注：居中，宋体，小一号，加黑。阅后删除此文本框。

**大连理工大学**

**自动化专业认识实习**

**基于STM32的简易激光雷达**

注：此处是论文中英文题目，中文题目，不超过20个汉字，居中，字体：华文细黑，加黑，字号：二号，行距：多倍行距1.25，间距：段前、段后均为0行，取消网格对齐选项。英文题目，与中文题目对应，居中，字体：Times New Roman，字号：三号，加黑，行距：多倍行距1.25，间距：段前、段后均为0行，取消网格对齐选项。阅后删除此文本框。

阅后删除此文本框。

注：此处按照实际情况填写即可。打印（宋体，小三）或手写都可以。阅后删除此文本框。

学 部：电子信息与电气工程学部

专 业：自动化

学 生 姓 名：朱思源

学 号：20202261013

指 导 教 师：

完 成 日 期：

大连理工大学

Dalian University of Technology

# 摘 要

注：页眉，居中，宋体，五号，论文中文题目。阅后删除此文本框。

“摘要”是摘要部分的标题，不可省略。

【

】

标题“摘要”选用模板中的样式所定义的“标题1”，再居中；或者手动设置成字体：黑体，居中，字号：小三，1.5倍行距，段后11磅，段前为0。

摘要是报告的缩影，文字要简练、明确。内容要包括项目背景、采用的技术方法、结论。单位采用国际标准计量单位制，除特别情况外，数字一律用阿拉伯数码。文中不允许出现插图。

摘要正文选用模板中的样式所定义的“正文”，每段落首行缩进2个汉字；或者手动设置成每段落首行缩进2个汉字，字体：宋体，字号：小四，行距：多倍行距 1.25，间距：段前、段后均为0行，取消网格对齐选项。

摘要篇幅以一页为限，字数为400-500字。

摘要正文后，列出3-5个关键词。“关键词：”是关键词部分的引导，不可省略。关键词请尽量用《汉语主题词表》等词表提供的规范词。

**本项目Github地址：<https://github.com/zsychina/little-project>**

关键词与摘要之间空一行。关键词词间用分号间隔，末尾不加标点，3-5个；黑体，小四，加粗。

关键词**：**STM32单片机；激光雷达；有机发光二极管；

注：字体：仿宋\_GB2312，小四；关键词之间用分号间隔。阅后删除此文本框。

目 录

注：在该页面中点击鼠标右键，选择“更新域…”，在弹出窗口中选择“更新整个目录”，确定即可自动生成目录。标题“目录”，字体：黑体，字号：小三。章、节标题和页码，字体：宋体，字号：小四。阅后删除此文本框。

[摘 要 I](#_Toc112539774)

[1 简易激光雷达的研发意义 1](#_Toc112539775)

[1.1项目背景和需求分析 1](#_Toc112539776)

[1.2项目研发意义 1](#_Toc112539777)

[2 简易激光雷达的功能与设计方案 2](#_Toc112539778)

[2.1 系统设计任务 2](#_Toc112539779)

[2.2 系统设计时考虑的因素 2](#_Toc112539780)

[2.3 系统设计方案 2](#_Toc112539781)

[3 简易激光雷达的软硬件设计 3](#_Toc112539782)

[3.1 硬件设计 3](#_Toc112539783)

[3.1.1激光测距模块MS53L1M简介与用法 3](#_Toc112539784)

[3.1.2 OLED模块简介与用法 4](#_Toc112539785)

[3.2 软件设计 5](#_Toc112539786)

[3.2.1软件设计思路 5](#_Toc112539787)

[3.2.2软件开发环境简介与用法 9](#_Toc112539788)

[3.2.3 OLED驱动程序 13](#_Toc112539789)

[3.2.4 数据可视化程序 19](#_Toc112539790)

[4 项目调试 24](#_Toc112539791)

[4.1 调试过程 24](#_Toc112539792)

[4.2 调试结果 24](#_Toc112539793)

[5 总结与感想 25](#_Toc112539794)

[5.1项目总结 25](#_Toc112539795)

[5.2 实习感想 25](#_Toc112539796)

[参考文献 26](#_Toc112539797)

[附录A 硬件电路 27](#_Toc112539798)

[附录B 软件程序 28](#_Toc112539799)

# 简易激光雷达的研发意义

## 1.1项目背景和需求分析

此处介绍作品的用途，市场上的需求情况。

## 1.2项目研发意义

可从以下几方面去说明。

（1）分析当前市场上现有同类产品的情况，并同本项目作品作比较，说明本项目作品的优越性。

（2）本项目对环境保护和可持续发展的作用

正文：

（1） 纸 型：A4纸。

（2） 页边距：上3.5cm，下2.5cm，左2.5cm、右2.5cm。

（3） 页 眉：2.5cm，页脚：2cm，左侧装订。

（4） 字 体：正文全部宋体、小四。

（5） 行 距：多倍行距：1.25，段前、段后均为0，取消网格对齐选项。

正文各级标题编号的示例如图1.1所示。



图1.1 标题编号的示例

# 2 简易激光雷达的功能与设计方案

## 2.1 系统设计任务

此处主要介绍产品功能。必要时可用表格的形式说明，并配有文字解释。

表2.1 物流的概念和范围

|  |  |
| --- | --- |
| 本质 | 过程 |
| 途径或方法 | 规划、实施、控制 |
| 目标 | 效率、成本效益 |
| 活动或作业 | 流动与储存 |
| 处理对象 | 原材料、在制品、产成品、相关信息 |
| 范围 | 从原点（供应商）到终点（最终顾客） |
| 目的或目标 | 适应顾客的需求（产品、功能、数量、质量、时间、价格） |

## 2.2 系统设计时考虑的因素

此处说明在确定系统设计方案过程中要考虑的诸多因素。包括以下几个方面。

（1）项目开发过程中，对制作成本的控制，可以扩展写一些自动化领域工程管理原理与经济决策方法方面的内容。

（2）本项目对社会、健康、安全、法律、文化可能产生的影响，清楚这些制约因素对项目实施的影响。

（3）其他方面

## 2.3 系统设计方案

（1）此处叙述确定方案的整个过程，要提供几种可能的方案，通过比较确定其中一种。

（2）针对最后确定的系统设计方案，画出系统硬件结构框图，说明系统硬件结构，并配有文字解释。

# 3 简易激光雷达的软硬件设计

## 3.1 硬件设计

### 3.1.1激光测距模块MS53L1M简介与用法

（1） 简介

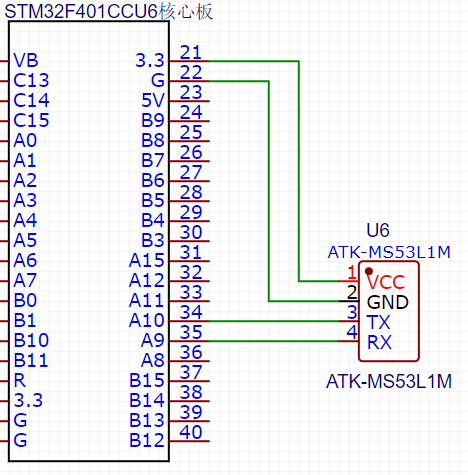
MS53L1M激光测距模块是正点原子推出的一款量程为4米的激光测距模块。可以通过串口直接输出测量距离。该模块主要有以下优点：

* 模块内部自带电压稳压电路，工作电压 3.3V~5V
* 模块支持Normal模式、Modbus模式、以及IIC模式
* 模块支持多种测量模式，测量最远 4 米
* 支持串口和 IIC 两种数字接口，串口方便配置模块，IIC 直接连接传感器，方便用 户选择最佳的开发连接方式
* 最高 100Hz 输出速率。输出速率 0.1~100Hz 可调
* 串口速率 2400~921600bps 可调

该模块有6根引脚，分别是电源线VCC、地线GND、串口输出TX、串口输入RX、I2C时钟线SCL、I2C数据线SDA。其中，由于本项目中使用串口通信的方式连接该模块与单片机，故SCL和SDA不做连接，处于悬空状态。

（2） 用法

接线图如下：



其中，单片机的PA9和PA10分别是单片机硬件串口1（UART1）的TX（发送端）和RX（接收端），根据“RX接TX，TX接RX”的原则，单片机的PA9则应该连接模块的RX，PA10则应该连接模块的TX。

### 3.1.2 OLED模块简介与用法

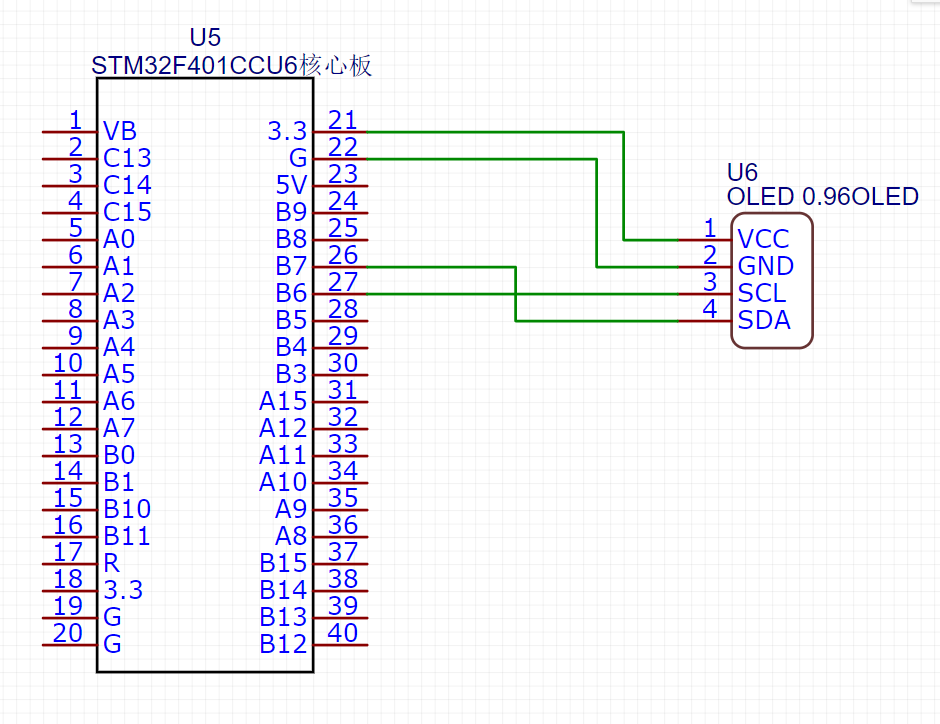
（1） 简介

OLED是一种先进的显示器，具有自发光、高对比度、快速响应、低功耗等优点。

本项目采用的OLED是128x64位像素、采用I2C通信、四线制的版本。四根引脚分别为电源线VCC、地线GND、时钟线SCL、数据线SDA。

（2） 用法

接线图如下：



其中，单片机的PA6和PA7分别是单片机的硬件I2C1的时钟线SCL和数据线SDA，分别连接模块的时钟线SCL和数据线SDA。

## 3.2 软件设计

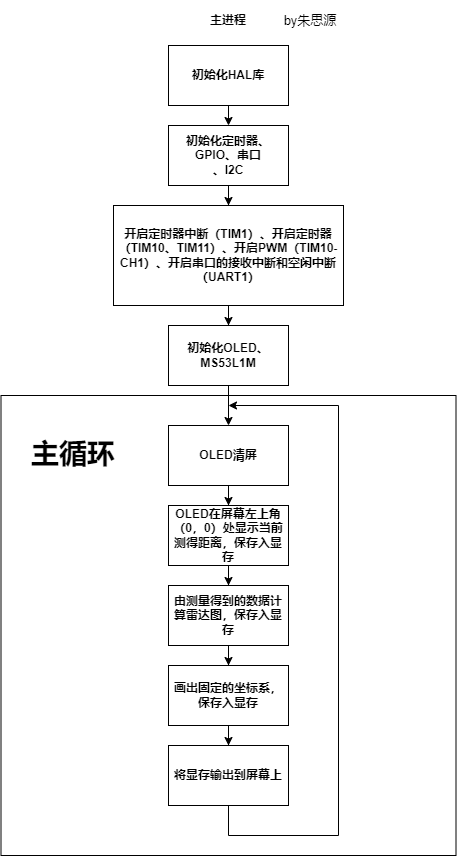
### 3.2.1软件设计思路

主体架构上，本项目采用主循环与定时器中断相结合的方式设计逻辑。

具体说来，就是将耗时的、运算量大的程序（图像显示相关代码）放在主循环中运行，将需要即时的、固定时间间隔的以及计算量比较小的程序（如定时从激光测距仪中读取数据、计算角度等代码）放在定时器中断中运行。主要的控制逻辑也是写在定时器中断之中的。

