**2022年第五届天津市大学生信息技术“新工科”工程实践创新技术竞赛**

**参赛作品文档**

**参赛作品名称：基于stm32的多功能智能门禁系统**

**学 校： 天津大学**

**学 院：精密仪器与光电子工程学院**

**团队名称： 鹏翔硅谷**

**指导教师： 许宝忠**

**参赛成员： 穆翔 黄思钦 张秋霏**

目 录

[目 录 2](#_Toc117804542)

[一、2022年第五届天津市大学生信息技术“新工科”工程实践创新技术竞赛登记表 4](#_Toc117804543)

[二、作品文档方案部分 5](#_Toc117804544)

[1.方案概述 5](#_Toc117804545)

[1.1门禁模式 5](#_Toc117804546)

[1.2防疫模式 5](#_Toc117804547)

[1.3人脸识别开锁模式 5](#_Toc117804548)

[2.作品的创新点 5](#_Toc117804549)

[2.1结构新 5](#_Toc117804550)

[2.2功能新 6](#_Toc117804551)

[2.3模式新 6](#_Toc117804552)

[3.系统实现原理 6](#_Toc117804553)

[3.1门禁模式 7](#_Toc117804554)

[3.2防疫模式 7](#_Toc117804555)

[3.3人脸识别开锁模式 7](#_Toc117804556)

[4.硬件设计 8](#_Toc117804557)

[4.1 基于STM32F103ZET6的战舰开发板 8](#_Toc117804558)

[4.1.1功能 8](#_Toc117804559)

[4.1.2性能 8](#_Toc117804560)

[4.2 0.96寸OLED显示屏 9](#_Toc117804561)

[4.2.1功能 9](#_Toc117804562)

[4.2.2性能 10](#_Toc117804563)

[4.3 LCD显示屏 10](#_Toc117804564)

[4.3.1功能 11](#_Toc117804565)

[4.3.2性能 11](#_Toc117804566)

[4.4 OV7670摄像头模块 11](#_Toc117804567)

[4.4.1功能 11](#_Toc117804568)

[4.4.2性能 11](#_Toc117804569)

[4.4 AS608指纹模块 12](#_Toc117804570)

[4.4.1 功能 12](#_Toc117804571)

[4.4.2 性能 12](#_Toc117804572)

[4.5 ATK\_BLE01蓝牙模块 13](#_Toc117804573)

[4.5.1功能 13](#_Toc117804574)

[4.5.2 性能 14](#_Toc117804575)

[4.6 SIPEED MAIX DOCK k210识别模块 15](#_Toc117804576)

[4.6.1功能 15](#_Toc117804577)

[4.6.2性能 16](#_Toc117804578)

[5.软件设计 16](#_Toc117804579)

[5.1单片机部分 16](#_Toc117804580)

[5.1.1照相机模块 16](#_Toc117804581)

[5.1.2 OLED模块 21](#_Toc117804582)

[5.1.3指纹模块 25](#_Toc117804583)

[5.1.4 LCD模块 29](#_Toc117804584)

[5.1.5蓝牙模块 33](#_Toc117804585)

[5.1.6 K210模块 37](#_Toc117804586)

[5.2 上位机部分 44](#_Toc117804587)

[5.2.1 手机软件 44](#_Toc117804588)

[6.系统测试及结果 45](#_Toc117804589)

[7.程序源代码 45](#_Toc117804590)

[三、照片及说明 49](#_Toc117804591)

[1. 基于stm32的多功能智能门禁系统整体外观 49](#_Toc117804592)

[2. 各模式效果图 50](#_Toc117804593)

一、2022年第五届天津市大学生信息技术“新工科”工程实践创新技术竞赛登记表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题目 | 基于stm32的多功能智能门禁系统 | | | | |
| 学校及学院 | 天津大学精密仪器与光电子工程学院 | | | | |
| 学生姓名1 | 穆翔 | 专业班级 | 工程科学实验班 | 联系方式  （手机、邮箱） | 18729869038 |
| 学生姓名2 | 张秋霏 | 专业班级 | 工程科学实验班 | 联系方式  （手机、邮箱） | 16622818823 |
| 学生姓名3 | 黄思钦 | 专业班级 | 电子科学与技术（光电子）1班 | 联系方式  （手机、邮箱） | 13307900285 |
| 指导教师 | 许宝忠 | | | | |
| 作品关键词 | 人脸识别，口罩识别，指纹识别，门禁，警报，图像采集，物联网，实时传输 | | | | |
| 作品摘要 | 随着人们生活水平和经济能力的逐步提升，自身的财产安全保护逐渐成为了当今社会的重要议题。而传统的简易锁头，存在许多的安全风险以及使用上的不便利。同时疫情期间，社会对于防控监管有着巨大需求，而传统的人工监管效率不高且费时费力。  受到共享单车车锁以及面部检测设备的自动启发，本款产品基于stm32单片机和K210芯片，使用OLED显示屏、TFTLCD显示屏、OV7607摄像头、AS608指纹模块、蜂鸣器、蓝牙模块等元器件，开发了智能门禁系统。门禁模式提供多种身份识别与安全防盗措施，大大提高装置的安全程度，满足了对个人财产进行有利保护的广大需求。防疫模式与人脸识别模式提供了在戴口罩情况下对来访人员身份识别以及检测口罩佩戴与否两个功能。减小了疫情期间监管人员工作强度，节约人力物力，提高了生产效率。 | | | | |
| 主要参考  文献 | 无 | | | | |
| 备注 | 文档要求思路清晰、描述详细而明确、无错别字、图文并茂。 | | | | |
| * 特别   声明 | **提交内容均原创，作品属于参赛选手，大赛组委会有发布权和使用权。** | | | | |

二、作品文档方案部分

# 1.方案概述

多功能门禁系统具有三个工作模式。

1.1门禁模式

使用OLED显示屏、TFTLCD显示屏、OV7607摄像头、AS608指纹模块、蜂鸣器等元器件，可实现采集和录入指纹，并实现对来客指纹的采集、录入，将来客指纹与原指纹对比。若经过对比是已录入指纹，则通过，若经对比判定为陌生指纹，则蜂鸣器鸣叫示警，同时摄像头拍下陌生访客照片，并存入SD卡内，同时发送至上位机（手机或者电脑）。方便管理者实时查看访客信息。

1.2防疫模式

通过遥控器切换到防疫模式，系统打开摄像头并实时显示取景画面，识别画面中的人脸并检测其口罩佩戴情况，一旦识别到未佩戴口罩情况，系统发出声音提醒，并闪烁灯光。防疫模式适合在早高峰之后，从门禁模式一键切换，提醒出门的员工按规佩戴口罩。

1.3人脸识别开锁模式

若疫情严重，需要尽量减少接触，可切换人脸识别门禁模式，系统识别来访者人脸信息并与数据库中的人脸信息做对比，若正确则显示通过信息，若未检测出人脸，则识别为陌生访客，系统将提取访客人脸信息存储至sd卡，发送至上位机，方便管理者判断；同时还会发出警报，提醒屋内人员。

# 2.作品的创新点

2.1结构新

创新性地将传统门禁的指纹识别，人脸识别等功能，结合到一台设备；此外，本产品在STM32F103ZET6芯片控制的基础上增加了K210芯片辅助计算识别，提高了系统的工作效率。相比传统门禁系统具有更加复杂的结构和更先进的功能。

2.2功能新

创新性地实现了功能的三合一，且支持远程切换模式，门禁模式下，可依据疫情防控需要选择人脸识别和指纹识别，（当然，指纹识别算法更加成熟，精度相对更高）；产品采集的人脸信息可以实时上传上位机电脑或者手机，管理者可随时查看，相比于传统门禁系统非开即关，有物联网功能加持的本产品有更高的灵活性和容错率。

2.3模式新

考虑到门禁系统只能在早上上工和下午退工时使用，本产品创新性地高效利用了时间的剩余价值，转变模式后可在正常工作时间提醒外出人员戴口罩，大大提高了产品的使用效率和使用场景。

# 3.系统实现原理

该系统依靠门禁模式、防疫模式、人脸识别开锁模式三大功能有机结合，模块实现上述功能。

****

3.1门禁模式

通过AS608指纹模块，实现对指纹信息的收集与比对，实现对身份信息的验证；实现对指纹信息的添加与删除。使用OLED显示屏显示系统状态以及简单操作指示，应用于管理员端。使用TFTLCD显示屏显示实时信息以及详细的使用指示，应用于用户端，以指导用户操作正确。通过OV7607摄像头，实现在身份验证失败后面部信息的获取。通过蜂鸣器，实现对于身份验证正误的提示。通过蓝牙模块，实现面部图像信息从单片机到上位机的传输。

该模式下有信息编辑和身份验证两大功能。

信息编辑中，可以录入新指纹信息（录入两次，对比有误，再次录入；对比无误后，保存为一则指纹信息，避免误差）；删除旧有指纹。

身份验证中，指纹模块对用户指纹进行扫描，如与已录入指纹相符，则文字提示正确，LED亮绿灯，蜂鸣器发声；如与已录入指纹不符，则文字提示错误，，蜂鸣器发声，照相机获取用户面部信息，蓝牙模块上传面部信息文件至上位机。

以上每一步操作，OLED与TFTLCD均有文字提示。

3.2防疫模式

通过K210模块，实现对人面部信息的分析，判断口罩佩戴与否。通过蜂鸣器，实现对于身份验证正误的提示。

照相机获取用户面部信息，经由K210分析判断，如佩戴口罩，不做特殊反馈；如未佩戴口罩，蜂鸣器发声，LED亮红灯。

以上每一步操作，TFTLCD均有文字提示。

3.3人脸识别开锁模式

通过K210模块，实现对人面部信息的分析，判断用户身份。通过OV7607摄像头，实现在身份验证失败后面部信息的获取。通过蜂鸣器，实现对于身份验证正误的提示。通过蓝牙模块，实现面部图像信息从单片机到上位机的传输。

