专业：生物医学工程
姓名：陈宣宇
学号：3230103134
日期：2025 年 9 月
地点：

浙江大学 实验报告

课程名称：电路综合创新实践 指导老师：刘雪松 成绩：
实验名称：基于 STM32 和 MLX90614 的测温仪实现 实验类型：设计制作型
专业：生物医学工程 姓名：陈宣宇 学号：3230103134

一、实验背景和内容

1.1 实验背景

本实验要求完成基于温度传感器 MLX90614 实现的测温仪。MLX90614 系列模块是一组通用的红外测温模块。在出厂前该模块已进行校验及线性化，具有非接触、体积小、精度高，成本低等优点。

本实验旨在开发一款以高性能 STM32 微控制器为核心、以 MLX90614 红外温度传感器为感知单元的非接触式测温仪。该设计不仅是对现代测温技术的一次具体实践，更是一个典型的物联网终端设备原型，涵盖了数据采集、数据处理、人机交互等嵌入式系统开发的核心环节，可以在疫情防控（体温筛查）、工业故障诊断（设备过热预警）、家电产品（智能空调、微波炉）、食品安全检测以及可穿戴设备等领域进行应用。

1.2.实验内容

完成基于 MLX90614 温度传感器进行的测温仪设计，包括完成电路硬件设计及软件控制设计两大部分。

二、实验原理和思路

2.1 芯片的选择

不同于本项目一贯选择的 STC89C52RC 单片机，我采用 STM32F103C8T6 作为主控芯片进行数据的采集和处理，选择原因如下：

①电赛培训期间一直在使用 STM32 开发板进行实验，对 STM32 和 ARM 架构的使用更加熟悉。

②在软件方面，STM32F103C8T6 可以利用 STM32cubeMX 图形化配置寄存器，分配引脚的功能，配置硬件 I2C、SPI 等协议，相对传统的 52 单片机效率更高，并且可以极大缩短项目开发周期，降低后续维护和功能升级的难度。

③在硬件方面，STM32F103C8T6 的主频高达 72MHz，相对 52 单片机的 11.0592 MHz 而言，速度更快，处理数据的能力更强，为未来功能扩展留足了空间。

④在调试方面，STM32F103C8T6 可以利用 ST-LINK（SWD/JTAG 接口）进行在线烧写调试，极大缩短调试时间，而 52 单片机只能通过串口打印调试，效率极低。

2.2 总体实验原理

MLX90614 将接收到的红外温度信号通过内部放大器和 ADC，转换为 17 位数字信号，通过 IIC 协议发送到主控芯片的 RAM，主控芯片处理温度信号并将温度信号通过 IIC（或 SPI 协议）发送到显示屏上。此外我们还设计了基于 HC05 设计了蓝牙通讯模块，能在手机 APP 上实时获取温度数据。

我们小组先在开发板上对软件进行测试，再在自己的板子上测试。

2.4 实验思路和方案

经过我们小组三人的讨论，我们基于不同的硬件资源和传输协议得出了多种不同的实验方案，并通过 STM32 开发板上的实验验证了几种方案的可行性。

几种方案主要不同点在于：

1.显示屏的选取，我们小组尝试对比了四种显示屏：LCD1602、4 引脚 0.96 寸 OLED(SSD1306）、7 引脚 0.96 寸 OLED、TFTLCD。如下图 1-4

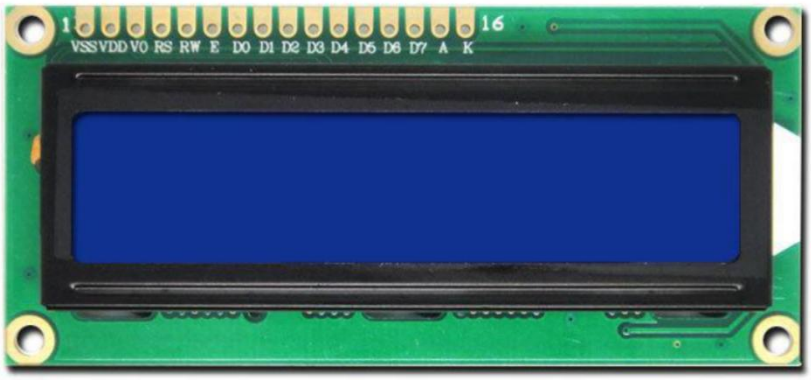


图 1　LCD1602

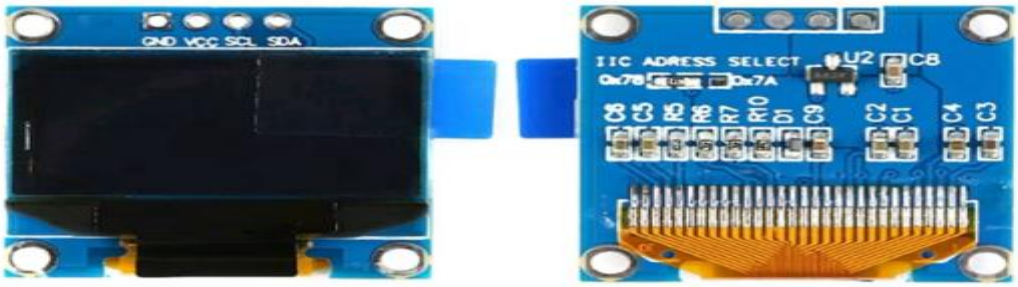


图 2　4 脚 0.96 寸 OLED

图 3　7 脚 0.96 寸 OLED

图 4　TFTLCD

	LCD1602	4 脚 0.96 寸 OLED	7 脚 0.96 寸 OLED	TFTLCD
显示内容	仅能显示字符/数字	单色位图/图形/汉字	单色位图/图形/汉字	图片、照片、彩色图标，能够构建复杂的用户界面（UI），且可支持触摸屏
视觉效果	需要背光，视觉效果差	128*64 分辨率，对比度高，视觉效果好	128*64 分辨率，对比度高，视觉效果好	视觉效果最好
引脚资源	并行 16 引脚，占用引脚多	4 引脚，占用引脚最少，有利于 PCB 布局且可留出引脚便以后扩展功能	7 引脚，占用引脚较少，有利于 PCB 布局且可留出引脚便以后扩展功能	32 引脚，引脚太多，占用过多资源
成本	低	中	中	高
体积	大	小	小	大
硬件接口	GPIO 模拟并行	1.MCU 内置 I2C 2.软件模拟 I2C	1.MCU 内置 I2C 2.软件模拟 I2C 3.MCU 内置 SPI	1.MCU 内置 I2C 2.软件模拟 I2C 3.MCU 内置 SPI 4.MCU 内置 FMC