流程图如下：

（1） 主循环



（2） 定时器中断

定时器中断主要是先获得一个1000Hz的中断源，然后根据任务的需求分为200Hz和50Hz的中断，实现“伪操作系统”的效果。

代码实现的方式如下例（**示例代码，非实际工程代码**）：

uint16\_t tim1clk=0; // 0~999 IT Counter

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

  if(htim==&htim1) // 1kHz  IT Source

  {

    tim1clk++;

    tim1clk%=1000;

    if(tim1clk%5==1) //200Hz

    {

        // TASK 1

    }

    if(tim1clk%20==1) //50Hz

    {

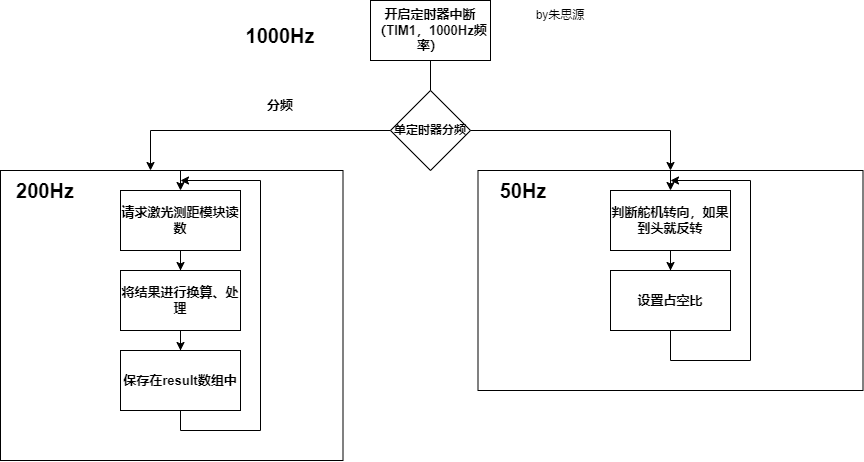
        // TASK 2

    }

  }

}

流程图如下图所示：



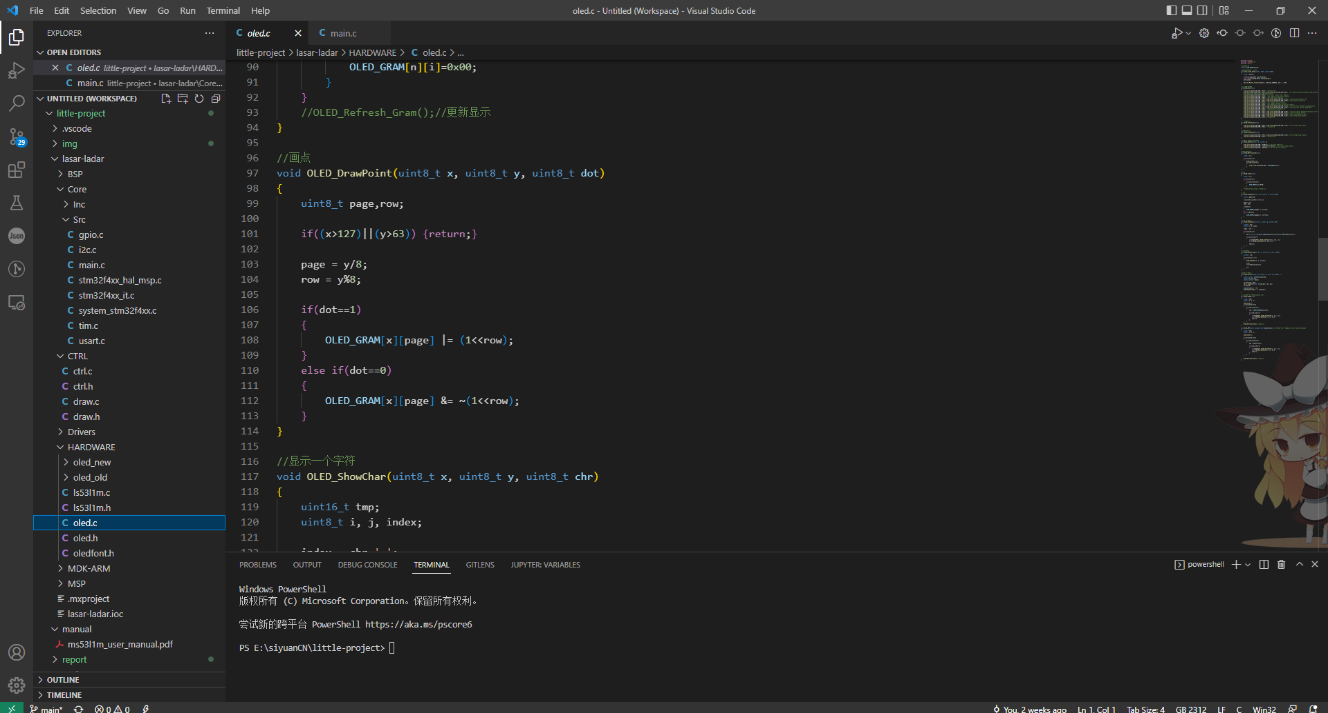
### 3.2.2软件开发环境简介与用法

（1） VSCode

是代码编辑器，通过安装C/C++相关插件可以提供比较好的代码提示、高亮等功能。

本项目主要通过此编辑器进行开发。

截图如下：

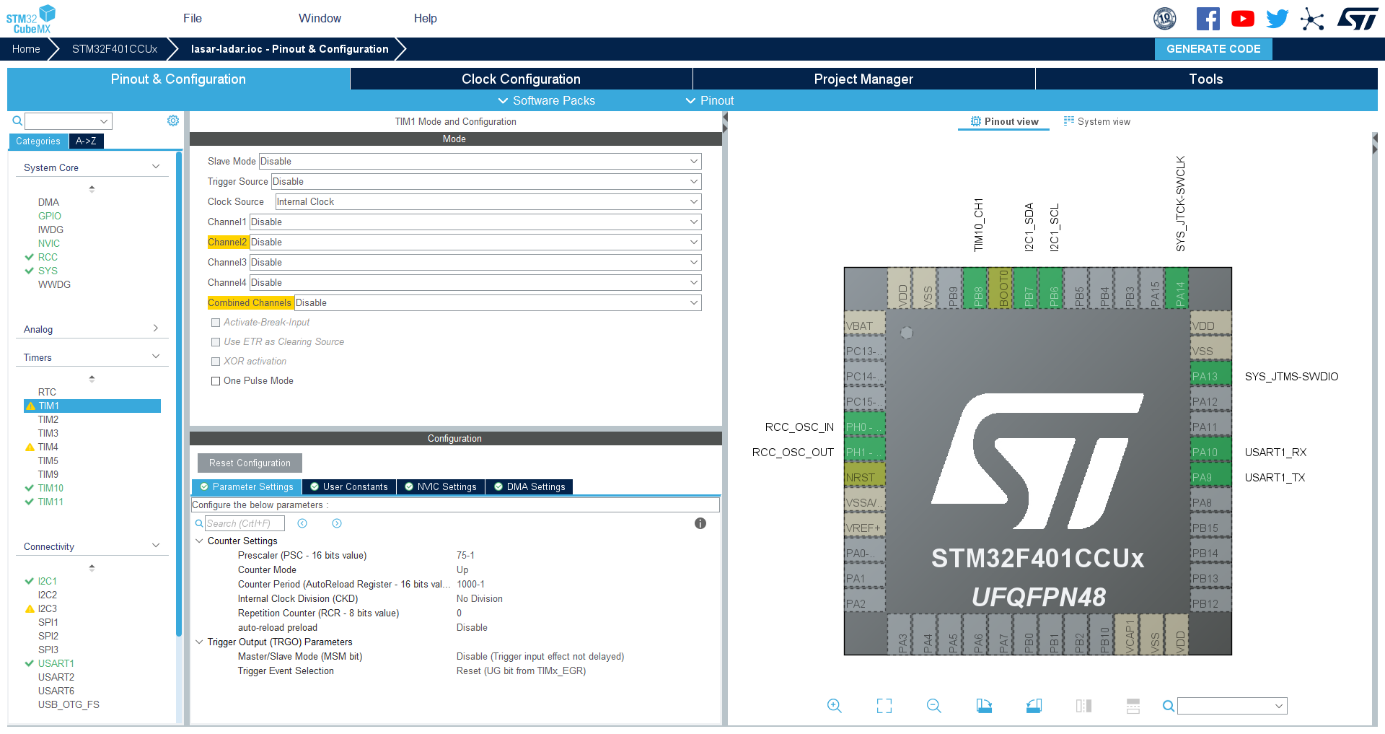


（2） STM32 CubeMX

意法半导体开发的专门用于自家芯片（STM32）的集成开发平台，可以比较方便的配置一些片上资源，并提供HAL库。

在本项目中用于前期生成一些初始化相关的代码。

截图如下：

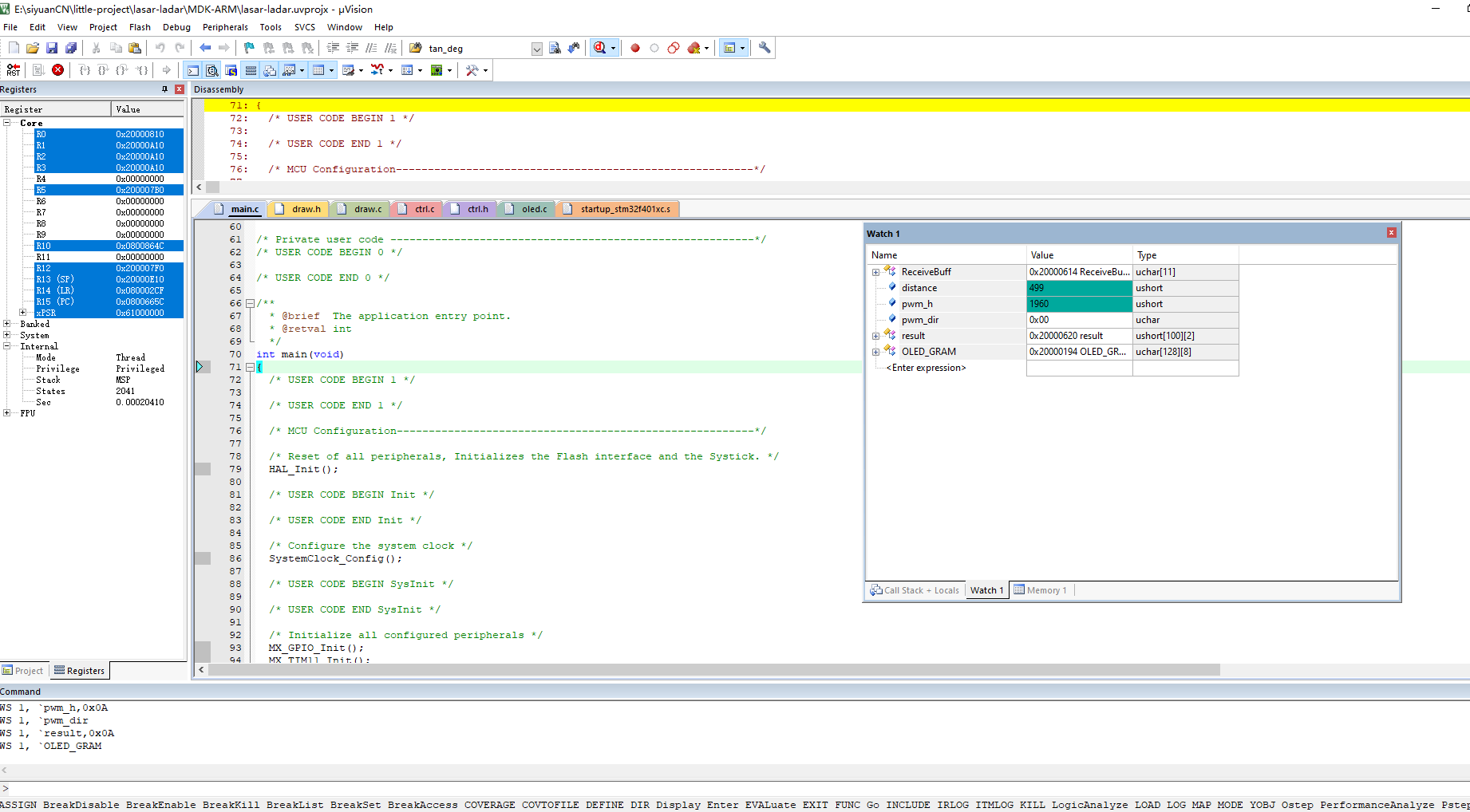


（3） Keil（MDK5）

集成的嵌入式开发平台，提供编辑、编译、调试、烧录等常见功能。

本项目中主要用来调试系统和烧录。

调试截图如下：



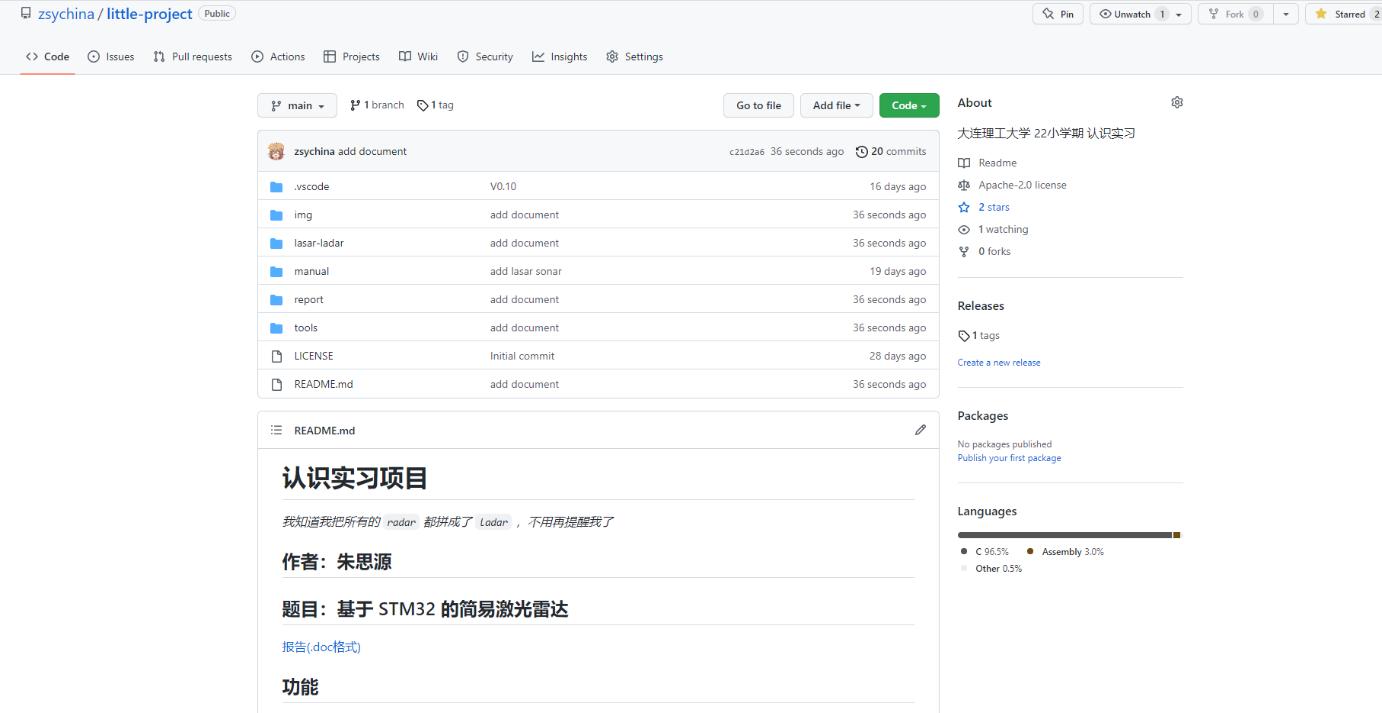
（4） Git

一个开源的代码版本管理工具。

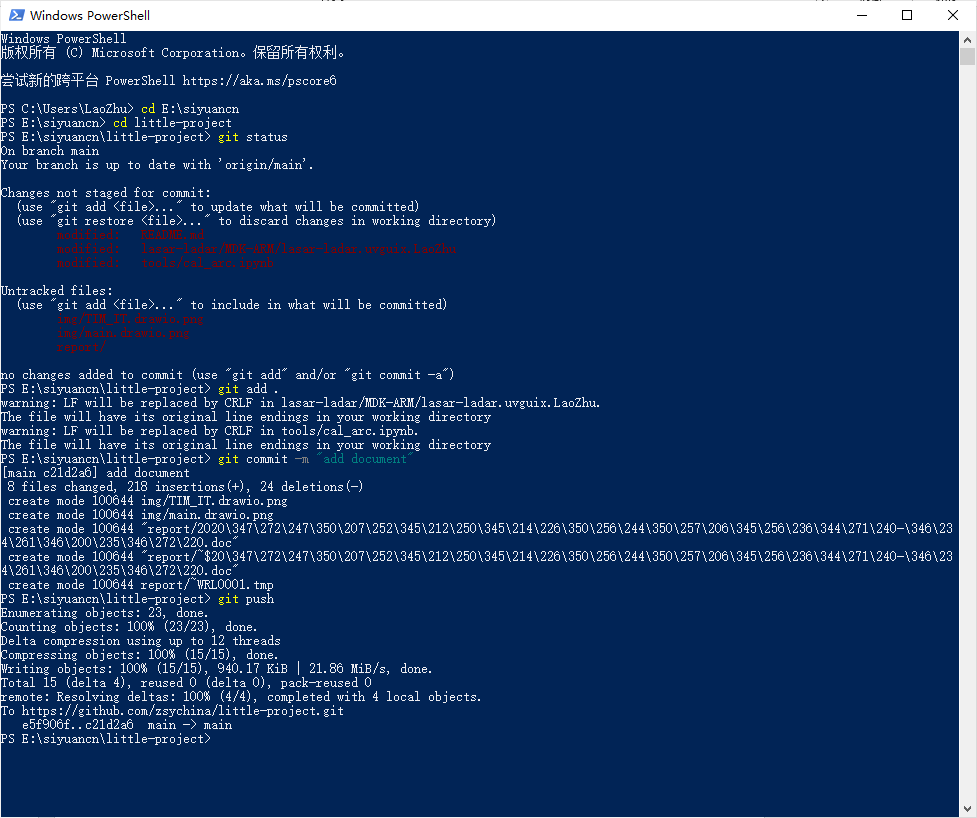
本项目中主要用来做版本管理，并将代码上传到Github中。

**本项目Github地址：<https://github.com/zsychina/little-project>**

Github截图如下：



Git提交代码的过程截图如下：

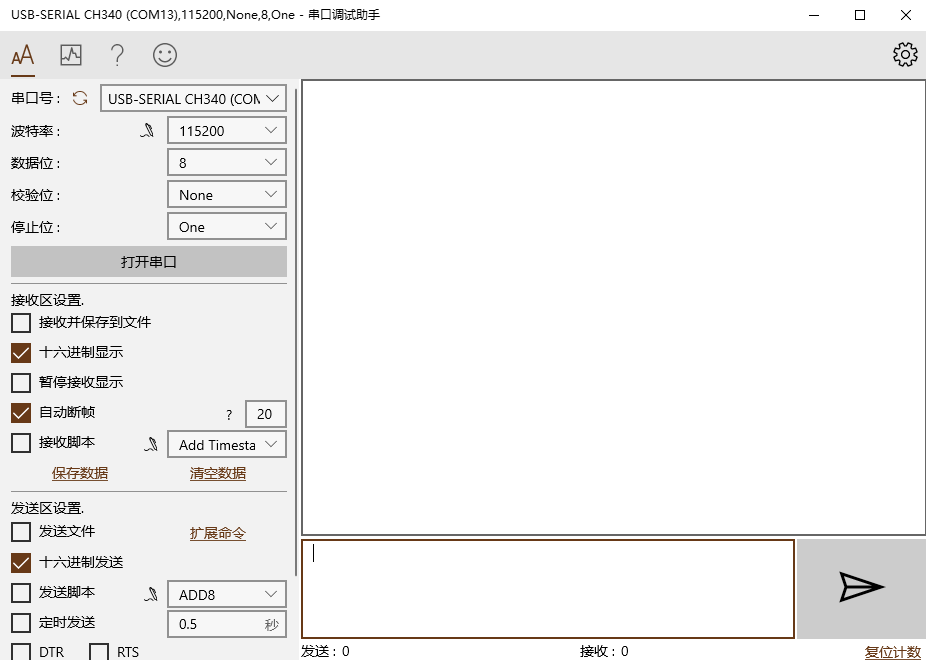


（5） 串口调试助手

连接USB转TTL模块后，能够调用电脑的USB向串口发送信息。

本项目中用来测试单片机串口和模块的串口通信是否正常。

截图如下：



### 3.2.3 OLED驱动程序

本项目使用的OLED是0.96寸、使用I2C通信的版本。

驱动程序由以下三部分组成：

1. oled.h 头文件，定义了一些函数名和跟通信地址有关的宏；
2. oledfont.h 字模文件，提供了OLED显示字母和汉字时必要的字体；
3. oled.c 源文件，定义了OLED初始化、OLED的I2C通信、在屏幕上描点等必要的代码。

以下就oled.c中最重要的几个函数进行简要的介绍。

（1） 画点函数

void OLED\_DrawPoint(uint8\_t x, uint8\_t y, uint8\_t dot)

{

    uint8\_t page,row;

    if((x>127)||(y>63)) {return;}

    page = y/8;

    row = y%8;

    if(dot==1)

    {

        OLED\_GRAM[x][page] |= (1<<row);

    }

    else if(dot==0)

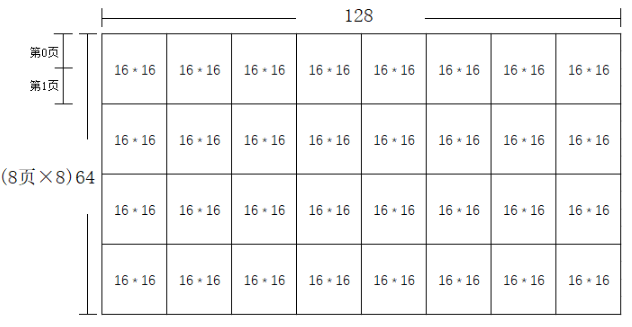
    {

        OLED\_GRAM[x][page] &= ~(1<<row);

    }

}

首先，根据相关模块的数据手册，本项目通过初始化的方式已经将OLED的屏幕划分成了如下形式：



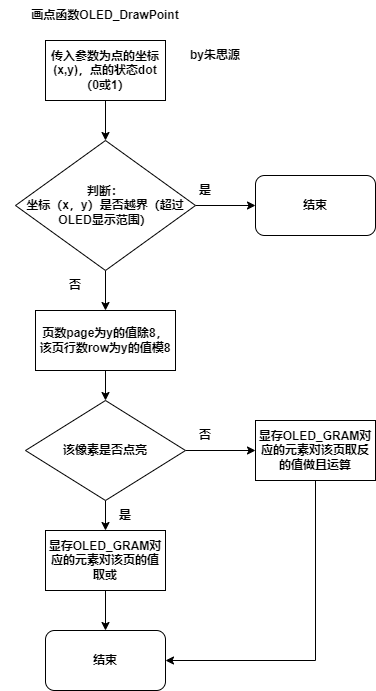
这样，我们可以很方便地将“同页、不同列”的8个像素点保存到一个8位的 “uint8\_t” 变量中。

然后通过定义一个二维数组，就可以将OLED显示屏上的所有像素状态保存在一个数组里：。

这样，我们就可以通过数组直接修改屏幕上某一个像素点的状态。

具体方法是，先定位到“页”，然后根据行数做位运算，即可修改该像素点的值（0或1）。

流程图如下：



（2） OLED的I2C通信函数

#define OLED\_CMD        0x00

#define OLED\_DATA       0x01

#define OLED\_I2C\_ADDRESS    0x78

void oled\_write\_byte(uint8\_t ctrl, uint8\_t dat)

{

    uint8\_t buf[2];

    if(ctrl==OLED\_CMD) {buf[0]=0x00;}

    else if(ctrl==OLED\_DATA) {buf[0]=0x40;}

    buf[1]=dat;

    HAL\_I2C\_Master\_Transmit(&hi2c1, OLED\_I2C\_ADDRESS, buf, 2, 100);

}