该模式下有信息编辑和身份验证两大功能。

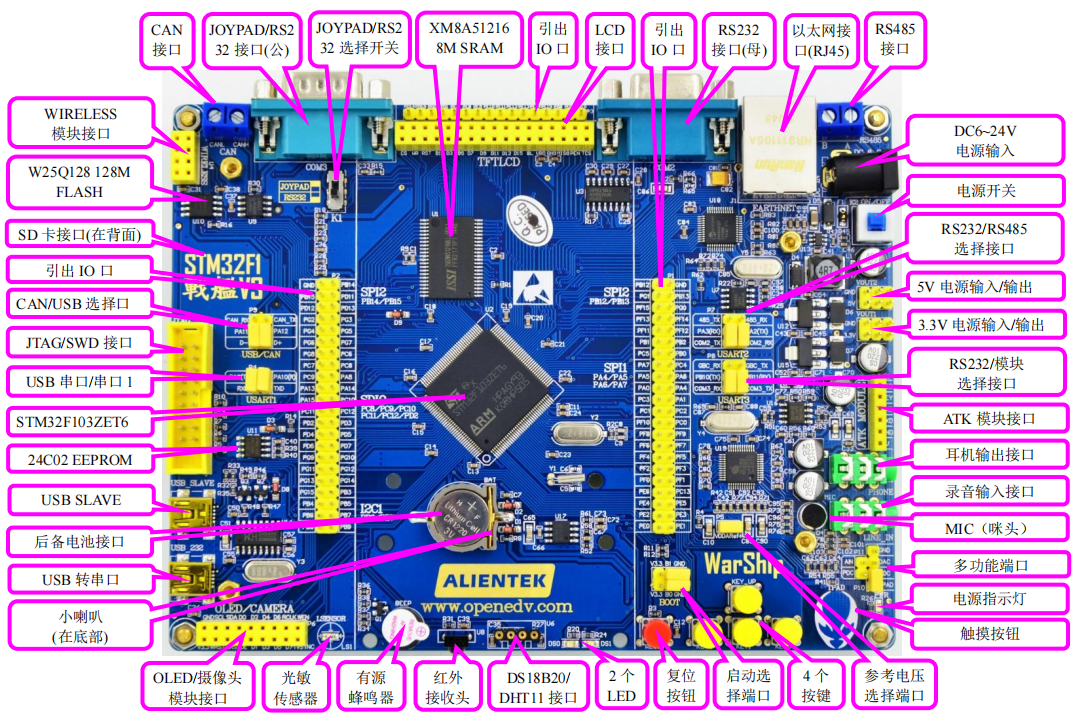
信息编辑中，可以录入新面部信息（录入两次，对比有误，再次录入；对比无误后，保存为一则面部信息，避免误差）；删除旧有面部信息。

身份验证中，K210模块对用户面部进行扫描，如与已录入面部信息相符，则文字提示正确，LED亮绿灯，蜂鸣器发声；如与已录入面部信息不符，则文字提示错误，蜂鸣器发声，照相机获取用户面部信息，蓝牙模块上传面部信息文件至上位机。

以上每一步操作，TFTLCD均有文字提示。

# 4.硬件设计

4.1 基于STM32F103ZET6的战舰开发板

****

4.1.1功能

主控制器，控制各个模块的工作配合；供电和连接器，给所有模块供电，连接K210芯片和副摄像头与副屏；信号发射器，发送图片给上位机；程序存储器：储存所有程序。

4.1.2性能

4.1.2.1存储

拥有W25Q128 128M FLASH闪存芯片容量为 128Mbit，也就是 16M 字节，可用于存储字库和其他用户数据，满足大容量数据存储要求。兼容SD卡，可以存储拍摄的照片。

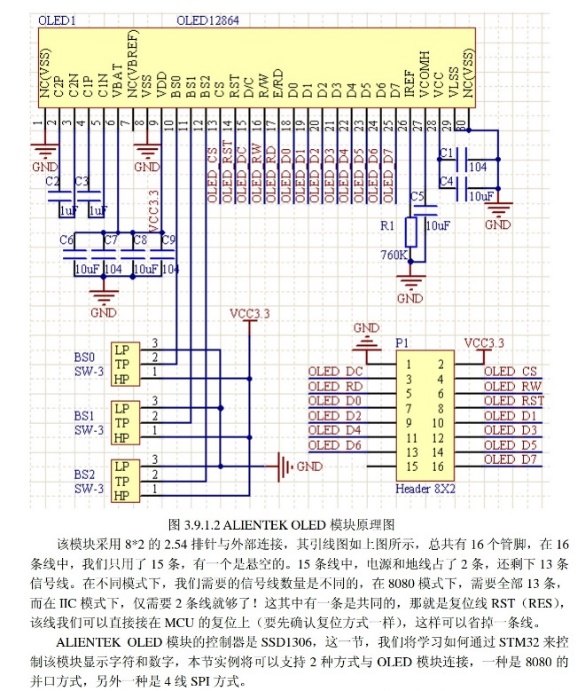
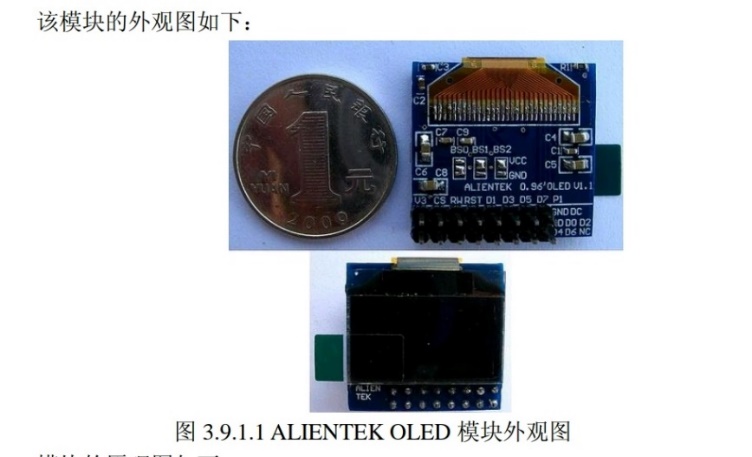
4.1.2.2接口丰富

