_t DS18B20_ReadBit(void)
{
    uint8_t dat;
    /* 读0和读1的时间至少要大于60us */
    DS18B20_Mode_Out_PP();
    /* 读时间的起始：必须由主机产生 >1us <15us 的低电平信号 */
    DS18B20_Dout_LOW();
    DS18B20_Delay(10);
    /* 设置成输入，释放总线，由外部上拉电阻将总线拉高 */
    DS18B20_Mode_IPU();
    //Delay_us(2);
    if( DS18B20_Data_IN() ==GPIO_PIN_SET )
        dat = 1;
    else
        dat = 0;
    /* 这个延时参数请参考时序图 */
    DS18B20_Delay(45);
    return dat;
}
/**
  * 函数功能: 从DS18B20读一个字节，低位先行
  * 输入参数: 无
```

```
  * 返 回 值: 读到的数据
  * 说    明: 无
  */
static uint8_t DS18B20_ReadByte(void)
{
    uint8_t i, j, dat = 0;
    for(i=0; i<8; i++)
    {
        j = DS18B20_ReadBit();
        dat = (dat) | (j<<i);
    }
    return dat;
}
/**
  * 函数功能: 写一个字节到DS18B20, 低位先行
  * 输入参数: dat: 待写入数据
  * 返 回 值: 无
  * 说    明: 无
  */
static void DS18B20_WriteByte(uint8_t dat)
{
