题目：基于人脸识别的多功能家居门禁系统

Multifunctional home access control system based on face recognition

摘要：

随着人工智能迅速发展，智能家居领域内的产品逐渐增多，但家用门锁的种类仍然以传统的钥匙解锁和指纹解锁为主。因此我们小组设计了一款基于人脸识别的多功能智能门禁系统，该门禁系统利用 MTMN 人脸检测模型，使用卷积神经网络对人脸进行检测。此外，本设计还有红外检测，亮度检测，智能开灯，报警等功能。本设计使用的硬件为芯片 esp32-cam 与芯片 stm32，esp32-cam 进行图像获取、处理与门锁的操控；stm32 对电路其他部分进行处理。

关键词：

MTMN、esp32-cam、stm32、卷积神经网络、门禁系统

Abstract:

With the rapid development of artificial intelligence, the number of products within the smart home sector is gradually increasing, but the types of home door locks are still dominated by traditional key unlocking and fingerprint unlocking. Therefore our group has designed a Multifunctional home access control system based on face recognition, which utilises the MTMN face detection model and uses convolutional neural networks for face detection. In addition, this design has functions such as infrared detection, brightness detection, automatic Light Control and alarm. The hardware used in this design is esp32-cam and stm32. esp32-cam carries out image acquisition, processing and door lock manipulation; stm32 processes the rest of the circuit.

Key words:

MTMN, esp32-cam, stm32, convolutional neural networks, access control systems.

目录

一、选题背景和系统指标

1.1 选题背景

过去的几年里，智能家居领域有了飞跃式的发展。许多新产品可以为家庭提供更加便利和安全的生活。

人脸识别门锁是一种基于面部识别技术的智能安全系统，通过检测面部提供访客的实时信息，实现对于门锁的控制。

新冠病毒疫情的爆发极大地影响了整个世界。为了预防病毒传播，如今对人脸识别门锁和智能门铃等非接触式智能安全解决方案的需求大大增长。全球各公司目前也致力于开发创新的智能门锁产品。

随着深度学习、物联网和人工智能等先进技术的整合，安全系统在检测人脸时的准确度有了明显的提升，必将提供更多便利。

1.2 系统指标

自动化及便捷性：面部识别门锁应是完全自动化的，值得信赖的。相机传感器应能自动捕捉面部图像，并与存储的图像算法进行匹配。

高安全性：脸部与算法匹配的人才可以开门，错误率减至 1%以下。

传感器可靠性：红外传感器判断人体存在，光感判断弱光控制开灯，可靠率 95%以上。

二、系统方案设计

2.1 系统整体功能设计

功能简述一：人脸识别进行开锁，可以录入多个人脸数据，开门后语音播报提示。

功能简述二：红外传感器检测有人时，开启状态提示灯，显示器显示；若长时间检测有人且未开门，系统关闭。

功能简述三：门铃通过按键开关控制，按下后蜂鸣器响。同时门铃通过波动开关控制，可以设置是否开启门铃。

功能简述四：光敏电路检测夜晚时，可以自动开启夜间 LED 灯进行打光。

2.1.1 系统整体框架

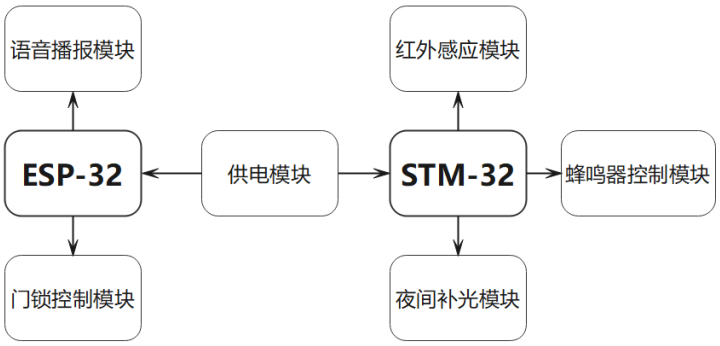


图 2-1 系统模块结构图

2.1.2 系统各部分功能简介

语音播报模块：在 ESP-32 识别到人脸后，进行语音播报提示

继电器模块：在 ESP-32 识别到存储的人脸后，传递信号，继电器模块控制门锁打开。

红外感应模块：感应一定范围内的人体活动，开启状态提示灯，控制显示器显示；若长时间检测有人且未开门，进行提醒并闪灯。

蜂鸣器控制模块：通过按键控制蜂鸣器实现门铃功能。

光敏电路模块：对光照进行检测，实现对周围环境的亮度和光强进行检测，进入夜晚后，控制夜间照明 LED 进行面部识别时的补光。

2.2 系统元件选择

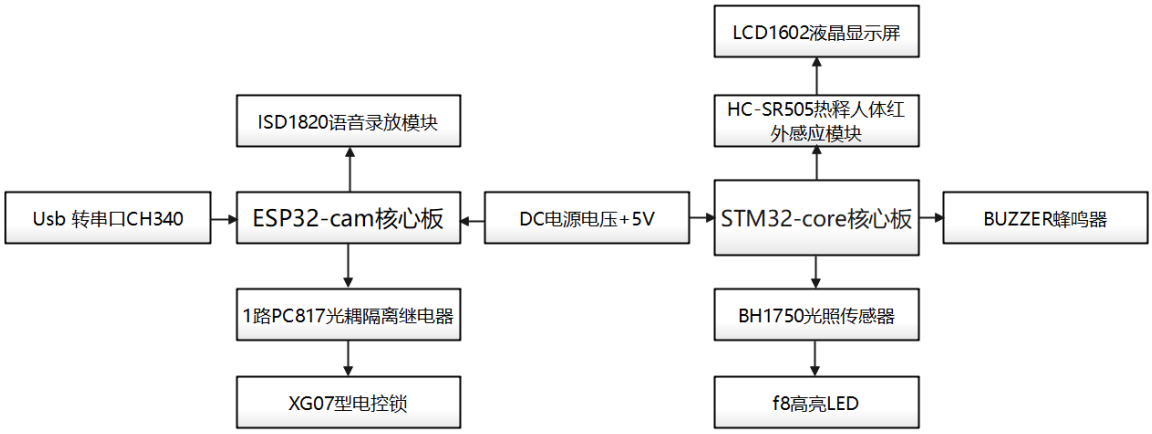


图 2-2 系统元件结构图

2.2.1 STM32-core 核心板

STM32 处理器是一种基于 ARM 7 架构的 32 位、支持实时仿真和跟踪的微控制器。此款控制芯片在完成单片机课程的学习后上手较为容易，具有很好的学习、实验研究价值。

STM32 的主要优点有：（1）优异的实时性能（2）出色的功耗控制（3）易于开发（4）可以在不修改原始框架及软件的条件下，升级至不同规格。

选择此款控制芯片是因为本系统设计并非追求成本的最低或更小的功耗，而是在实现本设计功能的前提下能够提供更丰富的接口和功能以便于设计实验系统各实验项目所需的外围扩展电路。

图 2-3 STM32-core 核心板

2.2.2 Esp32-cam 核心板

ESP32-CAM 作为小尺寸摄像头模组，尺寸仅为 27*40.5*4.5mm，深度睡眠电流

最低为 6mA，同时其采用 DIP 封装，直接插上底板即可使用，连接方式具有极高可靠性。

本项目中利用 Arduino IDE 对 esp32 进行开发，实现人脸识别功能，通过 esp322 号引脚输出电平信号，控制继电器开合，进而控制电子锁的闭合。

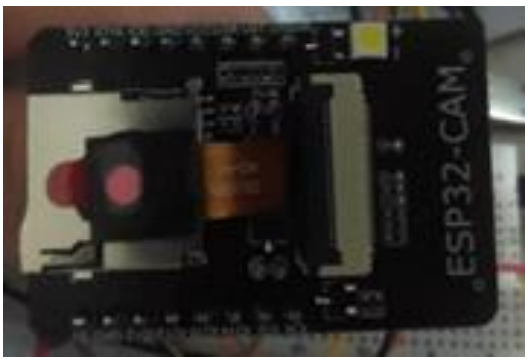


图 2-4 Esp32-cam 核心板

2.2.3 1 路 PC817 光耦隔离继电器

1 路 5V 继电器模块，以高电平触发，采用了贴片光耦隔离，驱动能力强、性能稳定，同时支持高低电平触发开发板，可以节省空间、减少中间接线环节，提高效率及性能。

图 2-5 1 路 PC817 光耦隔离继电器

2.2.4 XG07 型电控锁

材质为钢材质外壳、碳钢精铸零件，有 2 根 5V 电源线及 2 根信号反馈线，设计荷载条件下，可使用大于 30 万次。使用时可在通电瞬间开锁，断电关门上锁，同时有机械开锁装置，在断电的情况下按下锁杆即可开锁。在本产品设计中，通过继电器的开合控制电子锁是否上电，上电即开锁。

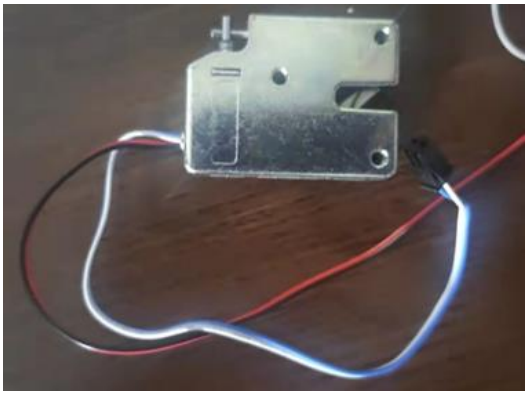


图 2-6 XG07 型电控锁

2.2.5 ISD1820 语音录放模块

采用 CMOS 技术，内含振荡器，话筒前置放大，自动增益控制，防混淆滤波器，扬声器驱动及 FLASH 阵列。其中 PLY 为播放接口，接单片机 IO，通过电平变化来控制放音，可直接驱动 8 欧 0.5W 小喇叭。

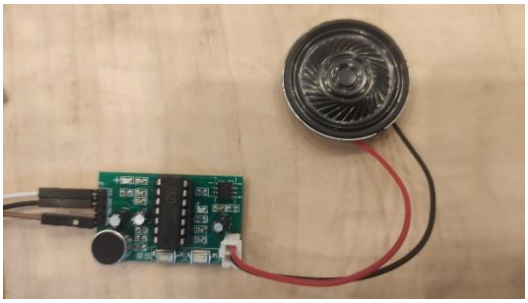


图 2-7 ISD1820 语音录放模块

2.2.6 HC-SR505 热释人体红外感应模块

基于红外线技术的自动控制产品，灵敏度高，可靠性强，体积小。

红外感应特性：（1）全自动，人进入感应范围则输出高电平，人离开感应范围则自动延时关闭高电平，输出低电平（2）可重复触发，感应输出高电平后，在延时时间段内，如果有人体在其感应范围活动，其输出将一直保持高电平，直到人离开后才延时将高电平变为低电平（3）工作电压范围宽，静态电流小，功耗低（4）输出高电平信号，便于与各类电路对接。

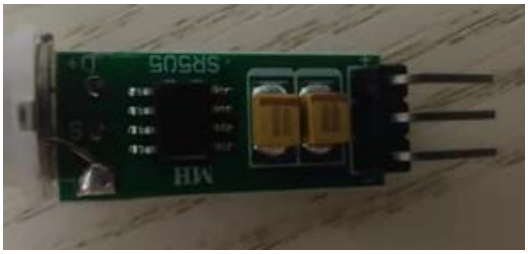


图 2-8 HC-SR505 热释人体红外感应模块

2.2.7 LCD1602 液晶显示屏

与传统的 LED 数码管显示器件相比，液晶显示模块体积小、功耗低、显示内容丰富，而且不需要外加驱动电路。

LCD1602 显示容量为 16×2 个字符，采用标准的 14 脚，其中第 4 脚 RS 为寄存器选择，高电平时选择数据寄存器、低电平时选择指令寄存器。第 5 脚 RW 为读写信号线，高电平时进行读操作，低电平时进行写操作。当 RS 和 RW 同为低电平时可以写入指令或者显示地址，当 RS 为低电平 RW 为高电平时可以读忙信号，当 RS 为高电平 RW 为低电平时可以写入数据。