这个函数主要是调用HAL库自带的I2C通信协议的API，即

HAL\_StatusTypeDef HAL\_I2C\_Master\_Transmit(I2C\_HandleTypeDef \*hi2c, uint16\_t DevAddress, uint8\_t \*pData, uint16\_t Size, uint32\_t Timeout)

函数的几个参数含义分别为：硬件I2C的指针，设备地址，传输信息的指针，传输信息的长度，允许超时的时间。

再根据该模块的数据手册可知，此OLED挂载在I2C总线上是地址总是0x78，故第二个参数传入0x78。又根据数据手册得知，如果要从单片机向OLED传入命令，应将发送的第一个字节设置为0x00；如果要从单片机向OLED传入数据，还应将发送的第一个字节设置为0x01。所以，OLED的I2C通信函数内部会先进行判断，决定这一次调用是命令还是数据，然后用HAL库的I2C API发送出去。

（3） 更新显存函数

void OLED\_Refresh\_Gram(void)

{

    uint8\_t i,n;

    for(i=0;i<8;i++)

    {

        OLED\_SetPos(0,i);

        for(n=0;n<128;n++)

        {

            oled\_write\_byte(OLED\_DATA, OLED\_GRAM[n][i]);

        }

    }

}

其中，是调用前文所述的OLED I2C通信函数；是个选定某页的函数，其内容如下：

void OLED\_SetPos(uint8\_t x, uint8\_t y)

{

    oled\_write\_byte(OLED\_CMD, (0xB0+y)); //set page address y

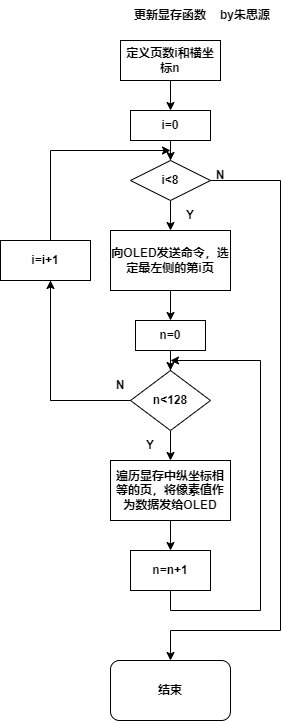
    oled\_write\_byte(OLED\_CMD, ((x&0xF0)>>4)|0x10); //set column high address

    oled\_write\_byte(OLED\_CMD, (x&0x0F)); //set column low address

}

主要是直接向OLED发送命令，选定某一页，为下面的数据传输做准备。

流程图如下：



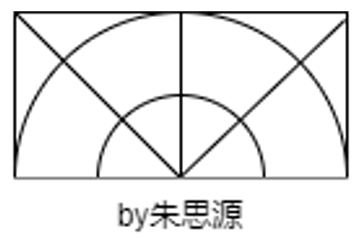
### 3.2.4 数据可视化程序

这部分程序的目的在于，在上述OLED驱动程序的基础上，开发出一套数据可视化的代码。

本项目中的数据可视化是以雷达图的方式呈现的。

（1） 描绘刻度

雷达图刻度如下（示意图）：



为了生成如上图所示的雷达图刻度，本项目采用了描点连线的方法。即用代码计算出画出如上所示图形所需的各个点，然后用OLED驱动函数的分别将各个点写入显存中。

生成代码的程序是用Python写的，具体如下：

1. 画圆弧：

# calculate a circle to display in OLED

import numpy as np

np.set\_printoptions(threshold = 1e6)

GRAM=np.zeros((128,64),dtype=np.int32)

cnt=0

for x in range(0,128):

    for y in range(0,64):

        if abs( (x-64)\*\*2+(y-64)\*\*2-32\*\*2 ) <= 20:

            GRAM[x][y]=1

            cnt+=1

            # print(x,y)

            print('  OLED\_DrawPoint({},{},1);'.format(x,y))

2. 画直线：

# calculate a line to display in OLED

import numpy as np

np.set\_printoptions(threshold = 1e6)

GRAM=np.zeros((128,64),dtype=np.int32)

cnt=0

for x in range(0,128):

    for y in range(0,64):

        if abs( x-64 ) <= 0:

            GRAM[x][y]=1

            cnt+=1

            # print(x,y)

            print('  OLED\_DrawPoint({},{},1);'.format(x,y))

由此得到的C语言代码即为draw.c中的部分代码，由于此函数过长因此下面只展示部分。

void draw\_bgd()

{

  // outer circle:

  OLED\_DrawPoint(0,59,1);

  //  省略

  OLED\_DrawPoint(127,54,1);

  // inner circle:

  OLED\_DrawPoint(32,60,1);

  //  省略

  OLED\_DrawPoint(96,63,1);

  // line y=x:

  OLED\_DrawPoint(0,0,1);

  //  省略

  OLED\_DrawPoint(63,63,1);

  // line y=-x+128:

  OLED\_DrawPoint(65,63,1);

  //  省略

  OLED\_DrawPoint(127,1,1);