    uint8_t i, testb;
    DS18B20_Mode_Out_PP();

    for( i=0; i<8; i++ )
    {
        testb = dat&0x01;
        dat = dat>>1;
        /* 写0和写1的时间至少要大于60us */
        if (testb)
        {
            DS18B20_Dout_LOW();
            /* 1us < 这个延时 < 15us */
            DS18B20_Delay(8);

            DS18B20_Dout_HIGH();
            DS18B20_Delay(58);
        }
        else
        {
            DS18B20_Dout_LOW();
            /* 60us < Tx 0 < 120us */
            DS18B20_Delay(70);

            DS18B20_Dout_HIGH();
            /* 1us < Trec(恢复时间) < 无穷大*/
            DS18B20_Delay(2);
        }
    }
}

/**
  * 函数功能: 跳过匹配 DS18B20 ROM
  * 输入参数: 无
  * 返 回 值: 无
  * 说    明: 无
  */
```

```
static void DS18B20_SkipRom ( void )
{
    DS18B20_Rst();
    DS18B20_Presence();
    DS18B20_WriteByte(0XCC);         /* 跳过 ROM */
}
/**
  * 函数功能: 执行匹配 DS18B20 ROM
  * 输入参数: 无
  * 返 回 值: 无
  * 说    明: 无
  */
static void DS18B20_MatchRom ( void )
{
    DS18B20_Rst();
    DS18B20_Presence();
    DS18B20_WriteByte(0X55);         /* 匹配 ROM */
}
/*
 * 存储的温度是16 位的带符号扩展的二进制补码形式
 * 当工作在12位分辨率时，其中5个符号位，7个整数位，4个小数位
 *
 *         |---------整数----------|-----小数 分辨率 1/(2^4)=0.0625----|
 * 低字节  | 2^3 | 2^2 | 2^1 | 2^0 | 2^(-1) | 2^(-2) | 2^(-3) | 2^(-4) |
 *
 *
 *         |-----符号位：0->正  1->负-------|-----------整数-----------|
 * 高字节  |  s  |  s  |  s  |  s  |    s   |   2^6  |   2^5  |   2^4  |
 *
 *
 * 温度 = 符号位 + 整数 + 小数*0.0625
 */
/**
  * 函数功能: 在跳过匹配 ROM 情况下获取 DS18B20 温度值
  * 输入参数: 无
  * 返 回 值: 温度值
  * 说    明: 无
  */
float DS18B20_GetTemp_SkipRom ( void )
{
    uint8_t tpmsb, tplsb;
    short s_tem;
    float f_tem;
    DS18B20_SkipRom ();