图 2-9 LCD1602 液晶显示屏

2.2.8 BH1750 光照传感器

光敏电阻基于内光电效应工作。无光照时，暗电阻一般可达 1.5MΩ，随着光照强度的升高，电阻值迅速降低，亮电阻值可小至 1KΩ 以下。

光敏特性：（1）灵敏度可调（图中蓝色数字电位器调节）（2）输出形式为 DO 数字开关量输出（0 和 1）和 AO 模拟电压输出（3）比较器输出，信号干净，波形好，驱动能力强，超过 15mA。同时比较器采用 LM393 芯片，工作稳定。

图 2-10 BH1750 光照传感器

2.2.9 f8 高亮 LED

超高亮 LED 是比一般 LED 发光二极管的亮度高近百倍的新型 LED，在本设计中，选择白色高亮 LED 灯作为照明灯使用。

白色高亮 LED 灯的优点：（1）寿命长，可靠耐用（2）效率高，其发光效率可达 80%～90%（3）点亮速度快。

2.2.10 BUZZER 蜂鸣器

5V 有源蜂鸣器模块采用三极管 9012 来驱动，控制引脚为高电平，蜂鸣器就会鸣叫报警，反之则不鸣叫，通过限流电阻起保护作用。

图 2-11 BUZZER 蜂鸣器

三、系统硬件设计

STM32F103C8T6 单片机核心板接口电路图如下图所示：

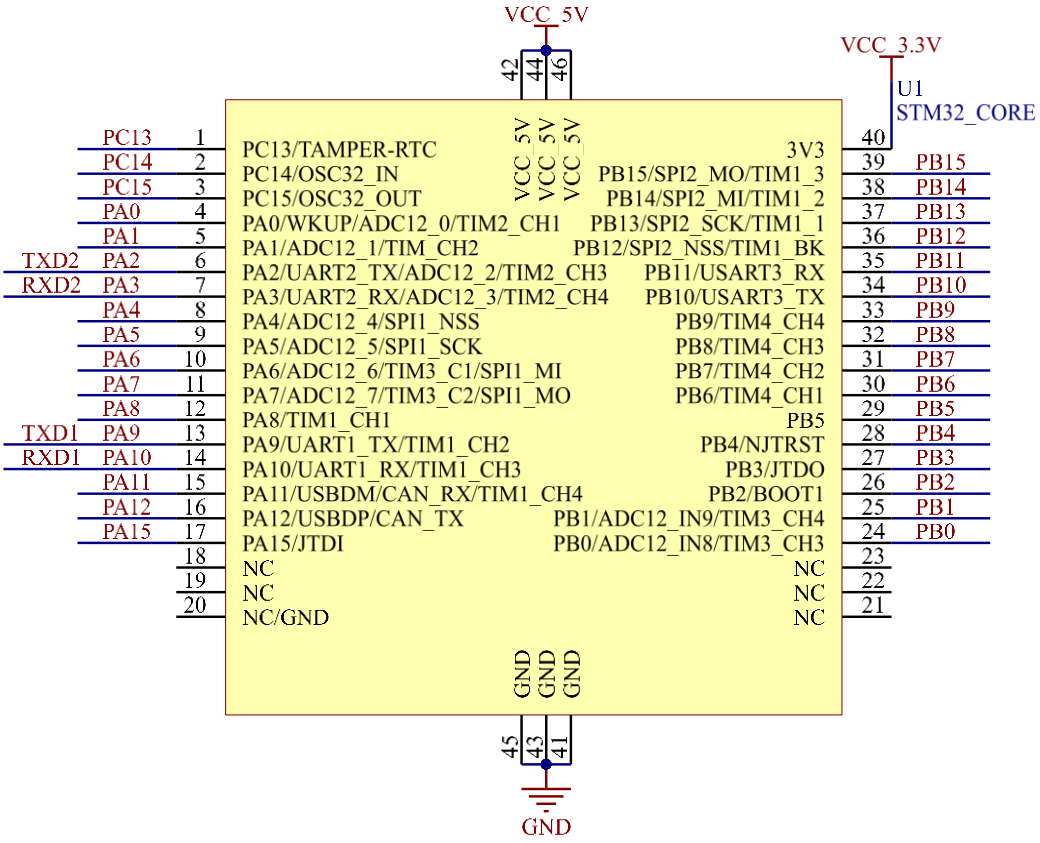


图 3-1 STM32F103C8T6 单片机核心板接口电路图

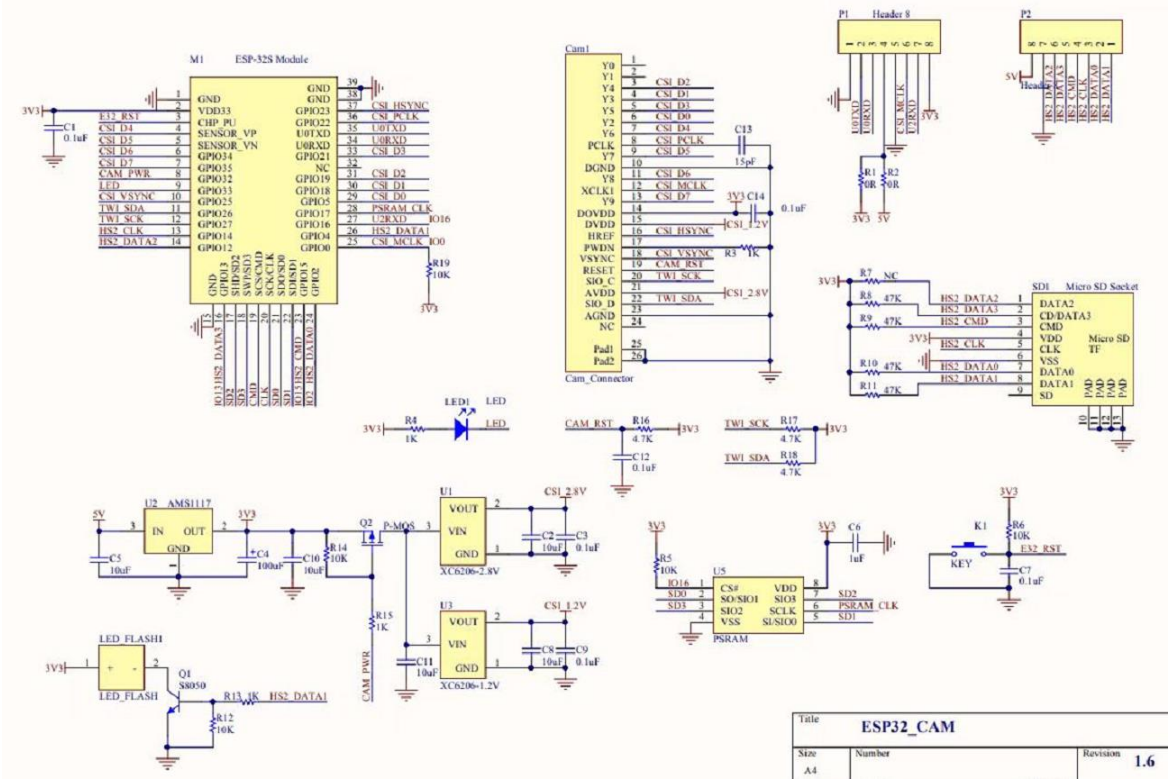


图 3-2 ESP32-CAM 接口电路图

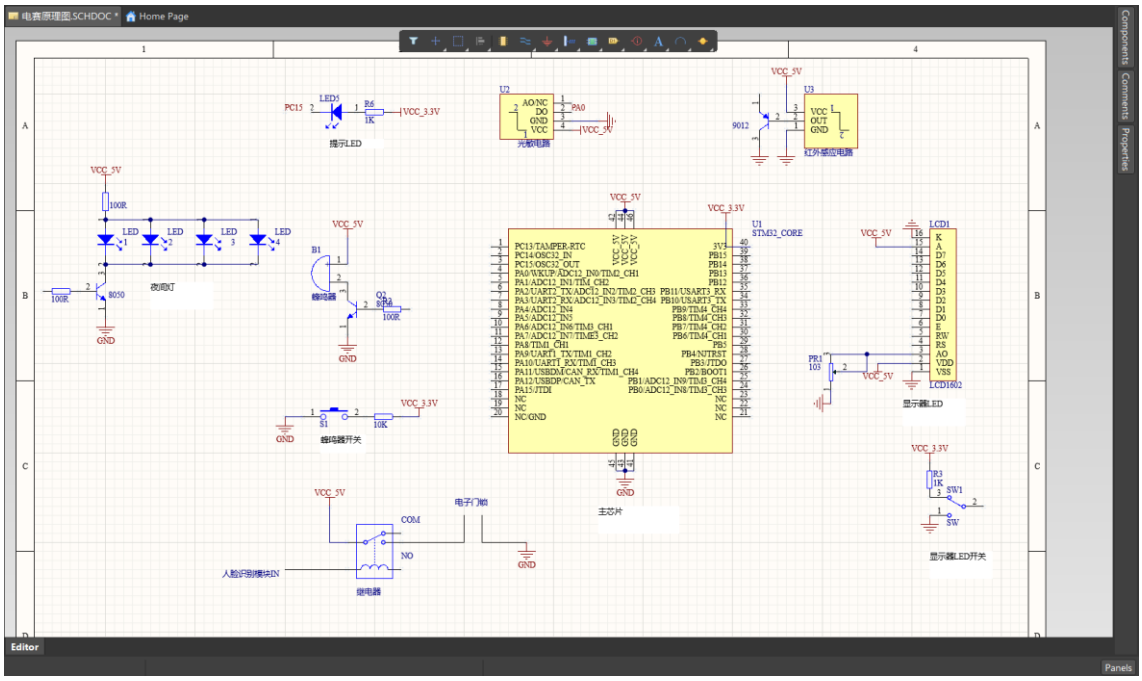


图 3-2 系统元件连接电路图

四、系统软件设计

4.1 软件说明

1. Keil：本设计中单片机开发环境是 Keil，Keil 是 51 系列兼容单片机 C 语言软件开发系统，可以完成从编辑、编译、到连接、调试的一套开发流程,与汇编相比，C 语言在功能上、结构性、可读性、可维护性上有明显的优势，易学易用。

2. FlyMcu：stm32 烧录程序软件，可以广泛应用于电路编程(ICP)和应用编程(IAP)领域，支持进行编程、校验、读器件信息。

3. CH340 串口烧写模块：本设计通过 CH340 串口烧写模块实现对单片机程序的烧写。CH340 串口烧写模块使用 USB 接口，价格低，性能高，是开发 STC 系列单片机的优秀工具。

4.2 人脸检测算法设计

人脸检测算法的目的就是在图像中判断物体是否为人脸，并返回该人脸对应的特征。本项目中的人脸检测用到了一种叫做 MTMN 的轻量级的人脸检测模型，它围绕一种名为 MobileNetV2 的新移动架构和多任务级联卷积网络构建，专为嵌入式设备而设计。

MTMN 由三个主要部分组成：

1、提案网络 Proposal Network（P-Net）：提出候选边界框，并将其发送到 R-Net;

2、优化网络 Refine Network（R-Net）：筛选 P-Net 中的边界框;

3、输出网络 Output Network（O-Net）：输出最终结果，即精确的边界框、置信系数和 5 点地标。

MTMN 的工作流程如下图：

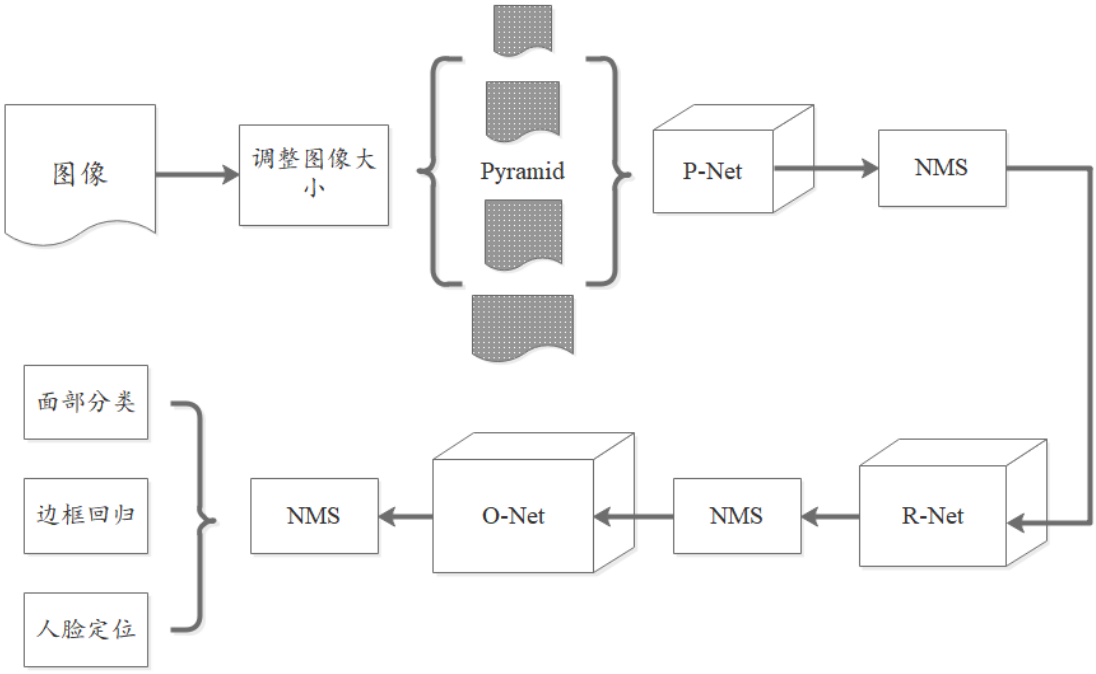


图 4-1 MTMN 工作流程图

整个人脸检测过程可以用其提供的 API 接口 face_detect()执行，它的函数原型为：box_array_t *face_detect(dl_matrix3du_t *image_matrix, mtmn_config_t *config);