  // line x=64:

  OLED\_DrawPoint(64,0,1);

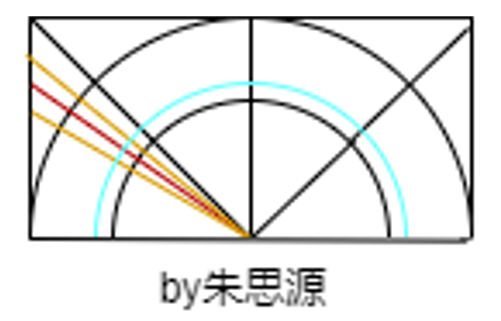
  //  省略

  OLED\_DrawPoint(64,63,1);

}

（2） 描绘目标

用来表示探测物体与雷达距离的弧线则更加复杂，核心是解一个二元一次不等式组。



如图所示，红线为某物体的方向（角度表示），黄线即为某物体方向的上下界（误差，角度表示），蓝色的半圆则表示与目标的距离。蓝色的半圆在两根黄线之内的小圆弧就是要计算的部分。

显然，要计算的不等式就是两根黄线中间的部分，再联立上表示距离的半圆方程即可。

由于遍历每个像素的时候，必然不可能很好地满足半圆方程，故必须对方程做一些数学处理：

通过改变round的值即可改变描绘的精度，达到近似的效果。

相关代码如下：

void draw\_arc(uint16\_t angle,uint16\_t distance)

{

  for(uint8\_t x=0;x<128;x++)

  {

    for(uint8\_t y=0;y<64;y++)

    {

      if(

        (abs\_64((x-64)\*(x-64)+(y-64)\*(y-64)-distance\*distance) <= 35) &&

        ((y-64)/tan\_deg(angle+WIDTH)+64-LENGTH <= x) &&

        (x <= (y-64)/tan\_deg(angle-WIDTH)+64+LENGTH)

      )

      {

        OLED\_DrawPoint(x,y,1);

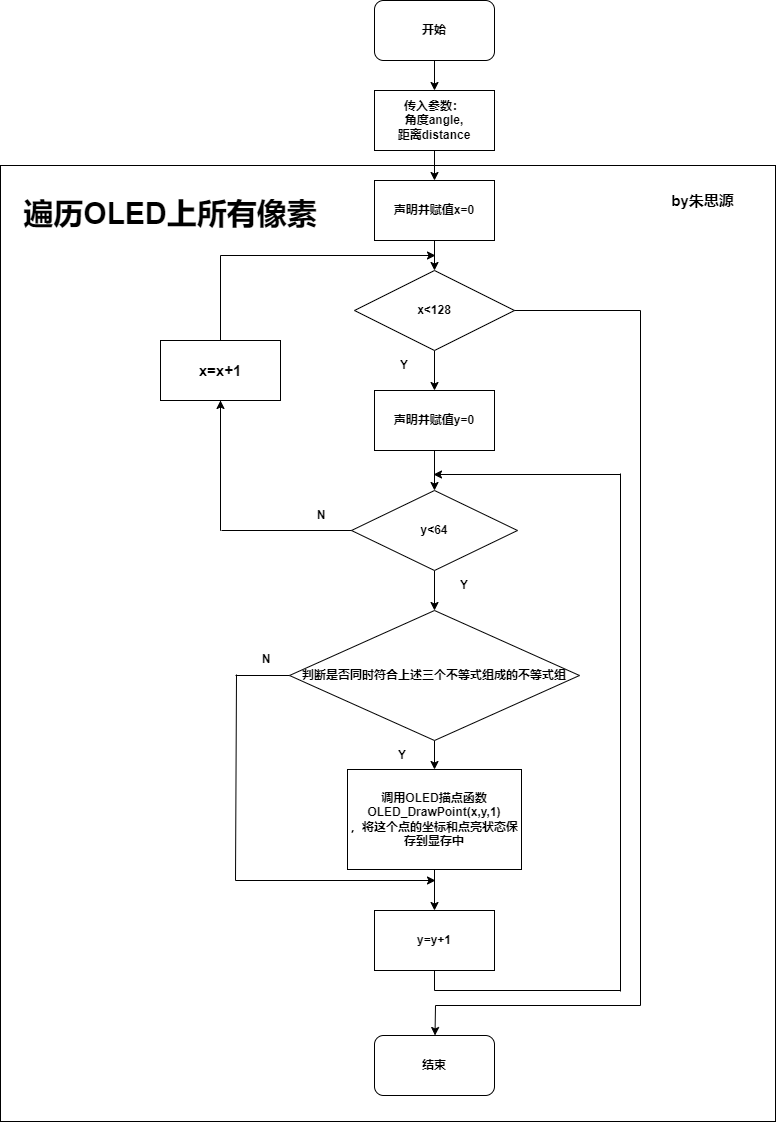
      }

    }

  }

}

流程图如下：



# 4 项目调试

## 4.1 调试过程

1. 运行卡顿

本项目一开始出现了运行卡顿的故障，经我推测，这可能的原因是将OLED刷新函数写在定时器中断内部的结果。

定时器中断本应该是瞬间完成的，里面不能有任何延时或时间较长的函数，否者可能会影响到下一次中断。

OLED的I2C通信本身速度比较慢，写在定时器中断内部则可能造成上述的BUG。

之后我将OLED的I2C通信相关代码写在主循环while(1)中则问题解决。

（2） 画图

如上文3.2.4所述，可视化在开发过程中也一度成为一个问题，关键就在于如何显示一张固定的背景，且保证不被其他信息覆盖。

（3） 固定

如何将激光测距模块固定到舵机上也一度成为一个问题。

如果固定不牢，那对测距也会造成很大的影响，最严重的情况会使测距结果完全不可靠，造成项目的失败。

起初是尝试用502胶水把激光测距模块粘到舵机上，可结果完全不行，可能是因为接触面比较少，根本粘不牢。

解决办法是买了热熔胶，用热熔胶粘效果很好。

## 4.2 调试结果

要提供作品的整个实物图。要有主要的程序运行结果，比如液晶显示屏画面等。

# 5 总结与感想

## 5.1项目总结

此处是对整个项目研发过程的整理与概括。谈一谈可以继续改进的地方。

## 5.2 实习感想

感想是实习后的感受，可以谈谈实习的收获、实习成果对未来工作或研究生学业的作用等等。

# 参考文献

1 普通图书

[1] 广西壮族自治区林业厅.广西自然保护区[M].北京:中国林业出版社,1993.

2 期刊中析出的文献

[1] 李炳穆.理想的图书馆员和信息专家的素质与形象[J].图书情报工作,2000,(2):5-8.

3 论文集、会议录

[1] 中国力学学会.第3届全国实验流体力学学术会议论文集[C].天津:[出版者不祥],1990.

5 学位论文

[1] 张志祥. 间断动力系统的随机扰动及其在守恒律方程中的应用[D].北京:北京大学数学学院,1998.

6 专利文献

[1] 刘加林. 多功能一次性压舌板:中国,92214985.2[P].1993,04,14.

8 报纸中析出文献

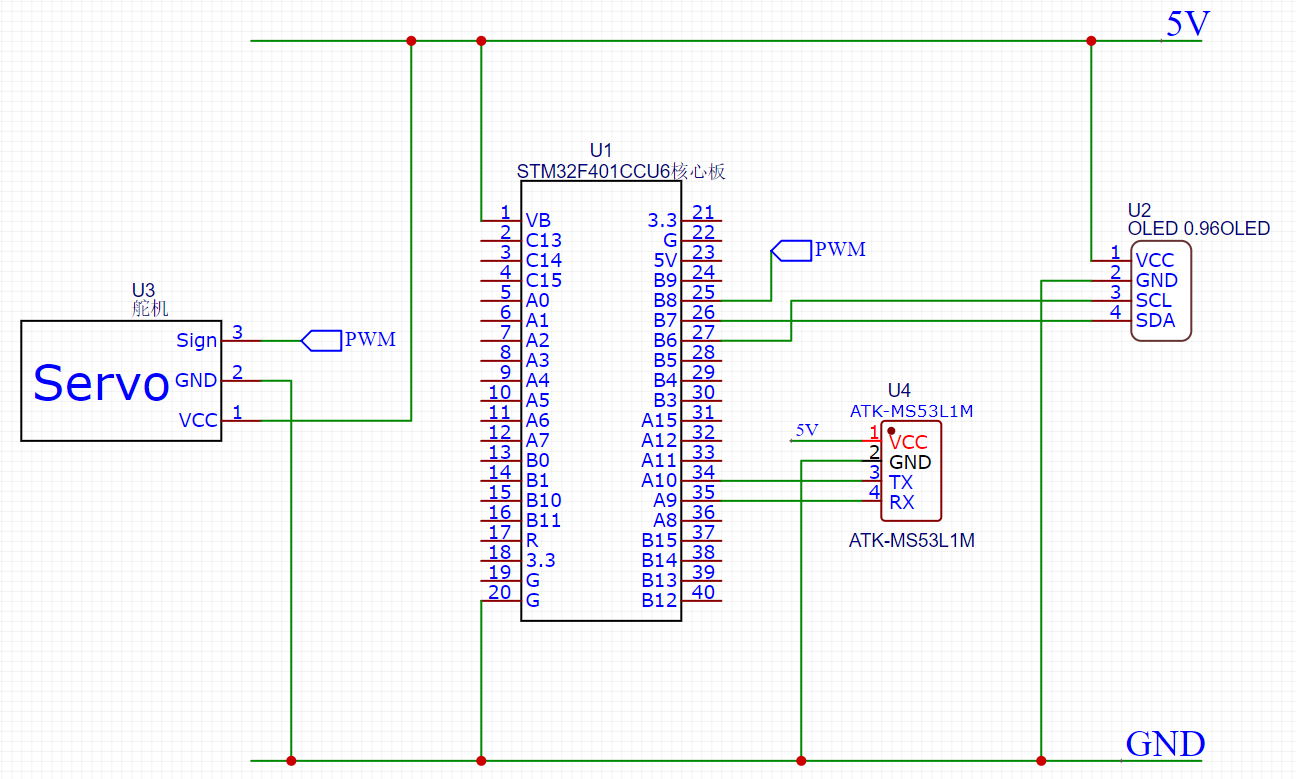
[1] 丁文祥. 数字革命与竞争国际化[N].中国青年报,2000 ,11,20(15).

9 电子文献（包括专著或连续出版物中析出的电子文献）

[1] 江向东.互联网环境下的信息处理与图书管理系统解决方案[J/OL].情报学报, 1999, 18(2):4[2000-01-18]. http://www.chinainfo.gov.cn/periodical/qbxb/qbxb99/qbxb990203.

# 附录A 硬件电路

硬件原理图如下



# 附录B 软件程序

本项目基于STM32 HAL（**H**ardware **A**bstraction **L**ayer）库进行开发，由于HAL是一个公开的库函数集合，故在此处不再赘述。

以下是本项目除HAL库以外，涉及到用户代码的部分。

**详见本项目的Github：<https://github.com/zsychina/little-project>**

oled.h

#ifndef \_OLED\_H\_

#define \_OLED\_H\_

#include "main.h"

#include <stdio.h>

#include <stdarg.h>

//

#define OLED\_I2C\_ADDRESS    0x78

#define OLED\_CMD        0x00

#define OLED\_DATA       0x01

//

extern void OLED\_Init(void);

extern void OLED\_Display\_On(void);

extern void OLED\_Display\_Off(void);

extern void OLED\_SetPos(uint8\_t x, uint8\_t y);

extern void OLED\_Refresh\_Gram(void);

extern void OLED\_Clear(void);

extern void OLED\_DrawPoint(uint8\_t x, uint8\_t y, uint8\_t dot);

extern void OLED\_ShowChar(uint8\_t x, uint8\_t y, uint8\_t chr);

extern void OLED\_ShowString(uint8\_t x, uint8\_t y, uint8\_t \*str);

extern void OLED\_printf(uint8\_t row, uint8\_t col, const char \*fmt,...);

extern const unsigned char ascii\_1206[95][12];

#endif

oledfont.h

#ifndef \_OLED\_FONT\_H\_

#define \_OLED\_FONT\_H\_

const unsigned char ascii\_1206[95][12]={

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*" ",0\*/

{0x00,0x00,0x00,0x00,0x3F,0x40,0x00,0x00,0x00,0x00,0x00,0x00},/\*"!",1\*/

{0x00,0x00,0x30,0x00,0x40,0x00,0x30,0x00,0x40,0x00,0x00,0x00},/\*""",2\*/

{0x09,0x00,0x0B,0xC0,0x3D,0x00,0x0B,0xC0,0x3D,0x00,0x09,0x00},/\*"#",3\*/

{0x18,0xC0,0x24,0x40,0x7F,0xE0,0x22,0x40,0x31,0x80,0x00,0x00},/\*"$",4\*/

{0x18,0x00,0x24,0xC0,0x1B,0x00,0x0D,0x80,0x32,0x40,0x01,0x80},/\*"%",5\*/

{0x03,0x80,0x1C,0x40,0x27,0x40,0x1C,0x80,0x07,0x40,0x00,0x40},/\*"&",6\*/

{0x10,0x00,0x60,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"'",7\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x80,0x20,0x40,0x40,0x20},/\*"(",8\*/

{0x00,0x00,0x40,0x20,0x20,0x40,0x1F,0x80,0x00,0x00,0x00,0x00},/\*")",9\*/

{0x09,0x00,0x06,0x00,0x1F,0x80,0x06,0x00,0x09,0x00,0x00,0x00},/\*"\*",10\*/

{0x04,0x00,0x04,0x00,0x3F,0x80,0x04,0x00,0x04,0x00,0x00,0x00},/\*"+",11\*/

{0x00,0x10,0x00,0x60,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*",",12\*/