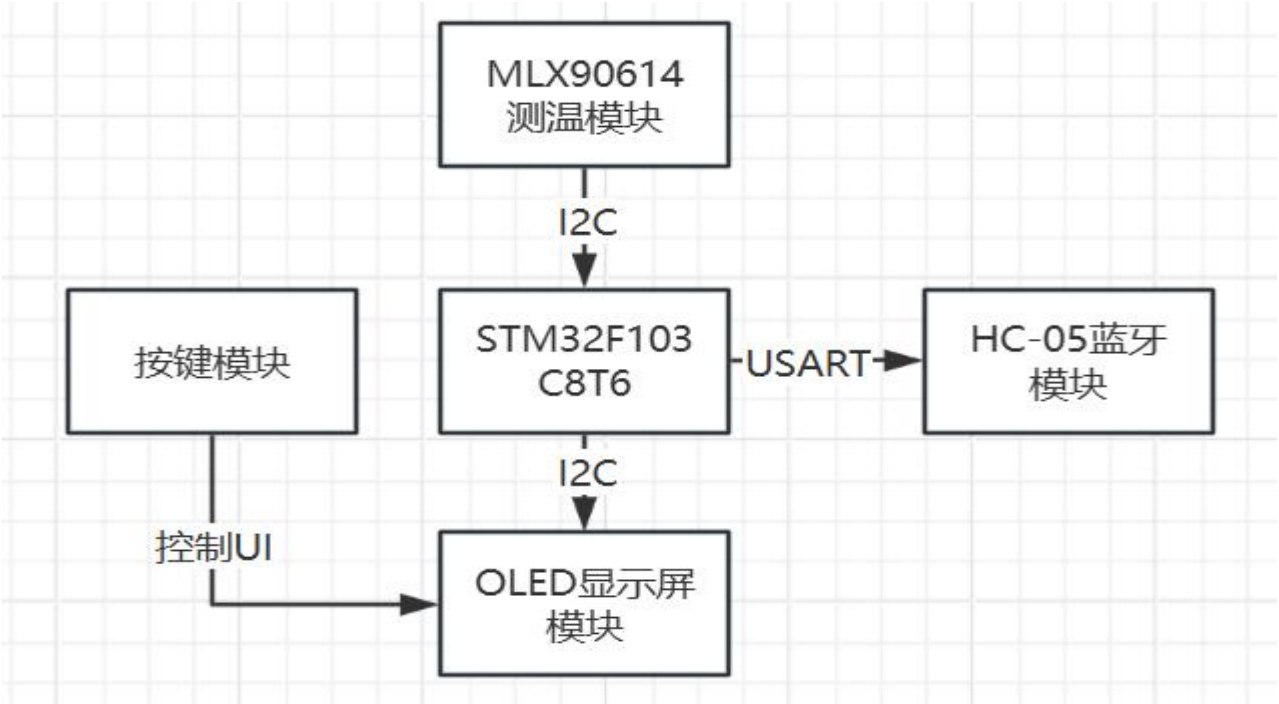
表 1 四种显示屏的优缺点对比

通过表 1 的分析，为了更好地显示效果并方便未来扩展，我最后选择了 4 脚 0.96 寸 OLED(OLED1305）。

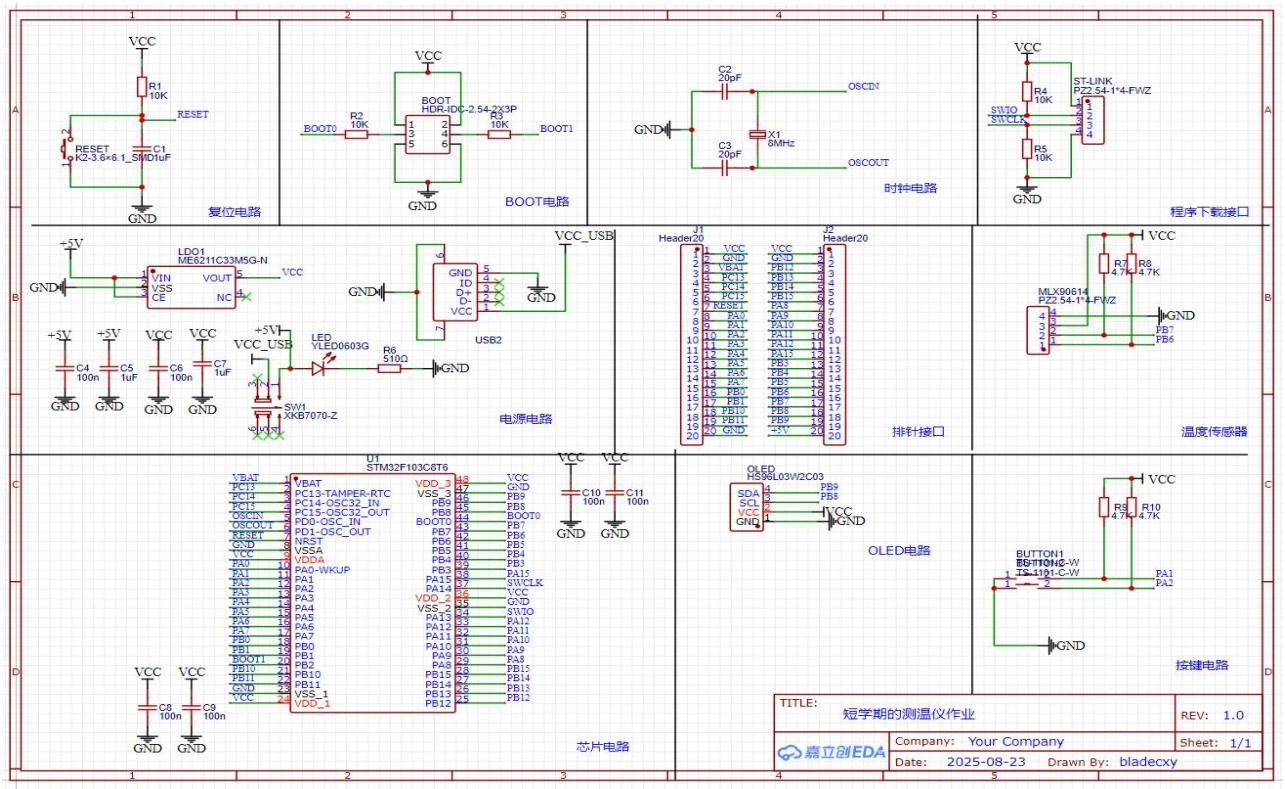
2. **显示屏的硬件接口**，我们小组通过在 STM32F103C8T6 上的实验得知，使用 MCU 内置 I2C 很容易出现卡顿的情况，数据传输一半就传输不过去了，于是我最后采用软件模拟 I2C 的方案（将引脚配成普通 GPIO 口，利用软件模拟 I2C），而小组内的另外两位同学则使用了 MCU 内置 SPI 与 7 脚 0.96 寸 OLED 通信，代码上更为简单。

