总 结 报 告

项目名称： 基于YOLOv2和传感器的多功能门禁系统

目录

[1.通信系统制作方案概述 3](#_Toc136690391)

[1.1系统设计的立意 3](#_Toc136690392)

[1.2系统的主要组成 4](#_Toc136690393)

[1.3系统的制作方案 4](#_Toc136690394)

[1.3.1制作方案框图 4](#_Toc136690395)

[1.3.2制作方案原理描述 4](#_Toc136690396)

[1.4系统方案的可行性论证 5](#_Toc136690397)

[2.具体模块分布图及集成原理图 6](#_Toc136690398)

[3.各个模块硬件调试中遇见难题及解决方案 7](#_Toc136690399)

[3.1蓝牙 7](#_Toc136690400)

[3.2蜂鸣器 8](#_Toc136690401)

[3.3红外感知模块 8](#_Toc136690402)

[3.4矩阵键盘 8](#_Toc136690403)

[3.5 RFID 8](#_Toc136690404)

[3.6人脸识别模块 9](#_Toc136690405)

[3.7舵机 9](#_Toc136690406)

[4.开发平台介绍及代码思路简介 10](#_Toc136690407)

[4.1 yolo v2算法介绍 10](#_Toc136690408)

[4.2 开发平台 12](#_Toc136690409)

[4.3代码（精简版）逐行解析 13](#_Toc136690410)

[5.各模块实操结果展示和实现指标 34](#_Toc136690411)

[6.可改进之处及拓展方向 37](#_Toc136690412)

[6.1 不足及改进之处 37](#_Toc136690413)

[6.2 系统方案可拓展方向 37](#_Toc136690414)

[7.课程总结与收获 38](#_Toc136690415)

[8.参考资料 39](#_Toc136690416)

[9.附录 39](#_Toc136690417)

[9.1 附录A 源代码 39](#_Toc136690418)

[9.2 代码流程图 72](#_Toc136690419)

1.系统制作方案概述

1.1系统设计的立意

《基于深度学习图像识别与传感器技术实现的人脸识别门禁系统》的通信系统综合设计，是一项集成了深度学习、图像识别和传感器技术的创新性应用。在这个项目中，通信系统的设计起到了至关重要的作用，它负责将传感器采集的数据传输到中央控制系统，使得门禁系统可以对人脸进行识别并且进行授权管理。因此，这个项目的立意是通过综合应用现代技术，提高门禁系统的安全性和便利性，为社会带来更加智能化的生活体验。

在该系统的实现过程中，需要涉及到多个课程中的知识和技术，例如通信原理、信号处理、计算机网络、数据库、人工智能等。需要对这些知识和技术进行综合应用和创新性的设计，以实现该系统的智能化和高效性。

对于通信系统的设计，这个项目给我们带来了一些启发。首先，这个项目的通信系统需要满足高速传输和低时延的要求，这需要我们在设计通信协议时要考虑到数据传输效率的同时，还要尽可能地减少延迟时间。其次，通信系统需要具备高度的可靠性和稳定性，因为一旦通信出现问题就会影响门禁系统的正常运行。因此，在通信系统的设计过程中，需要采用多种技术手段确保数据传输的可靠性和稳定性，例如冗余传输、错误检测和纠正等。

另外，这个项目还启示我们在通信系统的设计过程中要考虑到系统的可扩展性和兼容性。考虑到未来门禁系统的升级和扩展，我们需要设计灵活且可扩展的通信系统，以便于将来的升级和扩展。同时，通信系统还需要与其他系统兼容，以实现更加高效和智能的数据交互。

《基于深度学习图像识别与传感器技术实现的人脸识别门禁系统》的通信系统综合设计，是一项集成了深度学习、图像识别和传感器技术的创新性应用。在这个项目中，通信系统的设计起到了至关重要的作用，它负责将传感器采集的数据传输到中央控制系统，使得门禁系统可以对人脸进行识别并且进行授权管理。因此，这个项目的立意是通过综合应用现代技术，提高门禁系统的安全性和便利性，为社会带来更加智能化的生活体验。

1.2系统的主要组成

本次通信系统综合设计的基于深度学习图像识别与传感器技术实现的人脸识别门禁，主要由成品模块组成。具体包含：K210 Maix Bit、配套24PIN DVP摄像头及LCD屏、SG90舵机、HC-SR501人体红外感应模块、MFRC-522射频模块、HC-05蓝牙模块、有源蜂鸣器、32G SD内存卡及读卡器、4位独立按键。

1.3系统的制作方案

1.3.1制作方案框图

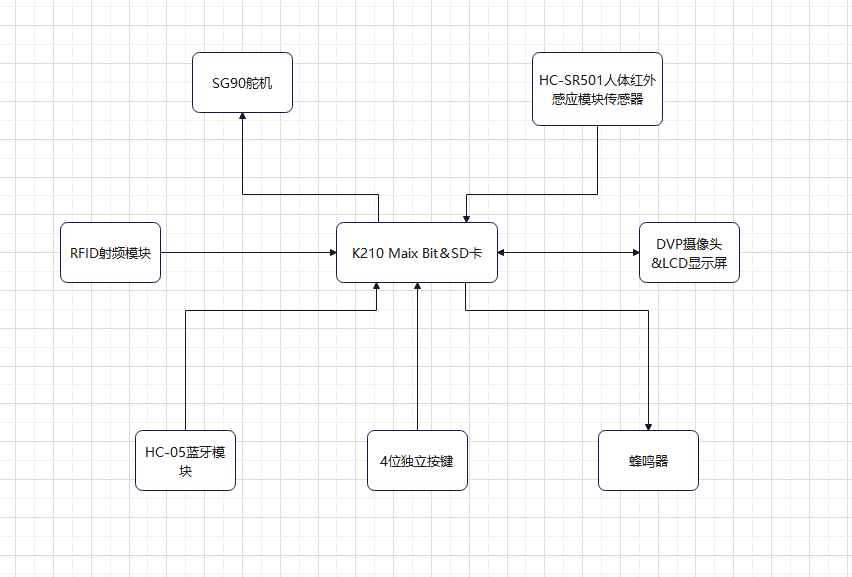


图1-1门禁系统最终实现方案图

1.3.2制作方案原理描述

初步设立门禁系统三大模式，等待、门禁及录入模式，等待模式无人时显示屏全黑只显示waiting......；门禁模式打开摄像头锁定人脸显示比对精度（百分制）与识别框，此时RFID数据通道打开，并在左上角标注所处模式；录入模式打开摄像头锁定人脸显示采样进度与识别框，此时蓝牙通道打开，并在左上角标注所处模式，默认等待模式。使用K210 MAIXBIT做主控，当红外感知模块检测到人时，屏幕亮起，可识别人脸但无反应（红外感知模块可调节灵敏度与延时，但过于灵敏会导致画面卡顿，反之则检测效果不理想）。使用独立按键切换到录入或门禁模式，30s未检测到人脸自动进入等待黑屏模式。录入模式时，使用移动端app调试全能王连接蓝牙模块并发送指令，输入“register\*\*\*\*”（\*\*\*\*代表录入人脸编号）后按照yolo v2算法采集196维人脸数据（3轮18次采样），并储存在SD卡文件中，且屏幕显示采样进度，录入完成屏幕显示“successful”，蜂鸣器发低声。输入“delete\*\*\*\*”则删除对应用户储存在SD卡的特征值，蜂鸣器发高声。输入“erase”则删除SD卡内所有用户人脸特征值，蜂鸣器发高声。输入“open”则无条件控制舵机旋转。门禁模式时，设定比对阈值，高于则蜂鸣器发低声且舵机旋转，低于则蜂鸣器发高声舵机无反应。此外RFID数据通道开通，我们已给两张S50标准卡与异形卡相同扇区注入不同信息，分别会有舵机与蜂鸣器的不同响应情形，当其他RFID卡靠近则默认发生系统错误并发出独有蜂鸣器声音（可以修改）。

1.4系统方案的可行性论证

1. 技术可行性分析:

深度学习算法和神经网络理论已有较成熟应用,图像识别、人脸识别等领域有较多成功案例,技术上实现人脸识别门禁系统是可行的。

传感器技术也比较成熟,可以采集人脸图像和相关生物特征,作为识别的输入。

系统集成方面,人脸识别算法、传感器技术和门禁系统的结合也是现有技术可以实现的。

2. 经济可行性分析:

相关硬件成本在可接受范围内,如高清摄像头等。软件实现可以采用开源算法和框架,开发成本不高。

人脸识别门禁系统可以降低人工成本,尤其适用于人流较大场景,具有较好的经济效益。

3. 操作可行性分析:

人脸识别门禁系统操作简单,易于推广使用,无需复杂训练,符合大多数人的接受习惯。

该系统也便于维护和管理,可靠性较高。

综上,从技术、经济和操作等多方面考虑,基于深度学习的人脸识别门禁系统是一种可行的智能门禁方案。但是,数据安全和隐私保护也需要重点考虑,总体而言该方案是值得研究和探索的。

2.具体模块分布图及集成原理图

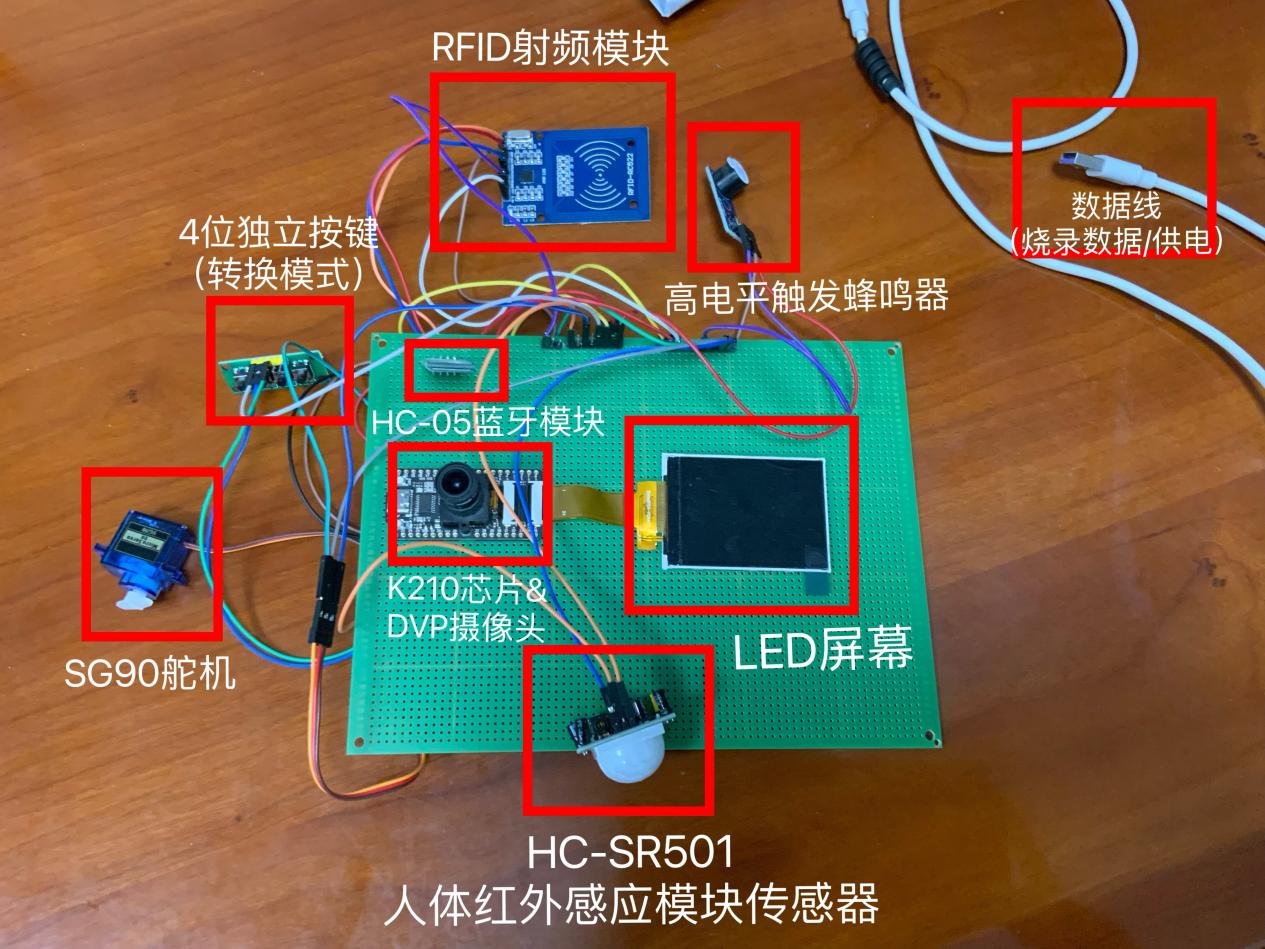


图2-1 人脸识别系统实操图

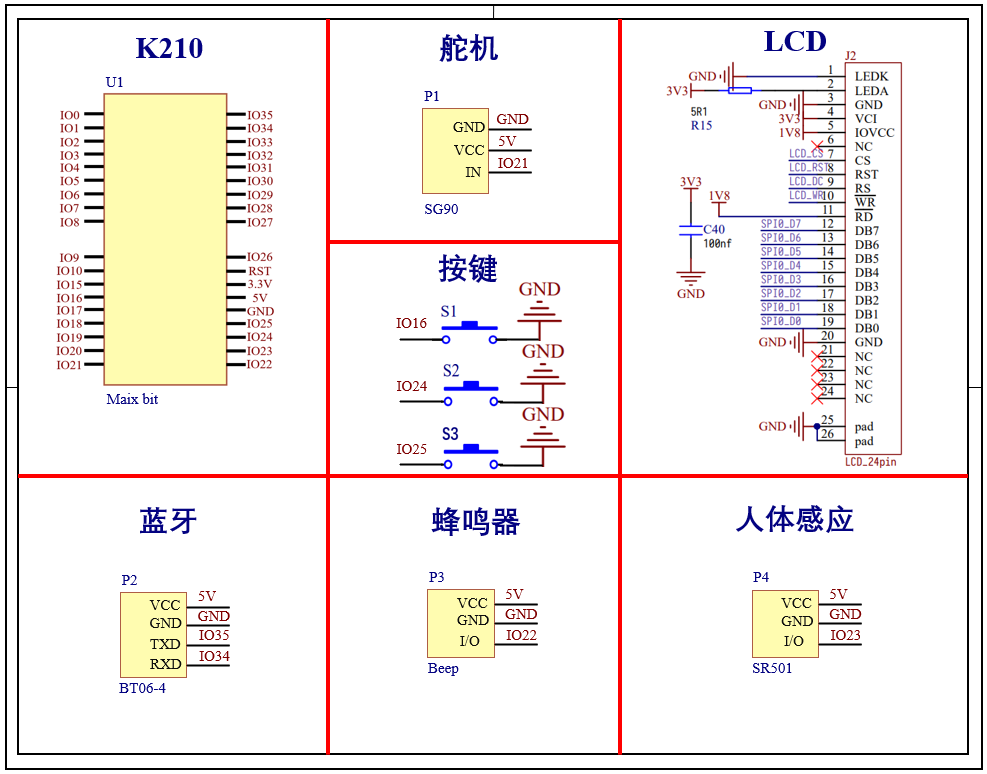


图2-2 通信系统原理图

3.各个模块硬件调试中遇见难题及解决方案

3.1蓝牙

HC-05蓝牙模块是一种基于蓝牙协议的简单无线通信设备。该模块基于BC417单芯片蓝牙IC，符合蓝牙v2.0标准，支持UART和USB接口。

具有两种工作模式：命令响应工作模式和自动连接工作模式。

当模块处于命令响应工作模式（或者AT模式）时能才能执行 AT 命令，用户可向模块发送各种 AT指令，为模块设定控制参数或发布控制命令。（AT指令就是我们PC与一些终端设备（例如蓝牙，WiFi模块）之间进行通信的，配置这些终端设备参数的一套指令。）在自动连接工作模式下模块又可分为主（Master）、从（Slave）和回环（Loopback）三种工作角色。当模块处于自动连接工作模式时，将自动根据事先设定的方式连接的数据传输。主模式：该模块可以主动搜索并连接其它蓝牙模块并接收发送数据。从模式：只能被搜索被其它蓝牙模块连接进行接收发送数据。回环：蓝牙模块就是将接收的数据原样返回给远程的主设备。