{0x04,0x00,0x04,0x00,0x04,0x00,0x04,0x00,0x04,0x00,0x00,0x00},/\*"-",13\*/

{0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*".",14\*/

{0x00,0x20,0x01,0xC0,0x06,0x00,0x38,0x00,0x40,0x00,0x00,0x00},/\*"/",15\*/

{0x1F,0x80,0x20,0x40,0x20,0x40,0x20,0x40,0x1F,0x80,0x00,0x00},/\*"0",16\*/

{0x00,0x00,0x10,0x40,0x3F,0xC0,0x00,0x40,0x00,0x00,0x00,0x00},/\*"1",17\*/

{0x18,0xC0,0x21,0x40,0x22,0x40,0x24,0x40,0x18,0x40,0x00,0x00},/\*"2",18\*/

{0x10,0x80,0x20,0x40,0x24,0x40,0x24,0x40,0x1B,0x80,0x00,0x00},/\*"3",19\*/

{0x02,0x00,0x0D,0x00,0x11,0x00,0x3F,0xC0,0x01,0x40,0x00,0x00},/\*"4",20\*/

{0x3C,0x80,0x24,0x40,0x24,0x40,0x24,0x40,0x23,0x80,0x00,0x00},/\*"5",21\*/

{0x1F,0x80,0x24,0x40,0x24,0x40,0x34,0x40,0x03,0x80,0x00,0x00},/\*"6",22\*/

{0x30,0x00,0x20,0x00,0x27,0xC0,0x38,0x00,0x20,0x00,0x00,0x00},/\*"7",23\*/

{0x1B,0x80,0x24,0x40,0x24,0x40,0x24,0x40,0x1B,0x80,0x00,0x00},/\*"8",24\*/

{0x1C,0x00,0x22,0xC0,0x22,0x40,0x22,0x40,0x1F,0x80,0x00,0x00},/\*"9",25\*/

{0x00,0x00,0x00,0x00,0x08,0x40,0x00,0x00,0x00,0x00,0x00,0x00},/\*":",26\*/

{0x00,0x00,0x00,0x00,0x04,0x60,0x00,0x00,0x00,0x00,0x00,0x00},/\*";",27\*/

{0x00,0x00,0x04,0x00,0x0A,0x00,0x11,0x00,0x20,0x80,0x40,0x40},/\*"<",28\*/

{0x09,0x00,0x09,0x00,0x09,0x00,0x09,0x00,0x09,0x00,0x00,0x00},/\*"=",29\*/

{0x00,0x00,0x40,0x40,0x20,0x80,0x11,0x00,0x0A,0x00,0x04,0x00},/\*">",30\*/

{0x18,0x00,0x20,0x00,0x23,0x40,0x24,0x00,0x18,0x00,0x00,0x00},/\*"?",31\*/

{0x1F,0x80,0x20,0x40,0x27,0x40,0x29,0x40,0x1F,0x40,0x00,0x00},/\*"@",32\*/

{0x00,0x40,0x07,0xC0,0x39,0x00,0x0F,0x00,0x01,0xC0,0x00,0x40},/\*"A",33\*/

{0x20,0x40,0x3F,0xC0,0x24,0x40,0x24,0x40,0x1B,0x80,0x00,0x00},/\*"B",34\*/

{0x1F,0x80,0x20,0x40,0x20,0x40,0x20,0x40,0x30,0x80,0x00,0x00},/\*"C",35\*/

{0x20,0x40,0x3F,0xC0,0x20,0x40,0x20,0x40,0x1F,0x80,0x00,0x00},/\*"D",36\*/

{0x20,0x40,0x3F,0xC0,0x24,0x40,0x2E,0x40,0x30,0xC0,0x00,0x00},/\*"E",37\*/

{0x20,0x40,0x3F,0xC0,0x24,0x40,0x2E,0x00,0x30,0x00,0x00,0x00},/\*"F",38\*/

{0x0F,0x00,0x10,0x80,0x20,0x40,0x22,0x40,0x33,0x80,0x02,0x00},/\*"G",39\*/

{0x20,0x40,0x3F,0xC0,0x04,0x00,0x04,0x00,0x3F,0xC0,0x20,0x40},/\*"H",40\*/

{0x20,0x40,0x20,0x40,0x3F,0xC0,0x20,0x40,0x20,0x40,0x00,0x00},/\*"I",41\*/

{0x00,0x60,0x20,0x20,0x20,0x20,0x3F,0xC0,0x20,0x00,0x20,0x00},/\*"J",42\*/

{0x20,0x40,0x3F,0xC0,0x24,0x40,0x0B,0x00,0x30,0xC0,0x20,0x40},/\*"K",43\*/

{0x20,0x40,0x3F,0xC0,0x20,0x40,0x00,0x40,0x00,0x40,0x00,0xC0},/\*"L",44\*/

{0x3F,0xC0,0x3C,0x00,0x03,0xC0,0x3C,0x00,0x3F,0xC0,0x00,0x00},/\*"M",45\*/

{0x20,0x40,0x3F,0xC0,0x0C,0x40,0x23,0x00,0x3F,0xC0,0x20,0x00},/\*"N",46\*/

{0x1F,0x80,0x20,0x40,0x20,0x40,0x20,0x40,0x1F,0x80,0x00,0x00},/\*"O",47\*/

{0x20,0x40,0x3F,0xC0,0x24,0x40,0x24,0x00,0x18,0x00,0x00,0x00},/\*"P",48\*/

{0x1F,0x80,0x21,0x40,0x21,0x40,0x20,0xE0,0x1F,0xA0,0x00,0x00},/\*"Q",49\*/

{0x20,0x40,0x3F,0xC0,0x24,0x40,0x26,0x00,0x19,0xC0,0x00,0x40},/\*"R",50\*/

{0x18,0xC0,0x24,0x40,0x24,0x40,0x22,0x40,0x31,0x80,0x00,0x00},/\*"S",51\*/

{0x30,0x00,0x20,0x40,0x3F,0xC0,0x20,0x40,0x30,0x00,0x00,0x00},/\*"T",52\*/

{0x20,0x00,0x3F,0x80,0x00,0x40,0x00,0x40,0x3F,0x80,0x20,0x00},/\*"U",53\*/

{0x20,0x00,0x3E,0x00,0x01,0xC0,0x07,0x00,0x38,0x00,0x20,0x00},/\*"V",54\*/

{0x38,0x00,0x07,0xC0,0x3C,0x00,0x07,0xC0,0x38,0x00,0x00,0x00},/\*"W",55\*/

{0x20,0x40,0x39,0xC0,0x06,0x00,0x39,0xC0,0x20,0x40,0x00,0x00},/\*"X",56\*/

{0x20,0x00,0x38,0x40,0x07,0xC0,0x38,0x40,0x20,0x00,0x00,0x00},/\*"Y",57\*/

{0x30,0x40,0x21,0xC0,0x26,0x40,0x38,0x40,0x20,0xC0,0x00,0x00},/\*"Z",58\*/

{0x00,0x00,0x00,0x00,0x7F,0xE0,0x40,0x20,0x40,0x20,0x00,0x00},/\*"[",59\*/

{0x00,0x00,0x70,0x00,0x0C,0x00,0x03,0x80,0x00,0x40,0x00,0x00},/\*"\",60\*/

{0x00,0x00,0x40,0x20,0x40,0x20,0x7F,0xE0,0x00,0x00,0x00,0x00},/\*"]",61\*/

{0x00,0x00,0x20,0x00,0x40,0x00,0x20,0x00,0x00,0x00,0x00,0x00},/\*"^",62\*/

{0x00,0x10,0x00,0x10,0x00,0x10,0x00,0x10,0x00,0x10,0x00,0x10},/\*"\_",63\*/

{0x00,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"`",64\*/

{0x00,0x00,0x02,0x80,0x05,0x40,0x05,0x40,0x03,0xC0,0x00,0x40},/\*"a",65\*/

{0x20,0x00,0x3F,0xC0,0x04,0x40,0x04,0x40,0x03,0x80,0x00,0x00},/\*"b",66\*/

{0x00,0x00,0x03,0x80,0x04,0x40,0x04,0x40,0x06,0x40,0x00,0x00},/\*"c",67\*/

{0x00,0x00,0x03,0x80,0x04,0x40,0x24,0x40,0x3F,0xC0,0x00,0x40},/\*"d",68\*/

{0x00,0x00,0x03,0x80,0x05,0x40,0x05,0x40,0x03,0x40,0x00,0x00},/\*"e",69\*/

{0x00,0x00,0x04,0x40,0x1F,0xC0,0x24,0x40,0x24,0x40,0x20,0x00},/\*"f",70\*/

{0x00,0x00,0x02,0xE0,0x05,0x50,0x05,0x50,0x06,0x50,0x04,0x20},/\*"g",71\*/

{0x20,0x40,0x3F,0xC0,0x04,0x40,0x04,0x00,0x03,0xC0,0x00,0x40},/\*"h",72\*/

{0x00,0x00,0x04,0x40,0x27,0xC0,0x00,0x40,0x00,0x00,0x00,0x00},/\*"i",73\*/

{0x00,0x10,0x00,0x10,0x04,0x10,0x27,0xE0,0x00,0x00,0x00,0x00},/\*"j",74\*/

{0x20,0x40,0x3F,0xC0,0x01,0x40,0x07,0x00,0x04,0xC0,0x04,0x40},/\*"k",75\*/

{0x20,0x40,0x20,0x40,0x3F,0xC0,0x00,0x40,0x00,0x40,0x00,0x00},/\*"l",76\*/

{0x07,0xC0,0x04,0x00,0x07,0xC0,0x04,0x00,0x03,0xC0,0x00,0x00},/\*"m",77\*/

{0x04,0x40,0x07,0xC0,0x04,0x40,0x04,0x00,0x03,0xC0,0x00,0x40},/\*"n",78\*/

{0x00,0x00,0x03,0x80,0x04,0x40,0x04,0x40,0x03,0x80,0x00,0x00},/\*"o",79\*/

{0x04,0x10,0x07,0xF0,0x04,0x50,0x04,0x40,0x03,0x80,0x00,0x00},/\*"p",80\*/

{0x00,0x00,0x03,0x80,0x04,0x40,0x04,0x50,0x07,0xF0,0x00,0x10},/\*"q",81\*/

{0x04,0x40,0x07,0xC0,0x02,0x40,0x04,0x00,0x04,0x00,0x00,0x00},/\*"r",82\*/

{0x00,0x00,0x06,0x40,0x05,0x40,0x05,0x40,0x04,0xC0,0x00,0x00},/\*"s",83\*/

{0x00,0x00,0x04,0x00,0x1F,0x80,0x04,0x40,0x00,0x40,0x00,0x00},/\*"t",84\*/

{0x04,0x00,0x07,0x80,0x00,0x40,0x04,0x40,0x07,0xC0,0x00,0x40},/\*"u",85\*/

{0x04,0x00,0x07,0x00,0x04,0xC0,0x01,0x80,0x06,0x00,0x04,0x00},/\*"v",86\*/

{0x06,0x00,0x01,0xC0,0x07,0x00,0x01,0xC0,0x06,0x00,0x00,0x00},/\*"w",87\*/

{0x04,0x40,0x06,0xC0,0x01,0x00,0x06,0xC0,0x04,0x40,0x00,0x00},/\*"x",88\*/

{0x04,0x10,0x07,0x10,0x04,0xE0,0x01,0x80,0x06,0x00,0x04,0x00},/\*"y",89\*/

{0x00,0x00,0x04,0x40,0x05,0xC0,0x06,0x40,0x04,0x40,0x00,0x00},/\*"z",90\*/

{0x00,0x00,0x00,0x00,0x04,0x00,0x7B,0xE0,0x40,0x20,0x00,0x00},/\*"{",91\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xF0,0x00,0x00,0x00,0x00},/\*"|",92\*/

{0x00,0x00,0x40,0x20,0x7B,0xE0,0x04,0x00,0x00,0x00,0x00,0x00},/\*"}",93\*/

{0x40,0x00,0x80,0x00,0x40,0x00,0x20,0x00,0x20,0x00,0x40,0x00},/\*"~",94\*/

};

#endif

oled.c

#include "oled.h"

#include "oledfont.h"

#include "i2c.h"

//OLED的显存

uint8\_t OLED\_GRAM[128][8];

//向SSD1306写入一个字节

void oled\_write\_byte(uint8\_t ctrl, uint8\_t dat)

{

    uint8\_t buf[2];

    if(ctrl==OLED\_CMD) {buf[0]=0x00;}

    else if(ctrl==OLED\_DATA) {buf[0]=0x40;}

    buf[1]=dat;

    HAL\_I2C\_Master\_Transmit(&hi2c2, OLED\_I2C\_ADDRESS, buf, 2, 100);

}

//初始化OLED模块

void OLED\_Init(void)

{

    oled\_write\_byte(OLED\_CMD, 0xAE); //Display OFF

    oled\_write\_byte(OLED\_CMD, 0x20); oled\_write\_byte(OLED\_CMD,0x02); //Set Memory Addressing Mode--0x00,Horizontal Addressing Mode;0x01,Vertical Addressing Mode;0x02,Page Addressing Mode;0x03,Invalid

    oled\_write\_byte(OLED\_CMD, 0x00); //set Lower Column Start Address

    oled\_write\_byte(OLED\_CMD, 0x10); //set higher Column Start Address

    oled\_write\_byte(OLED\_CMD, 0xB0); //Set Page Start Address

    oled\_write\_byte(OLED\_CMD, 0xA8); oled\_write\_byte(OLED\_CMD, 0x3F); //Set Multiplex Ratio--63

    oled\_write\_byte(OLED\_CMD, 0xD3); oled\_write\_byte(OLED\_CMD, 0x00); //Set Display Offset--0

    oled\_write\_byte(OLED\_CMD, 0x40); //Set Display Start Line--0

    oled\_write\_byte(OLED\_CMD, 0xA1); //Set Segment Re-map--reverse

    oled\_write\_byte(OLED\_CMD, 0xC8); //Set COM Output Scan Direction--reverse

    oled\_write\_byte(OLED\_CMD, 0xDA); oled\_write\_byte(OLED\_CMD, 0x12); //Set COM Pins Hardware Configuration

    oled\_write\_byte(OLED\_CMD, 0x81); oled\_write\_byte(OLED\_CMD, 0xFF); //set Contrast Control 0x00~0xFF

    oled\_write\_byte(OLED\_CMD, 0xA4); //Disable Entire Display On

    oled\_write\_byte(OLED\_CMD, 0xA6); //Set Normal Display

    oled\_write\_byte(OLED\_CMD, 0xD5); oled\_write\_byte(OLED\_CMD, 0xF0); //set Oscillator Frequency/Display Clock Divide Ratio

    oled\_write\_byte(OLED\_CMD, 0xD9); oled\_write\_byte(OLED\_CMD, 0x22); //Set Pre-charge Period

    oled\_write\_byte(OLED\_CMD, 0xDB); oled\_write\_byte(OLED\_CMD, 0x20); //Set Vcomh Deselect Level

    oled\_write\_byte(OLED\_CMD, 0x8D); oled\_write\_byte(OLED\_CMD, 0x14); //Set Charge Pump Enable

    oled\_write\_byte(OLED\_CMD, 0xAF); //Display ON

}

//开启OLED显示

void OLED\_Display\_On(void)