三、电路原理设计和分析

基本架构：

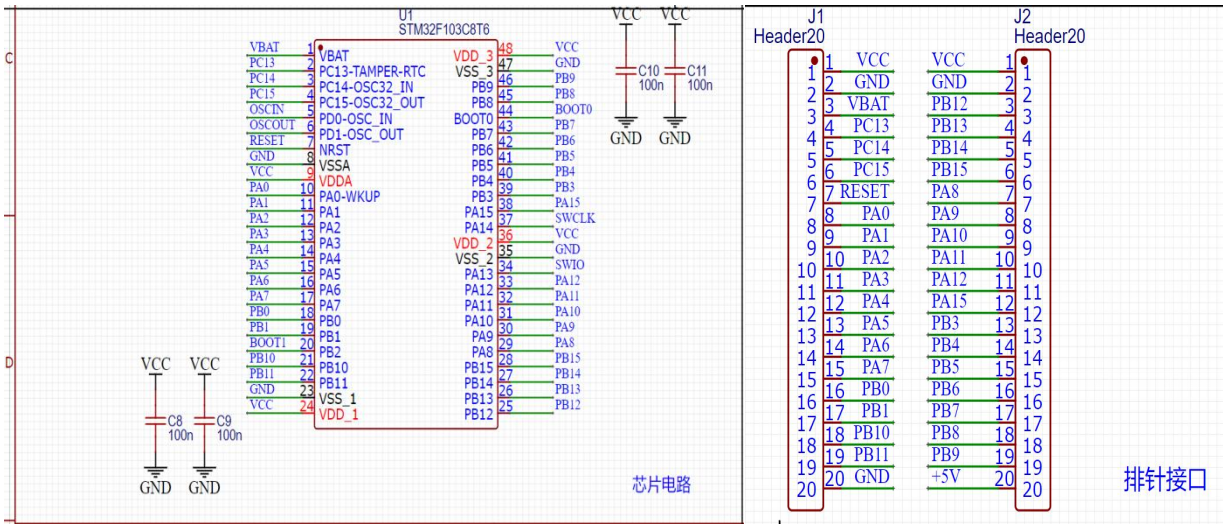


原理图总体布局：



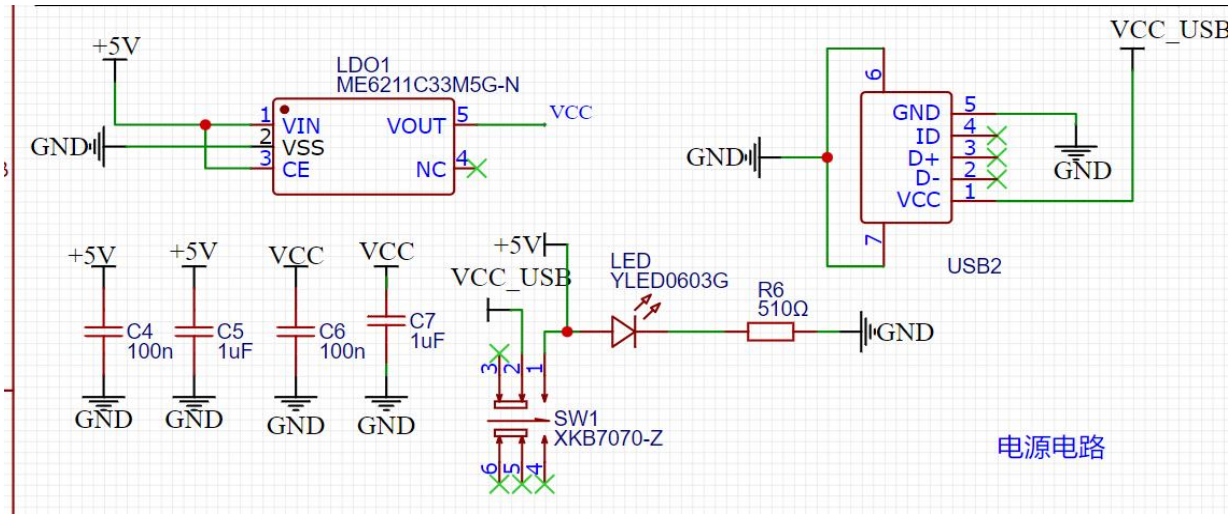
芯片模块：

本测温仪采用 STM32F103C8T6 芯片进行数据的采集与处理，芯片连接到排针方便连接外部扩展。100nF 电容为退耦电容。



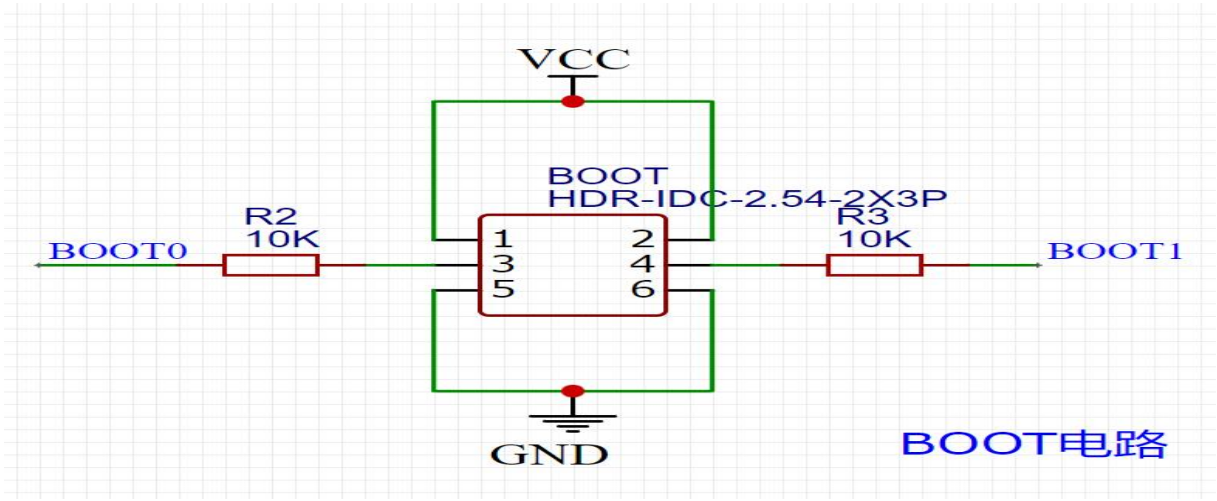
电源控制模块：

电源控制模块由四部分组成。1. MicroUSB 连接到外部电源。2. 电源按键控制整个系统是否上电，LED 灯以指示系统是否正常上电。3. ME6211C33M5G 芯片将 5V 电压转换为 3.3V 电压。4. 四个电容均为退耦电容，起滤波作用，进一步减少电源供电信号的噪声。



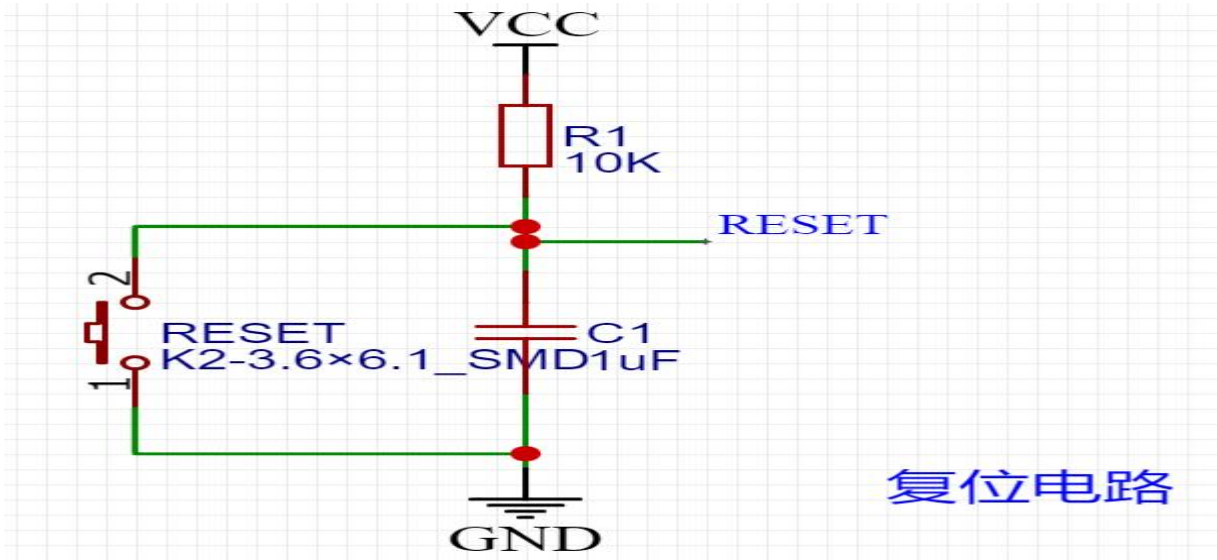
BOOT 电路：

STM32F103C8T6 的启动模式由 BOOT0 和 BOOT1 引脚的电平组合决定，控制芯片上电或复位后的代码执行来源。分别有主 Flash 启动模式（BOOT0=0）、系统存储器启动模式（BOOT0=1，BOOT1=0）、内置 SRAM 启动模式（BOOT0=1，BOOT1=1），我们一般采用主 Flash 启动模式（BOOT0=0）。