提供十来种标准接口，可以方便的进行各种外设的实验和开发。

4.2 0.96寸OLED显示屏

4.2.1功能

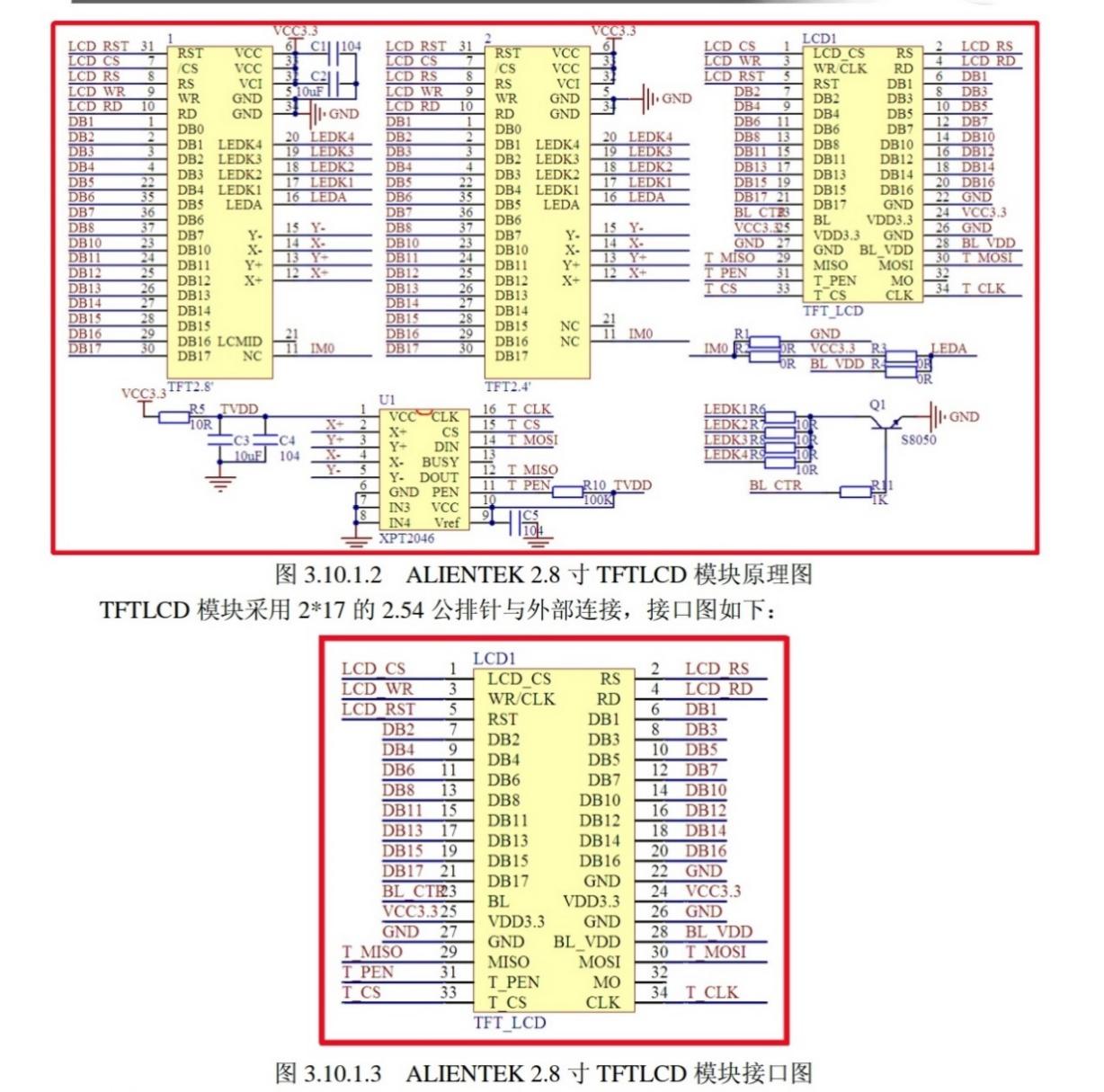
显示指纹系统状态及简单操作指示，与指纹模块配合。

4.2.2性能

能实时显示指纹系统提示信息，提示错误信息，指导使用者合规操作。

4.3 LCD显示屏



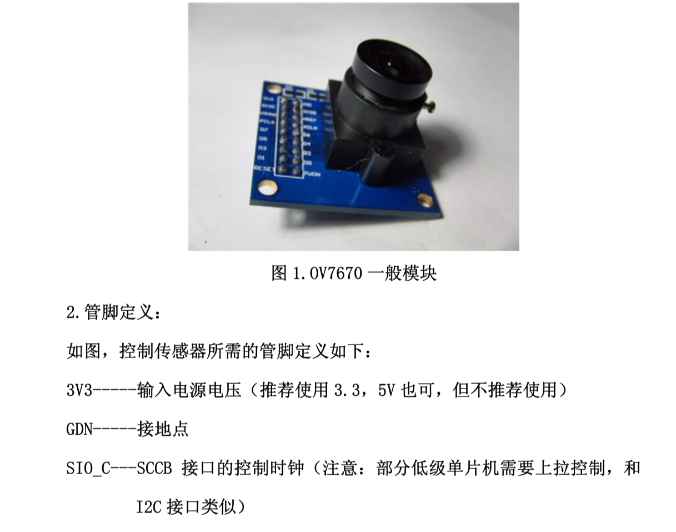
4.3.1功能

在调用摄像头时实时显示摄像头拍摄内容，防疫模式下作为电子日历显示时间日期，也可自定义显示内容；显示控制时的所有提示信息。

4.3.2性能

能长时间亮屏；能够支持240\*320画面30帧以上实时显示；色彩准确，反应灵敏。

4.4 OV7670摄像头模块

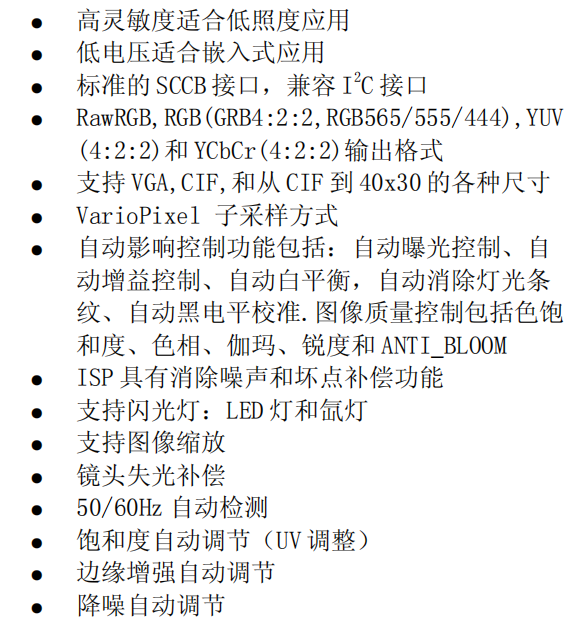


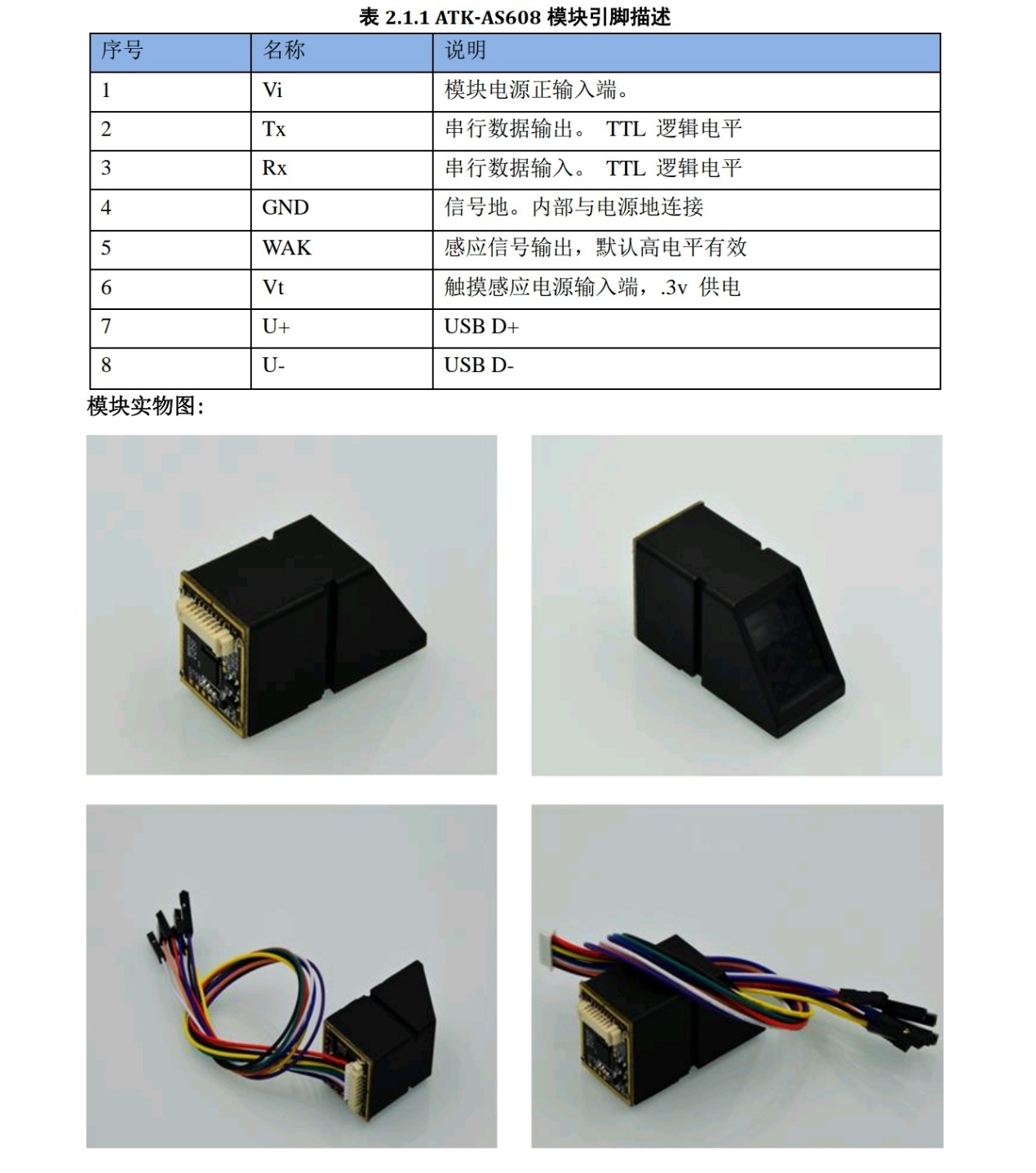
4.4.1功能

在得到指令后被调用，拍摄画面，存储进fifo中，等待调用。

4.4.2性能

响应迅速，自带fifo能更快存储图像数据。



4.4 AS608指纹模块

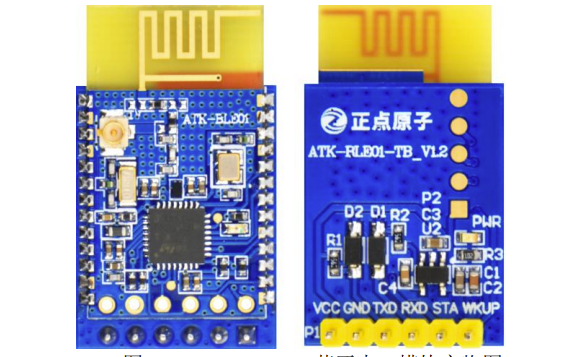
4.4.1 功能

指纹录入，指纹存储，指纹识别，指纹对比。

4.4.2 性能

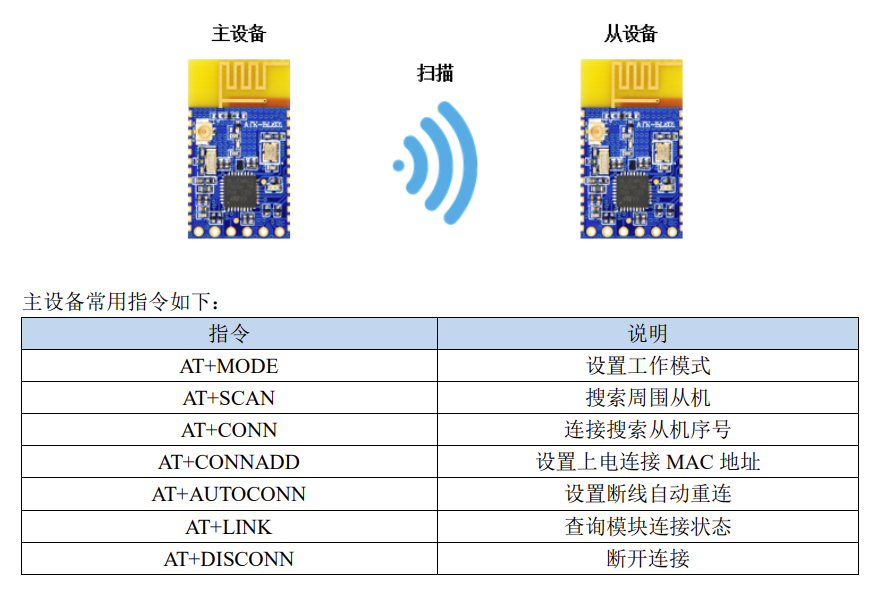
内置DSP运算单元，集成了指纹识别算法，能高效快速采集图像并识别指纹特征。配备了串口，USB通讯接口控制模块。

4.5 ATK\_BLE01蓝牙模块



4.5.1功能

主设备模式下，ATK-BLE01 模块可以与一个从设备模块进行连接。此模式下可以对周 围从设备搜索并且连接。也可以设置从设备的 MAC 地址，主设备模块上电会自动搜索该 MAC 地址的从设备并且进行连接。为保证连接的稳定性，预防断电、信号等异常问题导致 模块之间断开连接，可以开启断线重连功能，当异常干扰问题消失，模块工作环境恢复正常， 主设备会自动搜索刚刚断连的从设备，仅可能减少数据的丢失，提高系统稳定性。