其中，输入为：

image_matrix: 类型为 dl_matrix3du_t 的图像。

config: MTMN 模型的配置。

输出为：类型值包含面框，以及每个框的分数和标志。返回值类型为 box_array_t

为了更好的完成人脸检测任务，针对不对任务中图像的不同种类，我们人为地可以自定义 face_detect()的参数来更改检测配置。实际就是通过传入不同的 config 来更改配置：

- **min_face**:

  范围：[12，原始输入图像的最短边缘的长度）

  对于固定大小的原始输入图像，其越小，生成的不同大小的图像数量越大；可检测人脸的最小尺寸越小；处理时间越长。

- **pyramid**

  指定控制生成的金字塔的比例。

  范围：（0，1）

  对于固定大小的原始输入图像，其越大，生成的不同大小的图像数量越大；检测率越高；处理时间越长。

- **pyramid_times**

  指定控制生成的金字塔的数字。

  范围：[1，+inf）

- **type**

  选项：FAST/NORMAL

- **score threshold**

  范围：（0，1）

  对于固定大小的原始输入图像，其越大，滤出的候选边界框数越大；检测率越低。

- **nms threshold**

  范围：（0，1）

  对于固定大小的原始输入图像，其越大，检测到重叠人脸的可能性越高；检测到的同一人脸的候选边界框数越大。

- **candidate number**

  指定每个网络的输出候选框的编号。

  范围：P-Net: [1, 200]；R-Net: [1, 100]；O-Net: [1, 10]

  对于固定大小的原始输入图像，该值越大处理时间越长；O-Net 越大，检测到的人脸数量就越多。

我们通过调试结合官方文档给出的默认值结合得到了适合门禁系统中人脸检测最优配置：

```
mtmn_config.type = FAST;
mtmn_config.min_face = 80;
mtmn_config.pyramid = 0.707;
mtmn_config.pyramid_times = 4;
mtmn_config.p_threshold.score = 0.6;
mtmn_config.p_threshold.nms = 0.7;
mtmn_config.p_threshold.candidate_number = 20;
mtmn_config.r_threshold.score = 0.7;
mtmn_config.r_threshold.nms = 0.7;
mtmn_config.r_threshold.candidate_number = 10;
mtmn_config.o_threshold.score = 0.7;
mtmn_config.o_threshold.nms = 0.7;
mtmn_config.o_threshold.candidate_number = 1;
```

在项目中，将视频转化成人脸数据的具体的工作流程如流程图：

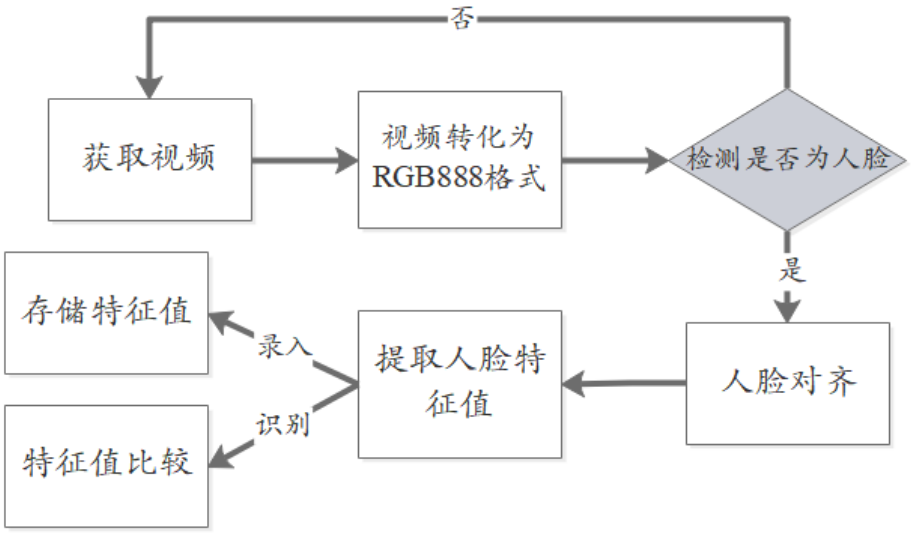


图 4-2 转化人脸数据工作流程图

4.2 摄像头算法设计

相机的驱动程序主要包括摄像头的初始化、传感器的配置以及启动摄像头服务器。主要流程如图：

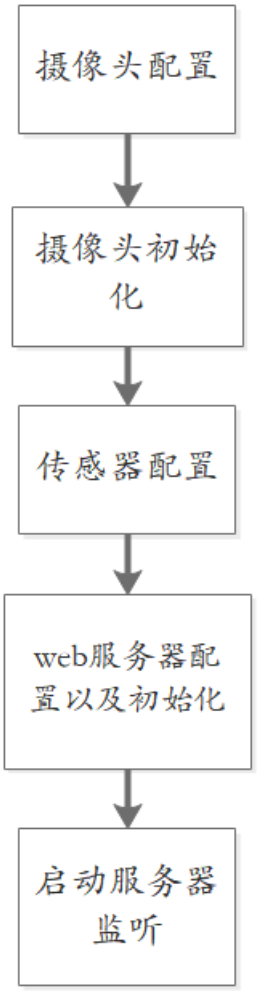


图 4-3 相机驱动程序流程

本项目使用的 esp_camera.h 包含了 Camera 的 SDK 方法，进行摄像头的驱动以及初始化配置。它通过将 camera_config_t 结构体写入到 esp_camera_init()函数，从而完成初始化配置。camera_config_t 结构体中主要包含以下几类参数：

- pin_xxxxx: 定义 Pin IO 口引脚
- ledc_timer、ledc_channe: 进行时钟配置
- xclk_freq_hz: 帧率，对于本项目使用的 ov2640，可以为 20KHz 或者 10KHz
- pixel_forma: 像素数据格式，包括 RGB565、JPEG 等等
- frame_siz: 图片输出大小，包括 QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA 等等
- jpeg_quality: PEG 图片质量，范围为 0-63，数字越小质量越高
- fb_count: 帧缓存数量，默认情况下都是 1

传感器配置使用了 esp_camera_sensor_get()函数获取当前摄像头的传感器，类型为 sensor_t，进而调用该类中的函数进行传感器的配置。一些重要的参数：

- sensor_id_t: 代表特定摄像头模块信息
- pixformat_t: 像素格式
- framesize_t: 图像帧数据大小(长 * 宽)
- gainceiling_t: 相机图像增益上限

摄像头服务器为 web 服务器，本项目中创建了一个 web 服务器的实例 camera_httpd，为其了注册 URI 处理程序，通过在 web 页面中发起请求进而控制摄像头的工作状态 g_state。为此我们自定义了 app_httpserver_init ()函数来初始化这个 web 服务器。

4.3 门禁系统程序

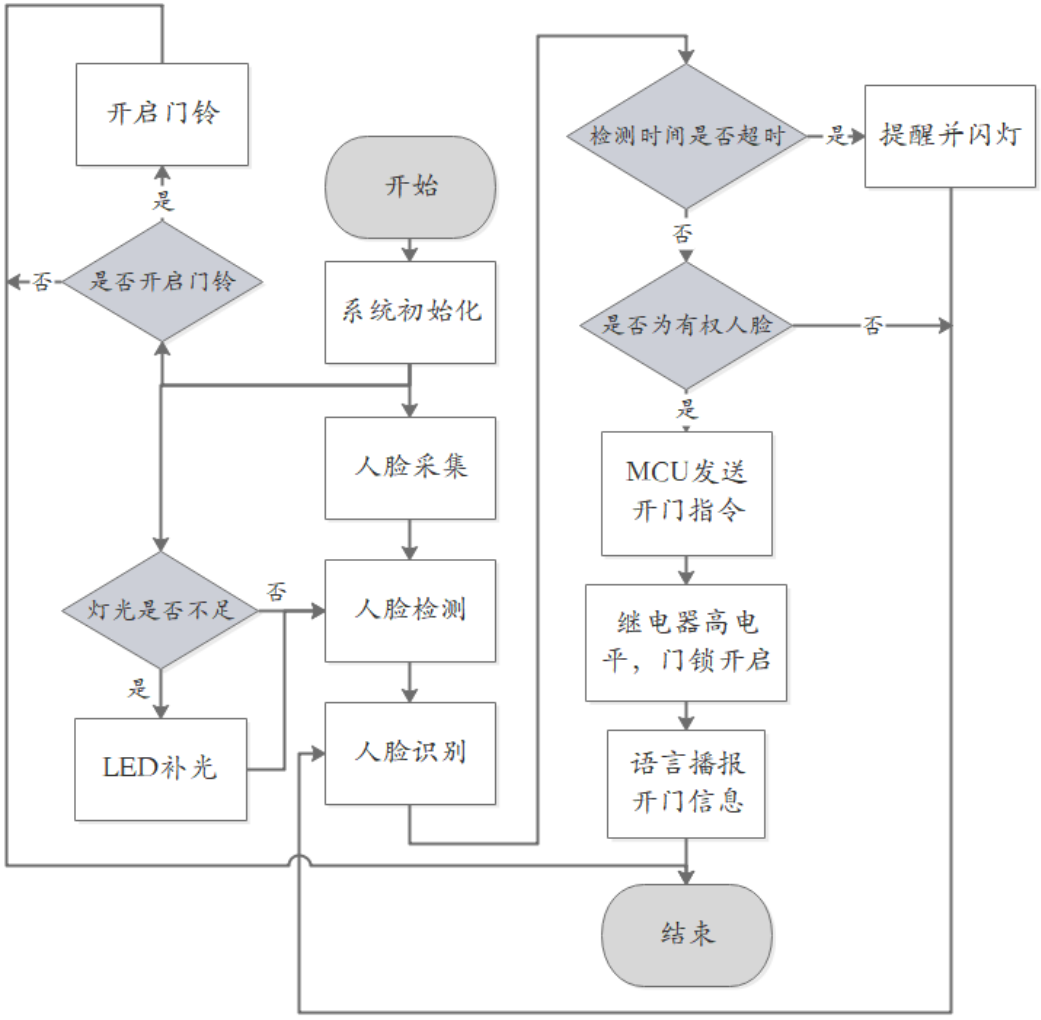


图 4-4 门禁系统流程图

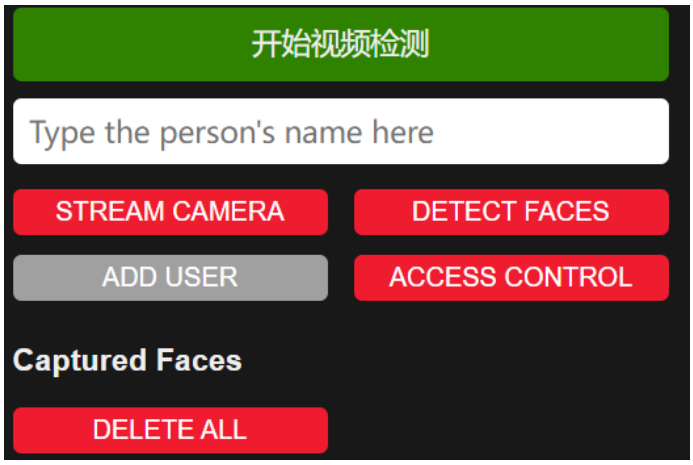
上电后，系统开始初始化配置。初始化分为三个相对独立的流程：一是选择是否开启门铃，若设置为开启门铃，则可以通过按键开关控制蜂鸣器；二是判断是否灯光不足，若灯光不足则开启 LED 补光功能；三是进行 MTMN 模型、相机的驱动程序的初始化。然后在通过 web 服务器端进行模式的选择，一般流程为人脸采集-人脸检测-人脸识别，当进行人脸识别时，若检测时间过长且长时间未开门，则进行提示并