    DS18B20_WriteByte(0X44);                    /* 开始转换 */
    DS18B20_SkipRom ();
    DS18B20_WriteByte(0XBE);                    /* 读温度值 */
    tplsb = DS18B20_ReadByte();
    tpmsb = DS18B20_ReadByte();
    s_tem = tpmsb<<8;
    s_tem = s_tem | tplsb;

    if( s_tem < 0 )         /* 负温度 */
        f_tem = (~s_tem+1) * 0.0625;
    else
        f_tem = s_tem * 0.0625;
    return f_tem;
```

```
}
/**
  * 函数功能: 在匹配 ROM 情况下获取 DS18B20 温度值
  * 输入参数: ds18b20_id：用于存放 DS18B20 序列号的数组的首地址
  * 返 回 值: 无
  * 说    明: 无
  */
void DS18B20_ReadId ( uint8_t * ds18b20_id )
{
    uint8_t uc;

    DS18B20_WriteByte(0x33);        //读取序列号
    for ( uc = 0; uc < 8; uc ++ )
      ds18b20_id [ uc ] = DS18B20_ReadByte();
}
/**
  * 函数功能: 在匹配 ROM 情况下获取 DS18B20 温度值
  * 输入参数: ds18b20_id：存放 DS18B20 序列号的数组的首地址
  * 返 回 值: 温度值
  * 说    明: 无
  */