初次使用模块由于不知道蓝牙模块密码，故需要再使用TTL转串口模块，使用AT指令查询和修改蓝牙名称与密码。

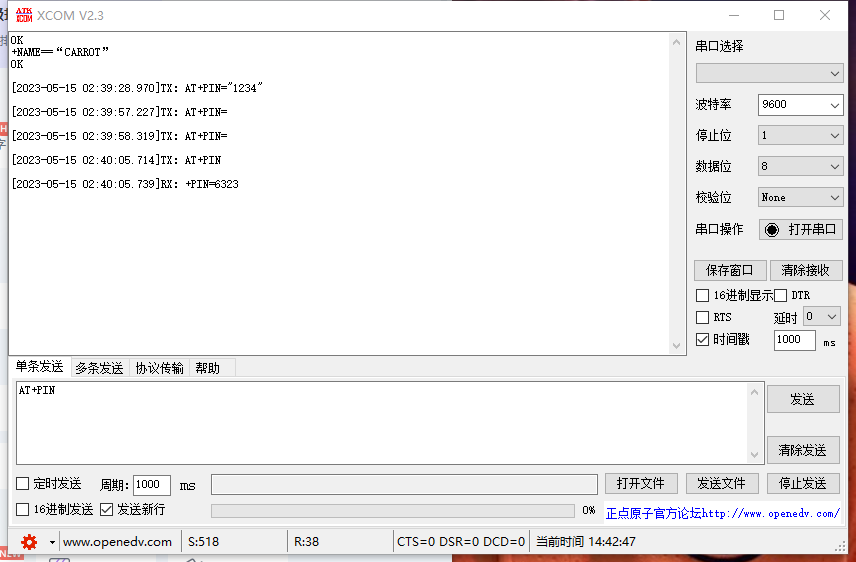


图3-1正点原子XCOM串口调试助手图

3.2蜂鸣器

蜂鸣器主要用于功能实现的提示音，一开始使用了低电平的蜂鸣器，导致一直响以为是代码不当导致的功能卡顿，多次检查后发现和代码无关，后来选取了高电平触发的蜂鸣器得以成功得到提示音。

3.3红外感知模块

红外模块用来感知是否有人在使用，从而开启或者关闭识别元件，红外检测器件的使用难点在于灵敏性调整，需要仔细修改，否则就会导致过快或者不进入休眠模式，从而影响使用。除此之外，不准确的红外检测还会导致器件反应过慢等等的问题，所以在红外器件的修改上下了很多时间才得到比较准确的识别数据。

3.4矩阵键盘

用于模式转换，录入人脸模式和正常使用模式，安装需要注意连接线。

3.5 RFID

RFID模块：标签进入磁场后，接收解读器发出的射频信号，凭借感应电流所获得的能量发送出存储在芯片中的产品信息（Passive Tag，无源标签或被动标签），或者由标签主动发送某一频率的信号（Active Tag，有源标签或主动标签），解读器读取信息并解码后，再进行有关数据处理。在RFID模块的设计中，为了方便地将标签的信息进行区分，首先设计了不同权限卡之间的对应规则，分别设计了存储信息中Y开头的万能通用卡，N开头的带有序号的普通用户权限卡，以及未规定的其他RFID卡，然后按照设定写了读卡器的写入程序，将对应的规则写入测试卡片，并且对卡片进行了测试。具体要将RFID验证模块加入到哪个环节之中，我们也对应地做了测试，所有流程都会回到普通模式，因此最开始是将RFID是加入在普通模式下，但是经过实际地测试与检验后，发现普通模式并不适合加入RFID模块，应该在检测到有人存在时再启动RFID。 其一，当门禁系统不被使用时，启动RFID模块会浪费大量的能源。通过检测到有人存在再启动RFID模块，可以最大程度地节省能源，降低系统的运行成本。其二，在门禁系统中，启动RFID模块是为了实现对进出人员的身份识别和控制。通过检测到有人存在后再启动RFID模块，可以确保门禁系统只对授权人员开放，提高门禁系统的安全性。最后综合考虑，将RFID模块放置在门禁模式之下。

3.6人脸识别模块

由k210芯片，摄像头和LED显示屏组成，用于识别用户脸部数据决定是否开启舵机。原理是YOLO 人脸识别，是一种基于深度学习的目标检测算法，它可以在一张图像中同时检测出多个目标，并且实时性能非常好。YOLO 模型的识别原理是通过将图像分成名个网格，然后对每个网格进行预测，最终将所有网格的预测结果合并起来得到最终的检迎结果。YOLO 模型的输入是一张图像，输出是每个目标的类别、位置和置信度。在训练阶段，YOLO 模型会学习如何将图像分成多个网格，并且对每个网格进行预测。每个网格的大小可以根据图像的大小 或者目标的大小进行调整。对于每个网格，YOLO 模型会预测出多个边界框，每个边界框包含了一个日标的位置和大小信息。同时，YOLO模型还会预测出每个边界框对应的目标的类别和置信度。除了写入训练的数学模型外，还需要设置识别的准确数据分数，超过标准才可以运行下一步功能。还需协调模型的数据存储和循环量，以免占用过多的数据空间，造成大量数据冗余，导致死机。

3.7舵机

SG 90舵机需要使用正确的通信协议与门禁系统的其他部件进行通信。在门禁系统中使用SG 90舵机时，应该选择适当的通信协议，例如PWM或者串口通信，并确保通信协议的准确性和稳定性。

4.开发平台介绍及代码思路简介

4.1 yolo v2算法介绍

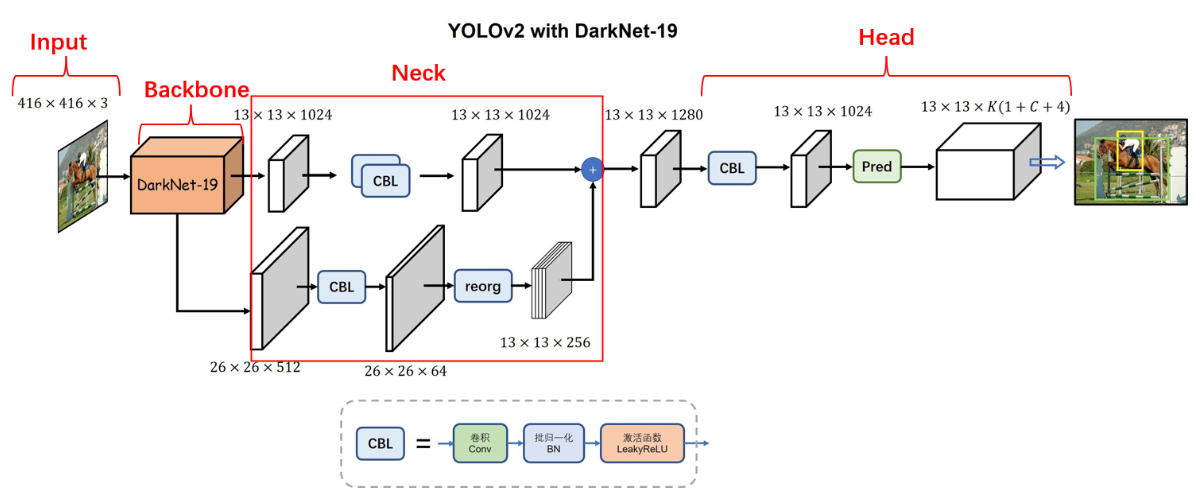


图4-1 YOLOv2原理图

1.输入图片被分成S x S个网格(cell)，每个网格负责检测该网格内是否存在目标物体。

2.对于每个网格，预测出B个边界框（bounding box），每个边界框包括(x, y, w, h, confidence)五个元素，其中(x, y)是边界框中心的坐标，w和h是边界框的宽和高，confidence表示该边界框中包含目标物体的置信度。

3.对于每个边界框，计算它与ground truth（真实标注的物体位置）的IoU（交并比），并选取IoU最大的边界框作为该物体的预测框。

4.使用非极大值抑制（NMS）方法来剔除重叠的预测框，最终输出预测结果。

**关于IoU：**

IoU是交并比（Intersection over Union）的缩写，是一种用于衡量目标检测算法检测准确性的指标。它是通过计算预测框和ground truth（真实标注的物体位置）之间的重叠程度来得出的。具体来说，IoU是预测框和ground truth的交集面积除以它们的并集面积，即：

IoU = Intersection over Union = Intersection / Union

其中，Intersection表示预测框和ground truth的交集面积，Union表示它们的并集面积。IoU的取值范围在0到1之间，取值越大表示检测准确性越高。一般来说，当IoU大于某个阈值时，我们就认为预测框和ground truth匹配成功，可以将其作为检测结果；反之则认为匹配失败。

在目标检测算法中，通常会使用IoU作为评价指标来衡量算法的准确性。比如，在YOLOv2算法中，会使用IoU来选取每个边界框中与ground truth匹配的框。在训练过程中，如果预测框与ground truth之间的IoU大于某个阈值，我们就认为这个预测框是正确的，并计算它的损失函数；反之则认为它是错误的，不参与损失函数的计算。通过不断地调整模型参数，最终可以得到一个准确性较高的目标检测算法。

**关于NMS：**

非极大值抑制（Non-Maximum Suppression，NMS）是一种用于目标检测算法中的后处理方法，主要用于剔除重叠的预测框，保留最准确的预测结果。在目标检测算法中，一些目标可能会被多个预测框所检测到，这些预测框之间可能存在重叠。在这种情况下，我们需要对这些预测框进行筛选，保留最准确的预测结果。NMS方法就是用来完成这个任务的。NMS的基本原理如下：

1.对所有预测框按照其置信度（confidence）进行排序，从置信度最高的预测框开始遍历。

2.对于当前遍历到的预测框，计算它与之前已经选中的预测框的IoU（交并比），如果IoU大于某个阈值（如0.5），则将该预测框剔除，否则保留该预测框。

3.重复步骤2和3，直到遍历完所有的预测框。

通过这个过程，NMS方法可以将多个重叠的预测框剔除，保留最准确的预测结果。NMS方法可以有效地解决目标检测算法中的多框问题，提高检测准确性。需要注意的是，NMS方法的效果受到阈值的影响。如果阈值较高，那么将会保留更少的预测框，可能会漏检一些目标；如果阈值较低，那么将会保留更多的预测框，可能会增加误检的概率。

**同时，为了提高准确性和速度，YOLOv2采用了一些技巧：**

1.Darknet-19网络：YOLOv2使用了一个19层的卷积神经网络Darknet-19来提取特征。该网络比VGG16等经典网络更轻量，同时具有更好的准确性。

2.Anchor Boxes：YOLOv2在每个网格上预测B个边界框，而不是像YOLOv1一样只预测一个。此外，YOLOv2使用了Anchor Boxes，即预定义的边界框，来提高检测准确性。在训练过程中，YOLOv2通过调整Anchor Boxes和预测框的位置和大小来进行目标检测。

3.Batch Normalization：在Darknet-19网络中使用Batch Normalization来加速收敛和提高准确性。

4.High Resolution Classifier：YOLOv2使用了一个高分辨率的分类器来提高检测准确性。具体来说，在训练时，YOLOv2将输入图片的分辨率提高到608x608，而在测试时，将其缩小到416x416，以加快处理速度。

5.Convolutional With Anchor Boxes (CWA)：YOLOv2引入了一种新的卷积层，称为Convolutional With Anchor Boxes (CWA)。该层同时计算多个Anchor Boxes的位置和置信度，以提高检测准确性和速度。

总的来说，YOLOv2是一个快速且准确的目标检测算法，它的核心思想是将目标检测问题转化为一个回归问题，并使用神经网络来解决。通过使用Anchor Boxes、Batch Normalization、High Resolution Classifier和Convolutional With Anchor Boxes等技巧，YOLOv2在准确性和速度方面都有所提高。

4.2 开发平台

**开发环境：MaixPy IDE**

MaixPy IDE是一款基于Python语言的集成开发环境（IDE），主要用于开发和调试MaixPy（开源的嵌入式人工智能框架）项目。MaixPy IDE提供了一系列功能，包括代码编辑、调试、编译、下载、串口调试、固件升级等，方便用户对MaixPy进行开发和调试。

**烧录环境：kflash\_gui**

kflash\_gui是一款开源的图形化固件烧录工具，用于将固件烧录到Kendryte K210芯片上。K210是一款基于RISC-V架构的高性能、低功耗的嵌入式处理器，广泛应用于物联网、人工智能等领域。kflash\_gui是Kendryte K210的官方固件烧录工具，提供了一系列功能，包括固件下载、擦除、烧录、调试等。kflash\_gui基于Python语言和PyQt库开发，支持命令行和GUI两种模式。用户可以通过命令行模式进行批量烧录等高级操作，也可以通过GUI模式进行简单易用的固件烧录。kflash\_gui还提供了一些高级功能，如自动检测串口、自动切换烧录模式、支持多种烧录设备等，方便用户进行高效的固件烧录。