闪灯，若检测为已登记的人脸，则通过 esp32-cam 发出高电平信号进而令继电器带动门锁开门，同时，将语音播报信息，预定任务完成。

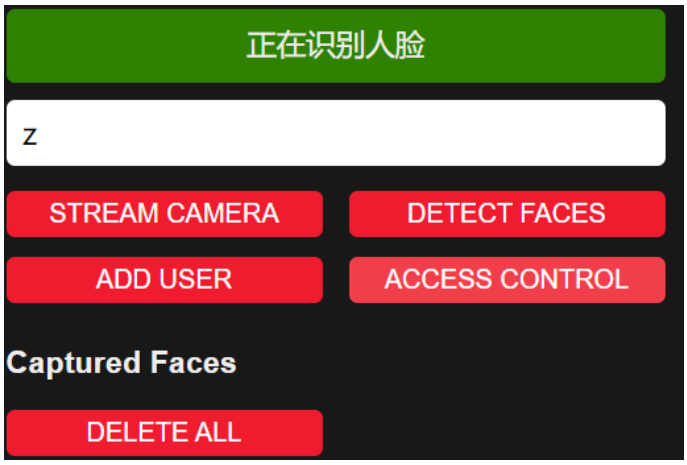
五、 作品测试

5.1 人脸数据录入

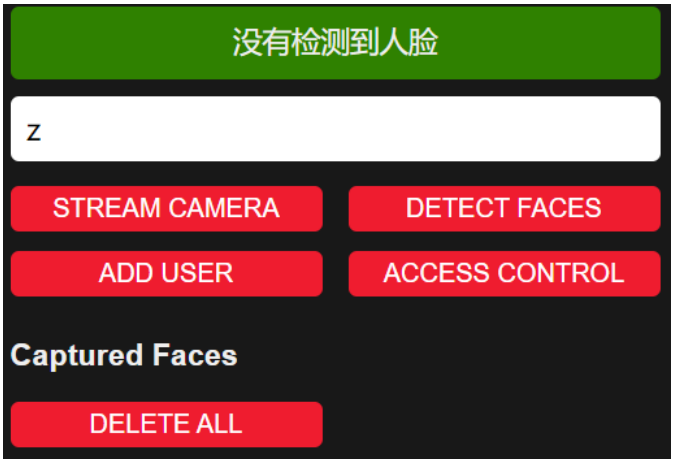
进入网站，系统提示开始视频检测。



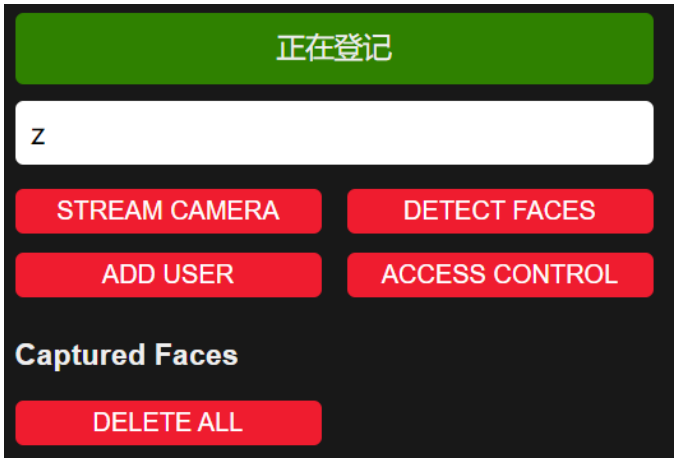
添加用户名（如下图 z），再点击 ADD USER 添加用户，如下图所示正在识别人脸。