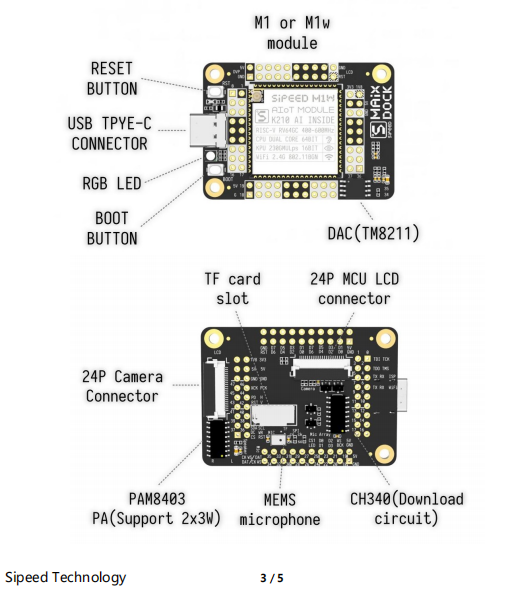


在本系统中，蓝牙模块主要承担单片机和电脑或者手机的通讯，传输图片。

4.5.2 性能



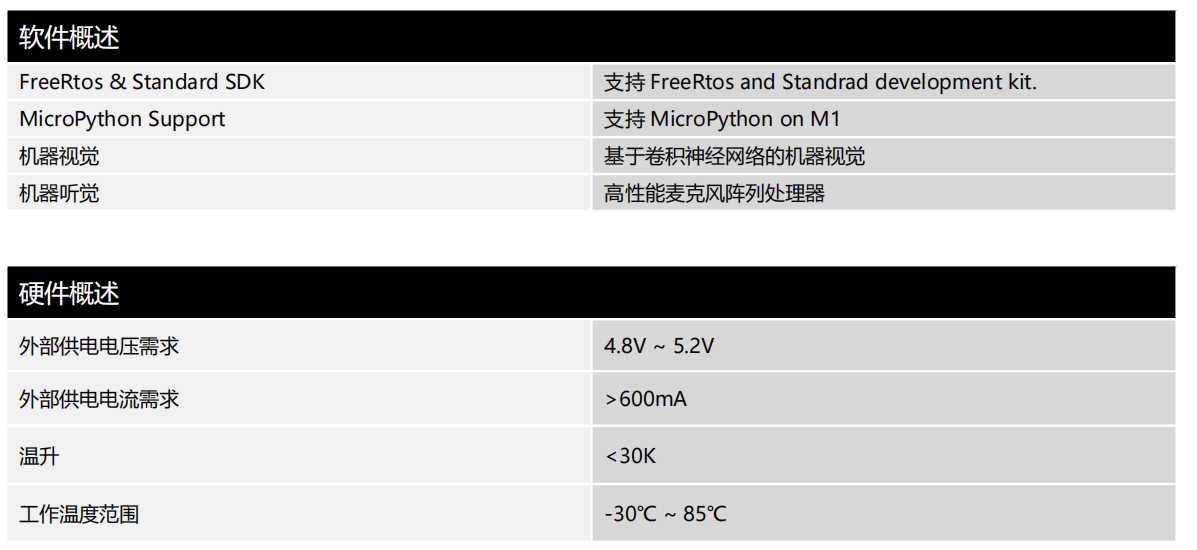
4.6 SIPEED MAIX DOCK k210识别模块



4.6.1功能

进行人脸识别辅助计算，提高系统整体算力，配合stm32完成人脸识别和人脸是否戴口罩识别。

4.6.2性能



# 5.软件设计

软件设计由单片机与上位机两大部分组成。单片机部分，stm32部分在KEIL开发环境下设计，K210在python开发环境下设计。上位机部分，应用了手机的蓝牙信息传输软件和电脑的蓝牙传输。

5.1单片机部分

5.1.1照相机模块

初始化ov7670，传感器读取图像数据，图像数据传入fifo，lcd读取fifo图象并且实时刷新，初始化sd卡，当收到指令，对屏幕指定区域截屏，进行bmp编码，将图像存储为rgb565格式，申请flash内存，将文件头信息和图像信息存入，最后使用fatfs系统在sd卡中新建文件夹和空文件，利用f\_write()函数将图片存入空bmp文件中，最后退出编码函数，lcd屏幕继续实时更新（单独照相机作用）

在系统中，拍照过程（bmp\_encode函数）由指纹识别出错来触发，因此封装整个“生成不重复新文件，lcd截屏，bmp编码，f\_write写入sd卡中”过程为photo\_taken函数，插入到指纹识别系统循环中。指纹模块持续扫描，若发现与已有指纹不同的指纹输入，就调动photo\_taken函数。其中三个核心函数：

5.1.1.1屏幕更新函数

实时显示摄像头拍到的画面，30帧，驱动lcd读取FIFO的数据

void OV7670\_camera\_refresh(void)

{

u32 j;

u16 color;

if(ov\_sta==2)

{

LCD\_Scan\_Dir(U2D\_L2R); //从上到下,从左到右

LCD\_SetCursor(0x00,0x0000); //设置光标位置

LCD\_WriteRAM\_Prepare(); //开始写入GRAM

OV7670\_CS=0;

OV7670\_RRST=0; //开始复位读指针

OV7670\_RCK=0;

OV7670\_RCK=1;

OV7670\_RCK=0;

OV7670\_RRST=1; //复位读指针结束

OV7670\_RCK=1;

for(j=0;j<76800;j++)

{

GPIOB->CRL=0X88888888;

OV7670\_RCK=0;

color=OV7670\_DATA; //读数据

OV7670\_RCK=1;

color<<=8;

OV7670\_RCK=0;

color|=OV7670\_DATA; //读数据

OV7670\_RCK=1;

GPIOB->CRL=0X33333333;

LCD\_WR\_DATA(color);

}

OV7670\_CS=1;

OV7670\_RCK=0;

OV7670\_RCK=1;

EXTI->PR=1<<15; //清除LINE8上的中断标志位

ov\_sta=0; //开始下一次采集

ov\_frame++;

LCD\_Scan\_Dir(DFT\_SCAN\_DIR); //恢复默认扫描方向

}

}

5.1.1.2文件自增函数

每次生成新的不重复的文件名，确保能连续拍照，这里利用了fatfs文件管理系统，这里用fa\_read还是fa\_creat\_new结果会大不相同，creat只能一直新建文件，但是跳不出循环，但是用read，利用了f\_open本身就能新增文件，read是用来判断是否生成了新的文件名而不是简单的读取。理解这段代码需要在fatfs官网学习每一句的含义。

void camera\_new\_pathname(u8 \*pname)

{

u8 res;

u16 index=0;

while(index<0XFFFF)

{

sprintf((char\*)pname,"0:PHOTO/PIC%05d.bmp",index);

res=f\_open(ftemp,(const TCHAR\*)pname,FA\_READ);//尝试打开这个文件

if(res==FR\_NO\_FILE)

break; //该文件名不存在=正是我们需要的.

index++;

}

}

5.1.1.3 bmp编码+拍照函数

这里是整个拍照的关键，这里把图像数据编码为二进制数据（rgb565）格式，同时形成图片信息文件，将lcd指定区域截屏，通过mollc内存管理系统保存在flash中，特别注意mini板的flash闪存芯片是w25q16，是一款比较老的芯片，而且迷你板没有外部内存，需要修改大量驱动代码才能驱动mini板的falsh。

完全理解这段函数，需要学习falsh内存管理实验，sd卡实验，特别注意mini是spi通讯方式，所以还要学习spi通讯；fatfs文件管理系统等。最开始不理解他需要通过闪存才能保存到sd卡中，走了大量弯路，最后学完所有基础才真正明白这个函数的工作原理。

u8 bmp\_encode(u8 \*filename,u16 x,u16 y,u16 width,u16 height,u8 mode)//参数：存放路径的文件名，截屏区域，mode表示创建文件的方式，覆盖或者重新创建。

{

FIL\* f\_bmp;

u16 bmpheadsize; //bmp头大小

BITMAPINFO hbmp; //bmp头

u8 res=0;

u16 tx,ty; //图像尺寸

u16 \*databuf; //数据缓存区地址

u16 pixcnt; //像素计数器

u16 bi4width; //水平像素字节数

if(width==0||height==0)return PIC\_WINDOW\_ERR; //区域错误

if((x+width-1)>lcddev.width)return PIC\_WINDOW\_ERR; //区域错误

if((y+height-1)>lcddev.height)return PIC\_WINDOW\_ERR; //区域错误

#if BMP\_USE\_MALLOC == 1 //使用malloc

databuf=(u16\*)mymalloc(2048); //开辟至少bi4width大小的字节的内存区域 ,对240宽的屏,480个字节就够了.

if(databuf==NULL)return PIC\_MEM\_ERR; //内存申请失败.

f\_bmp=(FIL \*)mymalloc(sizeof(FIL)); //开辟FIL字节的内存区域

if(f\_bmp==NULL) //内存申请失败.

{

myfree(databuf);

return PIC\_MEM\_ERR;

}

#endif

bmpheadsize=sizeof(hbmp);//得到bmp文件头的大小

mymemset((u8\*)&hbmp,0,sizeof(hbmp));//置零空申请到的内存.

hbmp.bmiHeader.biSize=sizeof(BITMAPINFOHEADER);//信息头大小

hbmp.bmiHeader.biWidth=width; //bmp的宽度

hbmp.bmiHeader.biHeight=height; //bmp的高度

hbmp.bmiHeader.biPlanes=1; //恒为1

hbmp.bmiHeader.biBitCount=16; //bmp为16位色bmp

hbmp.bmiHeader.biCompression=BI\_BITFIELDS;//每个象素的比特由指定的掩码决定。

hbmp.bmiHeader.biSizeImage=hbmp.bmiHeader.biHeight\*hbmp.bmiHeader.biWidth\*hbmp.bmiHeader.biBitCount/8;//bmp数据区大小

hbmp.bmfHeader.bfType=((u16)'M'<<8)+'B';//BM格式标志

hbmp.bmfHeader.bfSize=bmpheadsize+hbmp.bmiHeader.biSizeImage;//整个bmp的大小

hbmp.bmfHeader.bfOffBits=bmpheadsize;//到数据区的偏移

hbmp.RGB\_MASK[0]=0X00F800; //红色掩码

hbmp.RGB\_MASK[1]=0X0007E0; //绿色掩码

hbmp.RGB\_MASK[2]=0X00001F; //蓝色掩码

if(mode==0||res==0x04)res=f\_open(f\_bmp,(const TCHAR\*)filename,FA\_WRITE|FA\_CREATE\_NEW);//模式0,或者尝试打开失败,则创建新文件

LCD\_ShowString(30,150,200,16,16,"2");delay\_ms(1000);

if((hbmp.bmiHeader.biWidth\*2)%4)//水平像素(字节)不为4的倍数

{

LCD\_ShowString(30,150,200,16,16,"3");delay\_ms(1000);

bi4width=((hbmp.bmiHeader.biWidth\*2)/4+1)\*4;//实际要写入的宽度像素,必须为4的倍数.

}else bi4width=hbmp.bmiHeader.biWidth\*2; //刚好为4的倍数

LCD\_ShowString(30,150,200,16,16,"4");delay\_ms(1000);

res=f\_write(f\_bmp,(u8\*)&hbmp,bmpheadsize,&bw);//写入BMP首部

for(ty=y+height-1;hbmp.bmiHeader.biHeight;ty--)

{

pixcnt=0;

for(tx=x;pixcnt!=(bi4width/2);)

{

if(pixcnt<hbmp.bmiHeader.biWidth)databuf[pixcnt]=LCD\_ReadPoint(tx,ty);//读取坐标点的值

else databuf[pixcnt]=0Xffff;//补充白色的像素.

pixcnt++;

tx++;

}

hbmp.bmiHeader.biHeight--;

res=f\_write(f\_bmp,(u8\*)databuf,bi4width,&bw);//写入数据

}

f\_close(f\_bmp);

}

5.1.2 OLED模块

应用I2C通信协议，与单片机相连。程序启动时，IIC初始化，配置SDA和SCL，产生IIC起始信号，发送数据，等待应答信号，产生结束信号。将信号发送到SRAM,在STM32的内部建立一个缓存(共128\*64个字节)，在每次修改的时候只是修改STM32上的缓存(SRAM)在修改完了之后一次性把STM32上的缓存数据用OLED\_Refresh\_Gram写入到OLED的GRAM。OLED初始化，驱动IC初始化代码，开启显示，清0显存，开始显示