**串口调试环境：XCOM**

1. 支持多个常用波特率，支持自定义波特率
2. 支持5/6/7/8位数据，支持1/1.5/2个停止位
3. 支持奇/偶/无校验
4. 支持16禁止发送/接收显示，支持DTR/RTS控制
5. 支持窗口保存，并可以设置编码格式
6. 支持延时设置，支持时间戳功能
7. 支持定时发送，支持文件发送，支持发送新行
8. 支持多条发送，并关联数字键盘，支持循环发送
9. 支持无限制扩展条数，可自行增删
10. 支持发送条目导出/导入（excel格式）
11. 支持协议传输（类modbus）
12. 支持发送/接收区字体大小、颜色和背景色设置
13. 支持简体中文、繁体中文、英文三种语言
14. 支持原子软件仓库

4.3代码（精简版）逐行解析

图4-3 代码流程图

从左到右依次将代码模块化为三个模块，分别是**人脸数据采集模块，人脸信息录入模块，人脸蓝牙RFID综合验证模块**。下面依次对三个模块的思路进行解析：

首先是**人脸数据采集模块**，这段代码是一个人脸识别的应用程序，主要分为以下几个步骤：首先设置各种标志位和参数，包括是否检测到人、是否录入、查找模式、人脸对比置信度等。

然后执行以下操作：

a. 调用check\_key()函数，可能用于检测设备是否已经授权使用相关的模型。

b. 调用sensor.snapshot()函数获取一张图像。

c. 使用kpu.run\_yolo2()函数对输入的图像进行人脸检测，返回一个包含检测结果的列表code。

d. 如果检测到人脸，程序会遍历code中的每个检测结果，计算人脸框的面积，并选择面积最大的人脸进行处理。

e. 使用img.draw\_rectangle()函数在原图像上绘制人脸框，然后使用img.cut()函数对原图像进行裁剪，得到人脸图像。

f. 对人脸图像进行特征点检测，得到人脸的五个关键点坐标，使用img.draw\_circle()函数在原图像上绘制出这些关键点的位置。

g. 使用这些关键点的坐标计算人脸的仿射变换矩阵，并使用image.get\_affine\_transform()函数计算变换矩阵，将人脸图像进行对齐，得到标准的人脸图像。

h. 使用kpu.forward()函数对对齐后的人脸图像进行特征提取，再使用kpu.face\_encode()函数将特征向量进行编码。

i. 遍历预先录入的人脸特征列表，使用kpu.face\_compare()函数计算当前人脸特征向量与列表中每个特征向量的相似度得分，并选择得分最高的特征向量，返回其在列表中的索引，用于识别当前人脸是否为已知人脸。

j. 根据人脸识别结果，可以执行不同的操作，如录入新人脸特征、查找已知人脸特征，或者进行相应的提示和处理。

代码详细注释如下：

1.# 设置各种flag

2.no\_people = 0 # 是否无人

3.no\_flag = 0

4.delline = [] # 将要删除的行数

5.check\_num = 0

6.shibie\_num = 0

7.uart.read()

8.

9.# 设置人脸对比置信度

10.ACCURACY = 75

11.# 录入模式flag

12.luru\_flag = 0

13.# 查找模式flag

14.find\_flag = 0

15.

16.feature = ''

17.max\_score = 0

18.index = 0

19.while (1):

20.    # 程序首先调用了check\_key()函数，该函数可能用于检测设备是否已经授权使用相关的模型。

21.    check\_key()

22.    # 调用了sensor.snapshot()函数获取一张图像。

23.    img = sensor.snapshot()

24.    # 使用kpu.run\_yolo2()函数对输入的图像进行人脸检测，返回一个包含检测结果的列表code。

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

26.    max\_score = 0

27.    # 如果code不为空，即检测到人脸，程序会遍历code中的每个检测结果，计算人脸框的面积，并选择面积最大的人脸进行处理。

28.    if code:

29.        t = 0

30.        max\_face = 0

31.        totalRes = len(code)

32.        area = []

33.        # 如果检测到的人脸数大于1，则只处理面积最大的人脸。

34.        if totalRes > 1:  #多张人脸时，挑选面积最大的，一般即最前面的

35.            for i in code: # 迭代坐标框  多张人脸

36.                area.append(i.w()\*i.h())

37.            for j in range(len(area)):

38.                if max\_face < area[j]:

39.                    max\_face = area[j]

40.                    t = j    #保存最大脸面积下标

41.            totalRes = 1

42.            del area

43.        if totalRes == 1:  #只有一张脸

44.            i = code[t]

45.            # 首先使用img.draw\_rectangle()函数在原图像上绘制人脸框

46.            # Cut face and resize to 128x128

47.            a = img.draw\_rectangle(i.rect())

48.            # 然后使用img.cut()函数对原图像进行裁剪，得到人脸图像。

49.            face\_cut = img.cut(i.x(), i.y(), i.w(), i.h())

50.            # 使用img.resize()函数将人脸图像缩放为128x128大小

51.            face\_cut\_128 = face\_cut.resize(128, 128)

52.            # 使用img.pix\_to\_ai()函数将图像转换为KPU能够处理的格式。

53.            a = face\_cut\_128.pix\_to\_ai()

54.            # a = img.draw\_image(face\_cut\_128, (0,0))

55.            # Landmark for face 5 points

56.            # 程序使用kpu.forward()函数对人脸图像进行特征点检测，得到人脸的五个关键点坐标。

57.            fmap = kpu.forward(task\_ld, face\_cut\_128)

58.            plist = fmap[:]

59.            le = (i.x() + int(plist[0] \* i.w() - 10), i.y() + int(plist[1] \* i.h()))

60.            re = (i.x() + int(plist[2] \* i.w()), i.y() + int(plist[3] \* i.h()))

61.            nose = (i.x() + int(plist[4] \* i.w()), i.y() + int(plist[5] \* i.h()))

62.            lm = (i.x() + int(plist[6] \* i.w()), i.y() + int(plist[7] \* i.h()))

63.            rm = (i.x() + int(plist[8] \* i.w()), i.y() + int(plist[9] \* i.h()))

64.            # 使用img.draw\_circle()函数在原图像上绘制出这些关键点的位置。

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

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

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

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

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

70.            # align face to standard position

71.            src\_point = [le, re, nose, lm, rm]

72.            # 程序使用这些关键点的坐标计算人脸的仿射变换矩阵，并使用image.get\_affine\_transform()函数计算变换矩阵。

73.            T = image.get\_affine\_transform(src\_point, dst\_point)

74.            # 使用image.warp\_affine\_ai()函数将人脸图像进行对齐，得到标准的人脸图像。

75.            a = image.warp\_affine\_ai(img, img\_face, T)

76.            # 然后，程序使用img.ai\_to\_pix()函数将对齐后的图像转换为原图像能够显示的格式。

77.            a = img\_face.ai\_to\_pix()

78.            # a = img.draw\_image(img\_face, (128,0))

79.            del (face\_cut\_128)

80.            # calculate face feature vector

81.            # 程序使用kpu.forward()函数对对齐后的人脸图像进行特征提取

82.            fmap = kpu.forward(task\_fe, img\_face)

83.            # 使用kpu.face\_encode()函数将特征向量进行编码

84.            feature = kpu.face\_encode(fmap[:])

85.            reg\_flag = False

86.            scores = []

87.            # 程序遍历预先录入的人脸特征列表

88.            for j in range(len(record\_ftrs)):

89.                # 使用kpu.face\_compare()函数计算当前人脸特征向量与列表中每个特征向量的相似度得分。

90.                score = kpu.face\_compare(record\_ftrs[j], feature)

91.                scores.append(score)

92.            max\_score = 0

93.            index = 0

94.            # 最终，程序选择得分最高的特征向量，并返回其在列表中的索引。这个索引可以用于识别当前人脸是否为已知人脸。

95.            for k in range(len(scores)):

96.                if max\_score < scores[k]:

97.                    max\_score = scores[k]

98.                    index = k

其次是**人脸信息录入模块**，主要分为以下几个步骤：

1. 如果检测到当前画面中存在人脸，且识别率超过预设值，程序会在屏幕上显示提示信息“Face Exist”，然后继续循环，等待下一张图像的检测和处理。
2. 如果检测到人脸特征，程序会记录当前采集的人脸数量，每检测到一次人脸就加1，并在屏幕上显示当前采样进度。程序会间隔录入，每采集6次人脸特征就将当前特征加入到临时特征值列表中。
3. 如果已采集到18次人脸特征，表示录入结束，程序会将当前特征添加到已知特征列表中，并将编号添加到names列表中。然后程序会尝试打开SD卡上的faceinfo.txt文件，并以追加模式写入数据。写入完成后，程序会将当前特征添加到已知特征列表中，并将编号添加到names列表中。同时清空临时特征值列表和采集次数，显示录入成功或失败的提示信息，并进行蜂鸣器声音提示。最后退出录入任务并回到正常模式。
4. 如果保存到SD卡失败，则按键次数清零，check\_num清零，编号清空，并显示录入失败的提示信息。
5. 最后，程序会删除code变量，释放内存空间。

代码详细注释如下：

1.# 执行录入任务

2.if luru\_flag == 1:

3.    # 如果检测到当前画面中存在人脸，且识别率超过预设值

4.    if max\_score > ACCURACY:

5.        # 人脸采集次数清零

6.        check\_num = 0

7.        # 在屏幕上显示提示信息“Face Exist”

8.        a = img.draw\_string(200,0, b'Face Exist', color=(255,0,0),scale=1.6,mono\_space=1)  #提示人脸已存在

9.        # 在屏幕上显示图像

10.        a = lcd.display(img)

11.        # 继续循环，等待下一张图像的检测和处理

12.        continue

13.    # 如果检测到人脸特征

14.    if code:

15.        # 记录当前采集的人脸数量，每检测到一次人脸就加1

16.        check\_num = check\_num + 1     #检测到一次人脸则加 1

17.        # 在屏幕上显示当前采样进度

18.        a = img.draw\_string(5,40, b'%d'%check\_num, color=(0,255,0),scale=1.4,mono\_space=1) #显示采样进度

19.        # 间隔录入，每采集6次人脸特征就将当前特征加入到临时特征值列表中

20.        if check\_num % 6 == 0 and check\_num != 0:

21.            record\_ftrtemp.append(feature)   #加入当前人脸特征

22.        # 如果已采集到18次人脸特征，表示录入结束

23.        if check\_num == 18:

24.        # 录入成功标志位设为1

25.            save\_success = 1

26.            try:

27.                # 尝试打开SD卡上的faceinfo.txt文件，并以追加模式写入数据

28.                with open("/sd/faceinfo.txt", "a") as f:

29.                    for i in range(len(record\_ftrtemp)):  #循环遍历临时特征值列表，写入SD卡

30.                        f.write(stu\_num+'#'+str(record\_ftrtemp[i]))

31.                        # 每个特征值占一行

32.                        f.write("\n")

33.                    # 关闭文件

34.                    f.close()

35.            except Exception:

36.                save\_success = 0  #表示保存到SD卡失败了

37.                pass

38.            # 如果保存到SD卡成功，则将当前特征添加到已知特征列表中，并将编号添加到names列表中；同时清空临时特征值列表和采集次数，显示录入成功的提示信息，并进行蜂鸣器声音提示；最后退出录入任务并回到正常模式。

39.            if save\_success == 1:

40.                #将当前特征添加到已知特征列表，编号添加到names

41.                for i in record\_ftrtemp:

42.                    record\_ftrs.append(i)

43.                    names.append(stu\_num)

44.                record\_ftrtemp.clear()  #清空临时特征值列表

45.                # 在屏幕上显示录入成功的提示信息

46.                a = img.draw\_string(0,5, b'Success!', color=(255,0,0),scale=1.4,mono\_space=1)  #录入成功

47.                # 在屏幕上显示图像

48.                a = lcd.display(img)

49.

50.                #蜂鸣器声音提示

51.                beep.enable()

52.                beep.freq(1000)

53.                time.sleep(2)

54.                beep.disable()

55.                luru\_flag = 0  # 退出录入任务

56.                # 回到正常模式

57.                LuRu\_mode = False   # 录入模式

58.                Door\_mode = False   # 门禁模式

59.                Normal\_mode = True  # 正常模式

60.            #如果保存到SD卡失败，则按键次数清零，check\_num清零，编号清空

61.            else:

62.                a = img.draw\_string(0,5, b'Fail!', color=(255,0,0),scale=1.4,mono\_space=1)  #录入失败

63.                #蜂鸣器声音提示

64.                beep.enable()

65.                beep.freq(600)

66.                time.sleep(2)

67.                beep.disable()

68.                luru\_flag = 0  # 退出录入任务

69.                # 回到正常模式

70.                LuRu\_mode = False   # 录入模式

71.                Door\_mode = False   # 门禁模式

72.                Normal\_mode = True  # 正常模式

73.    # 删除code变量，释放内存空间。

74.    del code

最后是**人脸蓝牙RFID综合验证模块**，这个模块相较于前面两个模块，逻辑设计较为复杂，首先判断是否检测到人：程序首先会检测人是否出现在门禁的监控区域，如果有人，则将 no\_flag 置为1。同时进行垃圾回收，程序会定期进行垃圾回收，以释放不再使用的内存空间。再接收蓝牙数据：程序会读取蓝牙模块发送的数据，如果读取到了数据且长度大于等于2，则进行后续的处理。