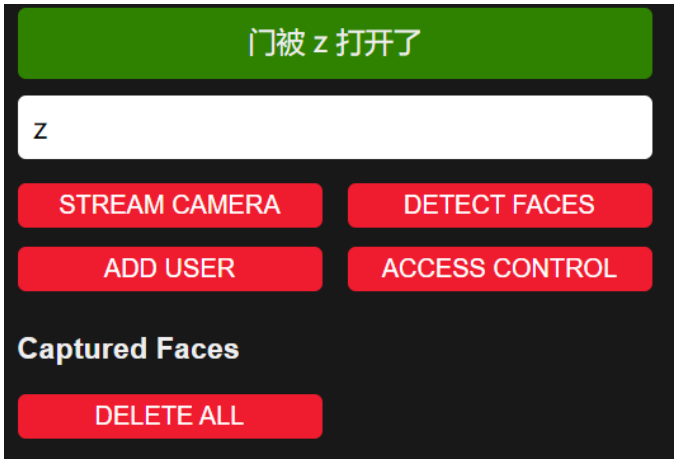
若摄像头未检测到人脸，则提示没有检测到人脸，如下图所示。



成功检测到后，显示正在登记，则提示正成功录入人脸数据。



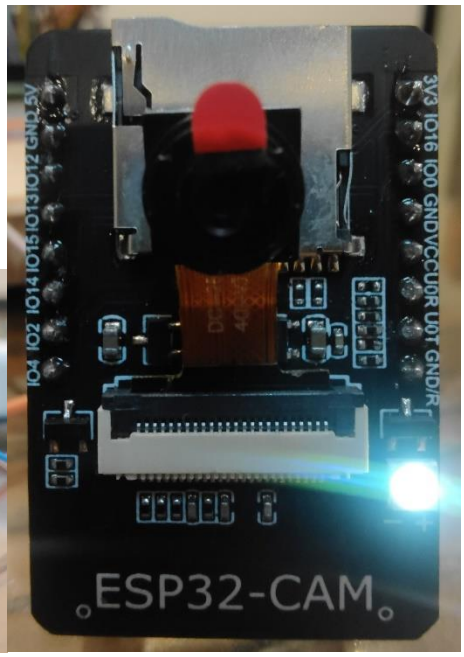
在使用人脸检测控制时，若人脸正确匹配，则显示门被用户打开了。



5.1 人脸识别功能

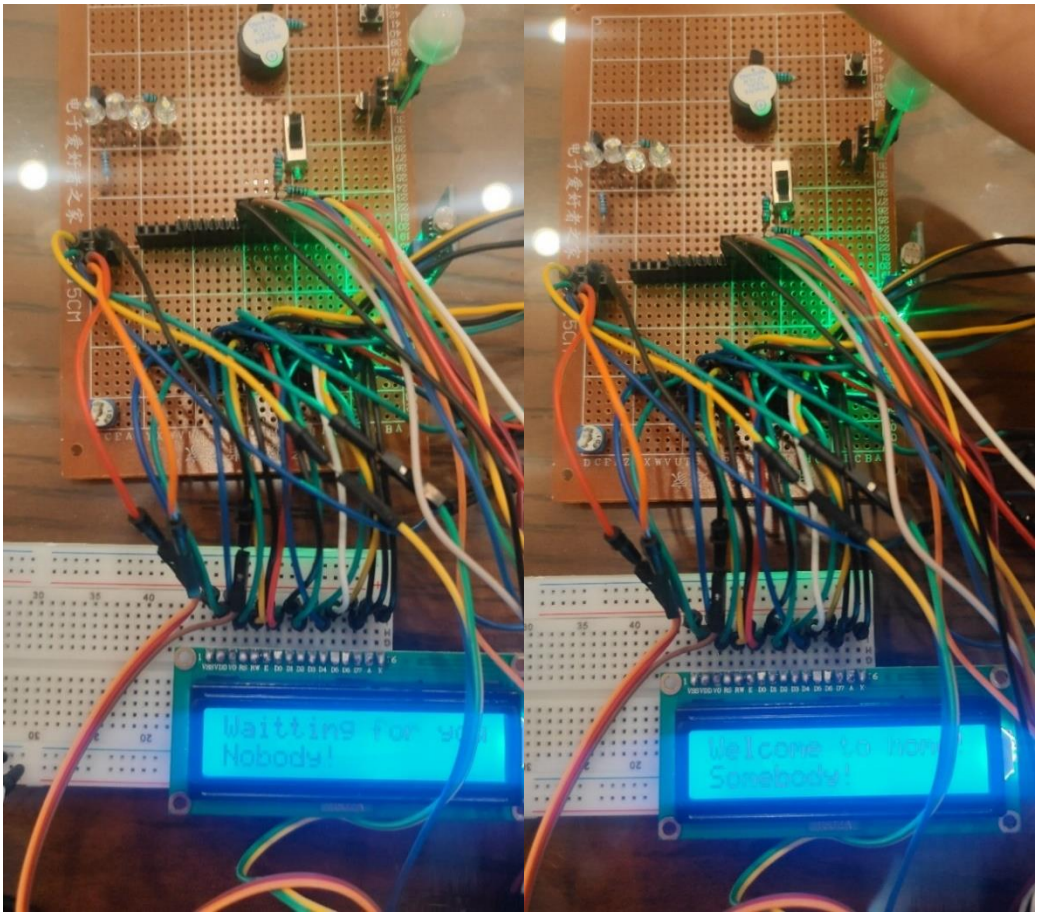
人脸识别未检测到人脸，门锁不打开。

检测到人脸，且认证成功，门锁自动弹开。



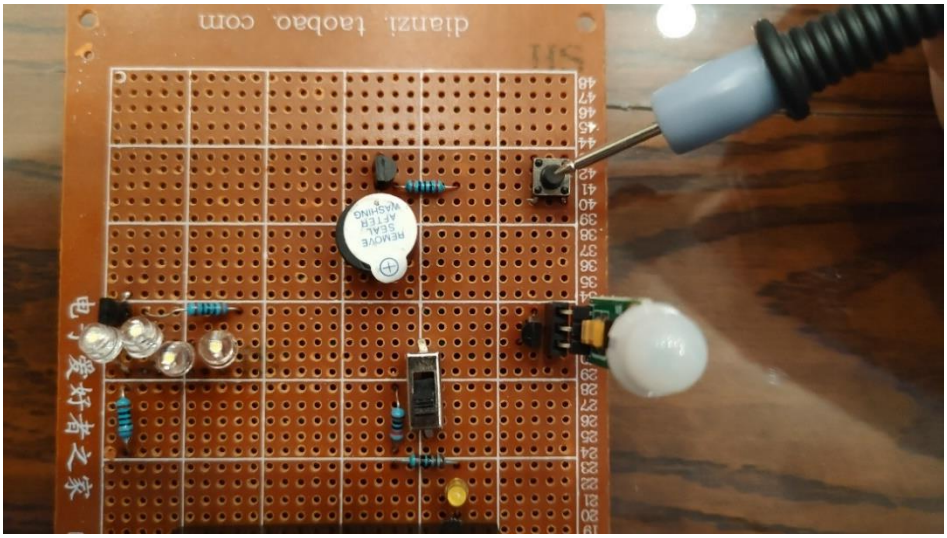
5.3 红外检测功能：

未检测到人体时显示 Waiting，检测到人体时，LED 显示屏显示 Welcome。



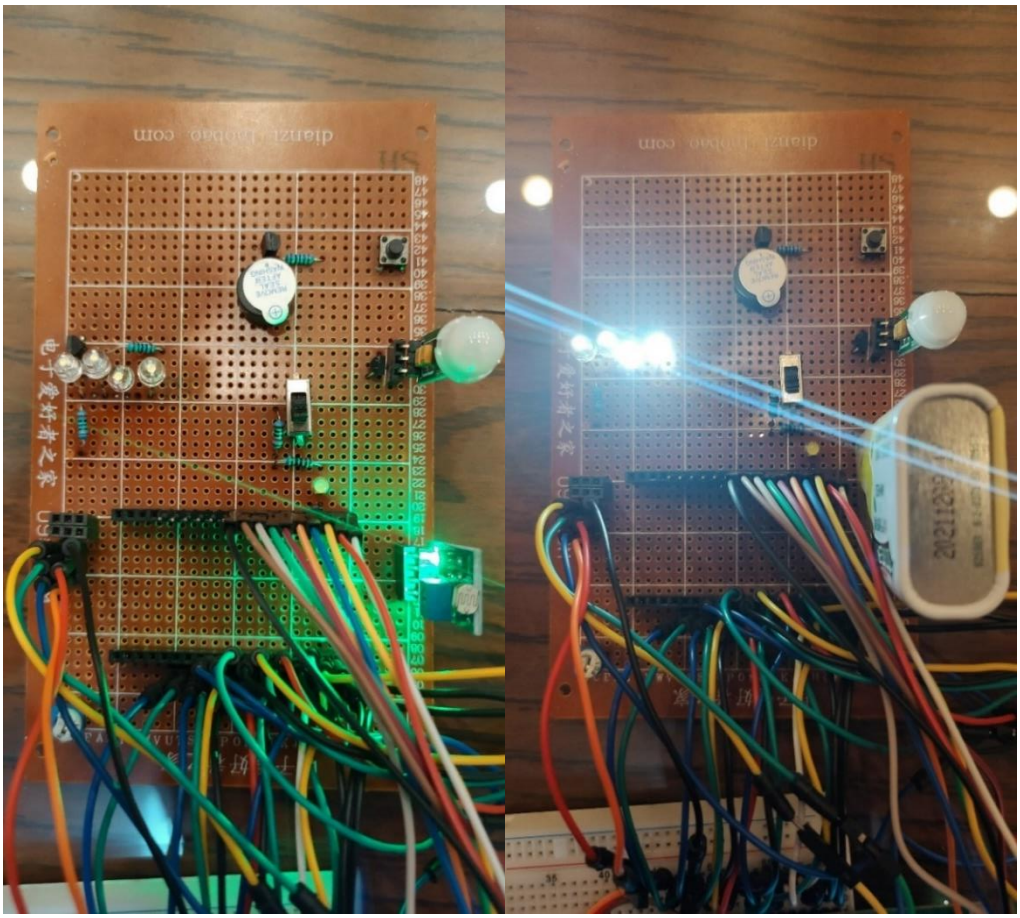
5.4 门铃功能

按住蜂鸣器控制开关则蜂鸣器持续发出声音



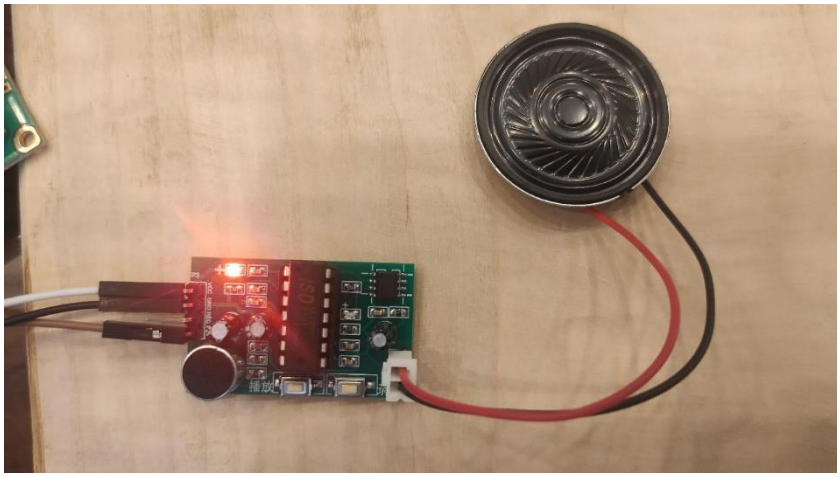
5.5 补光功能

正常室内光照条件下，高亮 LED 灯不亮，使用不透光物体遮挡后，高亮 LED 灯打开，并持续发光。



5.6 语音提示功能

开锁后，esp32-cam IO2 引脚电平信号驱动其进行播放语音提示，表示门已开锁。



六、结论与展望

6.1 存在的问题

1、系统中的人脸检测准确率还没有达到 100%。卷积网络模型仍需要更多数据集的训练。此外，由于该系统未加入活物检测的功能，所以存在误检验的问题，考虑

可以通过加入让被识别者张嘴做动作等方式实现活物检测。

2、系统的 stm32 芯片的成本相对较高，可以通过模块化生产、使用其他芯片进行替代（如 51 单片机）等方式逐步降低成本。

3、实际应用过程中可能遇到有人恶意进行识别的问题，会导致系统持续工作耗电，后续可以加入多次识别未果后将系统锁定一段时间这一类的功能。

6.2 智能门锁的未来

在智能家居的大浪潮和现代人注重隐私的大时代背景下，智能门锁出于以上两点考虑成为普通家庭也会安装的设备。市面上各式各样的智能门锁设计也就应运而生。我国的门锁设计侧重点从传统的机械结构设计发展到如今的智能化设计，不仅代表着我国工业设计水平进入一个新的台阶，还意味着我国消费者逐步上升的物质文化的需求。

而人脸识别技术作为一种新兴技术离完全应用可能尚早，但可以看出其未来前景非常广阔，尤其是在智能门锁这一领域。一方面，由于这几年的疫情，人们对无接触式门锁的热情高涨不下，越来越多的用户偏爱人脸识别的门锁：在线上消费端，根据 2021 京东双 11 数据显示，关于人脸锁相关的搜索同比增长 230%。而另一方面，这些智能门锁又普遍面临着识别准确率不高、功能不完善、产品不成熟的问题，这不正是我们这些电子类的学生投身于此、大展宏图的好机会吗？在可以遇见的将来，随着技术的逐步迭代与发展，智能化的浪潮将会也势必会席卷我们每一个人，参与到家家户户的衣食住行中来，或许，这人脸识别门锁就是下一只在风口上跳舞的猪。

七、附录

7.1 参考文献

[1]Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2019

[2]Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. 2016

[3] Ahmed, Karim;Torresani, Lorenzo. Connectivity Learning in Multi-Branch Networks. 2017

[4] Changpinyo, Soravit;Sandler, Mark;Zhmoginov, Andrey. The Power of Sparsity in Convolutional Neural Networks. 2017

[5] He, Kaiming;Zhang, Xiangyu;Ren, Shaoqing. Deep Residual Learning for Image Recognition. 2015

[6] Jia, Yangqing;Shelhamer, Evan;Donahue, Jeff. Caffe: Convolutional Architecture for Fast Feature Embedding. 2014

[7] Xie, Saining;Girshick, Ross;Dollár, Piotr. Aggregated Residual Transformations for Deep Neural Networks. 2016

7.2 使用说明

①将设备 esp32-cam 处通过 usb 转 ttl 接口连接到电脑上，启动电源对设备进行供电。

②进入 Aduino IDE 将电脑所连接 WIFI 无线网络的账号与密码输入到程序的 ssid 和 password 处。注意，设备目前不支持 5G 网络。

③保证设备与电脑同处一个网络后，断开 GPIO 0 与 GND 的连接。以 115200 的波特率打开串行监视器，按下 ESP32-CAM 板上复位按钮，ESP32 IP 地址应在串行监视器中打印，如下图所示。

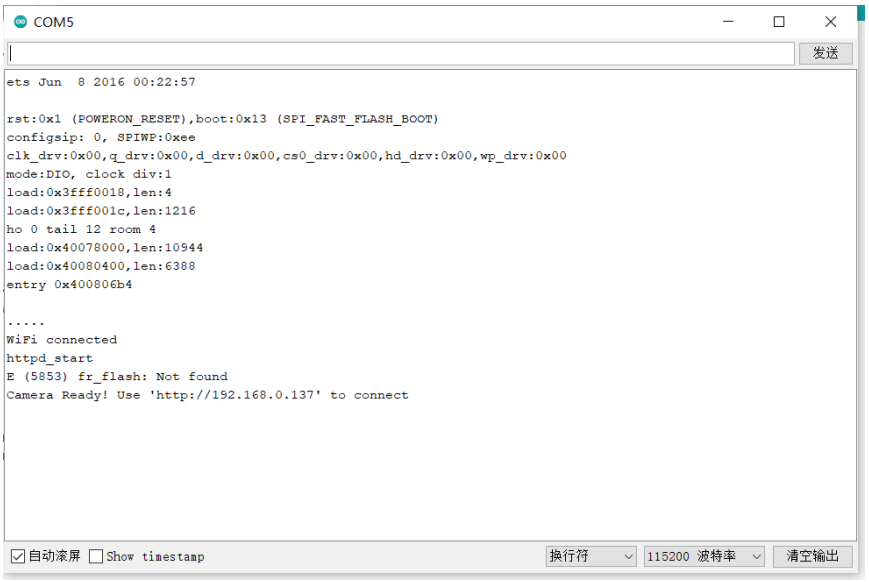


图 7-1 串行监视器显示

④打开浏览器，输入串口监视器上显示的 ESP32-CAM IP 地址，可以访问本地网络上的相机流服务器。在此网站上点击 ADD USER 并将人脸置于摄像头范围内即可录入人脸数据。

⑤准备工作结束后，点击 FACE CONTROL，进入人脸验证状态，此时保证设备接入 5V 电源。

⑥工作正常开始后，将电子锁上锁。当未录入人脸的用户刷脸后，设备无法认证，故保持关闭状态；待录入用户刷脸，设备认证成功后，即可自动打开门锁并弹出，且有语音播报提醒门锁已打开。

⑦设备上装有蜂鸣器门铃，其他人可按住按键开关进行响铃。且门铃装置设置有拨动总控开关，当关闭波动开关后，门铃关闭，即使按住按键开关也不会发声。

⑧设备仍有以下辅助功能，为全自动智能工作无需用户操作：

(1)设备外部设置有红外感应装置，未感应到人体时保持待机状态，显示器显示“Waiting”；感应到人体时显示器显示“Welcome”