{

    oled\_write\_byte(OLED\_CMD, 0x8D); oled\_write\_byte(OLED\_CMD, 0x14); //Set Charge Pump Enable

    oled\_write\_byte(OLED\_CMD, 0xAF); //Display ON

}

//关闭OLED显示

void OLED\_Display\_Off(void)

{

    oled\_write\_byte(OLED\_CMD, 0x8D); oled\_write\_byte(OLED\_CMD, 0x10); //Set Charge Pump Disable

    oled\_write\_byte(OLED\_CMD, 0xAE); //Display OFF

}

//设置光标起点,横轴x,纵轴y

void OLED\_SetPos(uint8\_t x, uint8\_t y)

{

    oled\_write\_byte(OLED\_CMD, (0xB0+y)); //set page address y

    oled\_write\_byte(OLED\_CMD, ((x&0xF0)>>4)|0x10); //set column high address

    oled\_write\_byte(OLED\_CMD, (x&0x0F)); //set column low address

}

//更新显存到LCD

void OLED\_Refresh\_Gram(void)

{

    uint8\_t i,n;

    for(i=0;i<8;i++)

    {

        OLED\_SetPos(0,i);

        for(n=0;n<128;n++)

        {

            oled\_write\_byte(OLED\_DATA, OLED\_GRAM[n][i]);

        }

    }

}

//清屏

void OLED\_Clear(void)

{

    uint8\_t i,n;

    for(i=0;i<8;i++)

    {

        for(n=0;n<128;n++)

        {

            OLED\_GRAM[n][i]=0x00;

        }

    }

    //OLED\_Refresh\_Gram();//更新显示

}

//画点

void OLED\_DrawPoint(uint8\_t x, uint8\_t y, uint8\_t dot)

{

    uint8\_t page,row;

    if((x>127)||(y>63)) {return;}

    page = y/8;

    row = y%8;

    if(dot==1)

    {

        OLED\_GRAM[x][page] |= (1<<row);

    }

    else if(dot==0)

    {

        OLED\_GRAM[x][page] &= ~(1<<row);

    }

}

//显示一个字符

void OLED\_ShowChar(uint8\_t x, uint8\_t y, uint8\_t chr)

{

    uint16\_t tmp;

    uint8\_t i, j, index;

    index = chr-' ';

    for(i=0;i<6;i++)

    {

        tmp = (uint16\_t)((ascii\_1206[index][2\*i]<<8)+ascii\_1206[index][2\*i+1]);

        for(j=0;j<12;j++)

        {

            if(tmp&0x8000) {OLED\_DrawPoint(x+i, y+j, 1);}

            else {OLED\_DrawPoint(x+i, y+j, 0);}

            tmp<<=1;

        }

    }

}

//显示字符串

void OLED\_ShowString(uint8\_t x, uint8\_t y, uint8\_t \*str)

{

    uint16\_t n=0;

    while(str[n]!='\0')

    {

        OLED\_ShowChar(x, y, str[n]);

        x+=6;

        if(x>120){x=0;y+=12;}

        n++;

    }

}

//OLED格式化输出

void OLED\_printf(uint8\_t x, uint8\_t y, const char \*fmt,...)

{

    static uint8\_t oled\_buf[128]={0};

    static va\_list ap;

    static uint16\_t len=0;

    va\_start(ap, fmt);

    len = vsprintf((char \*)oled\_buf, fmt, ap);

    va\_end(ap);

    oled\_buf[len] = '\0';

    OLED\_ShowString(x, y, oled\_buf);

}

//显示LOGO图片 ~90msunsigned char

void OLED\_LOGO(void)

{

    uint8\_t tmp;

    uint8\_t x, y, i;

    OLED\_Clear();

    for(y=0;y<64;y+=8)

    {

        for(x=0;x<128;x++)

        {

            tmp = LOGO2\_DUT0BUG[x][y/8];

            for(i=0;i<8;i++)

            {

                if(tmp&0x80) {OLED\_DrawPoint(x, y+i, 1);}

                else {OLED\_DrawPoint(x, y+i, 0);}

                tmp<<=1;

            }

        }

    }

    OLED\_Refresh\_Gram();//更新显示

}

void OLED\_BMP(const unsigned char bmp[128][8])//显示某张图片，图片需要在oled.h中给个extern声明

{

    uint8\_t tmp;

    uint8\_t x, y, i;

    OLED\_Clear();

    for(y=0;y<64;y+=8)

    {

        for(x=0;x<128;x++)

        {

            tmp = bmp[x][y/8];

            for(i=0;i<8;i++)

            {

                if(tmp&0x80) {OLED\_DrawPoint(x, y+i, 1);}

                else {OLED\_DrawPoint(x, y+i, 0);}

                tmp<<=1;

            }

        }

    }

    OLED\_Refresh\_Gram();//更新显示

}

ls53l1m.h

// lasar ranging module

#ifndef LS53L1M

#define LS53L1M

#include "main.h"

extern UART\_HandleTypeDef huart1;

extern void request\_distance(void);

#endif

ls53l1m.c

// lasar ranging module

#include "ls53l1m.h"

#define REQ\_LEN 9

void request\_distance()

{

  uint8\_t data[REQ\_LEN];

  data[0]=0x51;

  data[1]=0x0b;

  data[2]=0x00;

  data[3]=0x01;

  data[4]=0x00;

  data[5]=0x05;

  data[6]=0x02;

  data[7]=0x00;

  data[8]=0x64;

  HAL\_UART\_Transmit(&huart1,data,REQ\_LEN,0xfff);

}

delay.h

#ifndef DELAY\_H

#define DELAY\_H

#include "main.h"

extern TIM\_HandleTypeDef htim11;

extern void delay\_us(uint16\_t us);

extern void delay\_ms(uint16\_t ms);

#endif

delay.c

#include "delay.h"

void delay\_us(uint16\_t us)

{

  \_\_HAL\_TIM\_SET\_COUNTER(&htim11,0);

  while(\_\_HAL\_TIM\_GET\_COUNTER(&htim11)<us);

}

void delay\_ms(uint16\_t ms)

{

  while(ms--)delay\_us(1000);

}

print.h

#ifndef \_\_PRINT\_H\_\_

#define \_\_PRINT\_H\_\_

#include "main.h"

#include "string.h"

#include "stdio.h"

#include "stdarg.h"

#include "usart.h"

extern void usart\_printf(UART\_HandleTypeDef \*huart,char \*fmt, ...);

extern void drawPtintf(UART\_HandleTypeDef \*huart, uint8\_t name, float value);

extern void print\_string(UART\_HandleTypeDef \*huart,char \*str);

#endif

print.c

#include "print.h"

void usart\_printf(UART\_HandleTypeDef \*huart,char \*fmt, ...)

{

  static uint8\_t tx\_buf[300] = {0};

  static va\_list ap;

  static uint16\_t len;

  va\_start(ap, fmt);

  len = vsprintf((char \*)tx\_buf, fmt, ap);

  va\_end(ap);

  HAL\_UART\_Transmit(huart,tx\_buf,len,500);

}

void drawPtintf(UART\_HandleTypeDef \*huart, uint8\_t name, float value)

{

  char buf[20];

  sprintf(buf, "Line%d=%.2f,", name, value);

  usart\_printf(huart,buf);

}

void print\_string(UART\_HandleTypeDef \*huart,char \*str)

{

  usart\_printf(huart,str);

}

ctrl.h

#ifndef CTRL\_H

#define CTRL\_H

#include "main.h"

#include "ls53l1m.h"

#include "oled.h"

#include "draw.h"

extern TIM\_HandleTypeDef htim1;

extern UART\_HandleTypeDef huart1;

extern TIM\_HandleTypeDef htim10;

uint8\_t pwm\_cvt(uint8\_t dir)

{

  if(dir==1)return 0;

  else return 1;

}

#endif

ctrl.c

#include "ctrl.h"

#define BUF\_LEN 11

#define MOTOR\_STEP 5

uint16\_t tim1clk=0; //0~999

uint16\_t pwm\_h=2500; //500~2500

uint8\_t pwm\_dir=0; // 1:++ , 0:--

uint8\_t ReceiveByte;

uint8\_t ReceiveBuff[BUF\_LEN]={0};

uint8\_t ReceiveCnt=0;

uint16\_t angle; // theortically 0~180

uint16\_t distance; // millimeter

uint16\_t result[100][2]; // [angle][distance]

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

  if(htim==&htim1) // 1kHz  NO-LOOGER THAN 1 MS

  {

    tim1clk++;

    tim1clk%=1000;

    if(tim1clk%1==1) //1kHz

    {

    }

    if(tim1clk%5==1) //200Hz

    {

      request\_distance();

      angle=(uint16\_t)(180-(1800.0\*(pwm\_h/20000.0)-45.0));

      result[(uint8\_t)(angle\*(5.0/9))][0]=angle;

      result[(uint8\_t)(angle\*(5.0/9))][1]=distance;

      if(result[(uint8\_t)(angle\*(5.0/9))][1]>=1000)result[(uint8\_t)(angle\*(5.0/9))][1]=1000;

      if(result[(uint8\_t)(angle\*(5.0/9))][1]<=100)result[(uint8\_t)(angle\*(5.0/9))][1]=100;

    }

    if(tim1clk%20==1) //50Hz

    {

      if(pwm\_dir==1)pwm\_h+=MOTOR\_STEP;else pwm\_h-=MOTOR\_STEP;

      if(pwm\_h<=500){pwm\_h=500;pwm\_dir=pwm\_cvt(pwm\_dir);}

      if(pwm\_h>=2500){pwm\_h=2500;pwm\_dir=pwm\_cvt(pwm\_dir);}

      \_\_HAL\_TIM\_SetCompare(&htim10,TIM\_CHANNEL\_1,pwm\_h);

    }

    if(tim1clk%50==1) //20Hz

    {

      // OLED\_Clear();

      // OLED\_printf(0,0,"range=%4dmm",distance);

      // for(uint8\_t i=0;i<100;i++)

      // {

      //   draw\_arc((uint16\_t)result[i][0],(uint16\_t)(result[i][1]/15.625)); // 500mm => 32 pixels

      // }

      // draw\_bgd();

      // OLED\_Refresh\_Gram();

    }

  }

}

void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart)

{

  if(huart==&huart1)

  {

    ReceiveBuff[ReceiveCnt]=ReceiveByte;

    ReceiveCnt++;  //save data into ReceiveBuff

    if(ReceiveCnt==BUF\_LEN)ReceiveCnt=0;

    HAL\_UART\_Receive\_IT(&huart1,&ReceiveByte,1);

  }

}

void UART\_IDLECallBack(UART\_HandleTypeDef \*huart)

{

  if(huart==&huart1)

  {

    if(\_\_HAL\_UART\_GET\_FLAG(&huart1,UART\_FLAG\_IDLE)!=RESET)  // IDLE interrupt happened

    {

      \_\_HAL\_UART\_CLEAR\_IDLEFLAG(&huart1); //clear IDLE flag

      // respond (do something)

      //HAL\_UART\_Transmit(&huart1,ReceiveBuff,BUF\_LEN,0xfff);

      distance=ReceiveBuff[8]<<8|ReceiveBuff[9];

      // clear the buff afterwards

      ReceiveCnt=0;

      for(uint8\_t i=0;i<BUF\_LEN;i++)  //clear buff

      {

          ReceiveBuff[i]=0;

      }

    }

  }

}

draw.h

#ifndef DRAW\_H

#define DRAW\_H

#include "main.h"

#include "oled.h"

#include "math.h"

#define DEG2RAD (3.141592653589793238462643/180.0)

// extern uint8\_t OLED\_GRAM[128][8];

void draw\_bgd(void);

void draw\_arc(uint16\_t angle,uint16\_t distance);

#endif

draw.c

#include "draw.h"

#define WIDTH 1 // angle

#define LENGTH 5 // parallel lines

int64\_t abs\_64(int64\_t num)

{

  if(num<0)return -num;

  else return num;

}

float\_t tan\_deg(float\_t deg)

{

  if(deg==90)return 1e10;

  else if(deg<2)return tan(2\*DEG2RAD);

  else if(deg>178)return tan(178\*DEG2RAD);

  else return tan(deg\*DEG2RAD);

}

void draw\_arc(uint16\_t angle,uint16\_t distance)

{

  for(uint8\_t x=0;x<128;x++)

  {

    for(uint8\_t y=0;y<64;y++)

    {

      if(

        (abs\_64((x-64)\*(x-64)+(y-64)\*(y-64)-distance\*distance) <= 35) &&

        ((y-64)/tan\_deg(angle+WIDTH)+64-LENGTH <= x) &&

        (x <= (y-64)/tan\_deg(angle-WIDTH)+64+LENGTH)

      )

      {

        OLED\_DrawPoint(x,y,1);

      }

    }

  }

}

void draw\_bgd()

{

  // outer circle

  OLED\_DrawPoint(0,59,1);

  OLED\_DrawPoint(0,60,1);

  OLED\_DrawPoint(0,61,1);

  OLED\_DrawPoint(0,62,1);

  OLED\_DrawPoint(0,63,1);

  OLED\_DrawPoint(1,52,1);