OLED\_WR\_Byte(u8 dat,u8 cmd)，向SSD1306写入一个字节dat:要写入的数据/命令。cmd:数据/命令标志 0,表示命令;1,表示数据;

void OLED\_Refresh\_Gram(void) 更新显存到gram，实现显示。

开启 OLED 显示 void OLED\_Display\_On(void

关闭 OLED 显示 void OLED\_Display\_Off(void)

清屏函数，清完屏，整个屏幕是黑色void OLED\_Clear(void)

显示字符void OLED\_ShowChar(u8 x，u8 y，u8 chr，u8 size，u8 mode)

显示字符串 void OLED\_ShowString(u8 x，u8 y，const u8 \*p)

初始化 SSD1303 void OLED\_Init(void)

void IIC\_Init(void)/\*函 数 名: IIC\_Init 功能 IIC初始化\*/

{

GPIO\_InitTypeDef GPIO\_InitStructure;//定义结构体名字

RCC\_APB2PeriphClockCmd(IIC\_SCL\_PORT\_RCC|IIC\_SDA\_PORT\_RCC,ENABLE);//打开APB2时钟，iic时钟定义

GPIO\_InitStructure.GPIO\_Pin=IIC\_SCL\_PIN;//设置结构体中各项参数

GPIO\_InitStructure.GPIO\_Speed=GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode=GPIO\_Mode\_Out\_PP;//推挽输出

GPIO\_Init(IIC\_SCL\_PORT,&GPIO\_InitStructure);//结构体的地址，读取结构体参数，写入配置寄存器

GPIO\_InitStructure.GPIO\_Pin=IIC\_SDA\_PIN;

GPIO\_Init(IIC\_SDA\_PORT,&GPIO\_InitStructure);//结构体的地址，读取结构体参数，写入配置寄存器

IIC\_SCL=1;

IIC\_SDA=1;

}

其中两个关键的初始化函数：

void OLED\_Init(void)

{

/\* GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd( RCC\_APB2Periph\_GPIOB|RCC\_APB2Periph\_GPIOC, ENABLE );

#if OLED\_MODE==1 //并口模式

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE); //使能AFIO时钟

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable , ENABLE); //JTAG-DP 失能 + SW-DP使能

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1|GPIO\_Pin\_2|GPIO\_Pin\_3|GPIO\_Pin\_4|GPIO\_Pin\_5|GPIO\_Pin\_6|GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

GPIO\_Write(GPIOB,0XFFFF);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6|GPIO\_Pin\_7|GPIO\_Pin\_8|GPIO\_Pin\_9;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

GPIO\_SetBits(GPIOC, GPIO\_Pin\_6|GPIO\_Pin\_7|GPIO\_Pin\_8|GPIO\_Pin\_9);//如果每一位决定一个GPIO\_Pin,则可以通过或的形式来初始化多个IO

#else

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_OD ; //推挽输出

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

GPIO\_Write(GPIOB,0X03);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_8|GPIO\_Pin\_9;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

GPIO\_SetBits(GPIOC, GPIO\_Pin\_8|GPIO\_Pin\_9);

#endif

void IIC\_Init(void)/\*函 数 名: IIC\_Init 功能 IIC初始化\*/

{

GPIO\_InitTypeDef GPIO\_InitStructure;//定义结构体名字

RCC\_APB2PeriphClockCmd(IIC\_SCL\_PORT\_RCC|IIC\_SDA\_PORT\_RCC,ENABLE);//打开APB2时钟，iic时钟定义

GPIO\_InitStructure.GPIO\_Pin=IIC\_SCL\_PIN;//设置结构体中各项参数

GPIO\_InitStructure.GPIO\_Speed=GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode=GPIO\_Mode\_Out\_PP;//推挽输出

GPIO\_Init(IIC\_SCL\_PORT,&GPIO\_InitStructure);//结构体的地址，读取结构体参数，写入配置寄存器

GPIO\_InitStructure.GPIO\_Pin=IIC\_SDA\_PIN;

GPIO\_Init(IIC\_SDA\_PORT,&GPIO\_InitStructure);//结构体的地址，读取结构体参数，写入配置寄存器

IIC\_SCL=1;

IIC\_SDA=1;

}

5.1.3指纹模块

指纹软件部分由以下一系列关键函数实现。

5.1.3.1 int main()

在main函数开头，初始化延时函数，设置中断分组，初始化LED及TIM4定时器。其中USART1串口初始化是上位机调试时所使用。

之后执行finger\_lock()函数。

5.1.3.2 finger\_lock()

整个程序的主要函数，进入此函数时，将初始化USART3串口，此为指纹模块与单片机进行通讯的串口，初始化时波特率为指纹模块波特率，这可以通过上位机修改。

初始化蜂鸣器，按键输入和OLED。

执行完以上代码进入while循环，内含主界面函数和设置界面函数，分别为main\_window\_show()和setup\_window\_show()。

5.1.3.3 main\_window\_show()

主界面函数，当第一次进入此函数时，因if语句中的finger.mode未被初始化（mode为结构体\_finger的变量，finger是一个结构体函数），被编译器自动初始化为0，从而执行if语句。

在if语句中，首先执行有关OLED的显示代码，并判断指纹模块是否与单片机“握手“，成功则显示（ ）

之后执行Press\_FP()函数，至其结束后goto MENU，即回到此函数开头。

5.1.3.4 Press\_FP()

默认刷指纹函数，进入后即进入其中的while循环，内含两个if语句执行对应功能，函数开头的ensure=PS\_GetImage()即为不断尝试获取指纹模块所检测到的指纹图像。

若探测到了手指，ensure将被赋值为返回码0，执行第一个if语句

在第一个if语句中，如果所检测指纹已存在于模块中则开锁成功，并且break打断while循环重返main\_window\_show()中的while循环并执行下一语句goto MENU回到主界面。若不存在则执行（ ）。

第一个if语句结束后执行key=KEY\_Scan(0)，该函数检测是否有按键按下，仅当KEY1按下时执行第二个if语句，此语句将finger.mode设置为10，然后break，回到goto MENU，进而回到main\_window\_show()，并且由于finger.mode不为0，main\_window\_show()直接执行完毕，回到finger\_lock()的while循环，进入setup\_window\_show()。

5.1.3.5 setup\_window\_show()

设置界面函数，从Press\_FP()函数按下KEY1后进入，此时finger.mode为10，可以执行if语句。

进入if语句后，OLED刷新为设置界面后进入while循环，不断检测是否有按键按下，按下KEY0为确定，KEY1为选择添加或删除指纹，KEY\_UP为返回主界面。

按键功能的实现由为finger.mode赋值实现，不同的值对应下方不同的if语句（之前将finger.mode设置为10后再进入设置界面函数的原因就是为了防止按下对应按键之前就发生了功能）。

5.1.3.6 Add\_FP()

录入指纹函数，进入后OLED刷新为录入指纹界面，然后不断检测是否有按键按下，并产生对应功能。按下KEY\_UP返回主界面，按下KEY1修改指纹ID，即选择将要录入的指纹对应的序号，按下KEY0确定，进入指纹录入阶段（在一个while循环内，从而可以一直等待直到检测到指纹），此时将手指按压在指纹模块前即可。

while循环中是一个switch语句，通过参数processnum的改变执行对应阶段。分别是录入图像生成特征并存储于缓冲区，二次录入图像生成特征并存储于缓冲区，对比两个缓冲区的特征，生成指纹模板存于缓冲区，最后将模板存于对应ID的flash数据库位置，最后return语句结束本函数（break在此只能打断switch，若在外再加上break则影响while循环和switch语句的配合），刷新OLED为设置界面。

5.1.3.7 Delete\_FP()

与Add\_FP()相似，无需赘述直接查看代码即可。

指纹模块原理：

参考文件AS60x指纹识别SOC用户手册V10

通过向AS608模块发送包头，芯片地址，包标识，包长度和指令码使AS608作出应答，应答表现为返回确认码（通过已被封装的函数调用出来），不同的确认码反映了不同情况，如录入失败，未检测到指纹，收包有错等。

5.1.3.8 一些来自as608.c的代码

//发送包头

static void SendHead(void)

{

MYUSART\_SendData(0xEF);

MYUSART\_SendData(0x01);

}

//发送地址

static void SendAddr(void)

{

MYUSART\_SendData(AS608Addr>>24);

MYUSART\_SendData(AS608Addr>>16);

MYUSART\_SendData(AS608Addr>>8);

MYUSART\_SendData(AS608Addr);

}

//发送包标识,

static void SendFlag(u8 flag)

{

MYUSART\_SendData(flag);

}

//发送包长度

static void SendLength(int length)

{

MYUSART\_SendData(length>>8);

MYUSART\_SendData(length);

}

//发送指令码

static void Sendcmd(u8 cmd)

{

MYUSART\_SendData(cmd);

}

//发送校验和

static void SendCheck(u16 check)

{

MYUSART\_SendData(check>>8);

MYUSART\_SendData(check);

}

注：查询得到校验和是指传输位数的累加，当传输结束时，接收者可以根据这个数值判断是否接到了所有的数据。如果数值匹配，那么说明传送已经完成。

我们直接调用的函数，如录入图像函数PS\_GetImage()

//录入图像 PS\_GetImage

//功能:探测手指，探测到后录入指纹图像存于ImageBuffer。

//模块返回确认字

u8 PS\_GetImage(void)

{

u16 temp;

u8 ensure;

u8 \*data;

SendHead();

SendAddr();

SendFlag(0x01);//命令包标识

SendLength(0x03);

Sendcmd(0x01);

temp = 0x01+0x03+0x01;

SendCheck(temp);

data=JudgeStr(2000);

if(data)

ensure=data[9];

else

ensure=0xff;

return ensure;

}

//生成特征 PS\_GenChar

//功能:将ImageBuffer中的原始图像生成指纹特征文件存于CharBuffer1或CharBuffer2

//参数:BufferID --> charBuffer1:0x01 charBuffer1:0x02

//模块返回确认字

5.1.4 LCD模块

以扫描成功的反馈为例，

void Screen\_True(void)

{

LCD\_Clear(WHITE);//清空LCD的显示内容，并给屏幕新的颜色

POINT\_COLOR=BLUE; //设置新显示内容的字体颜色

Show\_Str\_Mid(20,50,"Finger verification succeeded",16,0);//设置显示内容字符的起始坐标和大小，LCD显示部分的主要组成部分就是Show\_Str\_Mid函数

}

设置好每一反应的LCD显示内容，加入进主程序中与主程序配合，

OLED\_Clear();

OLED\_ShowString(0,0,"Verified!",16);

sprintf((char \*)strbuf,"ID:%d Score:%d",seach.pageID,seach.mathscore);//%d格式意为int型

OLED\_ShowString(0,20,strbuf,16);

OLED\_Refresh\_Gram();

Screen\_True();//屏幕显示正确

Show\_Str\_Mid(20,70,"ID:",16,0);