(2)设备外部设置有感光装置，在感应到有人且为黑暗状态下设备自动打开高亮度 LED 灯进行照明。

(3)在红外感应装置未感应到人体时，即使是黑暗状态下高亮 LED 灯不会开启，进而实现节约用电。

(4)设备在约 25s 未接收到任何指令时自动关闭，进而实现节约用电。

7.3 程序代码

```
/*
esr32-cam代码,包含:
camera_index.h
camera_pins.h
face_test.c
*/

//camera_index.h
//File: index_ov2640.html.gz, Size: 4316
//web网页的源码
#define index_ov2640_html_gz_len 4316
const uint8_t index_ov2640_html_gz[] = {
0x1f,0x8b,0x08,0x08,0x76,0x8e,0x5e,0x5d,0x00,0xff,0x69,0x6e,0x64,0x65,0x78,0x
2e,0x68,0x74,0x6d,0x6c,0x2e,0x67,0x7a,0x00,0xb5,0x58,0x6d,0x6f,0x22,0x39,0x12,0xf
e,0x3c,0xfc,0x0a,0x5f,0xef,0x9d,0x8e,0xdc,0x42,0x43,0x42,0x76,0x95,0x61,0x42,0x74,0
x4c,0x86,0xdc,0xac,0x94,0xb9,0x48,0x49,0x46,0x7b,0xdf,0x66,0x4d,0xb7,0x01,0xdf,0x
b8,0xdb,0xbd,0xb6,0x1b,0xc2,0xad,0xf2,0xdf,0xb7,0xca,0x2f,0x8d,0x9b,0x90,0x49,0x76
,0xa5,0x03,0x89,0x06,0xbb,0xea,0x71,0xd5,0xe3,0xaa,0x72,0x99,0xf3,0xbf,0xe4,0x32,0
x33,0xdb,0x8a,0x91,0x95,0x29,0xc4,0x45,0xe7,0x3c,0x3c,0x18,0xcd,0xe1,0x51,0x30,0x
43,0x49,0xb6,0xa2,0x4a,0x33,0x33,0x49,0x6a,0xb3,0xe8,0x9f,0x25,0x61,0xb8,0xa4,0x0
5,0x9b,0x24,0x6b,0xce,0x36,0x95,0x54,0x26,0x21,0x99,0x2c,0x0d,0x2b,0x41,0x6c,0xc3
,0x73,0xb3,0x9a,0xe4,0x6c,0xcd,0x33,0xd6,0xb7,0x3f,0x7a,0xbc,0xe4,0x86,0x53,0xd1,0
xd7,0x19,0x15,0x6c,0x72,0x8c,0x18,0x86,0x1b,0xc1,0x2e,0xae,0x68,0xc6,0xc8,0x2d,0x
cb,0xe4,0x12,0x25,0x64,0x49,0xa6,0x59,0xc6,0xb4,0x26,0x97,0x00,0xa6,0xa4,0x38,0x1
f,0x38,0xb1,0xce,0xb9,0x36,0x5b,0x7c,0xfe,0xb3,0x60,0x39,0xa7,0x44,0x96,0x62,0x4b,
0x74,0xa6,0x18,0x2b,0x09,0x2d,0x73,0xd2,0x2d,0x78,0xe9,0x96,0x1a,0x93,0xb3,0x1f,0
x86,0xd5,0xc3,0x11,0xf9,0xad,0xf3,0x66,0x2e,0xf3,0x2d,0x3e,0xdf,0x90,0x9c,0xeb,0x4
a,0xd0,0xed,0x98,0x2c,0x04,0x7b,0x78,0xd7,0x79,0xf3,0xd8,0x79,0x43,0xbe,0xf3,0x06,
0xf7,0x15,0x5f,0xae,0x8c,0x93,0x2b,0xa8,0x5a,0x02,0x92,0x60,0x0b,0x33,0x26,0xc7,0x
80,0x83,0xb2,0x84,0x74,0x1e,0x3b,0x1e,0x8b,0xe0,0x6b,0x01,0x8a,0xfd,0x05,0x2d,0xb
8,0x00,0xc4,0xa9,0x02,0xc7,0x7a,0xe4,0x23,0x13,0x6b,0x66,0x78,0x46,0x7b,0x44,0xd3
,0x52,0xf7,0x35,0x53,0x7c,0xf1,0xce,0xc9,0xcf,0x69,0xf6,0x75,0xa9,0x64,0x5d,0xe6,0x
63,0xf2,0xdd,0xf1,0x19,0xbe,0xfd,0x4c,0x26,0x85,0x54,0x30,0x38,0xbb,0xc2,0xb7,0x1f
,0xb4,0xf0,0x9a,0xff,0x8f,0x81,0x09,0x3f,0xa2,0x09,0x8f,0x9d,0x9d,0xb1,0x68,0x5a,0x
b0,0xa3,0xa0,0x0f,0xc1,0xe9,0xd3,0xa1,0x33,0xd6,0x3a,0x08,0x7a,0x6d,0xa5,0xe0,0xe1
,0xeb,0xb4,0xb4,0x51,0x8c,0x16,0x41,0xdc,0x8b,0x1e,0x0f,0x87,0x7f,0x0b,0xd3,0xd4,0
xd4,0xba,0xef,0x29,0x0d,0x62,0x2b,0x86,0x6b,0x8c,0xc9,0xc9,0x0f,0x88,0xe8,0xdc,0x9
6,0x2a,0x67,0xe0,0x5d,0x29,0x4b,0xe6,0x87,0x2a,0x9a,0xe7,0xbc,0x5c,0x06,0x6e,0x1b,
0x77,0x61,0xe0,0xac,0x7a,0x18,0x9c,0x9c,0x54,0x0f,0x4f,0xe9,0xf3,0x9b,0x32,0x97,0x
c6,0xc8,0xa2,0xa5,0xea,0x56,0xe8,0x2b,0x9a,0xf3,0x5a,0x8f,0x49,0xb4,0x74,0xc4,0xf8,
0x12,0xa3,0xc4,0x8f,0x1b,0xf6,0x60,0xfa,0x54,0xf0,0x65,0x39,0x26,0x19,0x30,0xc3,0x
94,0xf3,0xa9,0x62,0x4a,0x43,0xf0,0xb5,0x5c,0x76,0x1e,0xff,0x19,0xe7,0x4e,0xc0,0xc2,
0xff,0xb7,0x87,0x8a,0xb9,0x08,0x89,0xd6,0x9f,0xcb,0x07,0x0c,0x1b,0x6b,0x82,0xd7,0x
```

83,0x21,0xeb,0xe0,0xbc,0x06,0xe4,0xc6,0xbf,0x26,0x1b,0xe6,0x42,0x66,0x5f,0x5b,0x36
,0xd8,0x25,0xc8,0x70,0xdf,0xa3,0x21,0x39,0x06,0xc3,0xf7,0x5c,0x0f,0x52,0x21,0x98,0
xce,0x02,0x5d,0x82,0x97,0xac,0xdf,0x70,0x06,0x6e,0x87,0x68,0xaf,0x81,0x65,0x50,0xa
c,0x24,0x77,0xd4,0xb7,0x72,0x60,0xb1,0x38,0x98,0x2f,0x8b,0xc5,0x68,0x38,0x3a,0x7d
,0x89,0x8e,0x27,0x39,0x63,0x47,0x65,0x6d,0xd0,0x16,0x6b,0x2a,0xb0,0x90,0x3a,0x1a,
0xf4,0x5e,0xcc,0x9e,0x5a,0xba,0x1b,0x96,0xc6,0x2b,0xb9,0x66,0x2a,0xc8,0xec,0xd9,0x
72,0xfa,0xf6,0x34,0x8f,0x65,0x69,0x66,0xf8,0x9a,0x1d,0x16,0x3e,0x39,0xce,0x4e,0x5c,
0x4a,0x79,0x61,0xe0,0x9d,0xce,0x05,0xcb,0x83,0x78,0xe0,0x23,0x67,0x0b,0x5a,0x0b,0
x73,0xc8,0x7b,0x3a,0xc4,0xb7,0xb3,0x3e,0x4e,0xfd,0x85,0x90,0x14,0xc8,0xc5,0x21,0x
37,0xd9,0x4a,0x71,0x3f,0x6b,0xc7,0xdc,0x34,0x2f,0xe8,0x92,0xf5,0xb1,0x1e,0x50,0x20
,0xa4,0xf1,0xae,0x92,0xda,0x96,0x5d,0x90,0x65,0x82,0xa2,0x2b,0x4e,0xbc,0x5d,0x02,0
x9e,0x56,0x0c,0x90,0xa9,0x45,0x98,0x16,0x5c,0x03,0xf7,0x58,0x9f,0x0f,0xa7,0xc3,0x6
e,0x97,0x7c,0x8c,0x39,0x7f,0x04,0x6f,0xac,0x88,0x24,0xc3,0x56,0xe5,0x4c,0x30,0xf3,0
x0c,0xb1,0xdf,0x88,0x08,0x48,0xda,0x5d,0xb8,0x3d,0x09,0xac,0x43,0xe9,0xff,0x34,0x6
2,0x31,0x51,0x9f,0x8b,0x58,0x30,0x6d,0x35,0xda,0xf1,0xd2,0x4a,0xdb,0x91,0x63,0xe6,
0x7c,0xe0,0xcf,0xaa,0xf3,0x81,0x3f,0x45,0xf1,0xec,0x80,0x47,0xce,0xd7,0x84,0xe7,0x9
3,0x24,0x2e,0xe5,0x70,0x12,0x12,0xd2,0xcc,0x38,0xda,0x77,0xdb,0x04,0x87,0xaa,0xa0,
0x5a,0x4f,0x92,0xbd,0xed,0x4b,0x2e,0xc8,0x39,0x2f,0x96,0x91,0x4e,0x42,0xb4,0xca,0x
26,0x09,0x4e,0x0c,0x00,0x0d,0x17,0x77,0x8f,0xfd,0x45,0x6d,0x4c,0xec,0xaf,0x1a,0x17,
0x74,0x84,0xd0,0x15,0x2d,0x9d,0x56,0xad,0x14,0x6a,0x39,0x91,0xe4,0x02,0x7c,0x83,0
xa9,0x66,0x91,0x08,0xc4,0x55,0xd0,0x3e,0x36,0x05,0x16,0x1d,0xa6,0x78,0x59,0xd5,0x
26,0x9a,0x4c,0x08,0x36,0x1a,0x93,0x04,0x77,0x21,0x21,0x6b,0x2a,0x6a,0xf8,0x91,0x1
0,0x58,0x34,0x63,0x2b,0x29,0x60,0x1b,0x27,0xc9,0x3d,0xb6,0x22,0x66,0xc5,0x88,0x5
3,0xf9,0xbb,0xb6,0x6d,0x06,0x64,0xaa,0x72,0xb0,0xed,0x85,0x3d,0x3b,0x3e,0xad,0xc3,
0xba,0xbe,0xd8,0xe1,0xc2,0xee,0x2b,0x98,0xef,0x38,0xf2,0xf2,0x8e,0xf8,0xbb,0xfb,0xd
b,0xd9,0xf4,0x13,0xb9,0x9c,0x7e,0x9a,0xdd,0x4e,0xcf,0x07,0x4e,0xf4,0x59,0x88,0x1c,0
xa2,0x31,0xc3,0x2e,0xc7,0x41,0x78,0x1a,0x3f,0xcc,0xee,0x67,0x97,0xf7,0xe4,0x6a,0x7
a,0x39,0xbb,0x8b,0x21,0xfe,0x9c,0x99,0x19,0xad,0x4c,0x0d,0x8e,0xb6,0xec,0x24,0xb6,
0x03,0x9a,0x24,0x33,0x0c,0x40,0xe2,0xda,0x2e,0x42,0xe7,0x50,0xa3,0xc8,0x9c,0x2d,0
xa4,0x62,0xc4,0xa9,0x41,0x02,0xc1,0xec,0x02,0xb8,0x4c,0x2e,0xa6,0x1f,0x3e,0x90,0xc
f,0x77,0xb3,0xdb,0x17,0xbd,0x52,0xae,0xf3,0xd2,0xbb,0x35,0xbd,0x63,0xd3,0x4b,0xf0,
0xe8,0x8e,0x5c,0xde,0xfc,0xfb,0xfe,0xf6,0xe6,0xfa,0x45,0xd7,0x2a,0x26,0x2b,0xd1,0x6
c,0xfc,0x6a,0x74,0x71,0xe9,0x5c,0xc9,0x09,0x36,0x78,0x1a,0x32,0x61,0xe4,0xe7,0x6a,
0xe8,0x2e,0xed,0x97,0x81,0xfb,0xf6,0x07,0x99,0x72,0x65,0xe1,0x0b,0x15,0x02,0xc9,0
xbf,0x06,0xfa,0xc9,0xf4,0xfa,0x90,0x7d,0xe1,0x01,0x4d,0x22,0xaf,0xcc,0x45,0x07,0xda
,0xdc,0xba,0x80,0x50,0x4e,0xa1,0xd6,0xcc,0xd6,0xf0,0xe5,0x1a,0xaa,0x16,0x83,0x5c,0
xea,0x26,0x1f,0x6e,0x3e,0x61,0xbb,0x89,0x63,0x92,0xe6,0x2c,0x4f,0x7a,0x64,0x51,0x9
7,0x50,0xd8,0x65,0xd9,0x65,0x28,0x8a,0xcd,0x24,0x81,0x80,0x55,0x50,0x87,0x34,0xfb
,0x28,0xb5,0x21,0x13,0xd2,0x00,0xc2,0x19,0x0a,0x95,0x53,0x96,0xa9,0x04,0xee,0xb8,
0xed,0x34,0x50,0xd4,0xc5,0xdc,0x67,0x25,0x40,0xb6,0x51,0xfb,0x9e,0x24,0xe3,0xb3,0
xe3,0x04,0x65,0x20,0x21,0x61,0xe0,0xe7,0xbb,0x2f,0x9f,0x6f,0xaf,0x41,0x24,0xd9,0xe
8,0xf1,0x60,0x90,0x80,0xc4,0x86,0x97,0xb9,0xdc,0xec,0x60,0x57,0x8d,0xe2,0x49,0xa4,
0xb8,0xd1,0xa0,0x54,0xb2,0x0d,0xf9,0x99,0xcd,0xef,0xe0,0x14,0x67,0xa6,0xeb,0xb0,0x