  OLED\_DrawPoint(1,53,1);

  OLED\_DrawPoint(1,54,1);

  OLED\_DrawPoint(2,48,1);

  OLED\_DrawPoint(2,49,1);

  OLED\_DrawPoint(3,44,1);

  OLED\_DrawPoint(3,45,1);

  OLED\_DrawPoint(4,42,1);

  OLED\_DrawPoint(5,39,1);

  OLED\_DrawPoint(6,37,1);

  OLED\_DrawPoint(7,35,1);

  OLED\_DrawPoint(8,33,1);

  OLED\_DrawPoint(9,31,1);

  OLED\_DrawPoint(10,30,1);

  OLED\_DrawPoint(11,28,1);

  OLED\_DrawPoint(12,27,1);

  OLED\_DrawPoint(13,25,1);

  OLED\_DrawPoint(14,24,1);

  OLED\_DrawPoint(15,23,1);

  OLED\_DrawPoint(16,22,1);

  OLED\_DrawPoint(22,16,1);

  OLED\_DrawPoint(23,15,1);

  OLED\_DrawPoint(24,14,1);

  OLED\_DrawPoint(25,13,1);

  OLED\_DrawPoint(27,12,1);

  OLED\_DrawPoint(28,11,1);

  OLED\_DrawPoint(30,10,1);

  OLED\_DrawPoint(31,9,1);

  OLED\_DrawPoint(33,8,1);

  OLED\_DrawPoint(35,7,1);

  OLED\_DrawPoint(37,6,1);

  OLED\_DrawPoint(39,5,1);

  OLED\_DrawPoint(42,4,1);

  OLED\_DrawPoint(44,3,1);

  OLED\_DrawPoint(45,3,1);

  OLED\_DrawPoint(48,2,1);

  OLED\_DrawPoint(49,2,1);

  OLED\_DrawPoint(52,1,1);

  OLED\_DrawPoint(53,1,1);

  OLED\_DrawPoint(54,1,1);

  OLED\_DrawPoint(59,0,1);

  OLED\_DrawPoint(60,0,1);

  OLED\_DrawPoint(61,0,1);

  OLED\_DrawPoint(62,0,1);

  OLED\_DrawPoint(63,0,1);

  OLED\_DrawPoint(64,0,1);

  OLED\_DrawPoint(65,0,1);

  OLED\_DrawPoint(66,0,1);

  OLED\_DrawPoint(67,0,1);

  OLED\_DrawPoint(68,0,1);

  OLED\_DrawPoint(69,0,1);

  OLED\_DrawPoint(74,1,1);

  OLED\_DrawPoint(75,1,1);

  OLED\_DrawPoint(76,1,1);

  OLED\_DrawPoint(79,2,1);

  OLED\_DrawPoint(80,2,1);

  OLED\_DrawPoint(83,3,1);

  OLED\_DrawPoint(84,3,1);

  OLED\_DrawPoint(86,4,1);

  OLED\_DrawPoint(89,5,1);

  OLED\_DrawPoint(91,6,1);

  OLED\_DrawPoint(93,7,1);

  OLED\_DrawPoint(95,8,1);

  OLED\_DrawPoint(97,9,1);

  OLED\_DrawPoint(98,10,1);

  OLED\_DrawPoint(100,11,1);

  OLED\_DrawPoint(101,12,1);

  OLED\_DrawPoint(103,13,1);

  OLED\_DrawPoint(104,14,1);

  OLED\_DrawPoint(105,15,1);

  OLED\_DrawPoint(106,16,1);

  OLED\_DrawPoint(112,22,1);

  OLED\_DrawPoint(113,23,1);

  OLED\_DrawPoint(114,24,1);

  OLED\_DrawPoint(115,25,1);

  OLED\_DrawPoint(116,27,1);

  OLED\_DrawPoint(117,28,1);

  OLED\_DrawPoint(118,30,1);

  OLED\_DrawPoint(119,31,1);

  OLED\_DrawPoint(120,33,1);

  OLED\_DrawPoint(121,35,1);

  OLED\_DrawPoint(122,37,1);

  OLED\_DrawPoint(123,39,1);

  OLED\_DrawPoint(124,42,1);

  OLED\_DrawPoint(125,44,1);

  OLED\_DrawPoint(125,45,1);

  OLED\_DrawPoint(126,48,1);

  OLED\_DrawPoint(126,49,1);

  OLED\_DrawPoint(127,52,1);

  OLED\_DrawPoint(127,53,1);

  OLED\_DrawPoint(127,54,1);

  // inner circle

  OLED\_DrawPoint(32,60,1);

  OLED\_DrawPoint(32,61,1);

  OLED\_DrawPoint(32,62,1);

  OLED\_DrawPoint(32,63,1);

  OLED\_DrawPoint(33,55,1);

  OLED\_DrawPoint(33,56,1);

  OLED\_DrawPoint(33,57,1);

  OLED\_DrawPoint(34,52,1);

  OLED\_DrawPoint(34,53,1);

  OLED\_DrawPoint(35,50,1);

  OLED\_DrawPoint(35,51,1);

  OLED\_DrawPoint(36,48,1);

  OLED\_DrawPoint(36,49,1);

  OLED\_DrawPoint(37,47,1);

  OLED\_DrawPoint(38,45,1);

  OLED\_DrawPoint(39,44,1);

  OLED\_DrawPoint(40,43,1);

  OLED\_DrawPoint(41,42,1);

  OLED\_DrawPoint(42,41,1);

  OLED\_DrawPoint(43,40,1);

  OLED\_DrawPoint(44,39,1);

  OLED\_DrawPoint(45,38,1);

  OLED\_DrawPoint(47,37,1);

  OLED\_DrawPoint(48,36,1);

  OLED\_DrawPoint(49,36,1);

  OLED\_DrawPoint(50,35,1);

  OLED\_DrawPoint(51,35,1);

  OLED\_DrawPoint(52,34,1);

  OLED\_DrawPoint(53,34,1);

  OLED\_DrawPoint(55,33,1);

  OLED\_DrawPoint(56,33,1);

  OLED\_DrawPoint(57,33,1);

  OLED\_DrawPoint(60,32,1);

  OLED\_DrawPoint(61,32,1);

  OLED\_DrawPoint(62,32,1);

  OLED\_DrawPoint(63,32,1);

  OLED\_DrawPoint(64,32,1);

  OLED\_DrawPoint(65,32,1);

  OLED\_DrawPoint(66,32,1);

  OLED\_DrawPoint(67,32,1);

  OLED\_DrawPoint(68,32,1);

  OLED\_DrawPoint(71,33,1);

  OLED\_DrawPoint(72,33,1);

  OLED\_DrawPoint(73,33,1);

  OLED\_DrawPoint(75,34,1);

  OLED\_DrawPoint(76,34,1);

  OLED\_DrawPoint(77,35,1);

  OLED\_DrawPoint(78,35,1);

  OLED\_DrawPoint(79,36,1);

  OLED\_DrawPoint(80,36,1);

  OLED\_DrawPoint(81,37,1);

  OLED\_DrawPoint(83,38,1);

  OLED\_DrawPoint(84,39,1);

  OLED\_DrawPoint(85,40,1);

  OLED\_DrawPoint(86,41,1);

  OLED\_DrawPoint(87,42,1);

  OLED\_DrawPoint(88,43,1);

  OLED\_DrawPoint(89,44,1);

  OLED\_DrawPoint(90,45,1);

  OLED\_DrawPoint(91,47,1);

  OLED\_DrawPoint(92,48,1);

  OLED\_DrawPoint(92,49,1);

  OLED\_DrawPoint(93,50,1);

  OLED\_DrawPoint(93,51,1);

  OLED\_DrawPoint(94,52,1);

  OLED\_DrawPoint(94,53,1);

  OLED\_DrawPoint(95,55,1);

  OLED\_DrawPoint(95,56,1);

  OLED\_DrawPoint(95,57,1);

  OLED\_DrawPoint(96,60,1);

  OLED\_DrawPoint(96,61,1);

  OLED\_DrawPoint(96,62,1);

  OLED\_DrawPoint(96,63,1);

  // line y=x

  OLED\_DrawPoint(0,0,1);

  OLED\_DrawPoint(1,1,1);

  OLED\_DrawPoint(2,2,1);

  OLED\_DrawPoint(3,3,1);

  OLED\_DrawPoint(4,4,1);

  OLED\_DrawPoint(5,5,1);

  OLED\_DrawPoint(6,6,1);

  OLED\_DrawPoint(7,7,1);

  OLED\_DrawPoint(8,8,1);

  OLED\_DrawPoint(9,9,1);

  OLED\_DrawPoint(10,10,1);

  OLED\_DrawPoint(11,11,1);

  OLED\_DrawPoint(12,12,1);

  OLED\_DrawPoint(13,13,1);

  OLED\_DrawPoint(14,14,1);

  OLED\_DrawPoint(15,15,1);

  OLED\_DrawPoint(16,16,1);

  OLED\_DrawPoint(17,17,1);

  OLED\_DrawPoint(18,18,1);

  OLED\_DrawPoint(19,19,1);

  OLED\_DrawPoint(20,20,1);

  OLED\_DrawPoint(21,21,1);

  OLED\_DrawPoint(22,22,1);

  OLED\_DrawPoint(23,23,1);

  OLED\_DrawPoint(24,24,1);

  OLED\_DrawPoint(25,25,1);

  OLED\_DrawPoint(26,26,1);

  OLED\_DrawPoint(27,27,1);

  OLED\_DrawPoint(28,28,1);

  OLED\_DrawPoint(29,29,1);

  OLED\_DrawPoint(30,30,1);

  OLED\_DrawPoint(31,31,1);

  OLED\_DrawPoint(32,32,1);

  OLED\_DrawPoint(33,33,1);

  OLED\_DrawPoint(34,34,1);

  OLED\_DrawPoint(35,35,1);

  OLED\_DrawPoint(36,36,1);

  OLED\_DrawPoint(37,37,1);

  OLED\_DrawPoint(38,38,1);

  OLED\_DrawPoint(39,39,1);

  OLED\_DrawPoint(40,40,1);

  OLED\_DrawPoint(41,41,1);

  OLED\_DrawPoint(42,42,1);

  OLED\_DrawPoint(43,43,1);

  OLED\_DrawPoint(44,44,1);

  OLED\_DrawPoint(45,45,1);

  OLED\_DrawPoint(46,46,1);

  OLED\_DrawPoint(47,47,1);

  OLED\_DrawPoint(48,48,1);

  OLED\_DrawPoint(49,49,1);

  OLED\_DrawPoint(50,50,1);

  OLED\_DrawPoint(51,51,1);

  OLED\_DrawPoint(52,52,1);

  OLED\_DrawPoint(53,53,1);

  OLED\_DrawPoint(54,54,1);

  OLED\_DrawPoint(55,55,1);

  OLED\_DrawPoint(56,56,1);

  OLED\_DrawPoint(57,57,1);

  OLED\_DrawPoint(58,58,1);

  OLED\_DrawPoint(59,59,1);

  OLED\_DrawPoint(60,60,1);

  OLED\_DrawPoint(61,61,1);

  OLED\_DrawPoint(62,62,1);

  OLED\_DrawPoint(63,63,1);

  // line y=-x+128

  OLED\_DrawPoint(65,63,1);

  OLED\_DrawPoint(66,62,1);

  OLED\_DrawPoint(67,61,1);

  OLED\_DrawPoint(68,60,1);

  OLED\_DrawPoint(69,59,1);

  OLED\_DrawPoint(70,58,1);

  OLED\_DrawPoint(71,57,1);

  OLED\_DrawPoint(72,56,1);

  OLED\_DrawPoint(73,55,1);

  OLED\_DrawPoint(74,54,1);

  OLED\_DrawPoint(75,53,1);

  OLED\_DrawPoint(76,52,1);

  OLED\_DrawPoint(77,51,1);

  OLED\_DrawPoint(78,50,1);

  OLED\_DrawPoint(79,49,1);

  OLED\_DrawPoint(80,48,1);

  OLED\_DrawPoint(81,47,1);

  OLED\_DrawPoint(82,46,1);

  OLED\_DrawPoint(83,45,1);

  OLED\_DrawPoint(84,44,1);

  OLED\_DrawPoint(85,43,1);

  OLED\_DrawPoint(86,42,1);

  OLED\_DrawPoint(87,41,1);

  OLED\_DrawPoint(88,40,1);

  OLED\_DrawPoint(89,39,1);

  OLED\_DrawPoint(90,38,1);

  OLED\_DrawPoint(91,37,1);

  OLED\_DrawPoint(92,36,1);

  OLED\_DrawPoint(93,35,1);

  OLED\_DrawPoint(94,34,1);

  OLED\_DrawPoint(95,33,1);

  OLED\_DrawPoint(96,32,1);

  OLED\_DrawPoint(97,31,1);

  OLED\_DrawPoint(98,30,1);

  OLED\_DrawPoint(99,29,1);

  OLED\_DrawPoint(100,28,1);

  OLED\_DrawPoint(101,27,1);

  OLED\_DrawPoint(102,26,1);

  OLED\_DrawPoint(103,25,1);

  OLED\_DrawPoint(104,24,1);

  OLED\_DrawPoint(105,23,1);

  OLED\_DrawPoint(106,22,1);

  OLED\_DrawPoint(107,21,1);

  OLED\_DrawPoint(108,20,1);

  OLED\_DrawPoint(109,19,1);

  OLED\_DrawPoint(110,18,1);

  OLED\_DrawPoint(111,17,1);

  OLED\_DrawPoint(112,16,1);

  OLED\_DrawPoint(113,15,1);

  OLED\_DrawPoint(114,14,1);

  OLED\_DrawPoint(115,13,1);

  OLED\_DrawPoint(116,12,1);

  OLED\_DrawPoint(117,11,1);

  OLED\_DrawPoint(118,10,1);

  OLED\_DrawPoint(119,9,1);

  OLED\_DrawPoint(120,8,1);

  OLED\_DrawPoint(121,7,1);

  OLED\_DrawPoint(122,6,1);