Show\_Str\_Mid(20,90,"score",16,0);

LCD\_ShowNum(80,70,seach.pageID,4,16);

LCD\_ShowNum(80,90,seach.mathscore,4,16);

与OLED同步，但显示更为详细的内容，让LCD服务用户端，给用户更清晰的操作指导，其中LCD\_ShowNum也是另一主要使用的函数，用于显示数字，设置显示内容的坐标和字体大小。

if(mode>=3)//第二次按下KEY1移动箭头并将mode赋值为3 重复此过程

{

mode=1;

}

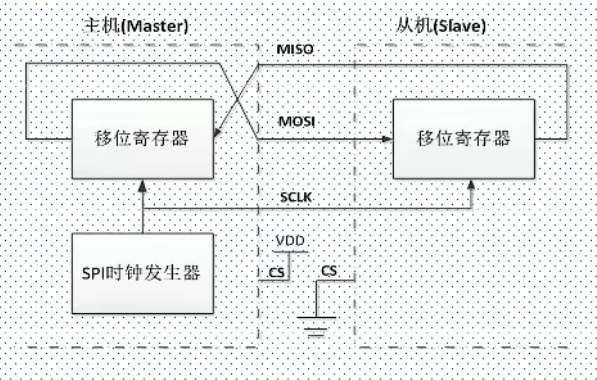
OLED\_ShowString(0,(mode)\*20,"-->",16);//箭头指向

OLED\_Refresh\_Gram();

Show\_Str\_Mid(0,(70+mode\*20),"-->",16,0);//箭头指向

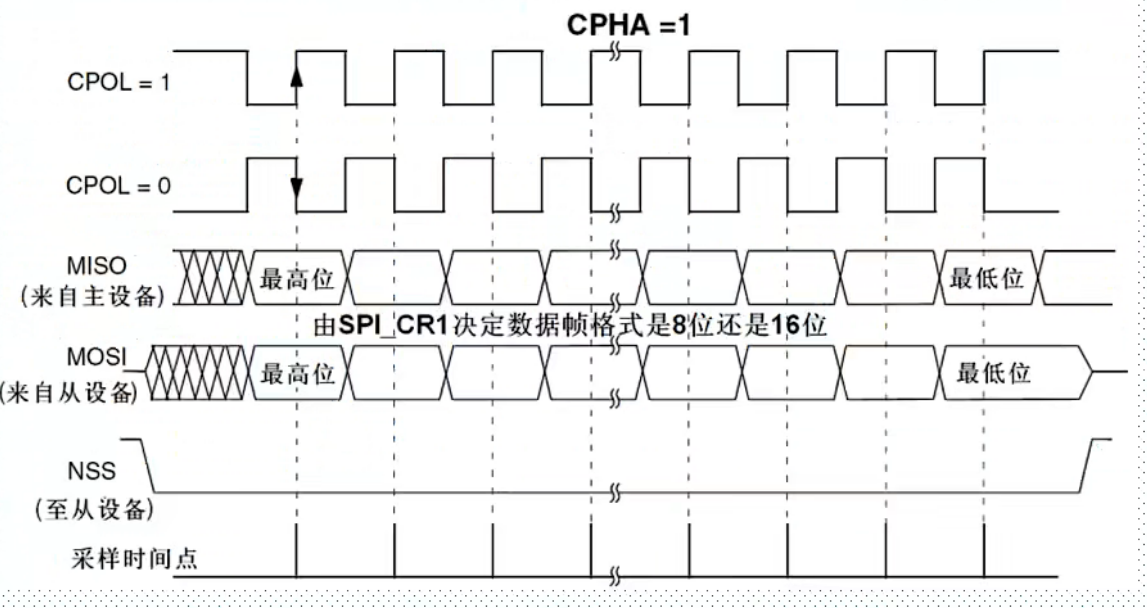
结合函数中的条件变量，可以达到随变量调整字符的起始坐标，达到移动图标的显示功能。

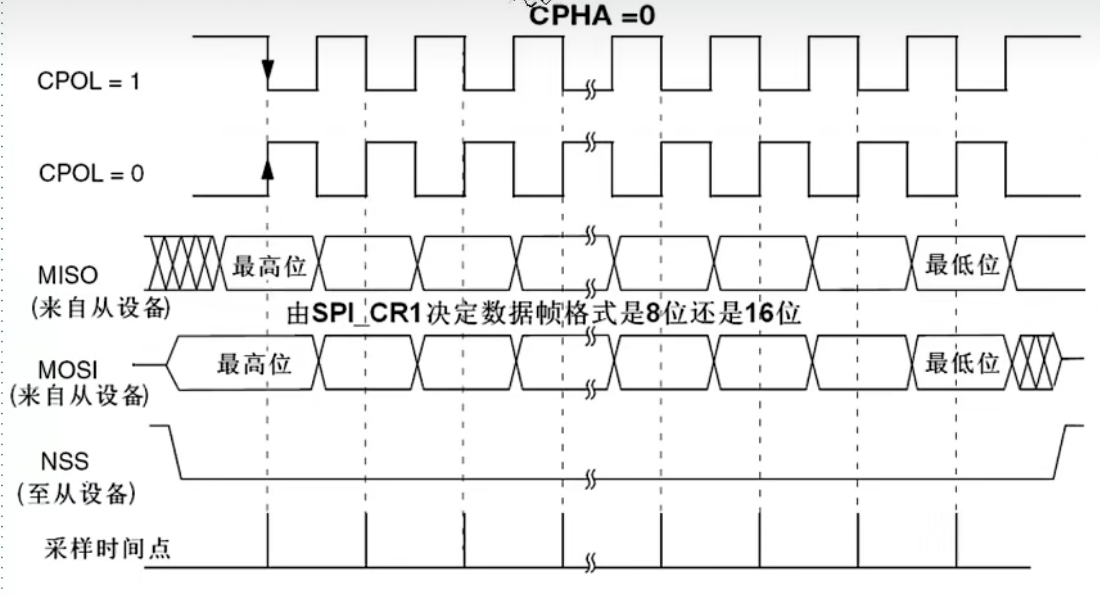
LCD使用SPI通讯协议。



SPI是一种四条线来做传输的一种通讯方式。MISO用于主设备数据输入，从设备数据输出；

MOSI用于主设备数据输出，从设备数据输入；SCLK作为时钟信号，由主设备产生；CS从设备片选信号，由主设备控制。





以void SPI\_Init为例，

void SPI1\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd( RCC\_APB2Periph\_GPIOA|RCC\_APB2Periph\_SPI1, ENABLE );

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5 | GPIO\_Pin\_6 | GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

GPIO\_SetBits(GPIOA,GPIO\_Pin\_5|GPIO\_Pin\_6|GPIO\_Pin\_7);

SPI\_InitStructure.SPI\_Direction = SPI\_Direction\_2Lines\_FullDuplex; //设置SPI单向或者双向的数据模式:SPI设置为双线双向全双工

SPI\_InitStructure.SPI\_Mode = SPI\_Mode\_Master; //设置SPI工作模式:设置为主SPI

SPI\_InitStructure.SPI\_DataSize = SPI\_DataSize\_8b; //设置SPI的数据大小:SPI发送接收8位帧结构

SPI\_InitStructure.SPI\_CPOL = SPI\_CPOL\_High; //选择了串行时钟的稳态:时钟悬空高

SPI\_InitStructure.SPI\_CPHA = SPI\_CPHA\_2Edge; //数据捕获于第二个时钟沿

SPI\_InitStructure.SPI\_NSS = SPI\_NSS\_Soft; //NSS信号由硬件（NSS管脚）还是软件（使用SSI位）管理:内部NSS信号有SSI位控制

SPI\_InitStructure.SPI\_BaudRatePrescaler = SPI\_BaudRatePrescaler\_256; //定义波特率预分频的值:波特率预分频值为256

SPI\_InitStructure.SPI\_FirstBit = SPI\_FirstBit\_MSB; //指定数据传输从MSB位还是LSB位开始:数据传输从MSB位开始

SPI\_InitStructure.SPI\_CRCPolynomial = 7; //CRC值计算的多项式

SPI\_Init(SPI1, &SPI\_InitStructure); //根据SPI\_InitStruct中指定的参数初始化外设SPIx寄存器

SPI\_Cmd(SPI1, ENABLE); //使能SPI外设

SPI1\_ReadWriteByte(0xff);//启动传输

}

依次设置好spi的主要参数，最后主机通过向它的SPI串行寄存器发送一个字节来发起一次传输。

5.1.5蓝牙模块

5.1.5.1 信息传输

蓝牙模块通过u3\_printf2函数，进行单片机对手机的信息传输。设置sendcnt初始值为0，传输内容为“INVADER”，用以提示管理员有未录入身份者进行了身份验证。当触发信息传输功能，u3\_printf2函数执行内容为“INVADER”的信息传输，一次后结束传输。

u8 sendcnt=0;

while(sendcnt<1)

{

u3\_printf2(" INVADER \r\n",sendcnt);

sendcnt++;

}

5.1.5.2 图片传输

在bmp解码函数stdbmp\_decode中获取名为color的图片信息，之后用Ble\_Sendpic对color进行处理。

color中每个像素是16位RGB565格式，而电脑上图片储存方式是16位bmp，像素是RGB555格式，于是要将像素格式进行转换，再发送。而且，串口发数据是每次发送一个字节。将一个像素分两次发送。于是先发送低八位，后发送高八位。

void Ble\_Sendpic(u8 color)

{

u8 r, g, b, temp;

u16 send;

r = 0x1f&(color>>11);

g = 0x1f&(color>>6);

b = 0x1f&color;

send = (r<<10)|(g<<5)|b;

temp = send&0x00ff; //准备发送数据，低八位

USART\_SendData(USART2,temp);

while(USART\_GetFlagStatus(USART2,USART\_FLAG\_TC) != SET);

temp = send>>8; //高八位

USART\_SendData(USART2,temp);

while(USART\_GetFlagStatus(USART2,USART\_FLAG\_TC) != SET);

}

将该函数包含到Ble\_Process函数中，在蓝牙模块连接无误时执行图片传输的功能。

void Ble\_Process(void)

{

u8 key = 0;

u8 t = 0;

u8 BLE\_LINK = 0; //1:蓝牙连接 0:未连接

static u8 flag1, flag2 = 0;

printf("\r\nKEY1:发送数据\r\n");

while (1)

{

key = KEY\_Scan(0);

if(key == KEY1\_PRESS) //发送数据

{

if (BLE\_LINK) /\*蓝牙已连接\*/

{

Ble\_SendData();/\*数据发送处理\*/

Ble\_Sendpic(0x00);

}

}

/\*数据接收处理\*/

Ble\_ReceData();

/\*BLE连接处理\*/

if (BLE\_STA)