1. 如果读取到的蓝牙数据中包含 'open'，则程序会将舵机旋转以打开门禁，以实现临时门禁的功能。
2. 如果读取到的蓝牙数据中包含 'erase'，执行删除全部用户的操作。具体来说，代码实现了以下功能：
3. 如果读取到的蓝牙数据中包含 'erase'，则执行内部代码块。
4. 清空名称列表、特征值列表和人脸信息文件。
5. 打开人脸信息文件，并将其内容清空。清空图像缓存。
6. 绘制矩形和字符串，并在 LCD 屏幕上显示图像。
7. 发出蜂鸣器声音提示。
8. 如果读取到的蓝牙数据中包含 'delete' ，执行删除指定用户的操作。具体来说，代码实现了以下功能：
9. 如果读取到的蓝牙数据中包含 'delete'，则执行内部代码块。截取出要删除的编号，并打印输出。
10. 逐行读取人脸信息文件，查找要删除的编号，并记录要删除的行数。
11. 如果找到了要删除的行，则打开人脸信息文件，删除指定行，并重新写入文件。
12. 从名称列表和特征值列表中删除指定的用户。
13. 绘制矩形和字符串，并在 LCD 屏幕上显示图像。发出蜂鸣器声音提示。
14. 将程序回到正常模式。
15. 如果当前处于录入模式，则在 LCD 屏幕上显示“Register Mode”字样。如果读取到的蓝牙数据中包含 'register'，则执行人脸注册操作。
16. 截取出要注册的编号，并打印输出。判断该编号是否已经被录入，如果已经被录入，则在 LCD 屏幕上显示“ID Exist!”字样，并在屏幕上显示红色矩形区域，发出蜂鸣器声音提示，并回到正常模式；否则，设置录入标志位为 1。
17. 如果录入标志位为 1，则进行人脸录入操作。
18. 检测到人脸后，提取人脸特征，并将其添加到特征值列表 record\_ftrs 中。将编号和姓名组合成一个字符串，并添加到名称列表 names 中。发出蜂鸣器声音提示。将录入标志位设置为 0。
19. 将程序回到正常模式。
20. 如果当前处于门禁模式，脚本初始化RFID模块并尝试从中读取数据。如果检测到有效的RFID卡片，代码将读取卡片数据并根据不同卡片内容执行相应的操作。如果卡片包含预期数据，则触发一个绿色LED和一个舵机来打开门锁。如果卡片包含无效数据，则触发一个红色LED和一个蜂鸣器来警告用户。此外，为了更加丰富使用场景，这里还预留了一个万能卡的对应信息，保证管理员可以在特殊情况下进行无限制开门。

除了RFID部分，“门禁模式”还同时运行人脸识别算法。如果检测到人脸，比较当前人脸与存储人脸相似的置信度，如果识别的置信度高于设定的阈值，则触发与有效RFID卡片相同的操作，控制舵机打开门禁并且显示绿灯。如果人脸识别分数低于设定的阈值，显示分数，并计数，如果计数超过3次则表示人脸识别失败，开启红灯并发出蜂鸣器声音

1. 如果当前处于正常模式，设置录入标志和检查数字为 0，使用 LCD 显示屏显示“等待......”,并将标志位 no\_people 设置为 0。如果当前没有人在操作，程序会将 LCD 清空并显示“休眠中......”,并将标志位 no\_people 设置为 1。程序会在适当的时候调用 del img 和 gc.collect() 来回收内存。其中，del img 用于删除 img 对象占用的内存，而 gc.collect() 用于回收未被使用的内存。

代码详细注释如下：

1.# 有人的时候LCD显示摄像头的拍摄画面

2.if people\_find.value() == 1:

3.    no\_flag = 1

4.

5.

6.

7.    # 接收蓝牙数据

8.    # 垃圾回收

9.    gc.collect()

10.    # 读取蓝牙数据

11.    text = uart.read()

12.    if text != None and len(text) >= 2: #如果读取到了数据，且大于等于2

13.        # 临时开门禁

14.        # 如果蓝牙数据中包含 'open'

15.        if 'open' in text:

16.            print("--------蓝牙开门--------")

17.            # 舵机旋转

18.            Servo(SS,90)

19.            time.sleep(2)

20.            Servo(SS,0)

21.

22.        # 删除全部用户命令

23.        # 如果蓝牙数据中包含 'erase'

24.        if 'erase' in text:

25.            print("--------删除全部用户--------")

26.            # 清空名称列表、特征值列表和人脸信息文件

27.            names.clear()

28.            record\_ftrs.clear()

29.            # 打开文件

30.            file\_new = open('/sd/faceinfo.txt', 'w')

31.            # 写入空字符串

32.            file\_new.write(''.join(''))

33.            # 关闭文件

34.            file\_new.close()

35.            # 清除图像

36.            img.clear()

37.            # 绘制矩形

38.            img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

39.            # 绘制字符串

40.            img.draw\_string(110, 112, "delete all", color=(255, 255, 255), scale=1.5, mono\_space=0)

41.            # 在LCD屏幕上显示图像

42.            lcd.display(img)

43.            # 蜂鸣器声音提示

44.            beep.enable()

45.            beep.freq(1000)

46.            time.sleep(2)

47.            beep.disable()

48.            a = img.clear()

49.

50.        # 删除用户命令

51.        # 如果蓝牙数据中包含 'delete'

52.        if 'delete' in text:

53.            print("--------删除用户--------")

54.            delnum = text[3:].decode('utf-8')  #截取出删除的编号

55.            print('删除的编号: ', delnum)

56.            print(delnum)

57.            del text

58.            k=-1

59.            # 打开人脸信息文件

60.            with open('/sd/faceinfo.txt', 'r') as f:

61.                while(1):

62.                # 逐行读取文件内容

63.                    thisline=f.readline()

64.                    if not thisline:

65.                        # 如果文件已读完

66.                        break

67.                    # 计数器加1

68.                    k=k+1

69.                    # 判断是否找到要删除的编号

70.                    result = delnum in thisline

71.                    # 如果找到了第一个匹配项

72.                    if result==True and find\_flag == 0:

73.                        # 标记为找到

74.                        find\_flag=1

75.                        print('ok')

76.                    # 如果找到了多个匹配项

77.                    if result==True and find\_flag == 1:

78.                        # 记录要删除的行数

79.                        delline.append(k)

80.                    # 如果匹配项已结束

81.                    if result==False and find\_flag == 1:

82.                        # 标记为未找到

83.                        find\_flag = 0

84.                        break

85.                # 关闭文件

86.                f.close()

87.            # 如果SD卡中有此人，那么就删除，

88.            if delline:

89.                # 按行读入，删除最后一行

90.                # 打开文件

91.                file\_old = open('/sd/faceinfo.txt', 'r')

92.                # 逐行读取文件内容

93.                lines = [i for i in file\_old]

94.                # 删除指定行数

95.                del lines[int(delline[0]):int(delline[-1])+1]

96.                # 关闭文件

97.                file\_old.close()

98.                del file\_old

99.                # 清空要删除的行数列表

100.                delline.clear()

101.                # 再覆盖写入

102.                # 打开文件

103.                file\_new = open('/sd/faceinfo.txt', 'w')

104.                # 将修改后的内容写入文件

105.                file\_new.write(''.join(lines))

106.                # 关闭文件

107.                file\_new.close()

108.                del lines

109.                ##删除姓名列表和特征值列表中的数据

110.                #print('之前',names)

111.                i = 0

112.                # 清空名称列表

113.                temp\_num = ''

114.                names = []

115.                # 清空特征值列表

116.                record\_ftrs = []

117.                # 垃圾回收

118.                gc.collect()

119.                try:

120.                    # 打开人脸信息文件

121.                    with open("/sd/faceinfo.txt", "r") as f:

122.                        while(1):

123.                            # 垃圾回收

124.                            gc.collect()

125.                            # 逐行读取文件内容

126.                            lin = f.readline()

127.                            # 如果文件已读完

128.                            if not lin:

129.                                break

130.                            stunum = lin[0:lin.index('#')]    #获取编号

131.                            lin = lin[lin.index('#')+1:]      #截取除了编号以外的字符串

132.                            stu\_name = lin[0:lin.index('#')]  #获取姓名

133.                            names.append(stunum+'#'+stu\_name) #追加到姓名列表

134.                            lin = lin[lin.index('#')+1:]      #截取人脸特征

135.                            record\_ftrs.append(eval(lin))     #向人脸特征列表中添加已存特征

136.                except:

137.                    pass

138.                # 垃圾回收

139.                gc.collect()

140.                # 清除图像

141.                img.clear()

142.                # 绘制矩形

143.                img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

144.                # 绘制字符串

145.                img.draw\_string(110, 112, "delete %s"%delnum, color=(255, 255, 255), scale=1.5, mono\_space=0)

146.                # 在LCD屏幕上显示图像

147.                lcd.display(img)

148.                #蜂鸣器声音提示

149.                beep.enable()

150.                beep.freq(1000)

151.                time.sleep(2)

152.                beep.disable()

153.                a = img.clear()

154.

155.                # 回到正常模式

156.                LuRu\_mode = False   # 录入模式

157.                Door\_mode = False   # 门禁模式

158.                Normal\_mode = True  # 正常模式

159.            else:

160.                a = img.clear()

161.                # 绘制矩形

162.                a = img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

163.                # 绘制字符串

164.                a = img.draw\_string(118, 112, "No people!", color=(255, 255, 255), scale=1.5, mono\_space=0)

165.                # 在LCD屏幕上显示图像

166.                a = lcd.display(img)

167.                #蜂鸣器声音提示

168.                beep.enable()

169.                beep.freq(600)

170.                time.sleep(2)

171.                beep.disable()

172.                a = img.clear()

173.                # 回到正常模式

174.                LuRu\_mode = False   # 录入模式

175.                Door\_mode = False   # 门禁模式

176.                Normal\_mode = True  # 正常模式

177.

178.    # 检测按键

179.    # 录入模式按键

180.    if LuRu\_mode:

181.        try:

182.            # 在屏幕上显示“录入模式”字样

183.            a = img.draw\_string(0,0, b'Register Mode', color=(0,255,0),scale=1.6,mono\_space=1)

184.        except:

185.            pass

186.        # 判断蓝牙数据是否读取到并且长度大于等于2

187.        if text != None and len(text) >= 2: #如果读取到了数据，且大于等于2

188.            # 判断是否收到“register”命令，如果是则进行人脸注册

189.            if 'register' in text:

190.                print("--------人脸注册--------")

191.                stu\_num = text[2:].decode('utf-8')  # 截取出编号

192.                print('编号: ', stu\_num)

193.

194.                # 判断该编号是否已被录入

195.                # 如果该编号已经被录入，显示“ID Exist!”字样，并在屏幕上显示红色矩形区域

196.                if stu\_num in names:

197.                    a = img.clear()

198.                    a = img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

199.                    a = img.draw\_string(118, 112, "ID Exist!", color=(255, 255, 255), scale=1.5, mono\_space=0)

200.                    a = lcd.display(img)

201.                    # 蜂鸣器声音提示

202.                    beep.enable()

203.                    beep.freq(600)

204.                    time.sleep(2)

205.                    beep.disable()

206.                    # 回到正常模式

207.                    LuRu\_mode = False   # 录入模式

208.                    Door\_mode = False   # 门禁模式

209.                    Normal\_mode = True  # 正常模式

210.                # 如果编号没有被录入，进行下一步录入

211.                else:

212.                    # 设置录入标志位为1

213.                    luru\_flag = 1

214.

215.    # 门禁模式按键

216.    if Door\_mode:

217.    # 录入标志位设置为0

218.        luru\_flag = 0

219.        # 校验次数设置为0

220.        check\_num = 0

221.        # 在屏幕上显示“门禁模式”字样

222.        a = img.draw\_string(0,0, b'Entrance Mode', color=(0,255,0),scale=1.6,mono\_space=1)

223.        # ================================== RFID ================================

224.

225.        # time.sleep(2)

226.        # from micropython import const

227.        ################### config ###################

228.        CS\_NUM = const(18)

229.        SPI\_FREQ\_KHZ = const(600)

230.        SPI\_SCK = const(19)

231.        SPI\_MOSI = const(8)

232.        SPI\_MISO = const(15)

233.

234.        #############################################

235.

236.        #continue\_reading = True

237.

238.        # 20: CS\_NUM;

239.        fm.register(CS\_NUM, fm.fpioa.GPIOHS20, force=True)

240.

241.        # set gpiohs work mode to output mode

242.        cs = GPIO(GPIO.GPIOHS20, GPIO.OUT)

243.

244.        spi1 = SPI(SPI.SPI\_SOFT, mode=SPI.MODE\_MASTER, baudrate=SPI\_FREQ\_KHZ \* 1000,

245.                polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck=SPI\_SCK, mosi=SPI\_MOSI, miso=SPI\_MISO)

246.

247.        # Create an object of the class MFRC522

248.        MIFAREReader = MFRC522(spi1, cs)

249.

250.        # time.sleep\_ms(300)

251.        # Scan for cards

252.        (status, ataq) = MIFAREReader.MFRC522\_Request(MIFAREReader.PICC\_REQALL)

253.

254.        # If a card is found

255.        if status == MIFAREReader.MI\_OK:

256.            # 如果卡片被找到，打印卡片类型和UID

257.            print("Card detected type: ",hex(ataq[0]<<8|ataq[1]))

258.            # Get the UID of the card

259.            (status, uid) = MIFAREReader.MFRC522\_Anticoll()

260.

261.            # If we have the UID, continue

262.            if status == MIFAREReader.MI\_OK:

263.

264.                # Print UID

265.                print("Card read UID: " +

266.                    str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3]))

267.

268.                # This is the default key of M1(S50) for authentication

269.                # M1卡片的默认密钥

270.                key = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]

271.

272.                # Select the scanned tag

273.                # 选择扫描到的标签

274.                MIFAREReader.MFRC522\_SelectTag(uid)

275.

276.                # Authenticate

277.                # 验证卡片密钥

278.                status = MIFAREReader.MFRC522\_Auth(

279.                    MIFAREReader.PICC\_AUTHENT1A, 0x12, key, uid)

280.

281.                # Check if authenticated

282.                # 检查是否验证成功

283.                #if status == MIFAREReader.MI\_OK:

284.                    # 示例：定义一个包含16个元素的列表，并将第一个元素设为'Y'

285.                    ##data = ['Y',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

286.                    ##data = ['N',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

287.                    ## Fill the data with 0~16

288.                    #for x in range(0, 16):

289.                        #data.append(x)

290.

291.                    ## Write the data

292.                    # 向扇区0x12写入数据

293.                    #print("Sector 11 will now be filled with 1~16:")

294.                    # 使用MFRC522\_Write方法将data列表写入扇区0x12

295.                    #status = MIFAREReader.MFRC522\_Write(0x12, data)

296.                # 如果验证成功，读取RFID模块中的数据

297.                try:

298.                    if status == MIFAREReader.MI\_OK:

299.                        print("start to read")

300.                        # read the data

301.                        # MIFAREReader.MFRC522\_Read(0x12)

302.                        datas = MIFAREReader.MFRC522\_Read(0x12)

303.                        #读出卡中的id号

304.                        strnum = ' '

305.                        strnum = chr(datas[0])+str(datas[1])+str(datas[2])+str(datas[3])+str(datas[4])

306.                        #如果第一个字符不是B则说明卡错误

307.                        if chr(datas[0]) == 'Y':

308.                            led\_g.value(0)  # 绿灯亮

309.                            Servo(SS,90)

310.                            time.sleep(2)

311.                            Servo(SS,0)

312.                            led\_g.value(1)  # 绿灯灭

313.