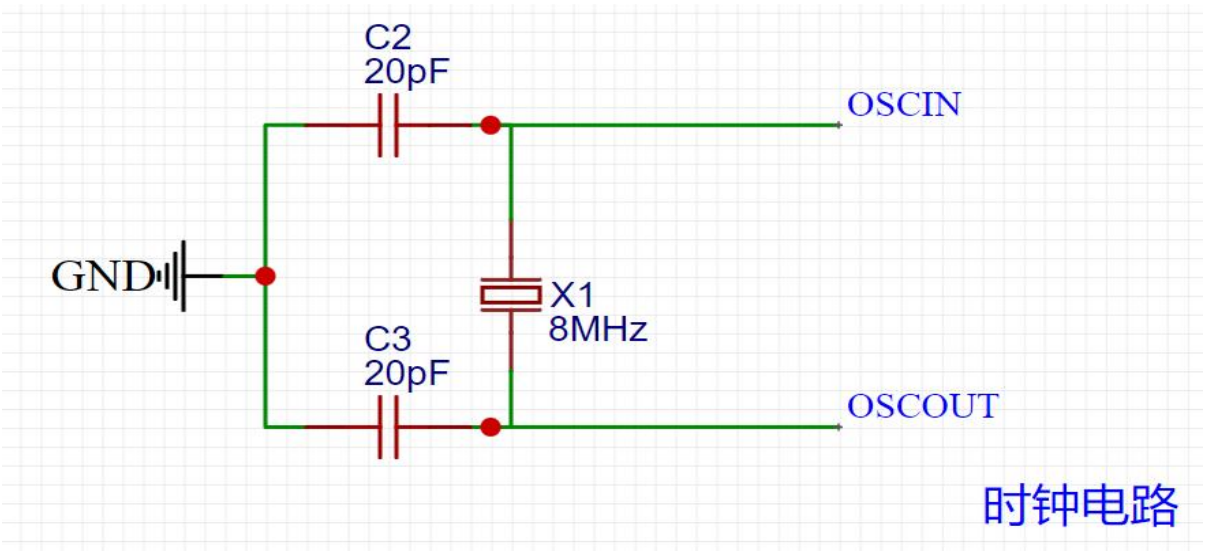
复位电路：

按下 RESET 按键即可复位。



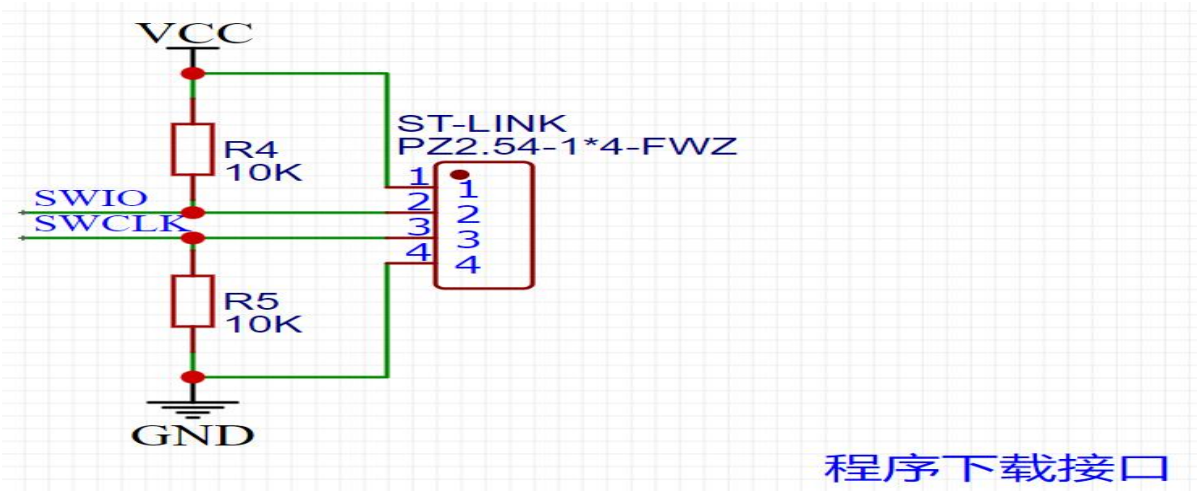
时钟电路：

STM32F103C8T6 采用 8MHZ 的晶振，并通过内部 PLL 将外部 8MHz 晶振倍频到 72MHz 系统时钟。



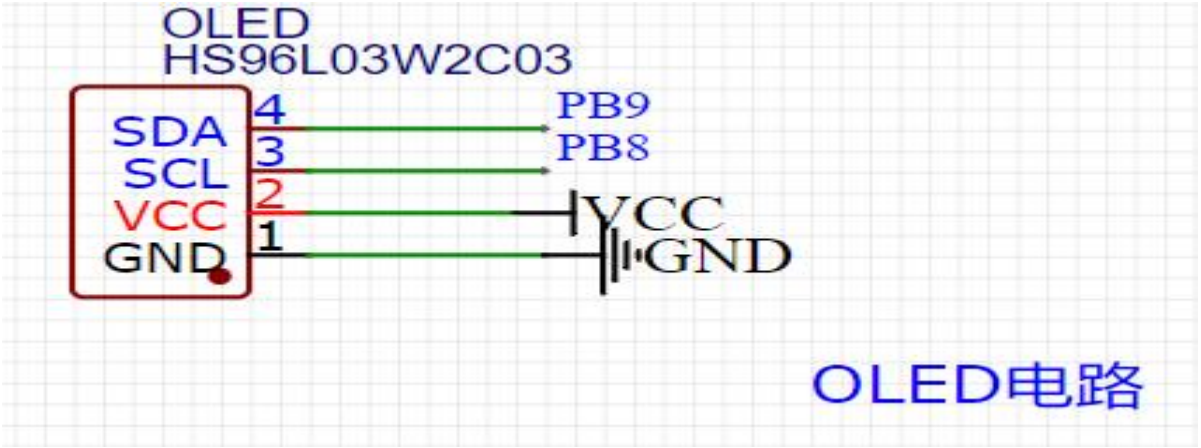
程序下载接口：

STM32F103C8T6 采用 ST-Link 烧写，可支持在线调试。



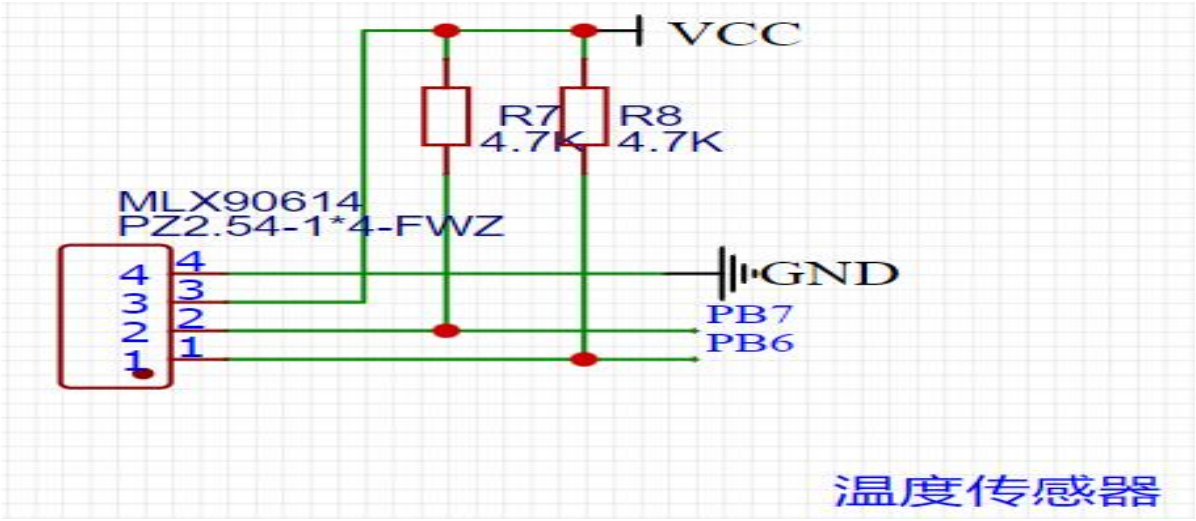
显示模块：

显示模块通过 4 脚 0.96 寸 OLED 液晶显示器实现，采用 I2C 协议通信，PB9 口作数据线 SDA，PB8 作控制线 SCL。



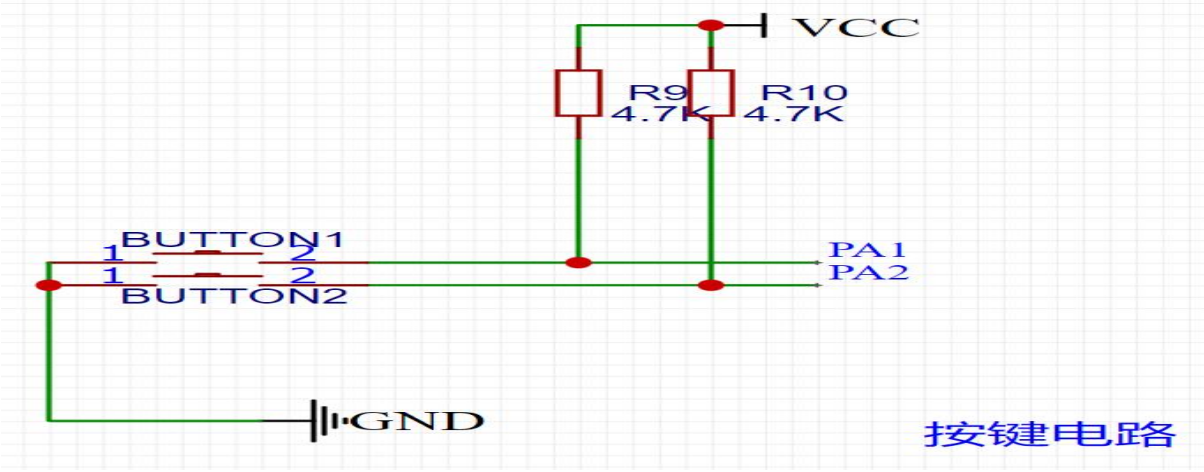
传感器模块：

传感器模块由 MLX90614 温度传感器构成，使用 I2C 方式进行通信。芯片引脚配置为开漏输出，无上下拉电阻，所以我们要人为地给 SCL、SDA 两根线接上拉电阻，确保没有设备主动拉低信号时，信号线处于高电平状态。而后为了方便测温，我们将 MLX90614 通过杜邦线连接到板子上。