{

BLE\_LINK = 1; //蓝牙连接

if (!flag1)

{

flag1 = 1;

flag2 = 0;

printf("\r\nble connect!\r\n");

}

}

else

{

BLE\_LINK = 0; //蓝牙断开

if (!flag2)

{

flag2 = 1;

flag1 = 0;

printf("\r\nble disconnect!\r\n");

}

}

t++;

if (t == 20)

{

t = 0;

LED1 = ~LED1;

}

delay\_ms(10);

}

}

最终将蓝牙模块初始化、参数设置和Ble\_Process函数整合为蓝牙主函数Ble\_Test，在需图片传输处调用函数即可。

void Ble\_Test(void)

{

Ble\_IoInit(); //BLE引脚初始化

Ble\_SetInit();//BLE参数配置

Ble\_Process();//数据测试

}

5.1.6 K210模块

5.1.6.1 模式切换

通过按键切换模式。调用按键扫描函数key=KEY\_Scan，若key1被按下，则回到门禁模式主菜单；若key2被按下，则进入人脸识别模式，并由LCD给出指示。

key=KEY\_Scan(0);

if(key==KEY1\_PRESS)//当KEY1键按下退出主菜单，进入设置菜单界面

{

finger.mode=10;//将mode设置为0至5之外的值 避开进入对应的功能

break;

}

if(key==KEY2\_PRESS)

{

LCD\_Clear(WHITE);

Show\_Str(30,50,200,16,"模式已切换",16,0);

Show\_Str(30,70,200,16,"进入人脸识别模式",16,0);

delay\_ms(5000);

LCD\_Clear(WHITE);

while(1)

{

Show\_Str(30,50,200,16,"请保持面部置于取景框内",16,0);

Show\_Str(30,70,200,16,"按下按键以录入人脸",16,0);

5.1.6.2 面部识别

基于三个面部识别模型：人脸检测模型、人脸五点关键点检测模型、人脸196维特征值模型，进行了如下设计，实现了对人面部信息识别的功能。

设计了两种方式，从flash或sd卡中加载所需的三个程序（最终使用了较为方便存储的flash方式）。

#从flash中加载模型

task\_fd = kpu.load(0x200000) # 从flash 0x200000 加载人脸检测模型

task\_ld = kpu.load(0x300000) # 从flash 0x300000 加载人脸五点关键点检测模型

task\_fe = kpu.load(0x400000) # 从flash 0x400000 加载人脸196维特征值模型

#从SD卡中加载模型

#task\_fd = kpu.load("/sd/FD\_face.smodel") # 加载人脸检测模型

#task\_ld = kpu.load("/sd/KP\_face.smodel") # 加载人脸五点关键点检测模型

#task\_fe = kpu.load("/sd/FE\_face.smodel") # 加载人脸196维特征值模型

首先进行硬件设置。设置所需硬件的初始化：初始化时钟，设置按键引脚及检测函数，初始化LCD，初始化sensor 摄像头，最后设置好一系列视觉点位后，初始化已加载人脸检测模型。

之后进行sd卡读写设置。

定义写入函数，将面部信息写入到/sd/features.txt文件中。

def save\_feature(feat):

with open('/sd/features.txt','a') as f:

record =ubinascii.b2a\_base64(feat)

f.write(record)

定义读取函数，判断特征文件是否存在，若存在则将写入的信息读取。

if(feature\_file\_exists):

print("start")

with open('/sd/features.txt','rb') as f:

s = f.readlines()

print(len(s))

for line in s:

record\_ftrs.append(bytearray(ubinascii.a2b\_base64(line)))

之后运行人脸识别的主程序部分。其主要逻辑为运行人脸检测模型，判断是否存在人脸，在存在人脸时，通过a = img.draw\_rectangle(i.rect())将人脸从全部图像中框出，通过face\_cut=img.cut(i.x(),i.y(),i.w(),i.h())裁剪框出内容，通过face\_cut\_128=face\_cut.resize(128,128)缩小所裁剪图像比例，通过a=face\_cut\_128.pix\_to\_ai()将图像转换为为kpu接受的格式。

之后运行人脸5点关键点检测模型，计算左眼、右眼、鼻子、左嘴角、右嘴角位置，生成矩阵。

le=(i.x()+int(plist[0]\*i.w() - 10), i.y()+int(plist[1]\*i.h())) # 计算左眼位置

re=(i.x()+int(plist[2]\*i.w()), i.y()+int(plist[3]\*i.h())) # 计算右眼位置

nose=(i.x()+int(plist[4]\*i.w()), i.y()+int(plist[5]\*i.h())) #计算鼻子位置

lm=(i.x()+int(plist[6]\*i.w()), i.y()+int(plist[7]\*i.h())) #计算左嘴角位置

rm=(i.x()+int(plist[8]\*i.w()), i.y()+int(plist[9]\*i.h())) #右嘴角位置

a = img.draw\_circle(le[0], le[1], 4)

a = img.draw\_circle(re[0], re[1], 4)

a = img.draw\_circle(nose[0], nose[1], 4)

a = img.draw\_circle(lm[0], lm[1], 4)

a = img.draw\_circle(rm[0], rm[1], 4) # 在相应位置处画小圆圈

# align face to standard position

src\_point = [le, re, nose, lm, rm] # 图片中 5 坐标的位置

T=image.get\_affine\_transform(src\_point, dst\_point) # 根据获得的5点坐标与标准正脸坐标获取仿射变换矩阵

a=image.warp\_affine\_ai(img, img\_face, T) #对原始图片人脸图片进行仿射变换，变换为正脸图像

最后运行人脸196维特征值模型，计算与存储特征的匹配程度，若匹配度得分大于70，则输出正确验证；否则输出错误验证。

if max\_score > 70: # 如果最大分数大于70， 可以被认定为同一个人

a = img.draw\_string(i.x(),i.y(), ("%s :%2.1f" % (names[index], max\_score)), color=(0,255,0),scale=2) # 显示人名 与 分数

uart.write('1\r\n')

else:

uart.write('0\r\n')

a = img.draw\_string(i.x(),i.y(), ("X :%2.1f" % (max\_score)), color=(255,0,0),scale=2) #显示未知 与 分数

5.1.6.3 添加面部信息

硬件初始化和面部识别部分相同，不做赘述。该部分的主要逻辑为：当按下按键1时，将此时检测到的面部信息保存以判断；调用save\_feature写入函数，将保存的面部信息写入sd卡，便于重置后信息保存。

if key\_pressed == 1: #如果检测到按键

key\_pressed = 0 #重置按键状态

record\_ftr = feature

record\_ftrs.append(record\_ftr) #将当前特征添加到已知特征列表

save\_feature(record\_ftr) #存到SD卡

start\_processing = False

5.1.6.4 口罩识别

基于口罩识别模型，进行了如下设计，实现了对人面部口罩识别的功能。

该模块可对图像进行判断，输出佩戴口罩可信任度\_confidence一值。基于这一判断值，定义drawConfidenceText判断文本函数，当classid == 1判断为已佩戴，图像中输出相关文本，并设置文本为绿色；当classid == 0判断为未佩戴，图像中输出相关文本，并设置文本为红色。

def drawConfidenceText(image, rol, classid, value):

text = ""

\_confidence = int(value \* 100)

if classid == 1:

text = 'mask: ' + str(\_confidence) + '%'

color\_text=color\_G

else:

text = 'no\_mask: ' + str(\_confidence) + '%'

color\_text=color\_R

image.draw\_string(rol[0], rol[1], text, color=color\_text, scale=2.5)

之后设置LCD、摄像头、时钟初始化，进入主程序。通过sensor.snapshot函数获取图像，执行kpu.run\_yolo2(task, img)以获取判断信息并进行信息筛选，得到配到口罩可信任度、是否判断口罩等信息。最终根据佩戴口罩可信任度，输出相应判断正误反馈。

while (True):

clock.tick()

img = sensor.snapshot()

code = kpu.run\_yolo2(task, img)

if code:

totalRes = len(code)

for item in code:

confidence = float(item.value())

itemROL = item.rect()

classID = int(item.classid())

if confidence < 0.52:

\_ = img.draw\_rectangle(itemROL, color=color\_B, tickness=5)

uart.write('0\r\n')

continue

if classID == 1 and confidence > 0.65:

\_ = img.draw\_rectangle(itemROL, color\_G, tickness=5)

uart.write('1\r\n')

if totalRes == 1:

drawConfidenceText(img, (0, 0), 1, confidence)

else:

\_ = img.draw\_rectangle(itemROL, color=color\_R, tickness=5)

if totalRes == 1:

drawConfidenceText(img, (0, 0), 0, confidence)

5.1.6.5 通信

应用usart串口通信将K210和stm32连接。K210在对比匹配度分数后，除通过屏幕显示正误外，还会输出一信号表示正误（正确为1，错误为0）。使用stm32的usart相关变量，如USART\_RX\_BUF[USART\_REC\_LEN]、USART\_RX\_STA等接收信号并条件判断，若信号为1，判断为正确，则通过LCD、蜂鸣器、LED给出正确指示；若信号为0，判断为错误，则通过LCD、蜂鸣器、LED给出错误指示。

if(USART\_RX\_STA&0x8000)

{

k210=USART\_RX\_BUF[0];

if(k210==0x31)//接收到1

{

LED1=1;

BEEP=1;

delay\_ms(1000);

BEEP=0;

LCD\_Clear(WHITE);

Show\_Str(30,50,200,16,"认证成功，",16,0);

Show\_Str(30,70,200,16,"欢迎进入！",16,0);

delay\_ms(3000);

k210=3;

}

if(k210==0x30)//接收到0

{

Show\_Str(30,50,200,16,"认证失败，",16,0);

Show\_Str(30,70,200,16,"请与管理员取得联系！",16,0);

while(g<5)

{

BEEP=1;

delay\_ms(300);

BEEP=0;

delay\_ms(300);

g+=1;

}

g=0;

k210=3;

}

USART\_RX\_STA=0;