314.                        if chr(datas[0]) == 'N':

315.                            #蜂鸣器声音提示

316.                            beep.enable()

317.                            beep.freq(300)

318.                            time.sleep(2)

319.                            beep.disable()

320.                            ##读出卡中的姓名

321.                            #if datas[11]==0: #名字两个字

322.                                #strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+"'"

323.                            #else :          #名字三个字

324.                                #strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+hex(datas[11])+hex(datas[12])+hex(datas[13])+"'"

325.                            #strname = eval(strname.replace('0','\\')).decode('utf8')

326.

327.                        # Stop

328.                        MIFAREReader.MFRC522\_StopCrypto1()

329.                    #else:

330.                        #print("Authentication error")

331.                except Exception as e:

332.                     beep.enable()

333.                     beep.freq(1000)

334.                     time.sleep(2)

335.                     beep.disable()

336.

337.

338.        try:

339.            # 如果人脸识别分数高于设定的阈值，显示编号与分数，并开启绿灯

340.            if max\_score > ACCURACY:

341.                a = img.draw\_string(i.x()+4,i.y()-20, ("%s: %2.1f" % (names[index], max\_score)), color=(0,255,0),scale=2) # 显示编号与分数

342.                a = lcd.display(img)

343.                print("--------人脸识别成功--------")

344.                led\_g.value(0)  # 绿灯亮

345.                Servo(SS,90)

346.                time.sleep(2)

347.                Servo(SS,0)

348.                led\_g.value(1)  # 绿灯灭

349.                shibie\_num = 0

350.            elif code:

351.                # 如果人脸识别分数低于设定的阈值，显示分数，并计数，如果计数超过3次则表示人脸识别失败，开启红灯并发出蜂鸣器声音

352.                img.draw\_string(i.x()+4,i.y()-20, ("%s: %2.1f" % ("No",max\_score)), color=(0,255,0),scale=2) # 显示分数

353.                a = lcd.display(img)

354.                shibie\_num = shibie\_num + 1

355.                if shibie\_num > 3:

356.                    shibie\_num = 0

357.                    print("--------人脸识别失败--------")

358.                    led\_r.value(0)  # 红灯亮

359.                    beep.enable()

360.                    beep.freq(600)

361.                    time.sleep(2)

362.                    beep.disable()

363.                    led\_r.value(1)  # 红灯灭

364.        except:

365.            pass

366.            # 删除变量code，释放内存空间

367.        del code

368.

369.    # 正常模式按键

370.    # 设置录入标志和检查数字为0，使用LCD显示屏显示“等待……”

371.    if Normal\_mode:

372.        luru\_flag = 0

373.        check\_num = 0

374.        a = img.draw\_string(0,0, b'Waiting......', color=(0,255,0),scale=1.6,mono\_space=1)

375.        ## ================================== RFID ================================

376.

377.        ## time.sleep(2)

378.        ## from micropython import const

379.        #################### config ###################

380.        #CS\_NUM = const(18)

381.        #SPI\_FREQ\_KHZ = const(600)

382.        #SPI\_SCK = const(19)

383.        #SPI\_MOSI = const(8)

384.        #SPI\_MISO = const(15)

385.

386.        ##############################################

387.

388.        ##continue\_reading = True

389.

390.        ## 20: CS\_NUM;

391.        #fm.register(CS\_NUM, fm.fpioa.GPIOHS20, force=True)

392.

393.        ## set gpiohs work mode to output mode

394.        #cs = GPIO(GPIO.GPIOHS20, GPIO.OUT)

395.

396.        #spi1 = SPI(SPI.SPI\_SOFT, mode=SPI.MODE\_MASTER, baudrate=SPI\_FREQ\_KHZ \* 1000,

397.                #polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck=SPI\_SCK, mosi=SPI\_MOSI, miso=SPI\_MISO)

398.

399.        ## Create an object of the class MFRC522

400.        #MIFAREReader = MFRC522(spi1, cs)

401.

402.        ## time.sleep\_ms(300)

403.        ## Scan for cards

404.        #(status, ataq) = MIFAREReader.MFRC522\_Request(MIFAREReader.PICC\_REQALL)

405.

406.        ## If a card is found

407.        #if status == MIFAREReader.MI\_OK:

408.            #print("Card detected type: ",hex(ataq[0]<<8|ataq[1]))

409.            ## Get the UID of the card

410.            #(status, uid) = MIFAREReader.MFRC522\_Anticoll()

411.

412.            ## If we have the UID, continue

413.            #if status == MIFAREReader.MI\_OK:

414.

415.                ## Print UID

416.                #print("Card read UID: " +

417.                    #str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3]))

418.

419.                ## This is the default key of M1(S50) for authentication

420.                #key = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]

421.

422.                ## Select the scanned tag

423.                #MIFAREReader.MFRC522\_SelectTag(uid)

424.

425.                ## Authenticate

426.                #status = MIFAREReader.MFRC522\_Auth(

427.                    #MIFAREReader.PICC\_AUTHENT1A, 0x12, key, uid)

428.

429.                ## Check if authenticated

430.                ##if status == MIFAREReader.MI\_OK:

431.                    ###data = ['Y',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

432.                    ###data = ['N',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

433.                    ### Fill the data with 0~16

434.                    ##for x in range(0, 16):

435.                        ##data.append(x)

436.

437.                    ### Write the data

438.                    ##print("Sector 11 will now be filled with 1~16:")

439.                    ##status = MIFAREReader.MFRC522\_Write(0x12, data)

440.                #try:

441.                    #if status == MIFAREReader.MI\_OK:

442.                        #print("start to read")

443.                        ## read the data

444.                        ## MIFAREReader.MFRC522\_Read(0x12)

445.                        #datas = MIFAREReader.MFRC522\_Read(0x12)

446.                        ##读出卡中的id号

447.                        #strnum = ' '

448.                        #strnum = chr(datas[0])+str(datas[1])+str(datas[2])+str(datas[3])+str(datas[4])

449.                        ##如果第一个字符不是B则说明卡错误

450.                        #if chr(datas[0]) == 'Y':

451.                            #led\_g.value(0)  # 绿灯亮

452.                            #Servo(SS,90)

453.                            #time.sleep(2)

454.                            #Servo(SS,0)

455.                            #led\_g.value(1)  # 绿灯灭

456.

457.                        #if chr(datas[0]) == 'N':

458.                            ##蜂鸣器声音提示

459.                            #beep.enable()

460.                            #beep.freq(300)

461.                            #time.sleep(2)

462.                            #beep.disable()

463.                            ###读出卡中的姓名

464.                            ##if datas[11]==0: #名字两个字

465.                                ##strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+"'"

466.                            ##else :          #名字三个字

467.                                ##strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+hex(datas[11])+hex(datas[12])+hex(datas[13])+"'"

468.                            ##strname = eval(strname.replace('0','\\')).decode('utf8')

469.

470.                        ## Stop

471.                        #MIFAREReader.MFRC522\_StopCrypto1()

472.                    ##else:

473.                        ##print("Authentication error")

474.                #except Exception as e:

475.                     #beep.enable()

476.                     #beep.freq(1000)

477.                     #time.sleep(2)

478.                     #beep.disable()

479.

480.    a = lcd.display(img)

481.

482.# 无人的时候LCD黑屏，休眠状态（录入的时候不能进入休眠状态）

483.# 如果没有人在操作，将LCD清空并显示“休眠中……”，并将标志位no\_people设置为1。

484.elif luru\_flag == 0:

485.    no\_people = 1

486.# 如果有人在操作，则将标志位no\_people和no\_flag都置为0，表示有人正在操作，不进行休眠。

487.if no\_people == 1 and no\_flag == 1:

488.    no\_people = 0

489.    no\_flag = 0

490.    a = img.clear()   # 清空一次LCD

491.    # 最后，将img对象在LCD上显示出来，并回收内存。

492.    a = img.draw\_string(40,110, b'Suspending......', color=(0,255,0),scale=2,mono\_space=1)

493.    a = lcd.display(img)

494.# del img用于删除img对象占用的内存

495.del img

496.# gc.collect()用于回收未被使用的内存。

497.gc.collect()

5.各模块实操结果展示和实现指标

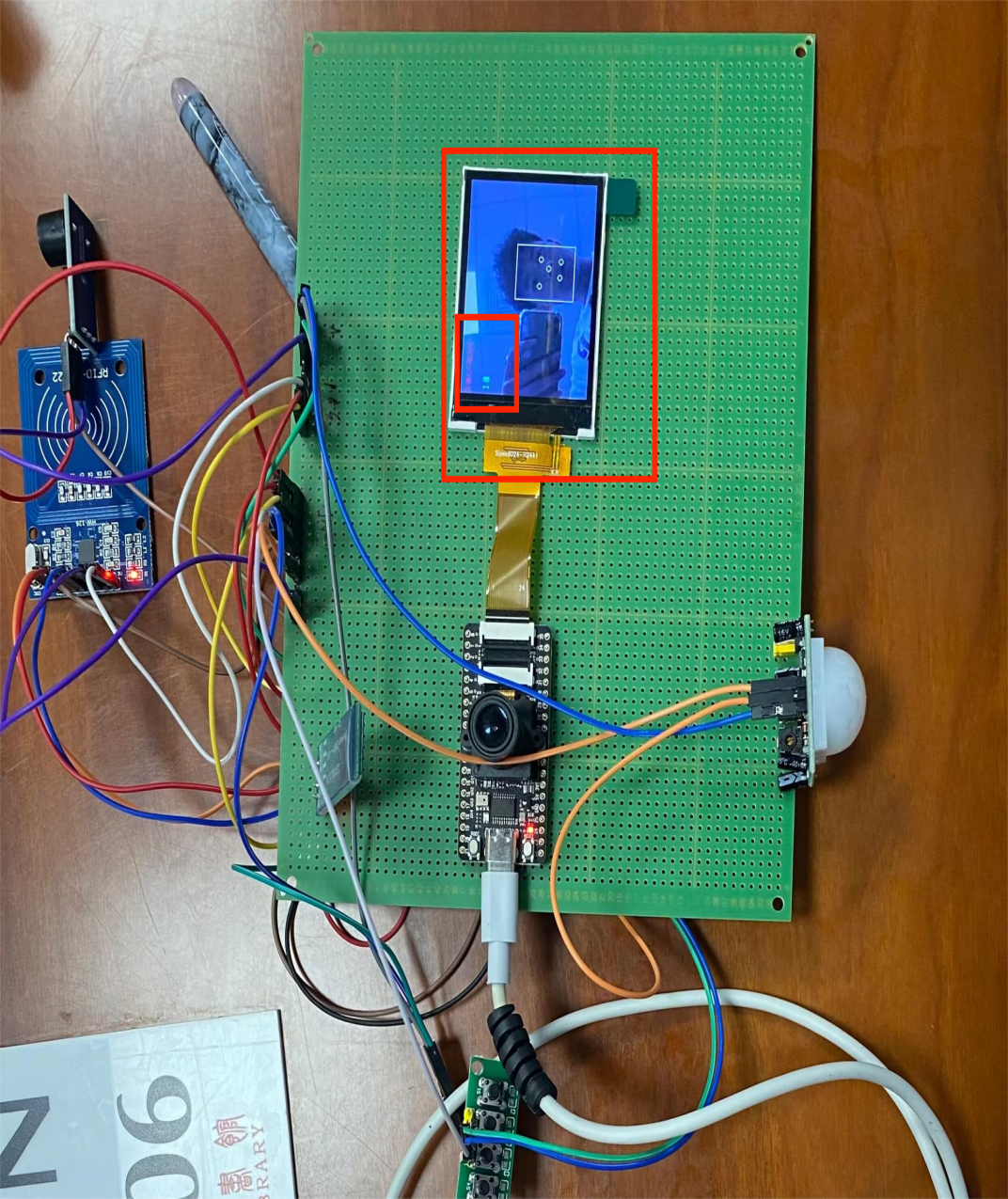


图5-1人脸录入成功

输入“register\*\*\*\*”（\*\*\*\*代表录入人脸编号）后按照yolo v2算法采集196维人脸数据（3轮18次采样），并储存在SD卡文件中，且屏幕显示采样进度，录入完成屏幕显示“successful”，蜂鸣器发低声。