按键模块：

芯片外接到两个按键，从而实现程序可控，为后续软件调试预留可观察空间。



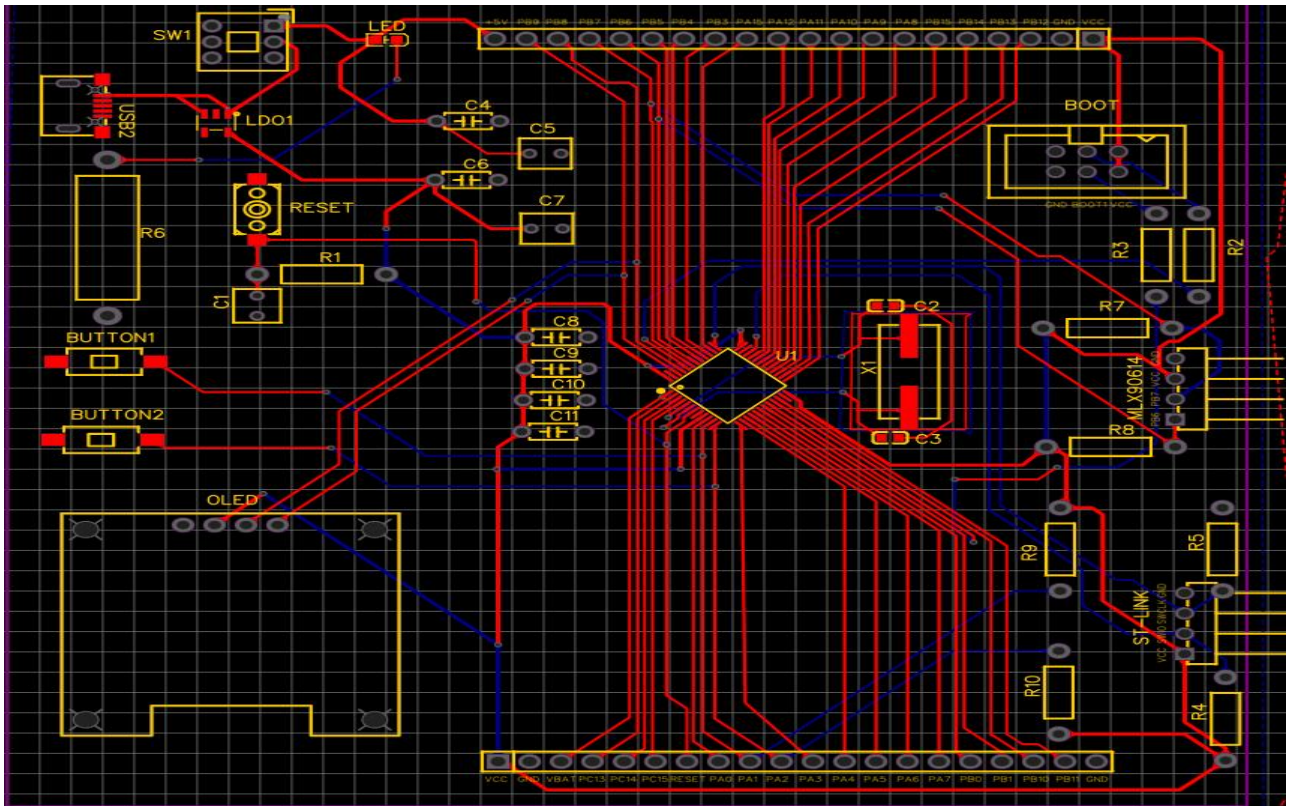
蓝牙模块：

蓝牙模块采用 HC-05 模块，通过杜邦线与板子连接，并通过串口进行通讯。

四、PCB 设计

通过原理图导出物理模型，放置元器件后进行连线，将所有模块安置在适当位置后并连接完毕得到 PCB 图。在设计过程中尽量完成 PCB 走线要求，包括布局总连线尽可能短，关键信号线最短；高电压、大电流信号与低电压、小电流的弱信号尽可能分开等等。此外，晶振需靠近芯片。

PCB 布局：

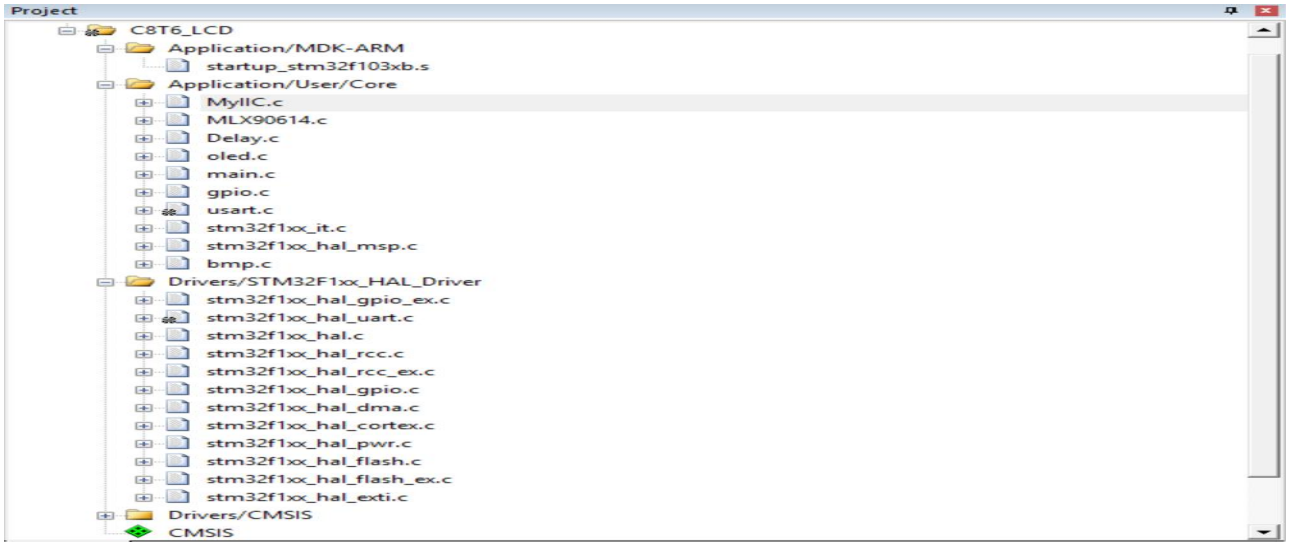


五、软件设计思路

软件主要分为 CubeMX 生成的初始化配置代码、MLX90614 测温驱动程序、OLED 屏 UI 程序、按键控制程序、蓝牙通讯程序。

5.1 软件架构

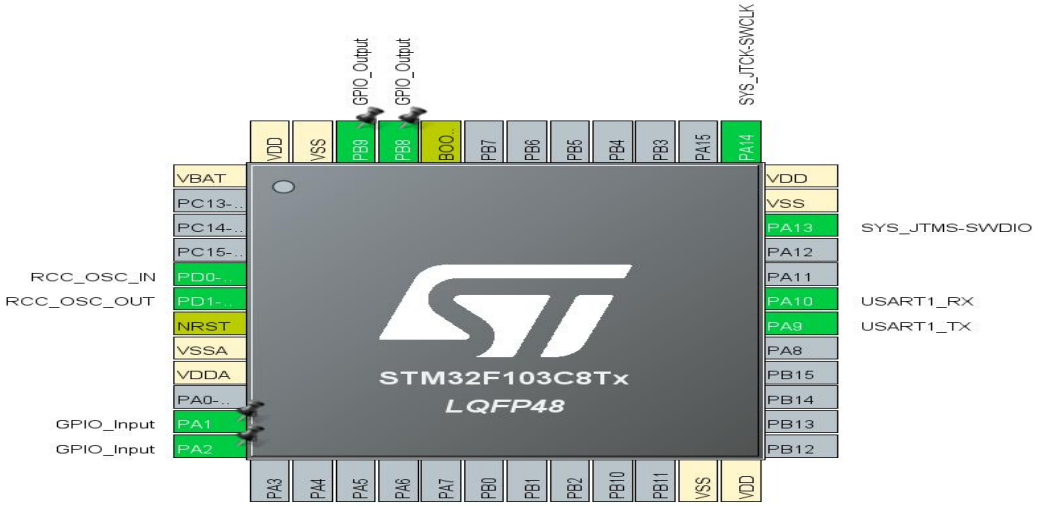
MyIIC.c 软件模拟 I2C。MLX90614.c 温度传感器驱动。Delay.c 延时程序。oled.c 显示屏驱动程序。main.c 主程序（测温算法和 UI 控制）。gpio.c 引脚驱动。usart.c 串口驱动。bmp.c 开机动画矩阵。