}

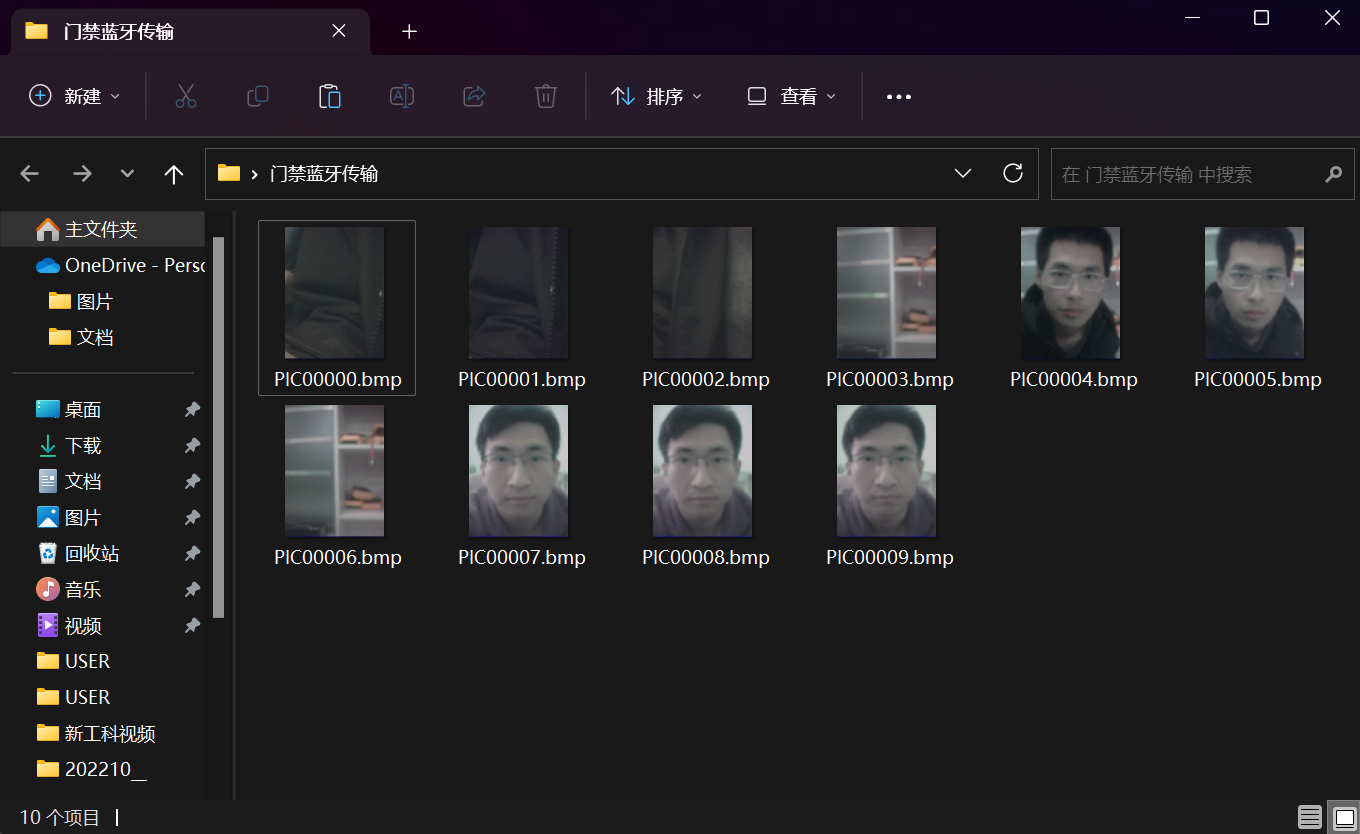
5.2 上位机部分

5.2.1 手机软件

应用IOS系统下的LightBlue软件以进行蓝牙连接信息传输。当蓝牙成功连接并由stm32成功发送信息给手机时，软件有如下显示。



应用Windows操作系统下的蓝牙，以进行蓝牙连接图片传输，当蓝牙成功连接并由stm32成功发送信息给电脑时，相关文件夹会得到相应图片。



# 6.系统测试及结果

经过多次模拟测试与实物操作实验，该系统下各功能运行正常，能够良好实现智能门禁、智能防疫的功能。

有线、无线传输相关功能运行正常，文件信息传输无误。信息判断相关功能运行正常，能对信息做出准确、灵敏的判断。显示与切换相关功能运行正常，能与各功能准确衔接。信息获取相关功能运行正常，能清晰、灵敏地获取图像信息。具体结果见演示视频。

# 7.程序源代码

全部代码详见代码文件。这里展示主程序。

int main(void)

{

delay\_init(); //延时函数初始化

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//设置中断优先级分组为组2：2位抢占优先级，2位响应优先级

uart\_init(115200); //串口初始化为115200

usmart\_dev.init(72); //初始化USMART

LED\_Init(); //初始化与LED连接的硬件接口

KEY\_Init(); //初始化按键

LCD\_Init(); //初始化LCD

Ble\_IoInit(); //BLE引脚初始化

Ble\_SetInit();//BLE参数配置

//

TIM3\_Int\_Init(1000-1,36000-1); //定时1000\*36000/72us=500ms从而使LED灯不断闪烁 表明系统正在运行

BEEP\_Init(); //蜂鸣器初始化

W25QXX\_Init(); //初始化W25Q128

my\_mem\_init(SRAMIN); //初始化内部内存池

exfuns\_init(); //为fatfs相关变量申请内存

f\_mount(fs[0],"0:",1); //挂载SD卡

f\_mount(fs[1],"1:",1); //挂载FLASH.

POINT\_COLOR=RED;

while(font\_init()) //检查字库

{

LCD\_ShowString(30,50,200,16,16,"Font Error!");

delay\_ms(200);

LCD\_Fill(30,50,240,66,WHITE);//清除显示

}

Show\_Str(30,50,200,16,"多功能智能门禁",16,0);

Show\_Str(30,70,200,16,"准备进行初始化...",16,0);

Show\_Str(30,110,200,16,"北洋大学鹏翔硅谷荣誉出品",16,0);

Show\_Str(30,130,200,16,"喜迎二十大，永远跟党走！",16,0);

res=f\_mkdir("0:/PHOTO"); //创建PHOTO文件夹

if(res!=FR\_EXIST&&res!=FR\_OK) //发生了错误

{

Show\_Str(30,150,240,16,"SD fail!",16,0);

delay\_ms(200);

Show\_Str(30,170,240,16,"cant take photo!",16,0);

sd\_ok=0;

}

else

{

Show\_Str(30,150,240,16,"SD卡准备就绪!",16,0);

delay\_ms(200);

Show\_Str(30,170,240,16,"指纹识别错误将会自动拍照",16,0);

sd\_ok=1;

}

pname=mymalloc(SRAMIN,30); //为带路径的文件名分配30个字节的内存

while(pname==NULL) //内存分配出错

{

Show\_Str(30,190,240,16,"error molloc!",16,0);

delay\_ms(200);

LCD\_Fill(30,190,240,146,WHITE);//清除显示

delay\_ms(200);

}

while(1)//初始化OV7725\_OV7670

{

if(OV7670\_Init()==0)

{

//sensor=OV7670;

LCD\_ShowString(30,150,200,16,16,"摄像头初始化成功！ ");

delay\_ms(1500);

OV7670\_Light\_Mode(0);

OV7670\_Color\_Saturation(0);

OV7670\_Brightness(0);

OV7670\_Contrast(0);

OV7670\_Special\_Effects(0);

OV7670\_Window\_Set(12,176,240,320);//设置窗口

break;

}

else

{

LCD\_ShowString(30,150,200,16,16,"OV7725\_OV7670 Error!!");

delay\_ms(200);

LCD\_Fill(30,150,200,246,WHITE);

delay\_ms(200);

}

}

OV7670\_CS=0;

TIM6\_Int\_Init(10000,7199); //10Khz计数频率,1秒钟中断

EXTI8\_Init(); //使能定时器捕获

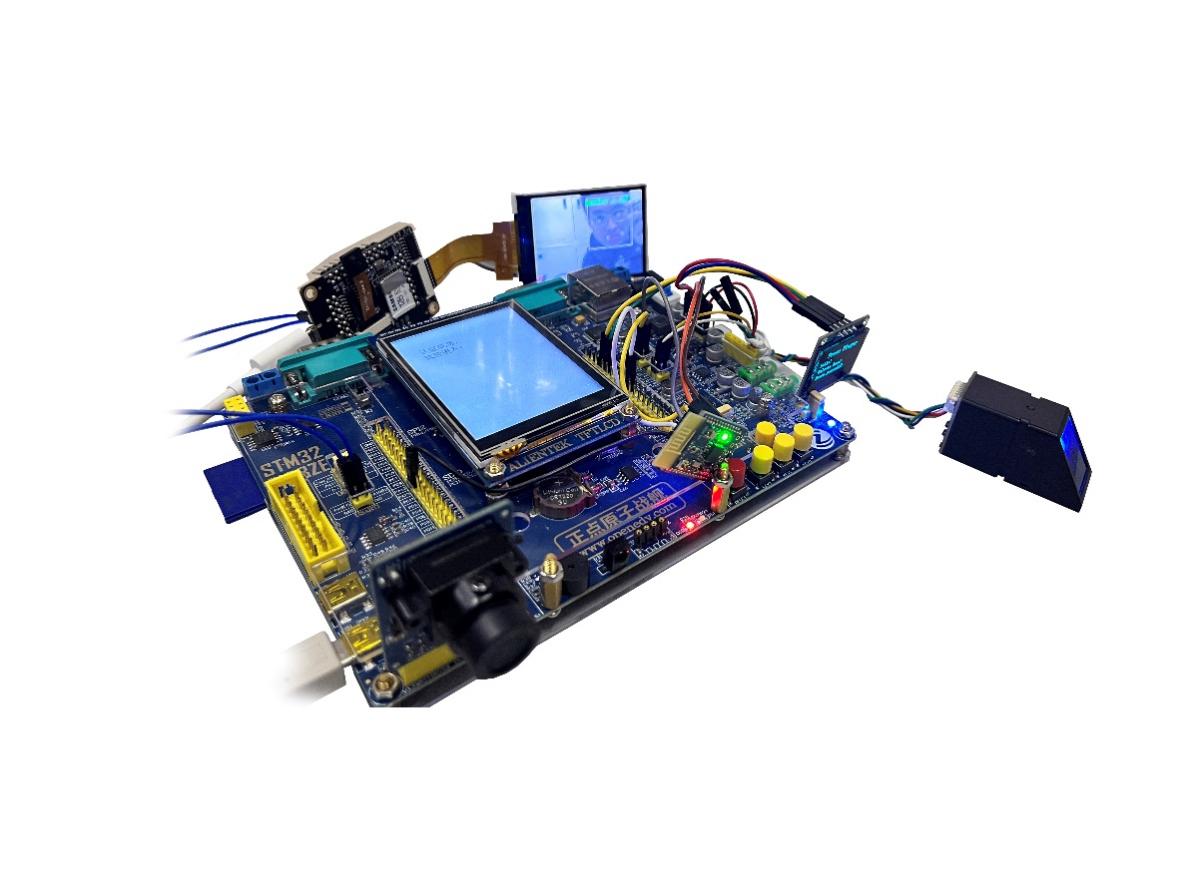
LCD\_Clear(WHITE);

finger\_lock();

}

三、照片及说明

# 1. 基于stm32的多功能智能门禁系统整体外观





# 2. 各模式效果图