  OLED\_DrawPoint(123,5,1);

  OLED\_DrawPoint(124,4,1);

  OLED\_DrawPoint(125,3,1);

  OLED\_DrawPoint(126,2,1);

  OLED\_DrawPoint(127,1,1);

  // line x=64

  OLED\_DrawPoint(64,0,1);

  OLED\_DrawPoint(64,1,1);

  OLED\_DrawPoint(64,2,1);

  OLED\_DrawPoint(64,3,1);

  OLED\_DrawPoint(64,4,1);

  OLED\_DrawPoint(64,5,1);

  OLED\_DrawPoint(64,6,1);

  OLED\_DrawPoint(64,7,1);

  OLED\_DrawPoint(64,8,1);

  OLED\_DrawPoint(64,9,1);

  OLED\_DrawPoint(64,10,1);

  OLED\_DrawPoint(64,11,1);

  OLED\_DrawPoint(64,12,1);

  OLED\_DrawPoint(64,13,1);

  OLED\_DrawPoint(64,14,1);

  OLED\_DrawPoint(64,15,1);

  OLED\_DrawPoint(64,16,1);

  OLED\_DrawPoint(64,17,1);

  OLED\_DrawPoint(64,18,1);

  OLED\_DrawPoint(64,19,1);

  OLED\_DrawPoint(64,20,1);

  OLED\_DrawPoint(64,21,1);

  OLED\_DrawPoint(64,22,1);

  OLED\_DrawPoint(64,23,1);

  OLED\_DrawPoint(64,24,1);

  OLED\_DrawPoint(64,25,1);

  OLED\_DrawPoint(64,26,1);

  OLED\_DrawPoint(64,27,1);

  OLED\_DrawPoint(64,28,1);

  OLED\_DrawPoint(64,29,1);

  OLED\_DrawPoint(64,30,1);

  OLED\_DrawPoint(64,31,1);

  OLED\_DrawPoint(64,32,1);

  OLED\_DrawPoint(64,33,1);

  OLED\_DrawPoint(64,34,1);

  OLED\_DrawPoint(64,35,1);

  OLED\_DrawPoint(64,36,1);

  OLED\_DrawPoint(64,37,1);

  OLED\_DrawPoint(64,38,1);

  OLED\_DrawPoint(64,39,1);

  OLED\_DrawPoint(64,40,1);

  OLED\_DrawPoint(64,41,1);

  OLED\_DrawPoint(64,42,1);

  OLED\_DrawPoint(64,43,1);

  OLED\_DrawPoint(64,44,1);

  OLED\_DrawPoint(64,45,1);

  OLED\_DrawPoint(64,46,1);

  OLED\_DrawPoint(64,47,1);

  OLED\_DrawPoint(64,48,1);

  OLED\_DrawPoint(64,49,1);

  OLED\_DrawPoint(64,50,1);

  OLED\_DrawPoint(64,51,1);

  OLED\_DrawPoint(64,52,1);

  OLED\_DrawPoint(64,53,1);

  OLED\_DrawPoint(64,54,1);

  OLED\_DrawPoint(64,55,1);

  OLED\_DrawPoint(64,56,1);

  OLED\_DrawPoint(64,57,1);

  OLED\_DrawPoint(64,58,1);

  OLED\_DrawPoint(64,59,1);

  OLED\_DrawPoint(64,60,1);

  OLED\_DrawPoint(64,61,1);

  OLED\_DrawPoint(64,62,1);

  OLED\_DrawPoint(64,63,1);

}

main.c

/\* USER CODE BEGIN Header \*/

/\*\*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @file           : main.c

  \* @brief          : Main program body

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @attention

  \*

  \* <h2><center>&copy; Copyright (c) 2022 STMicroelectronics.

  \* All rights reserved.</center></h2>

  \*

  \* This software component is licensed by ST under BSD 3-Clause license,

  \* the "License"; You may not use this file except in compliance with the

  \* License. You may obtain a copy of the License at:

  \*                        opensource.org/licenses/BSD-3-Clause

  \*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "i2c.h"

#include "tim.h"

#include "usart.h"

#include "gpio.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include "oled.h"

#include "draw.h"

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

extern uint8\_t ReceiveByte;

extern uint16\_t distance;

extern uint16\_t result[100][2];

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

  \* @brief  The application entry point.

  \* @retval int

  \*/

int main(void)

{

  /\* USER CODE BEGIN 1 \*/

  /\* 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\_TIM11\_Init();

  MX\_TIM1\_Init();

  MX\_USART1\_UART\_Init();

  MX\_I2C1\_Init();

  MX\_TIM10\_Init();

  /\* USER CODE BEGIN 2 \*/

  HAL\_TIM\_Base\_Start\_IT(&htim1);

  HAL\_TIM\_Base\_Start(&htim10);

  HAL\_TIM\_PWM\_Start(&htim10,TIM\_CHANNEL\_1);

  HAL\_TIM\_Base\_Start(&htim11);

  HAL\_UART\_Receive\_IT(&huart1,&ReceiveByte,1);

  \_\_HAL\_UART\_ENABLE\_IT(&huart1,UART\_IT\_IDLE);

  OLED\_Init();

  /\* USER CODE END 2 \*/

  /\* Infinite loop \*/

  /\* USER CODE BEGIN WHILE \*/

  //\_\_HAL\_TIM\_SetCompare(&htim10,TIM\_CHANNEL\_1,2000);  // 2000/20000 = 10%

  while (1)

  {

    /\* USER CODE END WHILE \*/

    /\* USER CODE BEGIN 3 \*/

    OLED\_Clear();

    OLED\_printf(0,0,"range=%4dmm",distance);

    for(uint8\_t i=0;i<100;i++)

    {

      draw\_arc((uint16\_t)result[i][0],(uint16\_t)(result[i][1]/15.625)); // 500mm => 32 pixels

    }

    draw\_bgd();

    OLED\_Refresh\_Gram();

  }

  /\* USER CODE END 3 \*/

}

/\*\*

  \* @brief System Clock Configuration

  \* @retval None

  \*/

void SystemClock\_Config(void)

{

  RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

  RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

  /\*\* Configure the main internal regulator output voltage

  \*/

  \_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

  \_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE2);

  /\*\* Initializes the RCC Oscillators according to the specified parameters

  \* in the RCC\_OscInitTypeDef structure.

  \*/

  RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

  RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

  RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

  RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

  RCC\_OscInitStruct.PLL.PLLM = 16;

  RCC\_OscInitStruct.PLL.PLLN = 192;

  RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV4;

  RCC\_OscInitStruct.PLL.PLLQ = 4;

  if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

  {

    Error\_Handler();

  }

  /\*\* Initializes the CPU, AHB and APB buses clocks

  \*/

  RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

                              |RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

  RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

  RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

  RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

  RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

  if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)

  {

    Error\_Handler();

  }

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

  \* @brief  This function is executed in case of error occurrence.

  \* @retval None

  \*/

void Error\_Handler(void)

{

  /\* USER CODE BEGIN Error\_Handler\_Debug \*/

  /\* User can add his own implementation to report the HAL error return state \*/

  \_\_disable\_irq();

  while (1)

  {

  }

  /\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef  USE\_FULL\_ASSERT

/\*\*

  \* @brief  Reports the name of the source file and the source line number

  \*         where the assert\_param error has occurred.

  \* @param  file: pointer to the source file name

  \* @param  line: assert\_param error line source number

  \* @retval None

  \*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

  /\* USER CODE BEGIN 6 \*/

  /\* User can add his own implementation to report the file name and line number,

     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

  /\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

stm32f4xx\_it.c

/\* USER CODE BEGIN Header \*/

/\*\*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @file    stm32f4xx\_it.c

  \* @brief   Interrupt Service Routines.

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @attention

  \*

  \* <h2><center>&copy; Copyright (c) 2022 STMicroelectronics.

  \* All rights reserved.</center></h2>

  \*

  \* This software component is licensed by ST under BSD 3-Clause license,

  \* the "License"; You may not use this file except in compliance with the

  \* License. You may obtain a copy of the License at:

  \*                        opensource.org/licenses/BSD-3-Clause

  \*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "stm32f4xx\_it.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN TD \*/

/\* USER CODE END TD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

/\* USER CODE BEGIN PFP \*/

extern void UART\_IDLECallBack(UART\_HandleTypeDef \*huart);

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\* External variables --------------------------------------------------------\*/

extern TIM\_HandleTypeDef htim1;

extern TIM\_HandleTypeDef htim10;

extern UART\_HandleTypeDef huart1;

/\* USER CODE BEGIN EV \*/

/\* USER CODE END EV \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*           Cortex-M4 Processor Interruption and Exception Handlers          \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

  \* @brief This function handles Non maskable interrupt.

  \*/

void NMI\_Handler(void)

{

  /\* USER CODE BEGIN NonMaskableInt\_IRQn 0 \*/

  /\* USER CODE END NonMaskableInt\_IRQn 0 \*/

  /\* USER CODE BEGIN NonMaskableInt\_IRQn 1 \*/

  while (1)

  {

  }

  /\* USER CODE END NonMaskableInt\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles Hard fault interrupt.

  \*/

void HardFault\_Handler(void)

{

  /\* USER CODE BEGIN HardFault\_IRQn 0 \*/

  /\* USER CODE END HardFault\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_HardFault\_IRQn 0 \*/

    /\* USER CODE END W1\_HardFault\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles Memory management fault.

  \*/

void MemManage\_Handler(void)

{

  /\* USER CODE BEGIN MemoryManagement\_IRQn 0 \*/

  /\* USER CODE END MemoryManagement\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_MemoryManagement\_IRQn 0 \*/

    /\* USER CODE END W1\_MemoryManagement\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles Pre-fetch fault, memory access fault.

  \*/

void BusFault\_Handler(void)

{

  /\* USER CODE BEGIN BusFault\_IRQn 0 \*/

  /\* USER CODE END BusFault\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_BusFault\_IRQn 0 \*/

    /\* USER CODE END W1\_BusFault\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles Undefined instruction or illegal state.

  \*/

void UsageFault\_Handler(void)

{

  /\* USER CODE BEGIN UsageFault\_IRQn 0 \*/

  /\* USER CODE END UsageFault\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_UsageFault\_IRQn 0 \*/

    /\* USER CODE END W1\_UsageFault\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles System service call via SWI instruction.

  \*/

void SVC\_Handler(void)

{

  /\* USER CODE BEGIN SVCall\_IRQn 0 \*/

  /\* USER CODE END SVCall\_IRQn 0 \*/

  /\* USER CODE BEGIN SVCall\_IRQn 1 \*/

  /\* USER CODE END SVCall\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles Debug monitor.

  \*/

void DebugMon\_Handler(void)

{

  /\* USER CODE BEGIN DebugMonitor\_IRQn 0 \*/

  /\* USER CODE END DebugMonitor\_IRQn 0 \*/

  /\* USER CODE BEGIN DebugMonitor\_IRQn 1 \*/

  /\* USER CODE END DebugMonitor\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles Pendable request for system service.

  \*/

void PendSV\_Handler(void)

{

  /\* USER CODE BEGIN PendSV\_IRQn 0 \*/

  /\* USER CODE END PendSV\_IRQn 0 \*/

  /\* USER CODE BEGIN PendSV\_IRQn 1 \*/

  /\* USER CODE END PendSV\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles System tick timer.

  \*/

void SysTick\_Handler(void)

{

  /\* USER CODE BEGIN SysTick\_IRQn 0 \*/

  /\* USER CODE END SysTick\_IRQn 0 \*/

  HAL\_IncTick();

  /\* USER CODE BEGIN SysTick\_IRQn 1 \*/

  /\* USER CODE END SysTick\_IRQn 1 \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* STM32F4xx Peripheral Interrupt Handlers                                    \*/

/\* Add here the Interrupt Handlers for the used peripherals.                  \*/

/\* For the available peripheral interrupt handler names,                      \*/

/\* please refer to the startup file (startup\_stm32f4xx.s).                    \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

  \* @brief This function handles TIM1 update interrupt and TIM10 global interrupt.

  \*/

void TIM1\_UP\_TIM10\_IRQHandler(void)

{

  /\* USER CODE BEGIN TIM1\_UP\_TIM10\_IRQn 0 \*/

  /\* USER CODE END TIM1\_UP\_TIM10\_IRQn 0 \*/

  HAL\_TIM\_IRQHandler(&htim1);

  HAL\_TIM\_IRQHandler(&htim10);

  /\* USER CODE BEGIN TIM1\_UP\_TIM10\_IRQn 1 \*/

  /\* USER CODE END TIM1\_UP\_TIM10\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles USART1 global interrupt.

  \*/

void USART1\_IRQHandler(void)

{

  /\* USER CODE BEGIN USART1\_IRQn 0 \*/

  /\* USER CODE END USART1\_IRQn 0 \*/

  HAL\_UART\_IRQHandler(&huart1);

  /\* USER CODE BEGIN USART1\_IRQn 1 \*/

  UART\_IDLECallBack(&huart1);

  /\* USER CODE END USART1\_IRQn 1 \*/

}

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

cal\_circle.ipynb

# calculate a circle to display in OLED

import numpy as np

np.set\_printoptions(threshold = 1e6)

GRAM=np.zeros((128,64),dtype=np.int32)

cnt=0

for x in range(0,128):

    for y in range(0,64):

        if abs( (x-64)\*\*2+(y-64)\*\*2-64\*\*2 ) <= 30:

            GRAM[x][y]=1

            cnt+=1

            # print(x,y)

            print('  OLED\_DrawPoint({},{},1);'.format(x,y))

# print('all %d counts'%cnt)

**日志部分**

|  |
| --- |
| 2022年 7 月 15 日~8月7日（准备阶段） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 8 月 8 日（第1天） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 8 月 9 日（第2天） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 8 月 10 日（第3天） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 8 月 11 日（第4天） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 8 月 12日（第5天） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 8 月 13 日（第6天） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 8 月 14 日（第7天） |
| 不少于150字。此处简单写即可，详细过程一定要放到报告部分。 |
| 2022年 月 日（项目验收） |
|  |