5.2CubeMX 配置

STM32CubeMX 是 ST（意法半导体）官方提供的一款图形化工具。不需要手动编写繁琐的寄存器配置代码，只需在软件界面上通过点击和选择来配置单片机。

总体配置：

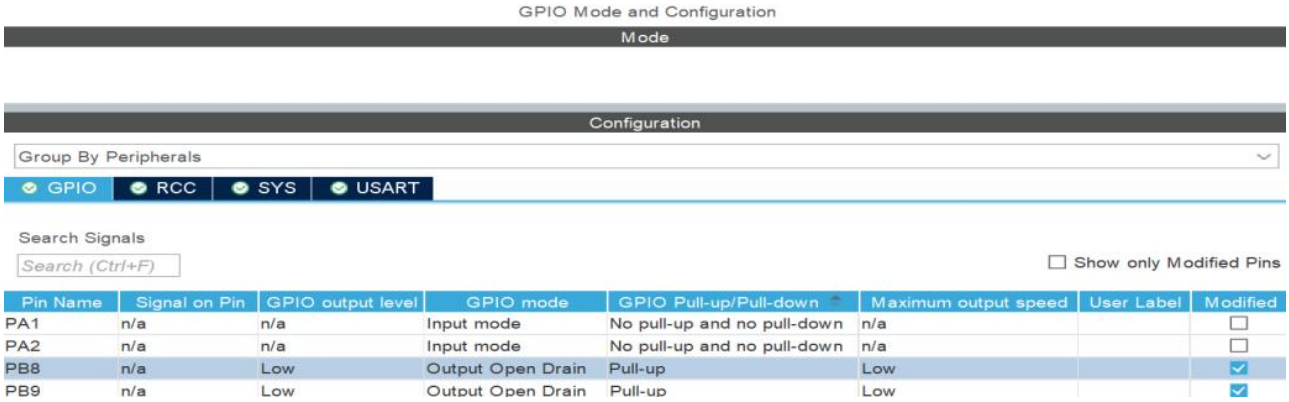


GPIO 配置：

PA1 和 PA2 连接到两个按钮，故设置为输入模式、无上下拉电阻。

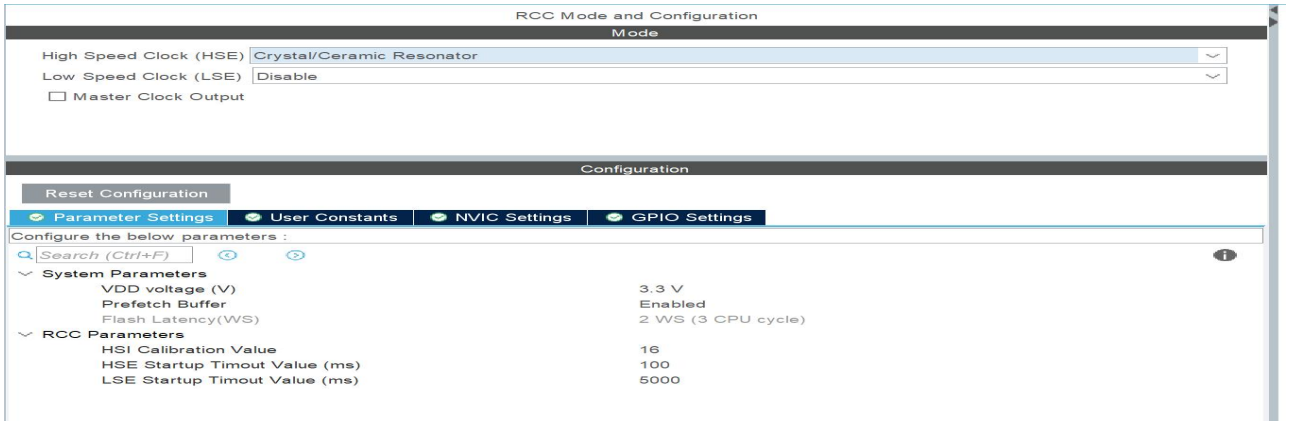
PB8 连接到 OLED 屏幕的 SCL，PB9 连接到 OLED 的 SDA，由于要 I2C 协议通讯，所以将两个引脚设置为开漏输出（I2C 协议规定要使用开漏输出，因为当多个主设备与多个从设备在一条总线上， 如果不用开漏输出， 而用推挽输出， 会出现主设备之间短路的情况）并接内部上拉电阻（使开漏输出具有输出高电平的能力）。

MLX90614 测温的驱动程序是从网站上移植下来的，由于驱动程序内部配好了引脚，我在 CubeMX 上就不配置了，由于是 I2C 协议传输，其配置方式也是开漏输出并外接上拉电阻。