float DS18B20_GetTemp_MatchRom ( uint8_t * ds18b20_id )
{
    uint8_t tpmsb, tplsb, i;
    short s_tem;
    float f_tem;
    DS18B20_MatchRom ();            //匹配ROM
  for(i=0;i<8;i++)
        DS18B20_WriteByte ( ds18b20_id [ i ] );
    DS18B20_WriteByte(0X44);                    /* 开始转换 */
    DS18B20_MatchRom ();            //匹配ROM
    for(i=0;i<8;i++)
        DS18B20_WriteByte ( ds18b20_id [ i ] );
    DS18B20_WriteByte(0XBE);                    /* 读温度值 */
    tplsb = DS18B20_ReadByte();
    tpmsb = DS18B20_ReadByte();

    s_tem = tpmsb<<8;
    s_tem = s_tem | tplsb;

    if( s_tem < 0 )         /* 负温度 */
        f_tem = (~s_tem+1) * 0.0625;
    else
        f_tem = s_tem * 0.0625;
    return f_tem;
}
```

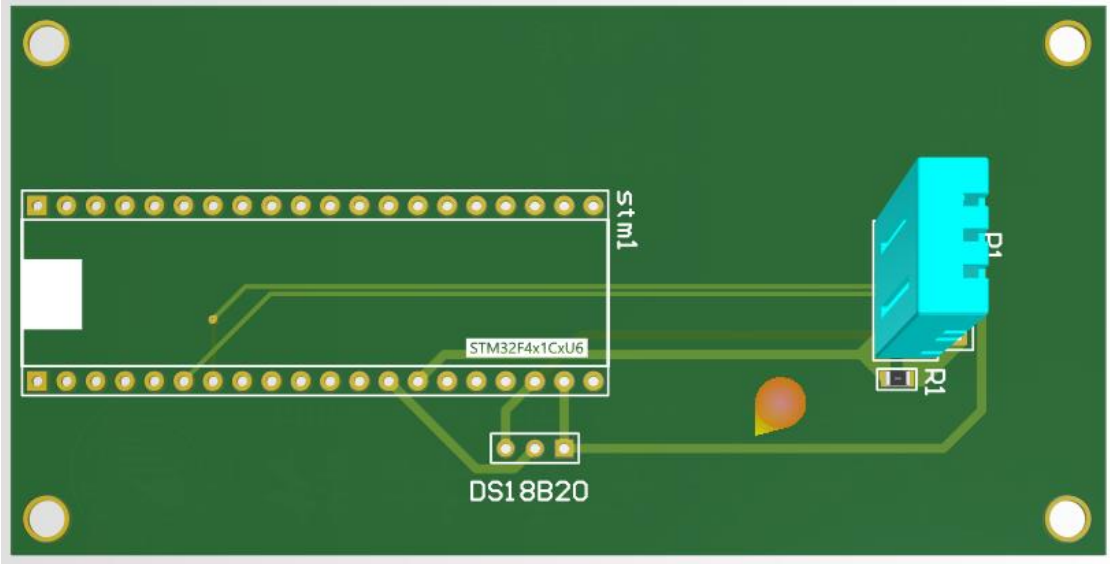
4　系统测试(调试)

4.1 硬件调试

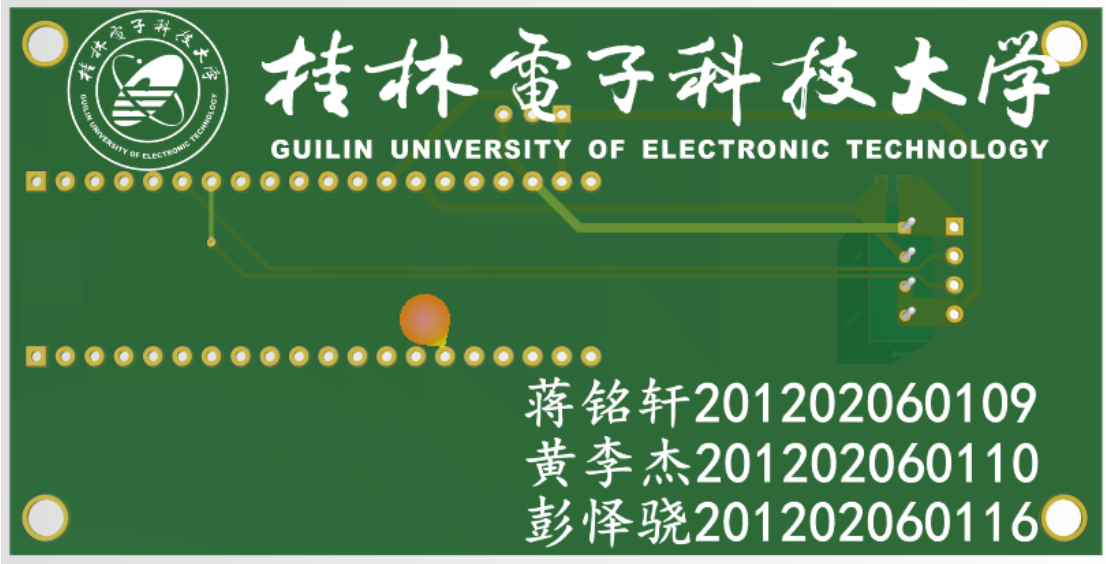