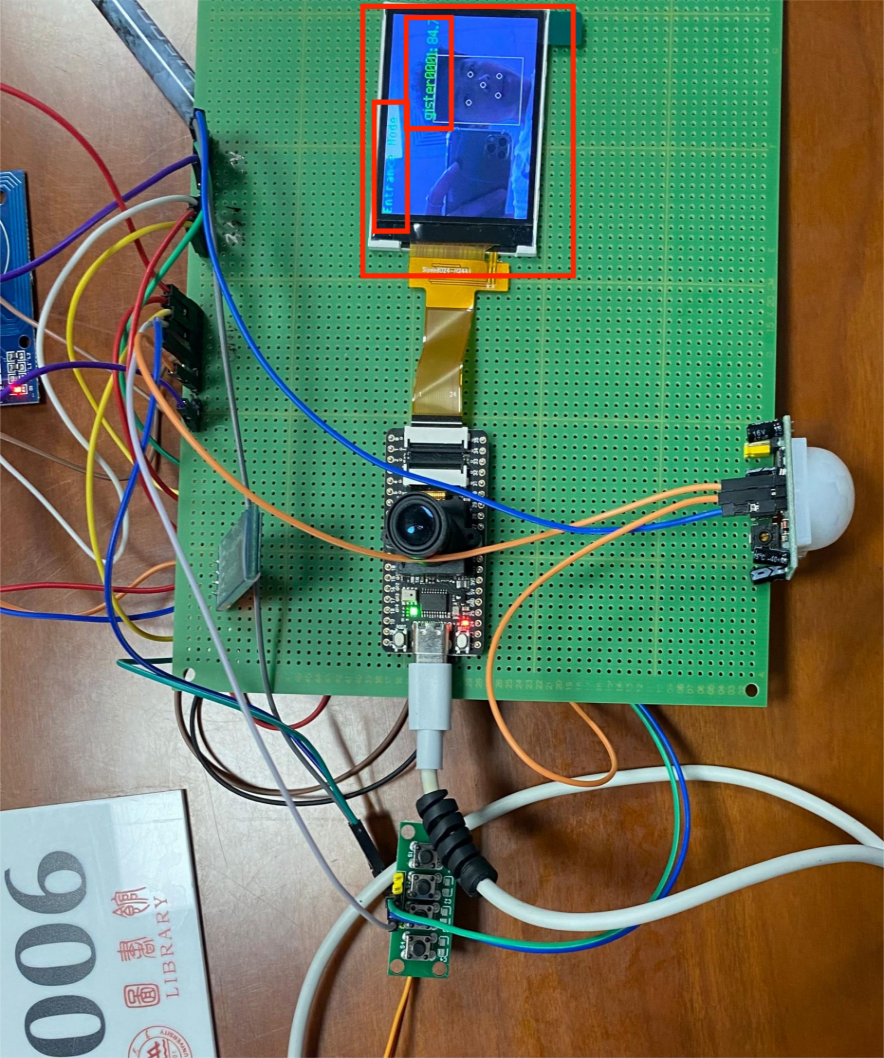


图5-2人脸识别成功

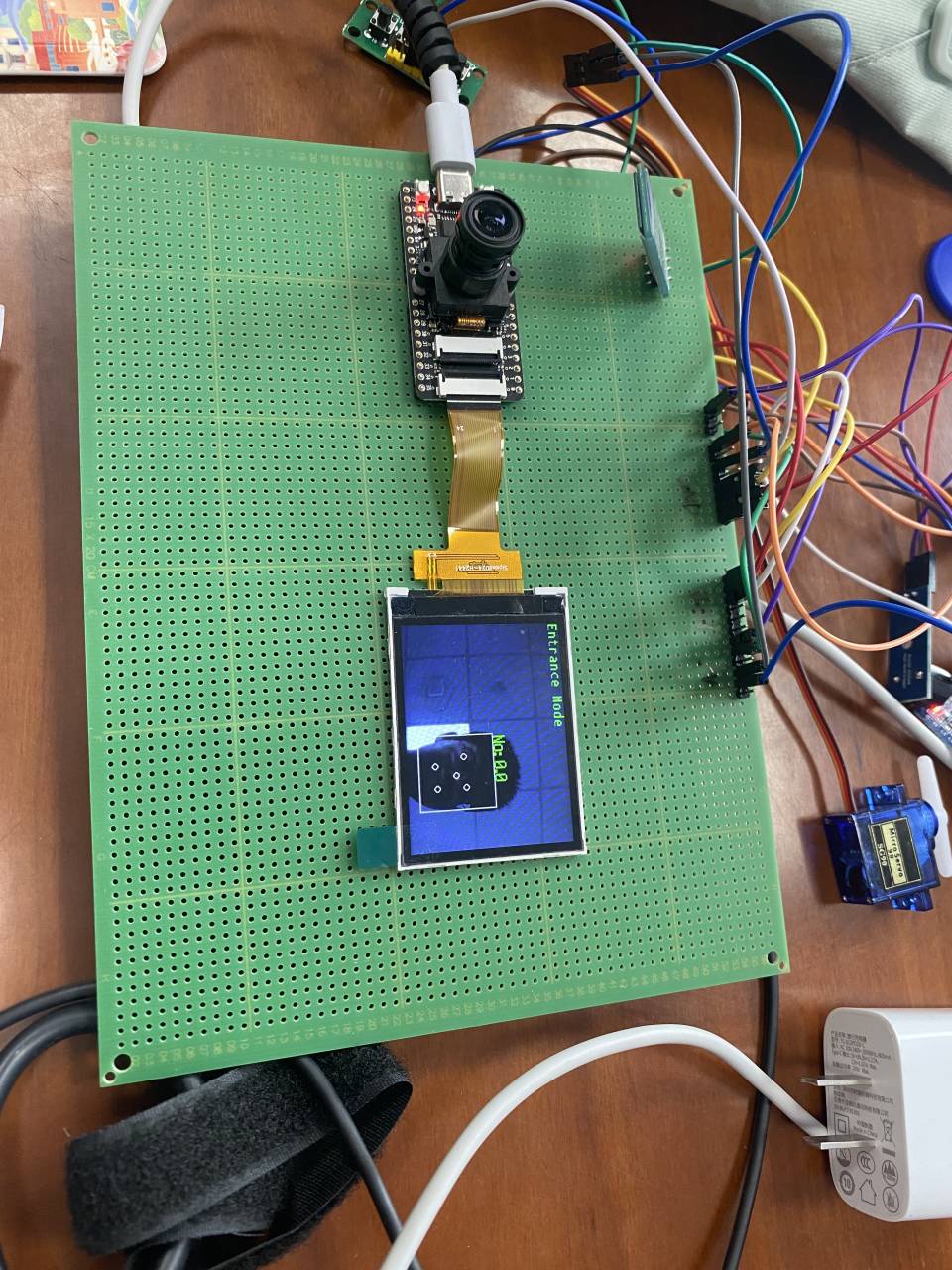


图5-3人脸识别比对失败

门禁模式时，人脸识别框上会显示序列号和相似度，设定比对阈值，此处设置75，高于则蜂鸣器发低声且舵机旋转，低于则蜂鸣器发高声舵机无反应。

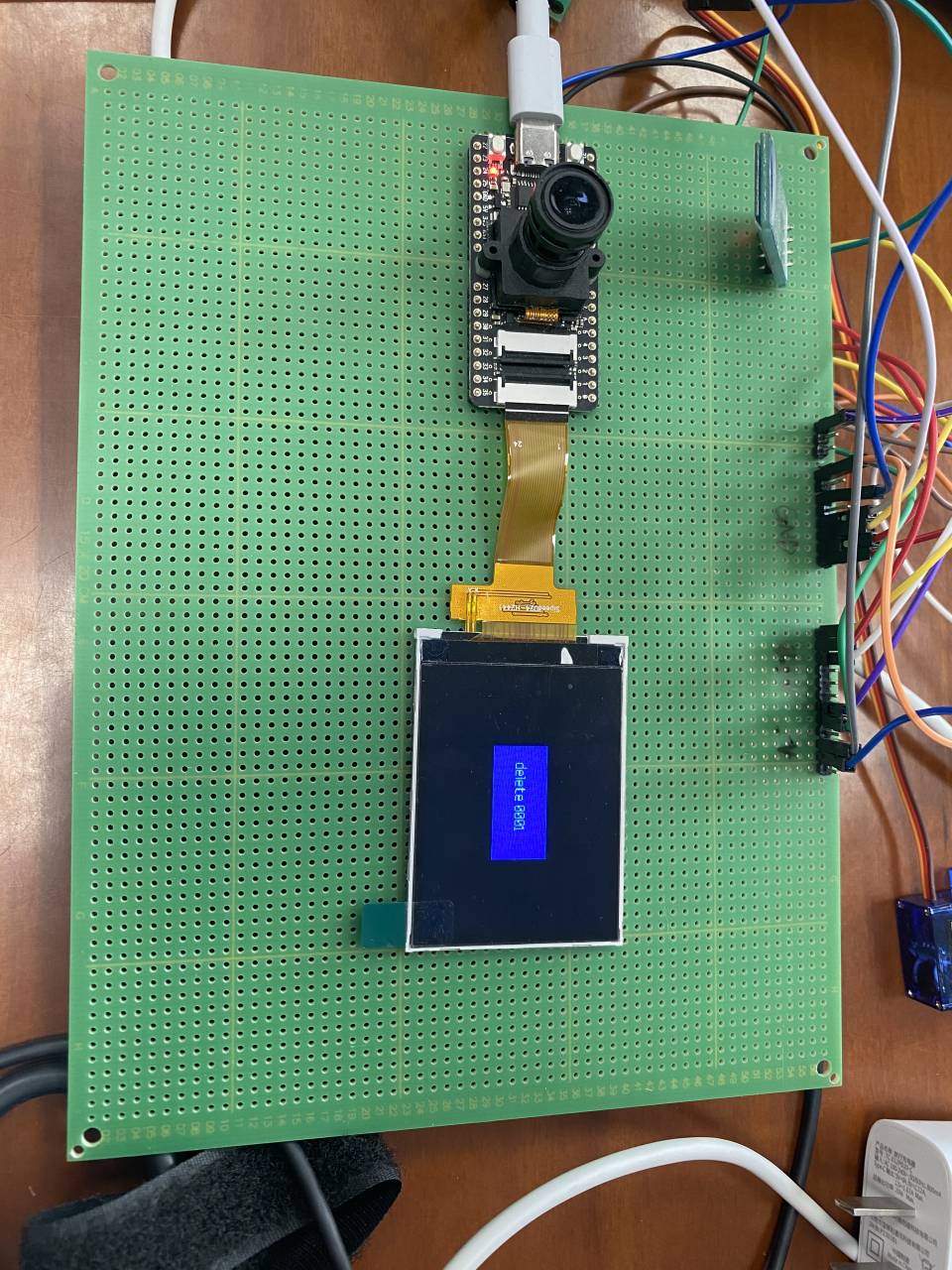


图5-4 人脸数据删除成功

输入“delete\*\*\*\*”则删除对应用户储存在SD卡的特征值，蜂鸣器发高声。且屏幕显示ok，输入“erase”则删除SD卡内所有用户人脸特征值，蜂鸣器发高声且屏幕显示delete all。输入“open”则无条件控制舵机旋转，用作临时开关门禁。



图5-5蓝牙app操作

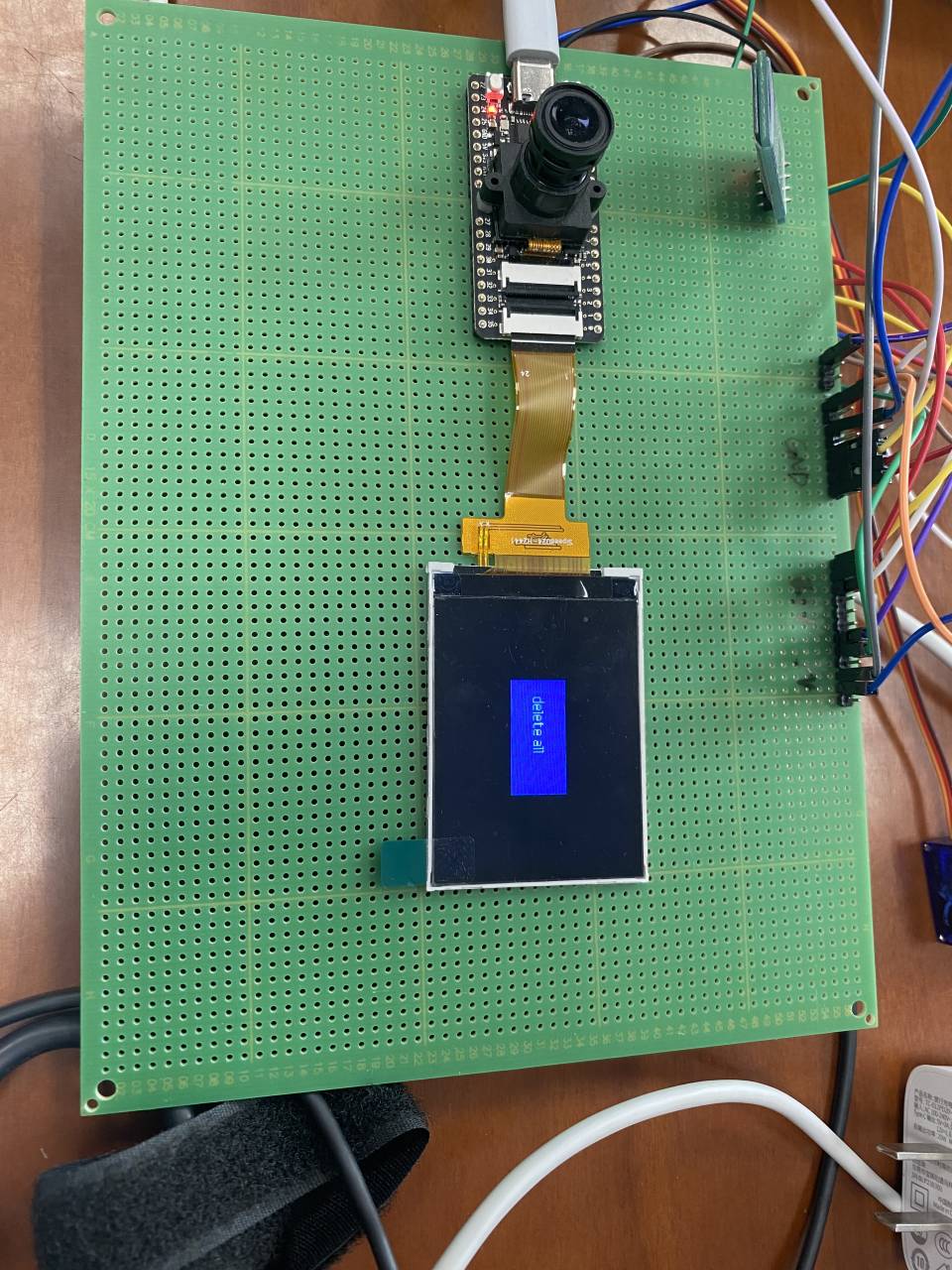


图5-6删除所有人脸数据

6.可改进之处及拓展方向

6.1 不足及改进之处

1、深度学习算法改进:可以采用更加先进的人脸检测方法和特征提取方法,如RetinaNet检测器和ResNet特征提取网络等,提高人脸识别的准确率。还可以采用多模型结合的方法提高系统性能。

2、采集更丰富的人脸数据:随着更多人员的使用,可以不断采集用户的人脸数据进行模型训练,提高对不同人脸的识别能力。还可以采集不同光照、表情等条件下的人脸,使人脸识别系统具有更好的适应性。

3、增加更多生物特征识别:人脸识别系统可以与其他生物特征识别技术结合,如指纹识别、虹膜识别等,采用多生物特征融合识别可以大大提高识别准确性和安全性。

6.2 系统方案可拓展方向

1、面部表情识别：门禁系统可以引入面部表情识别技术，实现对人员情绪和健康状况的识别和监测，提高门禁系统的应用价值和服务质量。需要考虑面部表情识别技术的稳定性和精度，以及对个人隐私和数据安全的保护等问题。

2、智能硬件应用：可以结合智能硬件技术，例如智能门锁、智能摄像头等，实现门禁系统的智能化升级和拓展。智能硬件应用需要对门禁系统的硬件设备进行升级和改进，例如引入智能门锁、智能摄像头等设备，以增强门禁系统的功能和性能。同时，还需要考虑硬件的兼容性、可靠性和安全性等问题。

3、云计算应用：可以将门禁系统与云计算技术相结合，实现数据存储、计算、分析和管理的云端化，从而实现门禁系统的远程监控和管理。云计算应用需要对门禁系统的数据进行云端存储和管理，利用云计算平台进行数据分析和处理，从而实现门禁系统的远程监控和管理。同时，还需要考虑数据隐私、信息安全和云计算平台的可靠性和稳定性等问题。