8e,0xde,0x75,0x1a,0x21,0x6c,0xce,0x63,0x53,0x97,0xcc,0xcc,0x04,0xc3,0xaf,0xef,0xb7,0x3f,0xe5,0xdd,0x50,0x3b,0x40,0x25,0x68,0xb8,0xb4,0xbb,0x92,0xaa,0xb8,0xe2,0x4c,0xe4,0xdf,0x52,0xf6,0x49,0x1d,0x29,0x3b,0xb8,0xf7,0x6e,0xb7,0xbe,0xa1,0xd9,0xce,0xca,0x08,0xc0,0x25,0xd9,0xab,0x01,0x7c,0x4e,0x46,0x00,0x3e,0x83,0x5e,0x8d,0x10,0x32,0x2e,0x82,0x68,0x52,0xe2,0xd5,0x20,0xbb,0x24,0x6a,0xb9,0x82,0x81,0x3a,0x15,0xe2,0x65,0x98,0x28,0xa6,0xdd,0xe6,0x0d,0x06,0x64,0x09,0x75,0x1e,0xc2,0x50,0xb1,0x5f,0x6b,0x56,0x66,0xdb,0x1e,0xc9,0x6b,0x65,0xc3,0x01,0xa6,0xe9,0x04,0xf7,0x7d,0x5a,0xe7,0x5c,0xda,0xb8,0x7d,0x30,0x5d,0xbb,0x70,0x08,0x5a,0x02,0x77,0x2c,0x65,0xee,0xf0,0xb4,0xec,0x6e,0x7a,0x0f,0xbd,0xed,0x91,0x3b,0xaf,0xd6,0x13,0x9a,0xc2,0x95,0x89,0x1a,0x76,0xa3,0x33,0x2e,0xe0,0xbc,0x97,0xca,0x69,0x12,0x52,0x37,0x73,0xff,0x82,0x95,0xc3,0xe8,0x3a,0x05,0x67,0x4a,0xe0,0xb8,0x5b,0x37,0x23,0x8d,0x4d,0xa9,0xab,0xe0,0xfe,0xa8,0x5c,0xa7,0xae,0xb8,0xeb,0x5f,0x6b,0x0a,0x84,0x7a,0xd4,0x46,0x9f,0xc2,0x81,0xae,0x0d,0x2f,0xad,0x0f,0x61,0xcd,0x14,0xbd,0xf4,0x30,0x9b,0x7f,0x0c,0xd3,0x21,0x74,0x4b,0x0e,0x0b,0xce,0x1b,0x85,0x4a,0xfe,0xfc,0xb9,0xe7,0x05,0x6b,0x0c,0xd0,0x46,0x56,0xed,0xa9,0xef,0xb7,0xa8,0x3c,0x3c,0xb6,0x22,0x8f,0xc8,0xe0,0x46,0xa7,0xb2,0x94,0x15,0xdc,0x0e,0x27,0xa4,0x7b,0x44,0x26,0x17,0xfe,0xc8,0xc6,0xbd,0x91,0x82,0x41,0x72,0x2d,0xbb,0xbf,0x00,0x79,0x68,0x1b,0x14,0x29,0x23,0xc9,0x5f,0x7f,0x73,0xe9,0xf3,0xf8,0x8b,0x43,0xc1,0x0f,0x8b,0x52,0xc0,0x75,0x14,0x0e,0x5f,0x00,0x6a,0xbe,0x05,0x34,0xbe,0x20,0x5d,0xf4,0x5a,0x2e,0xc2,0x5c,0x9a,0x53,0xb8,0x19,0x4f,0x26,0x90,0xcf,0x10,0xdd,0x50,0x92,0x13,0x57,0x3b,0x82,0x74,0x2c,0x96,0xea,0x7a,0x0e,0x42,0xdd,0x61,0x8f,0x9c,0x81,0x89,0xa0,0x82,0x8d,0x94,0x2d,0xe0,0x3b,0x25,0xd8,0xed,0x3c,0xc7,0x1a,0x7a,0x2f,0xef,0xec,0x75,0xf7,0x20,0xc4,0xdb,0x23,0xcf,0x0e,0x18,0x4e,0x98,0xd0,0xec,0xc9,0x62,0x16,0xdf,0x07,0x1a,0x2e,0xa1,0x5b,0x6b,0x34,0xc1,0x6a,0xab,0xf5,0x95,0x92,0x85,0x5f,0xec,0x75,0xb0,0x52,0xaa,0x2f,0x48,0x76,0x0b,0x13,0x2c,0xdf,0x85,0xe1,0xf1,0xb0,0x77,0x32,0x1a,0xf5,0xa0,0x3b,0x3b,0x7a,0x17,0x8f,0x8f,0x7a,0x3f,0x0e,0x47,0xbd,0x13,0x1c,0x6e,0x2f,0x14,0xe3,0x3c,0x9b,0x38,0x7b,0xdd,0xc9,0x51,0xca,0x61,0x43,0xd5,0xc7,0xfb,0x4f,0x58,0x50,0x63,0x33,0x03,0xf8,0x37,0xd1,0xf6,0xda,0xa0,0x23,0x08,0x34,0x68,0xe3,0xd2,0x5d,0xeb,0x89,0x55,0xda,0x5e,0x26,0x7d,0x74,0x63,0xac,0xed,0x3e,0x9f,0x50,0xc3,0xa1,0x0c,0xd0,0x32,0xc3,0xe8,0x78,0x2f,0xe4,0x7c,0xc7,0x0d,0x9e,0x0e,0xb5,0x12,0x37,0xf3,0xff,0x42,0xf4,0x01,0x28,0x44,0x5d,0xc8,0x4b,0x3b,0x04,0xbf,0x5b,0x48,0x0d,0x37,0x58,0xce,0x53,0xe8,0xf3,0x40,0xa7,0xd1,0x77,0x73,0x68,0x82,0x0d,0xfc,0xb8,0x04,0x43,0xf0,0x66,0x82,0x67,0x5f,0xf7,0x72,0x00,0xa2,0x5a,0x33,0xe0,0xbe,0x55,0xfd,0x6d,0xbc,0xc7,0xe5,0xf7,0x25,0xe5,0xb8,0xf6,0x5a,0xe5,0x56,0xe9,0x7d,0x46,0xdb,0x1d,0x19,0x5f,0x6c,0x0b,0xf3,0x8a,0x93,0xc5,0x95,0x06,0xe7,0x61,0xb3,0xb0,0x5f,0x67,0x8c,0x47,0x65,0x84,0xb7,0x33,0x64,0xaf,0x80,0xbf,0xe4,0xc8,0x5e,0xf5,0xf6,0x44,0xb4,0x8a,0xf7,0xcb,0x5c,0xc4,0x05,0xdc,0x63,0xec,0x9d,0xa4,0x80,0xf1,0x95,0x6d,0xeb,0x6a,0xbf,0x20,0xb5,0x58,0x6b,0x2e,0x88,0x13,0x68,0xe4,0x20,0x0d,0x3c,0x58,0x5c,0xdb,0x9f,0x4f,0x55,0x8f,0x08,0x87,0x87,0x93,0xd1,0xb6,0x19,0xb4,0xb7,0x33,0x88,0x45,0x32,0x57,0x72,0x03,0xff,0x39,0xd8,0xbf,0xce,0x9a,0x5a,0x68,0xac,0x0c,0xb6,0x42,0xf1,0x7e,0x40,0x7d,0x57,0xdb,0x3b,0x00,0xc9,0xf0,0x84,0x48,0x6a,0xef,0x16,0x78,0xbc,0xe2,0x82,0x91,0x6e,0xd0,0x49,0x17,0x5c,0x69,0x73,0x09,0x83,0xf9,0x2e,0xba,0x0f,0x4c,0xa6,0x8a,0x15,0xd0,0xaf,0x86,0x6a,0xe2,0xf2,0x65,0x9f,0x1f,0xbb,0xd7,0x98,0x62,0x3e,0xbb,0x9e,0x65,0xc6,0x28,

```
0x17,0x13,0x36,0xde,0x77,0x67,0xde,0x5e,0x99,0x8c,0x43,0x23,0x2a,0xfe,0x7f,0xd0,0x
61,0xe0,0xd1,0x12,0xf8,0x93,0x61,0x45,0xac,0xe0,0xd2,0xd5,0x07,0x6d,0x17,0x0a,0x7
7,0xac,0x90,0x09,0xa9,0xd9,0x0b,0x1a,0x78,0x8d,0x0a,0x3a,0x8d,0x7c,0x6a,0xdb,0x5f,
0x4b,0x1e,0xb8,0x13,0xe2,0xea,0xa9,0x18,0x47,0x1a,0x22,0x07,0xf7,0xe7,0x9f,0xb6,0x
b8,0x36,0x7c,0xe3,0xbe,0x76,0xb7,0x5f,0x51,0x22,0xe0,0x26,0x1d,0xcc,0x2b,0x20,0x3
b,0xf8,0xe7,0xc9,0x48,0x69,0x05,0x05,0x3f,0xb7,0xdb,0xdb,0xf5,0x48,0xcf,0xfa,0x0a,0
xff,0xe4,0xe2,0x49,0x68,0xc5,0x8e,0x52,0x6c,0x58,0x7c,0xbf,0x7d,0xc8,0x8d,0x83,0x2
b,0x34,0xbe,0xed,0xab,0x27,0xff,0xf1,0xe1,0xd2,0x84,0x5d,0xac,0x16,0xb0,0x76,0x8d,
0xc1,0x0b,0x51,0x85,0x6e,0xc2,0x1d,0xd7,0xdf,0x19,0xe0,0x5a,0xe1,0x6e,0xee,0x03,0x
fb,0xaf,0xf8,0xef,0x5f,0x9e,0x6a,0x05,0x2c,0x17,0x00,0x00
    };

    //camera_pins.h
    //定义摄像头引脚
    #if defined(CAMERA_MODEL_AI_THINKER)
    #define PWDN_GPIO_NUM     32
    #define RESET_GPIO_NUM    -1
    #define XCLK_GPIO_NUM      0
    #define SIOD_GPIO_NUM     26
    #define SIOC_GPIO_NUM     27
    #define Y9_GPIO_NUM       35
    #define Y8_GPIO_NUM       34
    #define Y7_GPIO_NUM       39
    #define Y6_GPIO_NUM       36
    #define Y5_GPIO_NUM       21
    #define Y4_GPIO_NUM       19
    #define Y3_GPIO_NUM       18
    #define Y2_GPIO_NUM        5
    #define VSYNC_GPIO_NUM    25
    #define HREF_GPIO_NUM     23
    #define PCLK_GPIO_NUM     22

    #else
    #error "摄像头种类未选择" //报错
    #endif

    //face_test.c
    #include "Arduino.h"
    #include "fd_forward.h"
    #include "fr_forward.h"
    #include "fr_flash.h"
    #include "esp_http_server.h"
    #include "esp_timer.h"
```

```
#include "esp_camera.h"
#include "camera_index.h"
#include <ArduinoWebsockets.h>

//WiFi账号密码
const char* ssid        = "---";     //网络名称
const char* password = "---"; //网络密码
#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

//选择摄像头种类(我们用的是AI_THINKER)
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"

using namespace websockets;
WebsocketsServer socket_server;

camera_fb_t * fb = NULL;

#define relay_pin 2                    //定义继电器引脚
unsigned long door_opened_millis = 0;
long interval = 5000;                  //  开锁时间5000ms
bool face_recognised = false;

long last_millis = 0;    //上一次检测时间

//esp32-cam初始化一个服务器
void server_init();

typedef struct
{
   uint8_t *image;
   box_array_t *net_boxes;
   dl_matrix3d_t *face_id;
} http_img_process_result;

//提前配置好MTMN人脸检测模型的参数，参阅通用方案
static inline mtmn_config_t mtmn_config_func()
{
   mtmn_config_t mtmn_config = {0};
   mtmn_config.type = FAST;
   mtmn_config.min_face = 80;
   mtmn_config.pyramid = 0.707;
```

```
    mtmn_config.pyramid_times = 4;
    mtmn_config.p_threshold.score = 0.6;
    mtmn_config.p_threshold.nms = 0.7;
    mtmn_config.p_threshold.candidate_number = 20;
    mtmn_config.r_threshold.score = 0.7;
    mtmn_config.r_threshold.nms = 0.7;
    mtmn_config.r_threshold.candidate_number = 10;
    mtmn_config.o_threshold.score = 0.7;
    mtmn_config.o_threshold.nms = 0.7;
    mtmn_config.o_threshold.candidate_number = 1;
    return mtmn_config;
}
mtmn_config_t mtmn_config = mtmn_config_func();

face_id_name_list st_face_list;
static dl_matrix3du_t *aligned_face = NULL;
httpd_handle_t camera_httpd = NULL;

//定义摄像头所处模式的类型camera_state
typedef enum
{
    START_STREAM,
    START_DETECT,
    SHOW_FACES,
    START_RECOGNITION,
    START_ENROLL,
    ENROLL_COMPLETE,
    DELETE_ALL,
} en_fsm_state;
en_fsm_state camera_state;

typedef struct
{
    char enroll_name[ENROLL_NAME_LEN];
} httpd_resp_value;

httpd_resp_value st_name;

//人脸识别初始化
void face_init();

void setup() {
    Serial.begin(115200);               //串口通讯时的数据传输速率为115200
```

```
  Serial.setDebugOutput(true);
  Serial.println();

//定义继电器模块的引脚模式，并将继电器初始化为低电平。
  pinMode(relay_pin, OUTPUT);
  digitalWrite(relay_pin, LOW);

//摄像头初始化配置,创建camera_config t类对象config
  camera_config_t config;
  //时钟配置
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  //Pin IO口引脚
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  //FPS帧率配置，对于ov2640，可以为20KHz或者10KHz
  config.xclk_freq_hz = 20000000;
  //数据像素格式设置为JPEG
  config.pixel_format = PIXFORMAT_JPEG;
  //
  if (psramFound()) {
    config.frame_size = FRAMESIZE_UXGA; //图像帧大小
    config.jpeg_quality = 10; //JPEG压缩
    config.fb_count = 2; //缓存帧数量
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
```

```
    // 用config对象初始化摄像头驱动，只调用一次
      esp_err_t err = esp_camera_init(&config);
      if (err != ESP_OK) {
        Serial.printf("摄像头初始化错误，错误代码：0x%x", err);
        return;
      }
    //获取摄像头传感器，对传感器做一些配置
      sensor_t * s = esp_camera_sensor_get();
      s->set_framesize(s, FRAMESIZE_QVGA);

    //连接wifi路由
      WiFi.begin(ssid, password);   //连接网络
      while (WiFi.status() != WL_CONNECTED) //等待网络连接成功
      {
        delay(500);
        Serial.print(".");
      }
      Serial.println("");
      Serial.println("WiFi连接成功"); //网络连接成功

    //启动web服务器
      server_init();
      face_init();
      socket_server.listen(82);//监听82端口
      Serial.print("摄像头初始化完成，请使用 'http://");
      Serial.print(WiFi.localIP());
      Serial.println("' 访问网站");
    }

    static esp_err_t index_handler(httpd_req_t *req) {
      httpd_resp_set_type(req, "text/html");
      httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
      return httpd_resp_send(req, (const char *)index_ov2640_html_gz,
index_ov2640_html_gz_len);
    }

    httpd_uri_t index_uri = {
      .uri       = "/",
      .method    = HTTP_GET,
      .handler   = index_handler,
      .user_ctx  = NULL
    };
```

```
//创建web服务器函数server_init
void server_init ()
{
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  if (httpd_start(&camera_httpd, &config) == ESP_OK) //创建 HTTP 服务器的实例camera_httpd