4.1.1 BOM 表

选中的	位号	注释	Current F...	设计项目ID	部件数量	图纸名
	串口屏	CKP	CKP	CKP	1	Sheet1.SchDoc
	DS18B20	Header 3	HDR1X3	Header 3	1	Sheet1.SchDoc
	P1	Header 4	HDR1X4	Header 4	1	Sheet1.SchDoc
	U1	DHT11	HT11A	DHT11	1	Sheet1.SchDoc
	R1	10K	R 0805_L	R	1	Sheet1.SchDoc
	stm1	U?	stm32f4x1cxu6_	stm32f4x1cxu6_core	1	Sheet1.SchDoc

4.1.2 PCB 三维效果图

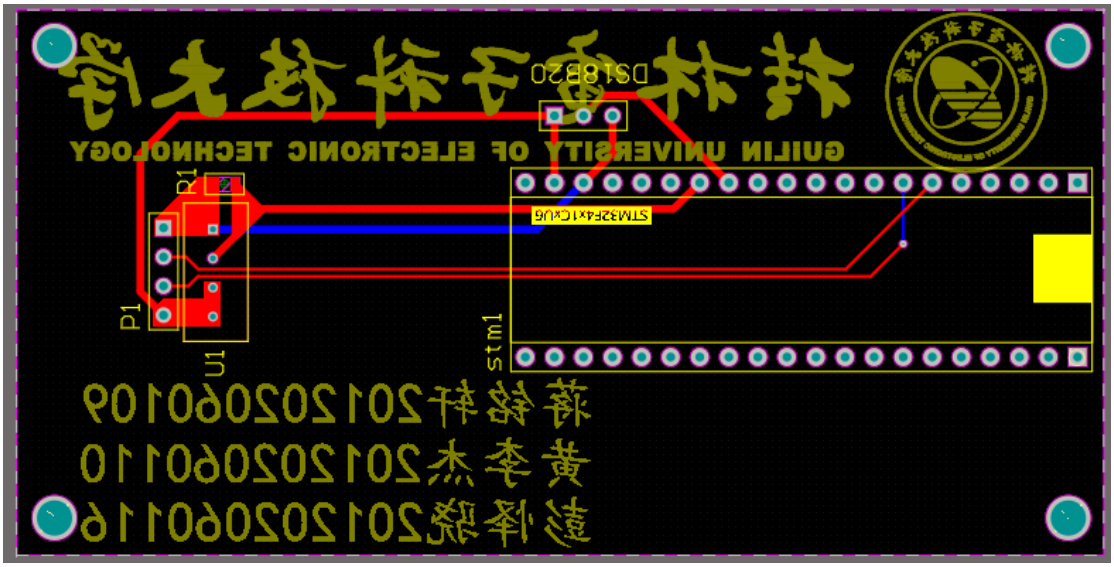


正面

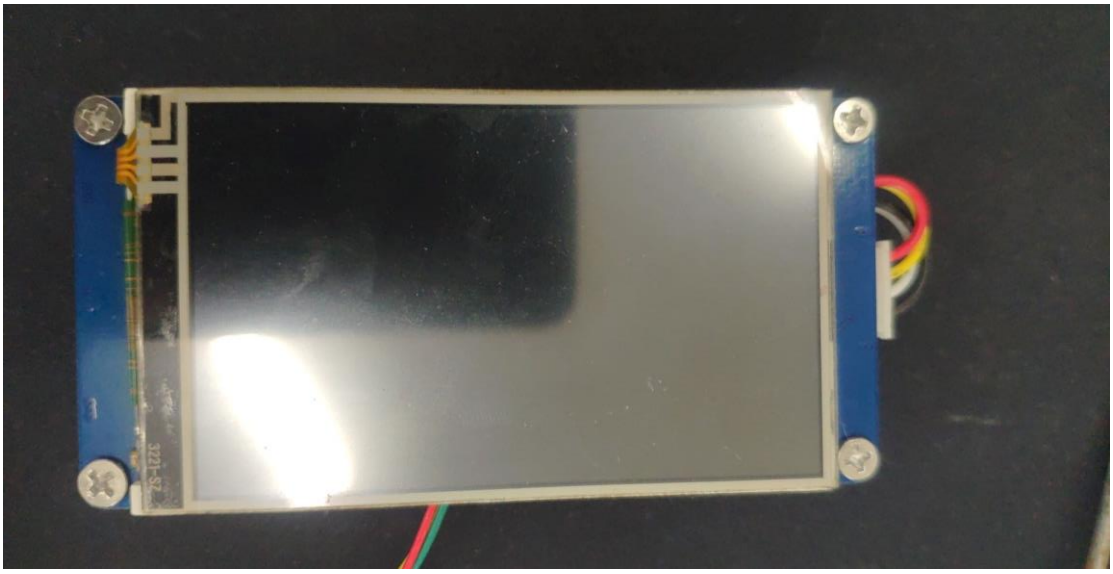


反面

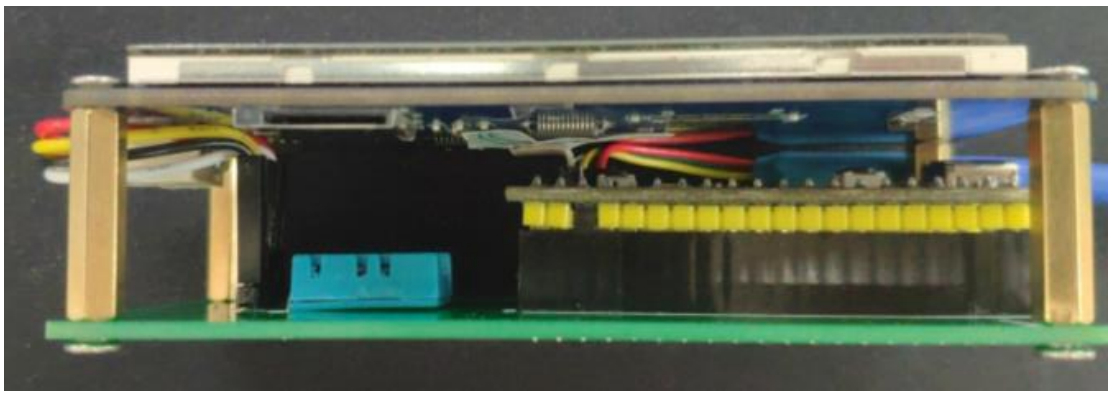
4.1.3 PCB 线路图



4.1.4 实物图



正面



侧面

反面

4.2 系统联调

4.2.1 模块测试

（1）电源模块 正常

（2）STM32F401CCU6 最小模块 正常

（3）LED 模块 正常

（4）DS18B20 模块 正常

（5）DHT11 模块 正常

4.2.2 整机测试

5V 供电下，整体电流为 103.5mA，温湿度显示正常，曲线记录显示正常。整机测试成功。

4.3 测试效果及结果分析

在 LCD 触摸屏上显示液体温度和环境温湿度。LCD 触摸屏能实时记录液体温度和环境温湿度共 3 种数据曲线并显示

5 实训总结

本设计主要是温湿度监控，。但是本设计还有许多可以改进和扩展的地方，比如增加温湿度传感器，设计成多路温湿度监控，还可以添加亚克力外壳和可充电电池供电使本设计更具有使用意义。