7.课程总结与收获

此次综合设计收获丰富，主要从以下几点来说：

人脸识别门禁系统：本部分主要关注于基于深度学习和图像识别技术实现的人脸识别系统。要学习如何使用深度学习算法（例如卷积神经网络）对人脸进行识别和分类，以及如何利用传感器技术（例如摄像头）实时捕捉人脸图像。

通信系统：本部分侧重于通信系统的设计和实现，包括无线通信技术（如蓝牙，Wi-Fi，NFC等）、有线通信技术（如以太网）以及相关的通信协议和标准。要学习如何设计和实现一个稳定、可靠的通信系统，以支持人脸识别门禁系统的运行。

综合设计：在掌握了人脸识别门禁系统和通信系统的基本知识之后，我们需要将这些知识整合起来，设计出一套适用于实际应用场景的综合系统。这包括了系统架构设计、硬件选择、软件开发、系统安装和维护等方面。通过此次实际项目来应用所学知识，提高了对实际工程问题的理解和解决能力。

总结来说，这项综合设计提高了我们在计算机视觉、深度学习、传感器技术以及通信系统设计等多个方面的综合能力，让我们能够在实际工程项目中应用所学知识，设计出高性能的人脸识别门禁系统和通信解决方案。

总之，《基于深度学习图像识别与传感器技术实现的人脸识别门禁系统》的通信系统综合设计，提供了一个智能化门禁系统的典型案例，它的立意是提高门禁系统的安全性和便利性，并提供了一些有益的启示。在未来的通信系统设计中，我们应该注重数据传输效率、可靠性和稳定性，并考虑到系统的可扩展性和兼容性，以实现更加高效、智能和安全的数据传输。同时，我们也应该不断探索和应用现代技术，为社会带来更多的创新性应用和智能化解决方案。这项综合设计将人脸识别门禁系统和通信系统这两个领域的知识融合在一起，实现了一种集成的、高效的、安全的门禁和通信解决方案。

8.参考资料

1、RFID学习资料<https://blog.csdn.net/HuangChen666/article/details/114024767?spm=1001.2014.3001.5506>

2、K210学习系列

<https://blog.csdn.net/Thousand_drive/article/details/123796878?spm=1001.2014.3001.5506>

1. 人脸识别学习资料

<https://blog.csdn.net/Aaron357/article/details/93485279?spm=1001.2101.3001.6650.5&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-5-93485279-blog-107153393.235%5Ev36%5Epc_relevant_default_base3&depth_1-utm_source=>

1. 固件烧录、信息储存学习资料

<https://blog.csdn.net/HuangChen666/article/details/113995079?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522168536128716800222862427%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168536128716800222862427&biz_id=0&utm_me>

1. 蓝牙模块学习资料

<https://blog.csdn.net/qq_44125275/article/details/128266379?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522168536196416800192217378%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=168536196416800192217378&biz_id>

[1] Zhu, X., Lei, Z., Liu, X., Shi, H., & Li, S. Z. (2018). Face recognition: From traditional to deep learning methods. Advances in Computers, 113, 1-69.

[2] AlBdairi, A.J.A.; Xiao, Z.; Alkhayyat, A.; Humaidi, A.J.; Fadhel, M.A.; Taher, B.H.; Alzubaidi, L.; Santamaría, J.; Al-Shamma, O. Face Recognition Based on Deep Learning and FPGA for Ethnicity Identification. Appl. Sci. 2022, 12, 2605. https://doi.org/10.3390/app12052605

[3] Chen, J., Liao, S., & Liu, Y. (2021). Intelligent access control system based on deep learning and IoT. Journal of Physics: Conference Series, 1821(1), 012074.

[4] Derbel, A., Vivet, D. and Emile, B. (2015), Access control based on gait analysis and face recognition. Electron. Lett., 51: 751-752. <https://doi.org/10.1049/el.2015.0767>

[5] A. Nag, J. N. Nikhilendra and M. Kalmath, "IOT Based Door Access Control Using Face Recognition," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 2018, pp. 1-3, doi: 10.1109/I2CT.2018.8529749.

[6] Radzi S A, Alif M K M F, Athirah Y N, et al. IoT based facial recognition door access control home security system using raspberry pi[J]. International Journal of Power Electronics and Drive Systems, 2020, 11(1): 417.

9.附录

9.1 附录A 源代码

1.from machine import SPI

2.import sensor

3.import image

4.import lcd

5.import KPU as kpu

6.import time

7.from Maix import FPIOA, GPIO

8.import gc

9.from fpioa\_manager import fm

10.from board import board\_info

11.import utime

12.from machine import Timer,PWM

13.from machine import UART

14.from machine import WDT

15.import os

16.from micropython import const

17.

18.# ==============================RFID CONFIG==============================

19.

20.# 定义了一个名为 MFRC522 的Python类，它包含了MFRC522 RFID读写器的各种常量。

21.class MFRC522:

22.    NRSTPD = 22

23.

24.    MAX\_LEN = 16

25.

26.    PCD\_IDLE = 0x00

27.    PCD\_AUTHENT = 0x0E

28.    PCD\_RECEIVE = 0x08

29.    PCD\_TRANSMIT = 0x04

30.    PCD\_TRANSCEIVE = 0x0C

31.    PCD\_RESETPHASE = 0x0F

32.    PCD\_CALCCRC = 0x03

33.

34.    PICC\_REQIDL = 0x26

35.    PICC\_REQALL = 0x52

36.    PICC\_ANTICOLL = 0x93

37.    PICC\_SElECTTAG = 0x93

38.    PICC\_AUTHENT1A = 0x60

39.    PICC\_AUTHENT1B = 0x61

40.    PICC\_READ = 0x30

41.    PICC\_WRITE = 0xA0

42.    PICC\_DECREMENT = 0xC0

43.    PICC\_INCREMENT = 0xC1

44.    PICC\_RESTORE = 0xC2

45.    PICC\_TRANSFER = 0xB0

46.    PICC\_HALT = 0x50

47.

48.    MI\_OK = 0

49.    MI\_NOTAGERR = 1

50.    MI\_ERR = 2

51.

52.    Reserved00 = 0x00

53.    CommandReg = 0x01

54.    CommIEnReg = 0x02

55.    DivlEnReg = 0x03

56.    CommIrqReg = 0x04

57.    DivIrqReg = 0x05

58.    ErrorReg = 0x06

59.    Status1Reg = 0x07

60.    Status2Reg = 0x08

61.    FIFODataReg = 0x09

62.    FIFOLevelReg = 0x0A

63.    WaterLevelReg = 0x0B

64.    ControlReg = 0x0C

65.    BitFramingReg = 0x0D

66.    CollReg = 0x0E

67.    Reserved01 = 0x0F

68.

69.    Reserved10 = 0x10

70.    ModeReg = 0x11

71.    TxModeReg = 0x12

72.    RxModeReg = 0x13

73.    TxControlReg = 0x14

74.    TxAutoReg = 0x15

75.    TxSelReg = 0x16

76.    RxSelReg = 0x17

77.    RxThresholdReg = 0x18

78.    DemodReg = 0x19

79.    Reserved11 = 0x1A

80.    Reserved12 = 0x1B

81.    MifareReg = 0x1C

82.    Reserved13 = 0x1D

83.    Reserved14 = 0x1E

84.    SerialSpeedReg = 0x1F

85.

86.    Reserved20 = 0x20

87.    CRCResultRegM = 0x21

88.    CRCResultRegL = 0x22

89.    Reserved21 = 0x23

90.    ModWidthReg = 0x24

91.    Reserved22 = 0x25

92.    RFCfgReg = 0x26

93.    GsNReg = 0x27

94.    CWGsPReg = 0x28

95.    ModGsPReg = 0x29

96.    TModeReg = 0x2A

97.    TPrescalerReg = 0x2B

98.    TReloadRegH = 0x2C

99.    TReloadRegL = 0x2D

100.    TCounterValueRegH = 0x2E

101.    TCounterValueRegL = 0x2F

102.

103.    Reserved30 = 0x30

104.    TestSel1Reg = 0x31

105.    TestSel2Reg = 0x32

106.    TestPinEnReg = 0x33

107.    TestPinValueReg = 0x34

108.    TestBusReg = 0x35

109.    AutoTestReg = 0x36

110.    VersionReg = 0x37

111.    AnalogTestReg = 0x38

112.    TestDAC1Reg = 0x39

113.    TestDAC2Reg = 0x3A

114.    TestADCReg = 0x3B

115.    Reserved31 = 0x3C

116.    Reserved32 = 0x3D

117.    Reserved33 = 0x3E

118.    Reserved34 = 0x3F

119.

120.    serNum = []

121.

122.# ==============================RFID SYSTEM FUNCTIONS==============================

123.    # \_\_init\_\_(self, spi, cs)：类的构造函数，接收2个参数，分别是SPI总线对象和片选引脚对象。

124.    # 在构造函数中，将SPI总线和片选引脚保存到类的成员变量中，并调用MFRC522\_Init()方法进行初始化。

125.    def \_\_init\_\_(self, spi, cs):

126.        # 初始化

127.        self.spi = spi

128.        self.cs = cs

129.        self.MFRC522\_Init()

130.    # Write\_MFRC522(self, addr, val)：写入MFRC522寄存器，接收2个参数，分别是寄存器地址和写入的值。

131.    # 在方法中，先将片选引脚置为低电平，然后通过SPI总线写入寄存器地址和写入的值，最后将片选引脚置为高电平。

132.    def Write\_MFRC522(self, addr, val):

133.        self.cs.value(0)

134.        self.spi.write(((addr << 1) & 0x7E))

135.        self.spi.write(val)

136.        self.cs.value(1)

137.    # Read\_MFRC522(self, addr)：读取MFRC522寄存器，接收1个参数，即寄存器地址。

138.    # 在方法中，先将片选引脚置为低电平，然后通过SPI总线写入寄存器地址，并读取该地址的值，最后将片选引脚置为高电平，并返回读取的值。

139.    def Read\_MFRC522(self, addr):

140.        self.cs.value(0)

141.        self.spi.write((((addr << 1) & 0x7E) | 0x80))

142.        val = self.spi.read(1)

143.        self.cs.value(1)

144.        return val[0]

145.    # SetBitMask(self, reg, mask)：设置MFRC522寄存器的位掩码，接收2个参数，即寄存器地址和位掩码。

146.    # 在方法中，先读取该寄存器的值，然后将位掩码与该寄存器的值进行按位或运算，最后将运算结果写入该寄存器。

147.    def SetBitMask(self, reg, mask):

148.        tmp = self.Read\_MFRC522(reg)

149.        self.Write\_MFRC522(reg, tmp | mask)

150.    # ClearBitMask(self, reg, mask)：清除MFRC522寄存器的位掩码，接收2个参数，即寄存器地址和位掩码。

151.    # 在方法中，先读取该寄存器的值，然后将位掩码按位取反，并与该寄存器的值进行按位与运算，最后将运算结果写入该寄存器。

152.    def ClearBitMask(self, reg, mask):

153.        tmp = self.Read\_MFRC522(reg)

154.        self.Write\_MFRC522(reg, tmp & (~mask))

155.

156.    # 通过RC522与卡片通信

157.    def MFRC522\_ToCard(self, command, sendData):

158.        backData = []

159.        backLen = 0

160.        status = self.MI\_ERR

161.        irqEn = 0x00

162.        waitIRq = 0x00

163.        lastBits = None

164.        n = 0

165.        i = 0

166.

167.        if command == self.PCD\_AUTHENT:     # Mifare认证

168.            irqEn = 0x12                    # 允许错误中断请求ErrIEn  允许空闲中断IdleIEn

169.            waitIRq = 0x10                  # 认证寻卡等待时候 查询空闲中断标志位

170.        if command == self.PCD\_TRANSCEIVE:  # 接收发送 发送接收

171.            irqEn = 0x77                    # 允许TxIEn RxIEn IdleIEn LoAlertIEn ErrIEn TimerIEn

172.            waitIRq = 0x30                  # 寻卡等待时候 查询接收中断标志位与 空闲中断标志位

173.

174.        # IRqInv置位管脚IRQ与Status1Reg的IRq位的值相反

175.        self.Write\_MFRC522(self.CommIEnReg, irqEn | 0x80)

176.        # Set1该位清零时，CommIRqReg的屏蔽位清零

177.        self.ClearBitMask(self.CommIrqReg, 0x80)

178.        # 写空闲命令

179.        self.Write\_MFRC522(self.CommandReg, self.PCD\_IDLE)

180.

181.        # 置位FlushBuffer清除内部FIFO的读和写指针以及ErrReg的BufferOvfl标志位被清除

182.        self.SetBitMask(self.FIFOLevelReg, 0x80)

183.

184.        while(i < len(sendData)):

185.            self.Write\_MFRC522(self.FIFODataReg, sendData[i])  # 写数据进FIFOdata

186.            i = i+1

187.

188.        self.Write\_MFRC522(self.CommandReg, command)  # 写命令

189.

190.        if command == self.PCD\_TRANSCEIVE:

191.            # StartSend置位启动数据发送 该位与收发命令使用时才有效

192.            self.SetBitMask(self.BitFramingReg, 0x80)

193.

194.        i = 1000 \* 3

195.

196.        # 认证 与寻卡等待时间

197.        while True:

198.            n = self.Read\_MFRC522(self.CommIrqReg)  # 查询事件中断

199.            i = i - 1

200.            if not ((i != 0) and (not (n & 0x01)) and (not (n & waitIRq))):

201.                break

202.

203.        self.ClearBitMask(self.BitFramingReg, 0x80)  # 清理允许StartSend位

204.

205.        if i != 0:

206.            # 读错误标志寄存器BufferOfI CollErr ParityErr ProtocolErr

207.            if not (self.Read\_MFRC522(self.ErrorReg) & 0x1B):

208.                status = self.MI\_OK

209.

210.                if n & irqEn & 0x01:

211.                    status = self.MI\_NOTAGERR

212.

213.                if command == self.PCD\_TRANSCEIVE:

214.                    n = self.Read\_MFRC522(self.FIFOLevelReg)

215.                    lastBits = self.Read\_MFRC522(self.ControlReg) & 0x07

216.                    # print("n: {}, {}".format(n, lastBits))

217.                    if lastBits != 0:

218.                        backLen = (n-1)\*8 + lastBits

219.                    else:

220.                        backLen = n\*8

221.

222.                    if n == 0:

223.                        n = 1

224.                    if n > self.MAX\_LEN:

225.                        n = self.MAX\_LEN

226.