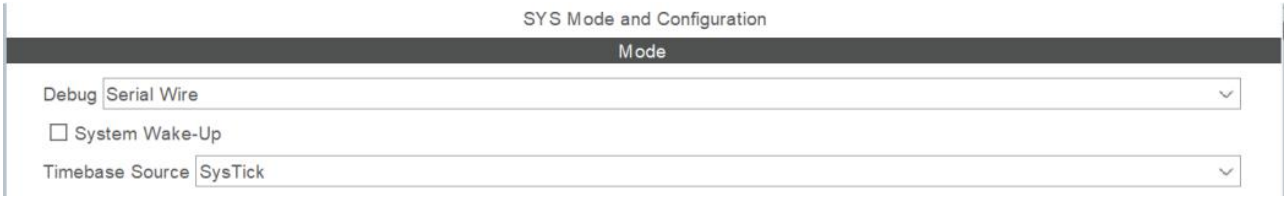
RCC（时钟来源）配置：

采用外部无源陶瓷晶振作为时钟源。



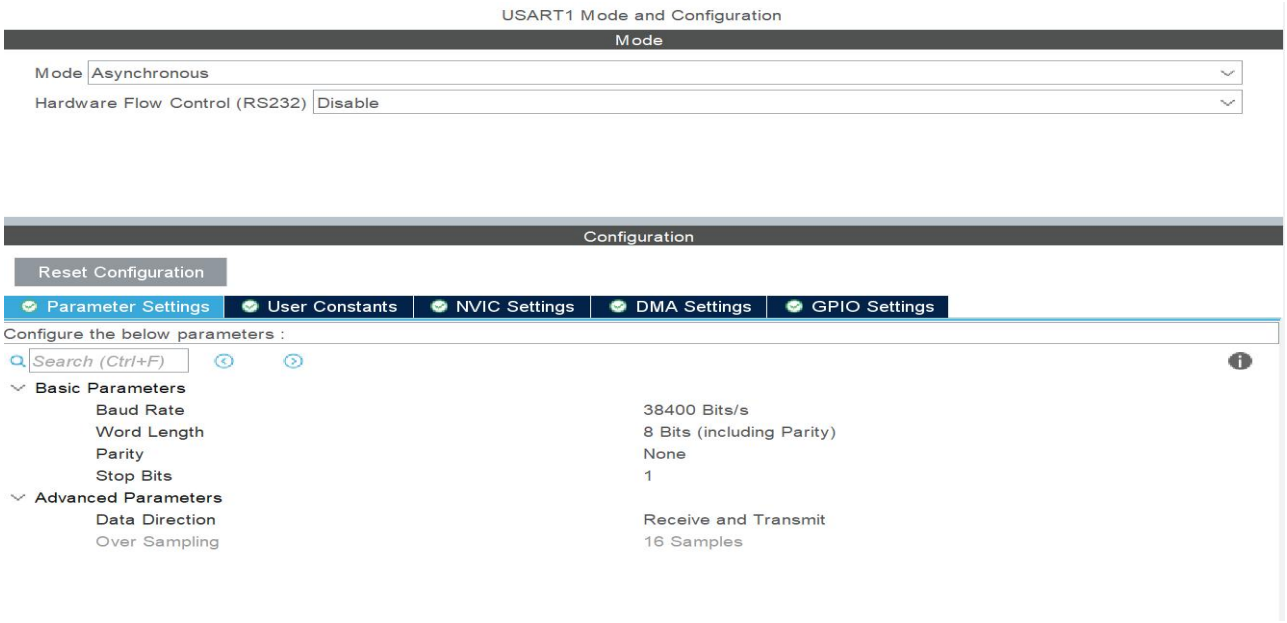
SYS 配置：

将调试方式设置为串行调试（Serial Wire Debug），即允许 4 引脚 ST-Link 在线调试，十分方便。



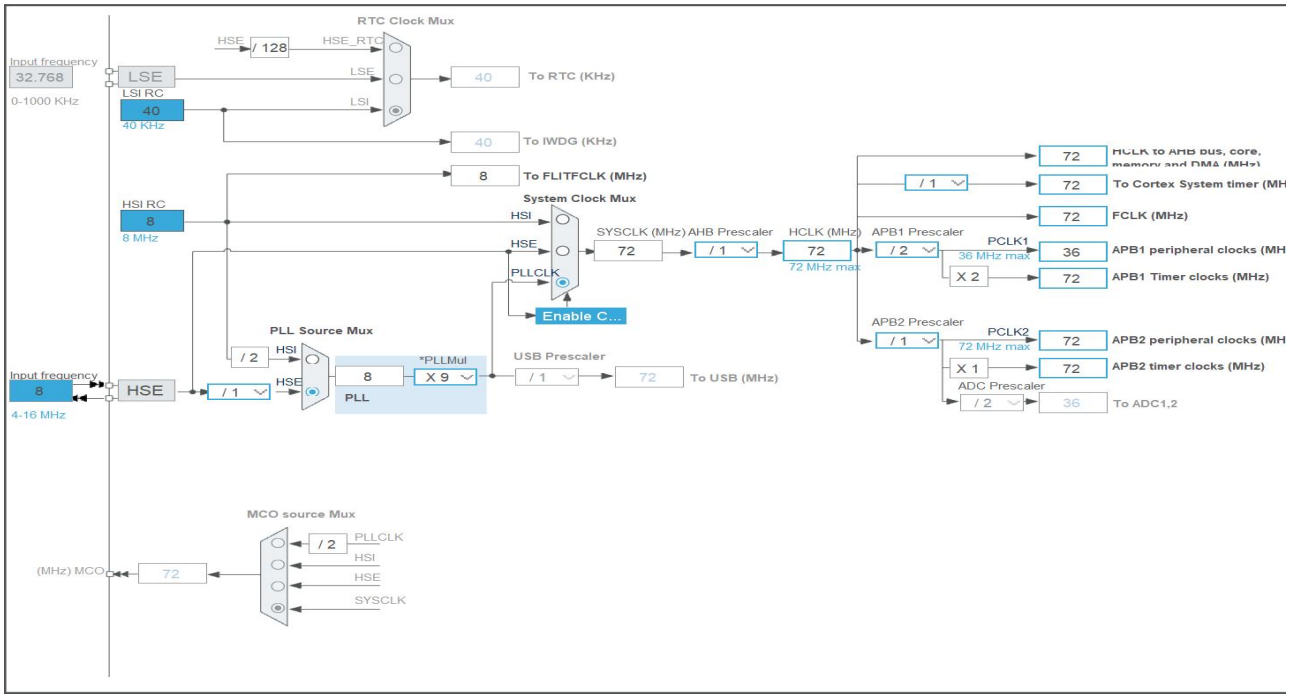
USART 配置：

将 USART1 串口设置为异步通信，波特率为 38400 方便与蓝牙模块通信，同时 CubeMX 会自动将串口配置为 PA9(TX）、PA10(RX）。



时钟树配置：

将主频设置为 72MHz（通过 PLL 将 8MHz 的外部时钟倍频为 72MHz）。



5.3MLX90614 测温驱动程序：

5.3.1 I2C 通讯

该模块内部传感器芯片被连接成 I2C 方式进行通讯，查询器件手册可知，在与传感器进行通讯时，单片机需要送出开始信号和中止信号，同时接收来自传感器发出的 ACK 和数据信号，具体表现为：

开始信号：SDA 下降沿，SCL 高电平转为低电平；

终止信号：SDA 低电平时，SCL 产生上升沿；

应答信号：在完成收发数据后，SCL 下降沿，此时 SDA 即为应答信号，其中低电平表示 ACK，高电平表示 NACK。

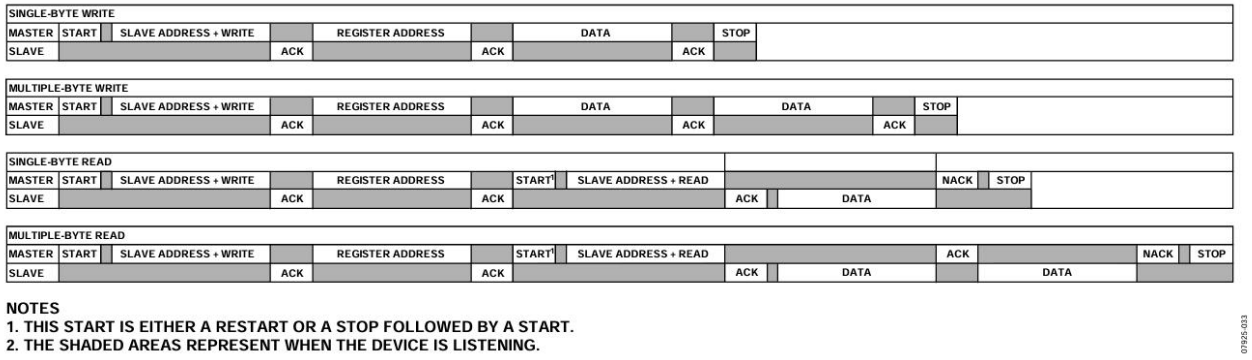


图40. I²C器件寻址

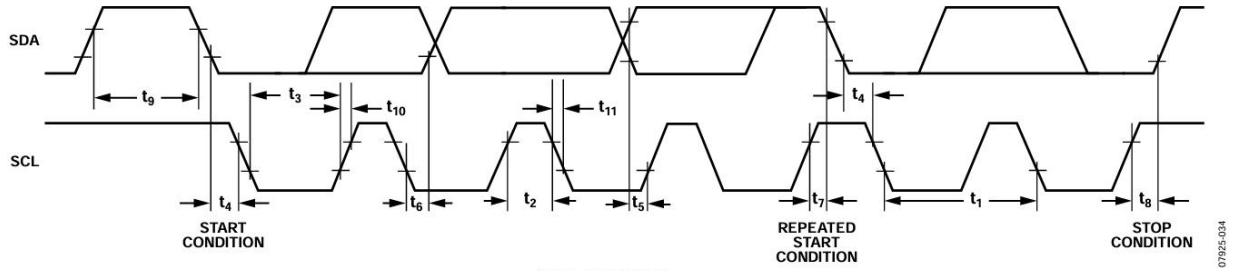


图41. I²C时序图

5.3.2 MLX90614 测温驱动

在具体实验过程中，编写对应的驱动程序可能相对麻烦，但对于每个器件模块，均有厂家提供的例程包括已封装好的驱动程序。在实际代码操作过程中，合理地调用相关函数可以帮助我们更好更快地完成目标需求，驱动函数如下图（MLX90614.h）。

```
#ifndef __MYI2C_H
#define __MYI2C_H

#include <stdint.h>

void MyI2C_Init(void);
void MyI2C_Start(void);
void MyI2C_Stop(void);
void MyI2C_SendByte(uint8_t Byte);
uint8_t MyI2C_ReceiveByte(void);
void MyI2C_SendAck(uint8_t AckBit);
uint8_t MyI2C_ReceiveAck(void);

#endif
```

5.3.3 测温算法

2.乘以 2：i * 2 - 将原始数据转换为实际的热力学温度值（开尔文）的 100 倍

3.减去 27315：- 27315 - 将开尔文转换为摄氏度（273.15K = 0°C），因为数据是 100 倍放大的，所以用 27315

4.除以 100：* 0.01 - 将 100 倍放大的数据还原为实际温度值

5.4 OLED 屏 UI 程序、按键控制程序

OLED 也是用 I2C 通讯，所以也可移植例程中的驱动

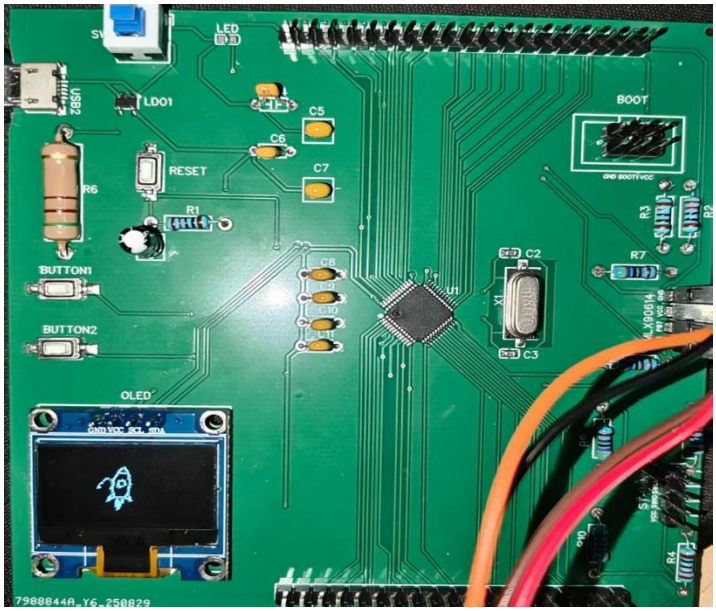
5.4.1 OLED 屏主要显示函数

```
#ifndef __OLED_H
#define __OLED_H


#include "main.h"


void OLED_Init(void);
void OLED_Clear(void);
void OLED_ShowChar(uint8_t Line, uint8_t Column, char Char);
void OLED_ShowString(uint8_t Line, uint8_t Column, char *String);
void OLED_ShowNum(uint8_t Line, uint8_t Column, uint32_t Number, uint8_t Length);
void OLED_ShowSignedNum(uint8_t Line, uint8_t Column, int32_t Number, uint8_t Length);
void OLED_ShowHexNum(uint8_t Line, uint8_t Column, uint32_t Number, uint8_t Length);
void OLED_ShowBinNum(uint8_t Line, uint8_t Column, uint32_t Number, uint8_t Length);
void OLED_DrawGIF(unsigned char x0, unsigned char y0, unsigned char x1,
                  unsigned char y1, unsigned char k, unsigned int m,
                  unsigned char GIF[][m]);
#endif
```