智能控制实训即将结束，回顾这几个星期来的辛勤工作，这是我在大二下学期学习阶段重要的一个环节，是对所学基础知识和专业知识的一种综合应用，是一种综合的再学习、再提高的过程，这一过程有助于培养我的学习能力和独立工作能力。我选的题目是温度计，这个题目对于我而言是一个全新的挑战。好多知识没有学过，但是通过自学和到图书馆查阅资料，最后还是把问题

解决了，不让自己带着问题遗憾的提交作品。另外这个题目也包含了许多平时所学的知识，利用在大二下学期，最重要的时间里，把所学知识活学活用，这也是对大二下学期的一次总结，一次汇报。我在完成设计时，把平时所学的知识应用到了设计中，虽然在实训设计的过程中存在许多问题，但通过自己不断的查阅书籍和上网查找资料，最后所有困难都迎刃而解。这对于培养我们的自学能力和独立工作能力是非常有帮助的。

通过本次实训设计，我感到自己应用基础知识及专业知识解决问题的能力有了很大的提高，并且这次实训设计的选题，是一个实际的项目工程，因此，是在我即将大三之前，它是一次重要演练。我想，通过这次实训设计，到了工作单位后，我将能够更快的适应工作岗位和工作要求。我对自己充满信心。总之，这次实训设计对我而言是受益匪浅的。

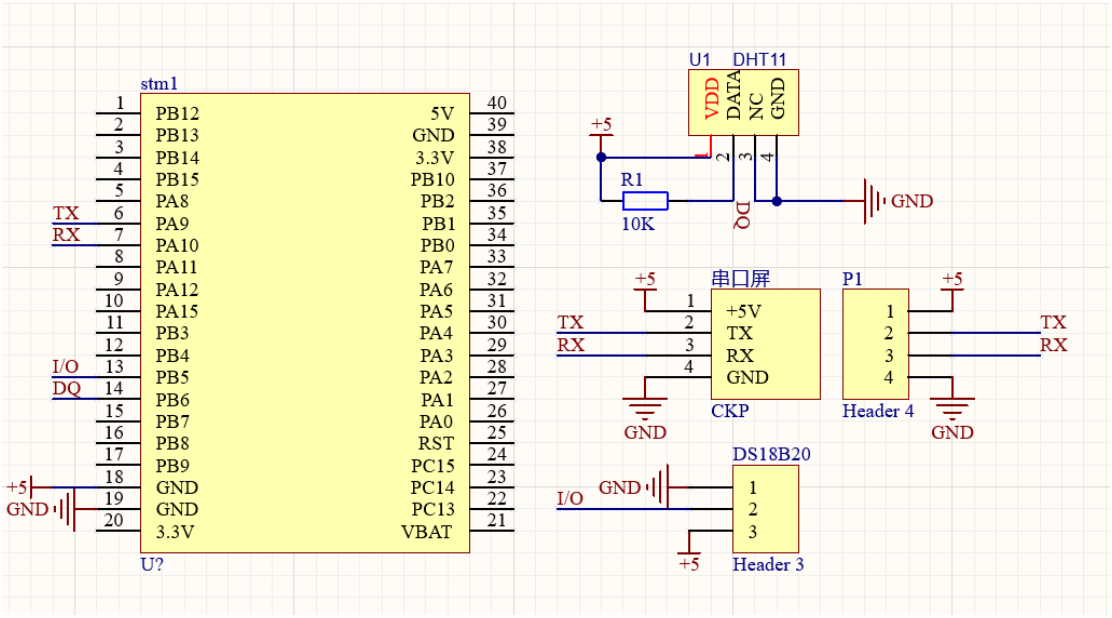
参 考 文 献

[1] 刘火良 杨森.基于 STM32+STM32 库开发实战指南.北京：机械工业出版社，2019

[2] Twistzz karrigan. STM32F4xx Chinese Reference Manual,2016

附　录

1 系统整机原理图



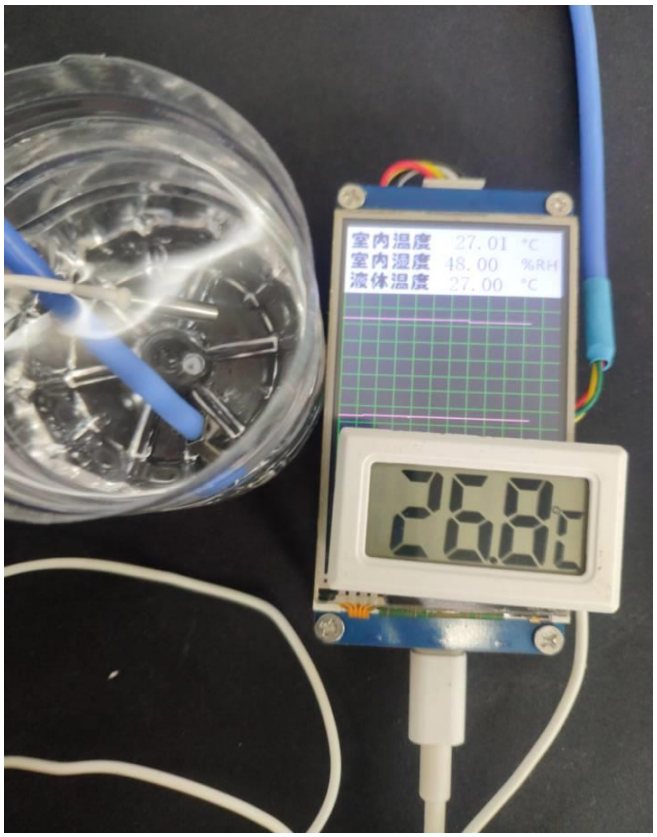
2 系统 PCB 图

3 BOM 表

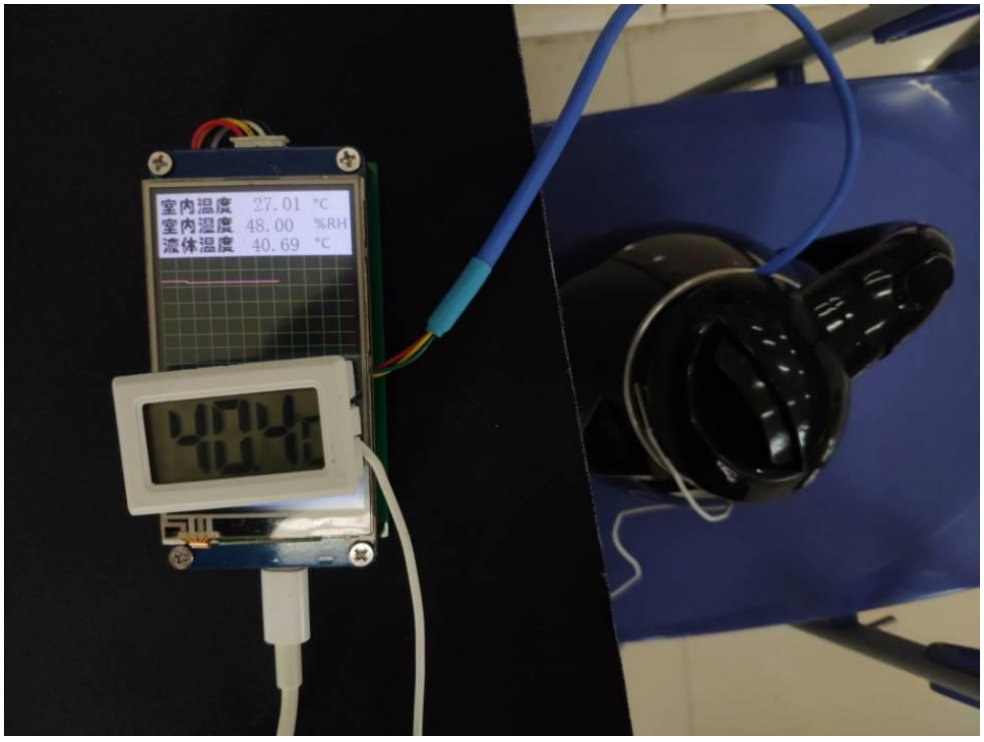