    Serial.println("网页开始服务");
  {
    httpd_register_uri_handler(camera_httpd, &index_uri); //注册 URI 处理程序
  }
}

//读取人脸数据函数face_init
void face_init()
{
  face_id_name_init(&st_face_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);
  aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);//为已登记的人脸分配内存
  read_face_id_from_flash_with_name(&st_face_list);
}

//登记人脸信息的函数do_enrollment
static inline int do_enrollment(face_id_name_list *face_list, dl_matrix3d_t *new_id)
{
  ESP_LOGD(TAG, "START ENROLLING");
  int left_sample_face = enroll_face_id_to_flash_with_name(face_list, new_id, st_name.enroll_name);
  ESP_LOGD(TAG, "面部ID：%s  样本编号：%d",
           st_name.enroll_name,
           ENROLL_CONFIRM_TIMES - left_sample_face);
  return left_sample_face;
}

void dealing_mes(WebsocketsClient &client, WebsocketsMessage msg)
{
  if (msg.data() == "stream") {
    camera_state = START_STREAM;
    client.send("开始视频检测");
  }
  if (msg.data() == "detect") {
    camera_state = START_DETECT;
```

```
        client.send("正在检测人脸");
    }
    if (msg.data().substring(0, 8) == "capture:") {
        camera_state = START_ENROLL;
        char person[FACE_ID_SAVE_NUMBER * ENROLL_NAME_LEN] = {0,};
        msg.data().substring(8).toCharArray(person, sizeof(person));
        memcpy(st_name.enroll_name, person, strlen(person) + 1);
        client.send("正在录入人脸");
    }
    if (msg.data() == "recognise") {
        camera_state = START_RECOGNITION;
        client.send("正在识别人脸");
    }
    if (msg.data().substring(0, 7) == "remove:") {
        char person[ENROLL_NAME_LEN * FACE_ID_SAVE_NUMBER];
        msg.data().substring(7).toCharArray(person, sizeof(person));
        delete_face_id_in_flash_with_name(&st_face_list, person);
        send_face_list(client); // reset faces in the browser
    }
    if (msg.data() == "delete_all") {
        delete_all_faces(client);
    }
}

//发送人脸函数
static esp_err_t send_face_list(WebsocketsClient &client)
{
    client.send("删除所有人脸");
    face_id_node *head = st_face_list.head;
    char add_face[64];
    for (int i = 0; i < st_face_list.count; i++) // loop current faces
    {
        sprintf(add_face, "人脸列表:%s", head->id_name); //字符串复制到add_face
        client.send(add_face); //发送人脸
        head = head->next;
    }
}

//删除人脸函数
static esp_err_t delete_all_faces(WebsocketsClient &client)
{
    delete_face_all_in_flash_with_name(&st_face_list);
    client.send("删除所有人脸");
```

```
}

//开锁函数
void open_door(WebsocketsClient &client) {
  if (digitalRead(relay_pin) == LOW) {
    digitalWrite(relay_pin, HIGH); //关闭（通电）继电器，使门解锁
    Serial.println("门已解锁!");
    client.send("门已解锁！ ");
    door_opened_millis = millis(); //  时间继电器关闭，门打开
  }
}


//开始循环工作
void loop() {
  auto client = socket_server.accept();
  client.onMessage(dealing_mes); //客户端接受数据通过调用dealing_mes()改变camera_state的值
  dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, 320, 240, 3); //为图像矩阵分配内存
  http_img_process_result out_res = {0}; //初始化out_res
  out_res.image = image_matrix->item;

  send_face_list(client);
  client.send("开始视频检测");

  while (client.available()) {
    client.poll();

    if (millis() - interval > door_opened_millis) { //当前时间-上一次开门时间>开锁时间间隔
      digitalWrite(relay_pin, LOW); //锁门

      fb = esp_camera_fb_get();//获取视频流

      if (camera_state == START_DETECT || camera_state == START_ENROLL || camera_state == START_RECOGNITION)
      {
        out_res.net_boxes = NULL;
        out_res.face_id = NULL;

        fmt2rgb888(fb->buf, fb->len, fb->format, out_res.image);//将视频转化为RGB888
```

```
            out_res.net_boxes = face_detect(image_matrix, &mtmn_config);//进行人脸
检测

            if (out_res.net_boxes)//如果检测到人脸
            {
               if (align_face(out_res.net_boxes, image_matrix, aligned_face) ==
ESP_OK)//如果人脸对齐正确
               {
                  out_res.face_id = get_face_id(aligned_face); //提取人脸特征值
                  last_millis = millis(); //将上一次检测时间改为当前时间

                  //如果是检测模式START_DETECT
                  if (camera_state == START_DETECT) {
                     client.send("人脸检测中");
                  }

                  //如果是录入模式START_ENROLL
                  if (camera_state == START_ENROLL)
                  {
                     int left_sample_face = do_enrollment(&st_face_list, out_res.face_id);
                     char enrolling_message[64];
                     sprintf(enrolling_message, "样本编号：%d  对应面部ID %s",
ENROLL_CONFIRM_TIMES - left_sample_face, st_name.enroll_name);
                     client.send("正在登记");

                     //如果人脸已被占用
                     if (left_sample_face == 0)
                     {
                        ESP_LOGI(TAG, "登记的人脸ID: %s",
st_face_list.tail->id_name);
                        camera_state = START_STREAM;
                        char captured_message[64];
                        sprintf(captured_message, "但该面部已被样本编号：%s占用",
st_face_list.tail->id_name);
                        client.send("该面部已被占用");
                        send_face_list(client);
                     }
                  }

                  //如果是识别模式START_RECOGNITION
                  if (camera_state == START_RECOGNITION    &&
(st_face_list.count > 0))
```

```
                {
                    face_id_node *f = recognize_face_with_name(&st_face_list,
out_res.face_id);//人脸特征进行对比

                    //如果识别成功
                    if (f)
                    {
                        char recognised_message[64];
                        sprintf(recognised_message, "门被 %s 打开了", f->id_name);
                        open_door(client); //打开门
                        client.send(recognised_message);
                    }
                    else
                    {
                        client.send("该脸未被认出");
                    }
                }
                dl_matrix3d_free(out_res.face_id); //在检测到的人脸特征中删除掉已
经识别过的人脸特征
            }

        }
        //检测人脸失败
        else
        {
            if (camera_state != START_DETECT) {
                client.send("没有检测到人脸");
            }
        }

        //正在检测但未检测到人脸
        if (camera_state == START_DETECT && millis() - last_millis > 500) //已经
检测了500ms以上
        {
            client.send("正在检测");
        }

    }

    client.sendBinary((const char *)fb->buf, fb->len); //传输视频流的buf(Pointer to
the pixel data)和len(Length of the buffer in bytes)到客户端

    esp_camera_fb_return(fb); //获取摄像头帧数据，复用帧数组
```

```
      fb = NULL; //fb置空，准备下一次循环
    }
  }
}
```

```
/*
以下为实现stm32对电路各项辅助功能的代码
*/
#include "delay.h"
#include "sys.h"
#include "usart.h"
#include "key.h"
#include "led.h"
#include "lcd1602.h"

u8 door_signal = 0;        //门检测信号
u8 people_signal = 0;    //人检测信号
u8 light_signal = 0;      //光照检测信号
u8 buzzer_signal=0;       //门铃检测信号
u8 trun_off=0;                  //自动关闭信号

 int main(void)
 {
    u32 time=0;
    delay_init();     //延时函数初始化

    KEY_Init();           //初始化按键
    LED_Init();           //初始化LED灯
    delay_ms(1);         //延时

    Lcd_GPIO_init();//初始化lcd引脚
    Lcd_Init();             //初始化lcd屏幕
    delay_ms(20);        //延时

     Lcd_Puts(0,0,"Waitting for");//初始化显示
    Lcd_Puts(0,1,"Turn off");

    buzzer=1;
```

```
    ligh=1;
    led_hot = 0;
    //打开各项功能，进行检测
    delay_ms(200); //延迟2s
    led_hot = 1;
    buzzer=0;
    ligh=0;
    //关闭各项功能，准备工作

    while(1)
    {
        delay_ms(10);//延时
        if (key_menling == 0)     //检测
        {
            buzzer_signal = 1;
        }
        else
        {
            buzzer_signal = 0;
        }  //否则复位标志

        if (key_redhot == 1)     //若红外传感器显示高电平
        {
            people_signal = 1;led_hot = 0;//标志检测到人体
        }
        else
        {
            people_signal = 0;led_hot = 1;
        }  //否则复位信号

        if (key_door == 1)     //检测门状态
        {
            door_signal = 1;//标志门打开
        }
        else
        {
            door_signal = 0;
        }  //否则复位标志

        if (key_ligh == 0)
//检测光度状态，若为黑暗（低电平信号）
        {
            light_signal = 1;//标志感光信号
```

```
        }
        else
        {
            light_signal = 0;
        }  //否则复位标志
        if(door_signal==1)
        {
           Lcd_Puts(0,1,"Turn on");
//打开门铃拨动开关，显示家中有人
            if(trun_off==0)
            {
                if(buzzer_signal==0) //如果按门铃
                {
                    buzzer=1;              //打开蜂鸣器，持续蜂鸣
                }
                else
                {buzzer=0;}              //关闭蜂鸣器，不发出声音
            }
        }
        else
        {
            Lcd_Puts(0,1,"Turn off              ");
//关闭门铃拨动开关，显示家中无人
            if(trun_off==0)
            {   buzzer=0;}
//关闭蜂鸣器，不发出声音
        }
        if (people_signal == 1)
//红外传感器信号显示有人时
        {
            Lcd_Puts(0, 0, "Welcome!");//lcd显示屏显示有人
            if (light_signal == 1) //若为黑暗状态
            {
                ligh = 1;//则打开灯
            }
            else
            {
                ligh = 0;
            }//否则关闭
            time++;
            if (time >= 3000)  //超过30s
            {
                trun_off = 1;    //自动关闭系统
```

```
                }
            }
            else
            {
                Lcd_Puts(0, 0, "Waitting for ");
//未检测到人时，显示等待
                ligh = 0;
                time = 0;
        }
}
}
```