5.4.2 开机动画

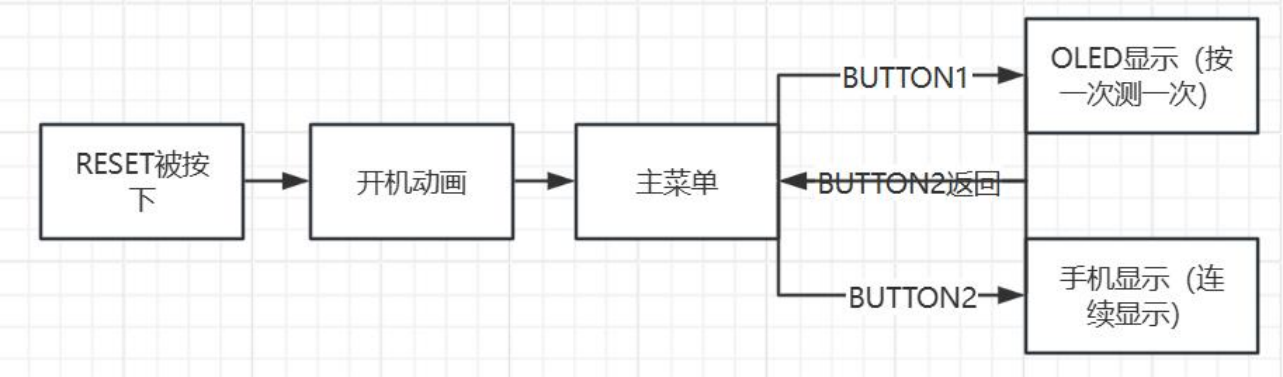
将图片数据硬编码到程序中：开机动画的每一帧图像都被预先转换为十六进制的像素数据，并存储在一个名为 Rocket 的二维数组中。

主程序调用 OLED_DrawGIF() 函数逐帧显示动画。显示效果如下



5.4.3 按键控制 UI 界面

显示逻辑如下图



显示效果如下图

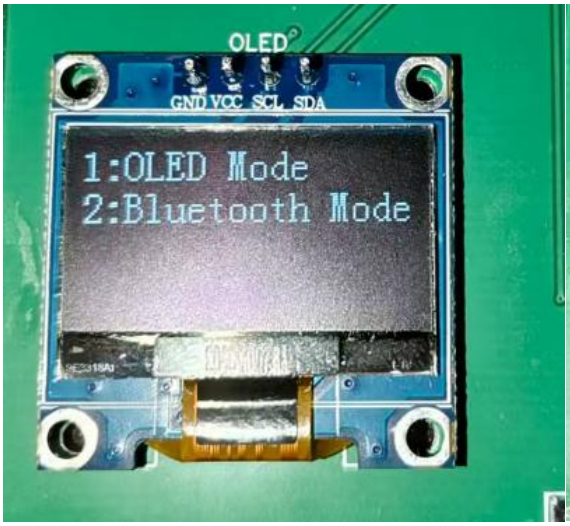


图 1 主菜单

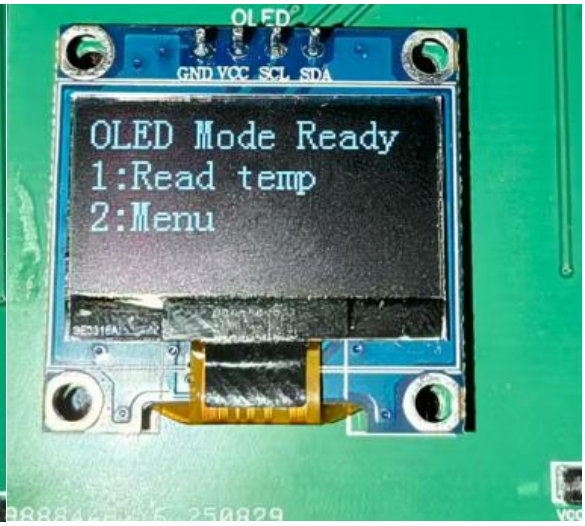


图 2 按键测温模式

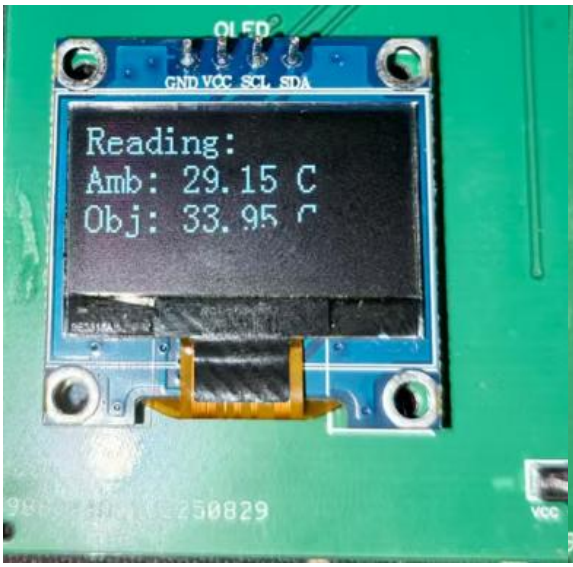


图 3 按键测温模式效果

图 4 蓝牙测温模式

5.5 蓝牙通讯程序

将 PA10（RX）接到蓝牙模块的 TX，PA9（TX）接到蓝牙模块的 RX，然后用 HAL 库内置的 HAL_UART_Transmit() 函数，将温度数据发到串口，并通过蓝牙转发到手机上，代码如下。

```
else if (mode == 2) //Bluetooth mode
{
      OLED_Clear(); //Initialize
      MLX90614_TA();
      Delay_ms(20);
      MLX90614_TO();
      Delay_ms(20);

      object_temperature = o_temp;
      snprintf(object_temp_str, sizeof(object_temp_str), "temp: %.2f C\r\n", object_temperature);

      HAL_UART_Transmit(&huart1, (uint8_t *)object_temp_str, strlen(object_temp_str), HAL_MAX_DELAY);   //Showing string on USART

      OLED_ShowString(1, 1, "Data Sent!");  //Showing string on screen
      Delay_ms(500);
}
```

六、设计过程中的问题

最终实验呈现结果较好，但仍存在许多可优化的点：

1. 电源电路的网络标签打错。按键两端的网络标签都被我打成了 VCC_USB，导致按键被短路（插上电源时直接上电，而不受按键控制），同时 5V 的引脚也无法使用。但由于板子已经打好，没法再修改，不过此错误对最终结果几乎没有影响。
2. 芯片自锁问题。CubeMX 配置时，DEBUG 未选择 Serial Wire，导致芯片锁住，程序烧录一次之后就无法再烧录进芯片。锁了好几块开发板的芯片，在网上查询很久之后，选择一直按住复位键，用 CubeMX Programmer 强制给芯片擦除 flash 并烧录一个正常的固件，再将正确程序烧录进芯片。
3. PCB 布局问题。我的电路板依靠 Micro-USB 供电，而布局时 Micro-USB 接口离 PCB 板过远，Micro-USB 无法插入。只能用刀将开发板边缘磨出一个缺口，才能让电源线完全插入。
4. 焊接问题。在焊接 Micro-USB 接口时，由于 VCC 和 GND 两端距离过近，焊接时一不小心短路，启动芯片时芯片就已经烧坏，只能换块板子重新焊。并且在焊接结束以后要及时检查各个焊点是否短路。
5. 蓝牙模块乱码问题。通过串口发送到蓝牙模块的数据在手机上是乱码的。我们先将板子的串口连接到电脑的串口，发现数据正常，说明硬件和软件没问题。我们又进入蓝牙模块的 AT 模式更改了许多次波特率，更换不同蓝牙模块进行测试，发现依旧是乱码，说明波特率没问题。我们修改了传输字符的编码方式，甚至自己重写了 HAL_UART_Transmit() 函数，发现依旧是乱码，我们认为是 STM32F103C8T6 向蓝牙模块传输数据的协议问题。只能模仿 51 每次发一个字节的模式传输数据。

七、体会和总结感想

本次基于 STM32 和 MLX90614 的测温仪实验，我完成了数据采集、处理和人机交互等核心环节，初步掌握了嵌入式系统开发的基本流程。通过调用和编写驱动程序，实现了非接触式测温的基本功能，并最终达到了较高的测量精度。

这次短学期的开发总体而言是比较顺利的，我们用 2 天在开发板上调试软件，用 4 天焊接电路并完成后续扩展功能设计（蓝牙，UI 界面）。我认为，我们在开发板上先调试软件方式是非常好的，能够较为独立地排查软件错误，节省较多时间。但是我们没有预先规划好所有拓展功能，导致后来硬件资源捉襟见肘，这是未来开发需要改进的。

在整个实践过程中，我将过去所学的知识应用到实际项目中，从零开始自主设计，根据需求完成了从硬件电路到软件控制的完整开发。我第一次系统性地接触了 I2C 等硬件通信协议，理解了驱动程序在硬件与软件交互中的关键作用，并独立完成了整个电路的硬件设计，包括绘制原理图、PCB 布局走线，以及之后的打样和焊接。这个过程不仅锻炼了我的动手实践能力，也让我更清楚地认识到理论与实践之间的差距，学会在实际操作中不断修正和完善设计。

此外，在项目中我也遇到了各种各样的挑战，包括硬件焊接短路、驱动程序编写逻辑错误等问题。在解决这些问题的过程中，我学会了如何系统性地查找资料、分析问题，并尝试多种方法来解决。这些宝贵的经验提升了我的自主学习能力和解决问题的能力，对于我未来的学习和工作都将非常有帮助。