选中的	位号	注释	Current F...	设计项目ID	部件数量	图纸名
	串口屏	CKP	CKP	CKP	1	Sheet1.SchDoc
	DS18B20	Header 3	HDR1X3	Header 3	1	Sheet1.SchDoc
	P1	Header 4	HDR1X4	Header 4	1	Sheet1.SchDoc
	U1	DHT11	HT11A	DHT11	1	Sheet1.SchDoc
	R1	10K	R 0805_L	R	1	Sheet1.SchDoc
	stm1	U?	stm32f4x1cxu6_	stm32f4x1cxu6_core	1	Sheet1.SchDoc

4 数据测试

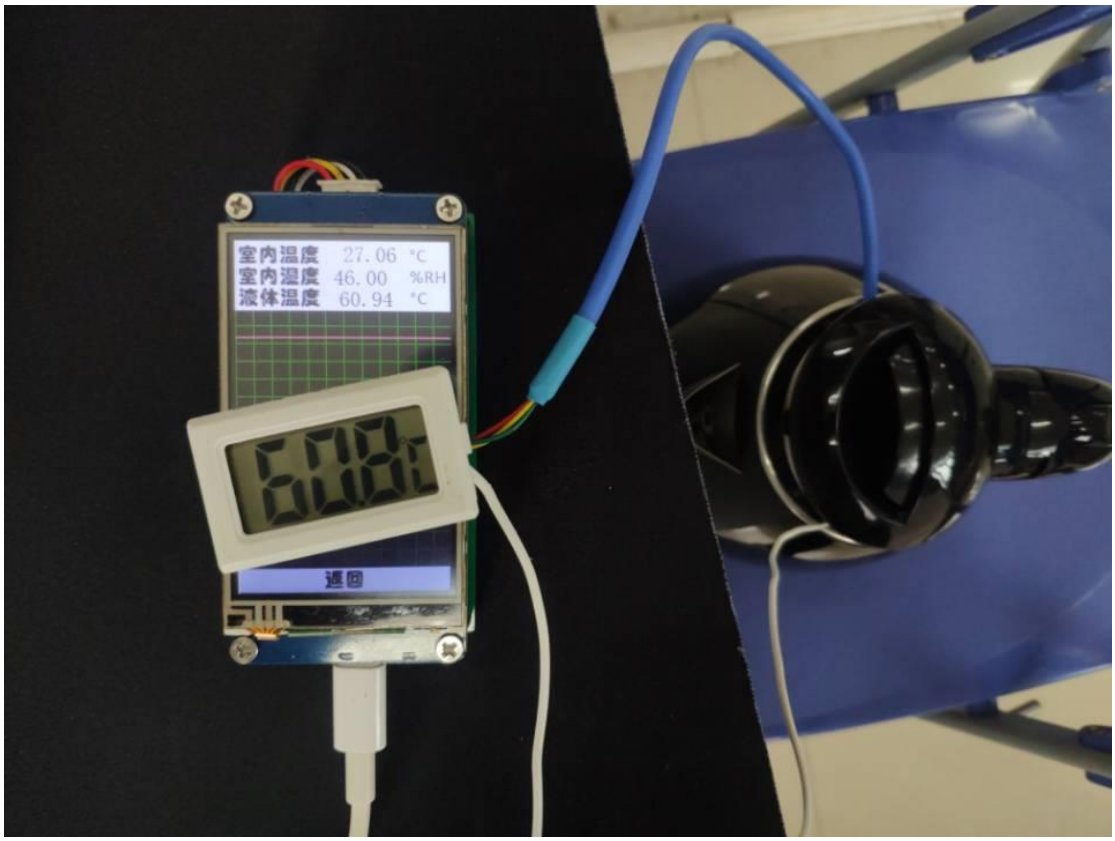
我们取一杯凉水，取一杯热水，在测量过程中我们逐渐加入热水，并记录了温度，我们对测量的温度做了拍摄，并截取了几张做为介绍。



常温



加热 2 分钟热水

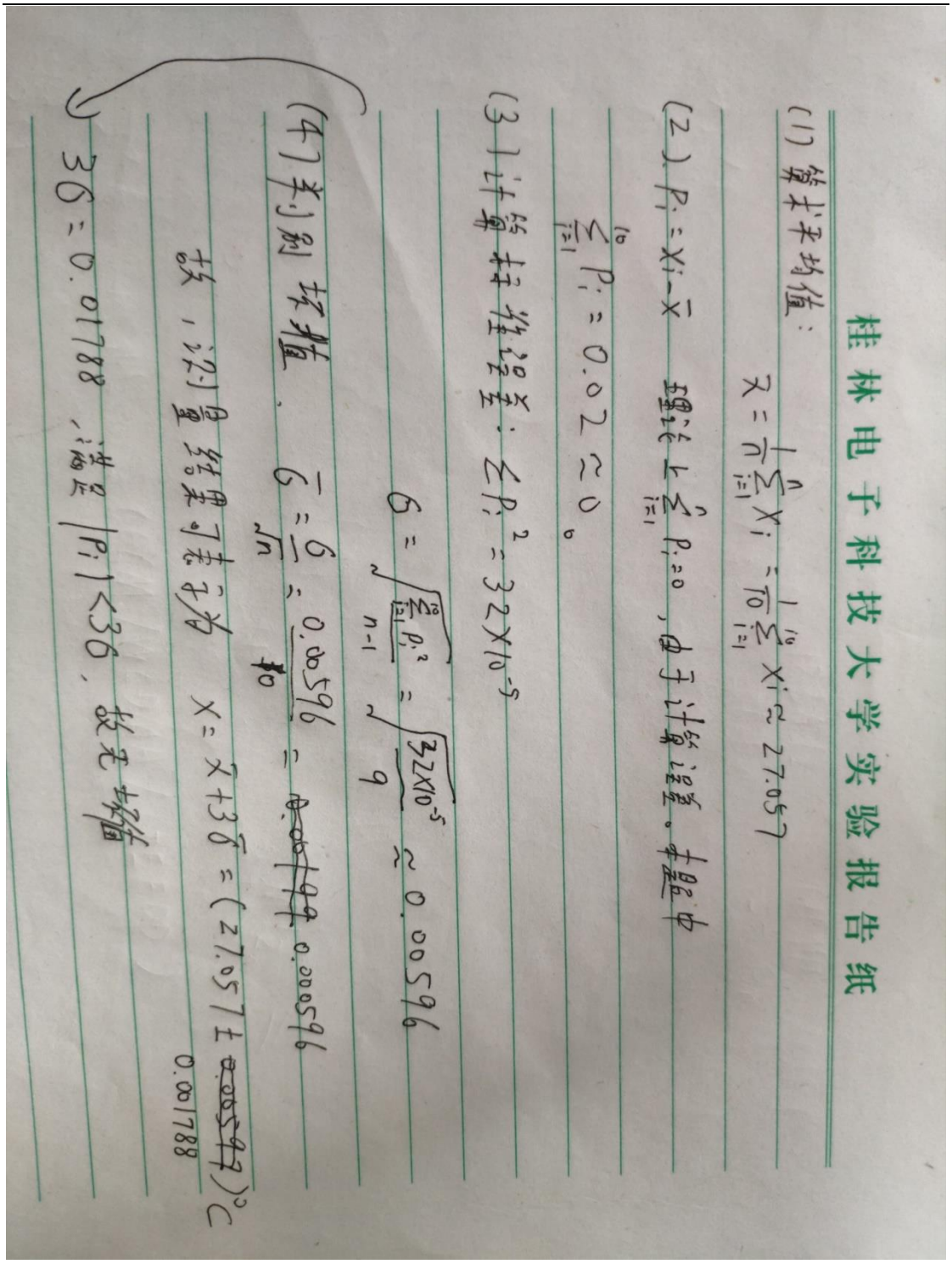


加热 4 分钟热水

为了进一步证明我们的实验误差很小，我们取了晚上某一时刻温水的数据，我们把等精度测量的数据记录以及处理分析结果做了如下表格：

序号	Xi	Pi	Pi^2
1	27.05	-0.01	0.001
2	27.03	-0.03	0.009
3	27.04	-0.02	0.004
4	27.06	0	0
5	27.07	0.01	0.001
6	27.07	0.01	0.001
7	27.08	0.02	0.004
8	27.06	0	0
9	27.07	0.01	0.001
10	27.04	-0.02	0.004
合计	算术平均值：27.057	误差： 0.02	0.00032

计算过程如下



故，经过分析我们此次温度测量机误差较小，符合设计要求。