227.                    i = 0

228.                    while i < n:

229.                        backData.append(self.Read\_MFRC522(self.FIFODataReg))

230.                        i = i + 1

231.            else:

232.                # print("erro: {}".format(hex(self.Read\_MFRC522(self.ErrorReg))))

233.                status = self.MI\_ERR

234.        # print("backlen: {}".format(backLen))

235.        self.SetBitMask(self.ControlReg, 0x80)

236.        # stop timer now

237.        self.Write\_MFRC522(self.CommandReg, self.PCD\_IDLE)

238.        return (status, backData, backLen)

239.

240.    def MFRC522\_Request(self, reqMode):

241.          status = None

242.          backBits = None

243.          TagType = []

244.

245.          # 清理指示MIFARECyptol单元接通以及所有卡的数据通信被加密的情况

246.          self.ClearBitMask(self.Status2Reg, 0x08)

247.          # 发送的最后一个字节的 七位

248.          self.Write\_MFRC522(self.BitFramingReg, 0x07)

249.          # TX1,TX2管脚的输出信号传递经发送调制的13.56的能量载波信号

250.          self.SetBitMask(self.TxControlReg, 0x03)

251.

252.          TagType.append(reqMode)

253.          (status, backData, backBits) = self.MFRC522\_ToCard(

254.              self.PCD\_TRANSCEIVE, TagType)

255.          # print("backBits: {}".format(backBits))

256.          if ((status != self.MI\_OK) | (backBits != 0x10)):

257.              status = self.MI\_ERR

258.

259.          return (status, backData)

260.

261.    def MFRC522\_Anticoll(self):

262.           # 定义返回数据列表

263.           backData = []

264.           # 定义序列号校验值

265.           serNumCheck = 0

266.           # 定义序列号列表

267.           serNum = []

268.           # 将BitFramingReg寄存器的值设为0x00

269.           self.Write\_MFRC522(self.BitFramingReg, 0x00)

270.           # 将PICC\_ANTICOLL命令添加到序列号列表中

271.           serNum.append(self.PICC\_ANTICOLL)

272.           # 将0x20添加到序列号列表中

273.           serNum.append(0x20)

274.           # 通过MFRC522\_ToCard方法向RC522模块发送序列号列表，并获取响应状态、返回数据和返回数据位数

275.           (status, backData, backBits) = self.MFRC522\_ToCard(

276.               self.PCD\_TRANSCEIVE, serNum)

277.           # 如果响应状态为MI\_OK

278.           if(status == self.MI\_OK):

279.               i = 0

280.               if len(backData) == 5:

281.                   while i < 4:

282.                       serNumCheck = serNumCheck ^ backData[i]

283.                       i = i + 1

284.                   if serNumCheck != backData[i]:

285.                       status = self.MI\_ERR

286.               else:

287.                   status = self.MI\_ERR

288.

289.           return (status, backData)

290.

291.    def CalulateCRC(self, pIndata):

292.           # 清除DivIrqReg寄存器的4位

293.           self.ClearBitMask(self.DivIrqReg, 0x04)

294.           # 设置FIFOLevelReg寄存器的7位

295.           self.SetBitMask(self.FIFOLevelReg, 0x80)

296.           i = 0

297.           while i < len(pIndata):

298.               # 将要进行CRC校验的数据写入FIFODataReg寄存器中

299.               self.Write\_MFRC522(self.FIFODataReg, pIndata[i])

300.               i = i + 1

301.           # 发送PCD\_CALCCRC命令，开始进行CRC校验

302.           self.Write\_MFRC522(self.CommandReg, self.PCD\_CALCCRC)

303.           i = 0xFF

304.           while True:

305.           # 读取DivIrqReg寄存器的值

306.               n = self.Read\_MFRC522(self.DivIrqReg)

307.               i = i - 1

308.                # 如果i为0或者DivIrqReg寄存器的4位为1，则退出循环

309.               if not ((i != 0) and not (n & 0x04)):

310.                   break

311.           # 定义输出列表

312.           pOutData = []

313.           # 将CRCResultRegL寄存器的值添加到输出列表中

314.           pOutData.append(self.Read\_MFRC522(self.CRCResultRegL))

315.           # 将CRCResultRegM寄存器的值添加到输出列表中

316.           pOutData.append(self.Read\_MFRC522(self.CRCResultRegM))

317.           # 返回输出列表

318.           return pOutData

319.

320.    def MFRC522\_SelectTag(self, serNum):

321.        # 定义返回数据列表

322.        backData = []

323.        # 定义发送数据列表

324.        buf = []

325.        # 将PICC\_SElECTTAG命令添加到发送数据列表中

326.        buf.append(self.PICC\_SElECTTAG)

327.        # 将0x70添加到发送数据列表中

328.        buf.append(0x70)

329.        i = 0

330.        # 将标签序列号的5个字节添加到发送数据列表中

331.        while i < 5:

332.            buf.append(serNum[i])

333.            i = i + 1

334.        # 计算发送数据列表的CRC校验码

335.        pOut = self.CalulateCRC(buf)

336.        # 将CRC校验码的第一个字节添加到发送数据列表中

337.        buf.append(pOut[0])

338.        # 将CRC校验码的第二个字节添加到发送数据列表中

339.        buf.append(pOut[1])

340.        # 通过MFRC522\_ToCard方法向RC522模块发送发送数据列表，并获取响应状态、返回数据和返回数据长度

341.        (status, backData, backLen) = self.MFRC522\_ToCard(self.PCD\_TRANSCEIVE, buf)

342.        # 如果响应状态为MI\_OK，并且返回数据长度为0x18

343.        if (status == self.MI\_OK) and (backLen == 0x18):

344.        # 打印返回数据的第一个字节（表示标签容量）

345.            print("Size: ", str(backData[0]))

346.            # 返回返回数据的第一个字节

347.            return backData[0]

348.        else:

349.            # 否则返回0

350.            return 0

351.

352.    def MFRC522\_Auth(self, authMode, BlockAddr, Sectorkey, serNum):

353.            buff = []

354.

355.            # First byte should be the authMode (A or B)

356.            buff.append(authMode)

357.

358.            # Second byte is the trailerBlock (usually 7)

359.            buff.append(BlockAddr)

360.

361.            # Now we need to append the authKey which usually is 6 bytes of 0xFF

362.            i = 0

363.            while(i < len(Sectorkey)):

364.                buff.append(Sectorkey[i])

365.                i = i + 1

366.            i = 0

367.

368.            # Next we append the first 4 bytes of the UID

369.            while(i < 4):

370.                buff.append(serNum[i])

371.                i = i + 1

372.

373.            # Now we start the authentication itself

374.            (status, backData, backLen) = self.MFRC522\_ToCard(self.PCD\_AUTHENT, buff)

375.

376.            # Check if an error occurred

377.            if not(status == self.MI\_OK):

378.                print("AUTH ERROR!!")

379.            if not (self.Read\_MFRC522(self.Status2Reg) & 0x08) != 0:

380.                print("AUTH ERROR(status2reg & 0x08) != 0")

381.

382.            # Return the status

383.            return status

384.

385.    def MFRC522\_Read(self, blockAddr):

386.        # 定义接收数据列表

387.        recvData = []

388.        # 将PICC\_READ命令添加到接收数据列表中

389.        recvData.append(self.PICC\_READ)

390.        # 将数据块地址添加到接收数据列表中

391.        recvData.append(blockAddr)

392.        # 计算接收数据列表的CRC校验码

393.        pOut = self.CalulateCRC(recvData)

394.        # 将CRC校验码的第一个字节添加到接收数据列表中

395.        recvData.append(pOut[0])

396.        # 将CRC校验码的第二个字节添加到接收数据列表中

397.        recvData.append(pOut[1])

398.        (status, backData, backLen) = self.MFRC522\_ToCard(

399.            self.PCD\_TRANSCEIVE, recvData)

400.        # 通过MFRC522\_ToCard方法向RC522模块发送接收数据列表，并获取响应状态、返回数据和返回数据长度

401.        if not (status == self.MI\_OK):

402.            # 如果响应状态不为MI\_OK，则打印错误信息

403.            print("Error while reading!")

404.        i = 0

405.        # 如果返回的数据长度为16，则说明读取到了完整的数据块

406.        if len(backData) == 16:

407.        # 打印读取到的数据块信息

408.            print("Sector "+str(blockAddr)+" "+str(backData))

409.            # 返回读取到的数据块

410.            return backData

411.

412.

413.    def MFRC522\_Init(self):

414.        # 调用M500PcdConfigISOType方法，传入' A'参数，表示使用ISO14443\_A协议进行通信

415.        self.M500PcdConfigISOType('A')

416.        # 延时2毫秒

417.        time.sleep\_ms(2)

418.

419.    def M500PcdConfigISOType(self, ucType):

420.        # 如果传入的参数为' A'，表示使用ISO14443\_A协议进行通信

421.        if ucType == 'A':  # ISO14443\_A

422.            # 清除Status2Reg寄存器的8位

423.            self.ClearBitMask(self.Status2Reg, 0x08)

424.            # 将ModeReg寄存器的值设为0x3D（3F）

425.            self.Write\_MFRC522(self.ModeReg, 0x3D)  # 3F

426.            # 将RxSelReg寄存器的值设为0x86（84）

427.            self.Write\_MFRC522(self.RxSelReg, 0x86)  # 84

428.            # 将RFCfgReg寄存器的值设为0x7F（4F）

429.            self.Write\_MFRC522(self.RFCfgReg, 0x7F)  # 4F

430.            # 将TReloadRegL寄存器的值设为30

431.            self.Write\_MFRC522(self.TReloadRegL, 30)

432.            # 将TReloadRegH寄存器的值设为0

433.            self.Write\_MFRC522(self.TReloadRegH, 0)

434.            # 将TModeReg寄存器的值设为0x8D

435.            self.Write\_MFRC522(self.TModeReg, 0x8D)

436.            # 将TPrescalerReg寄存器的值设为0x3E

437.            self.Write\_MFRC522(self.TPrescalerReg, 0x3E)

438.        else:

439.            # 如果传入的参数不是' A'，则打印错误信息

440.            print("unk ISO type\r\n")

441.

442.

443.

444.

445.#=================================  FACE  =====================================

446.

447.# 加载模型

448.

449.#task\_fd = kpu.load(0x100000) # 从flash 0x100000 加载人脸检测模型

450.#task\_ld = kpu.load(0x200000) # 从flash 0x200000 加载人脸五点关键点检测模型

451.#task\_fe = kpu.load(0x300000) # 从flash 0x300000 加载人脸196维特征值模型

452.task\_fd = kpu.load("/sd/FaceDetection.smodel")

453.task\_ld = kpu.load("/sd/FaceLandmarkDetection.smodel")

454.task\_fe = kpu.load("/sd/FeatureExtraction.smodel")

455.

456.

457.#####################配置蓝牙串口#####################

458.fm.register(34, fm.fpioa.UART1\_TX, force=True)

459.fm.register(35, fm.fpioa.UART1\_RX, force=True)

460.

461.uart = UART(UART.UART1, 9600, 8, 1, 0, timeout=1000, read\_buf\_len=4096)

462.

463.#####################配置舵机####################

464.#PWM通过定时器配置

465.tim = Timer(Timer.TIMER0, Timer.CHANNEL0, mode=Timer.MODE\_PWM)

466.SS = PWM(tim, freq=50, duty=0, pin=21)

467.#'''

468.#说明：舵机控制函数

469.#功能：180度舵机：angle:-90至90 表示相应的角度

470.     #360连续旋转度舵机：angle:-90至90 旋转方向和速度值。

471.    #【duty】占空比值：0-100

472.#'''

473.def Servo(servo,angle):

474.    SS.duty((angle+90)/180\*10+2.5)

475.Servo(SS,0)

476.

477.g\_led = 12

478.r\_led = 13

479.fm.register(r\_led, fm.fpioa.GPIO0)     # 将引脚 r\_led 映射为了 GPIO0 的功能

480.fm.register(g\_led, fm.fpioa.GPIO1)

481.led\_r=GPIO(GPIO.GPIO0, GPIO.OUT)

482.led\_g=GPIO(GPIO.GPIO1, GPIO.OUT)

483.# 一开始两个灯都灭

484.led\_r.value(1)

485.led\_g.value(1)

486.

487.

488.#####################配置蜂鸣器#####################

489.beep\_pin=22

490.fpioa = FPIOA()

491.#PWM 通过定时器配置，接到 IO15 引脚，蜂鸣器

492.tim2 = Timer(Timer.TIMER0, Timer.CHANNEL1, mode=Timer.MODE\_PWM)

493.beep = PWM(tim2, freq=1, duty=50, pin=beep\_pin)

494.beep.disable()

495.#beep.enable()

496.#beep.freq(800)

497.

498.#####################配置人体红外传感#####################

499.fpioa.set\_function(23,FPIOA.GPIO7)

500.people\_find = GPIO(GPIO.GPIO7,GPIO.IN)

501.

502.lcd.init()

503.sensor.reset()

504.sensor.set\_pixformat(sensor.RGB565)

505.sensor.set\_framesize(sensor.QVGA)

506.sensor.set\_hmirror(1)

507.sensor.set\_vflip(1)

508.sensor.run(1)

509.anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437,

510.          6.92275, 6.718375, 9.01025)  # anchor for face detect

511.dst\_point = [(44, 59), (84, 59), (64, 82), (47, 105),

512.             (81, 105)]  # standard face key point position

513.a = kpu.init\_yolo2(task\_fd, 0.5, 0.3, 5, anchor)

514.img\_lcd = image.Image()

515.img\_face = image.Image(size=(128, 128))

516.a = img\_face.pix\_to\_ai()

517.record\_ftr = []

518.record\_ftrs = []

519.names = []

520.record\_ftrtemp=[]  #临时存储录入时的人脸特征

521.

522.

523.

524.

525.

526.################## 开机时读取SD卡中的人脸信息 ##################

527.i = 0

528.temp\_num = ''

529.with open("/sd/faceinfo.txt", "r") as f:

530.    while(1):

531.        lin = f.readline()

532.        if not lin:

533.            break

534.        stu\_num = lin[0:lin.index('#')]    #获取编号

535.        names.append(stu\_num)              #追加到编号列表

536.        faceftr = lin[lin.index('#')+1:]   #截取后半段的人脸特征

537.        record\_ftrs.append(eval(faceftr))  #向人脸特征列表中添加SD卡中的已存特征

538.        if stu\_num == temp\_num:

539.            continue

540.        temp\_num = stu\_num

541.        i = i + 1

542.        print("%d:%s#%s" % (i,stu\_num,faceftr))

543.    f.close()

544.

545.

546.#####################配置按键####################

547.LuRu\_mode = False   # 录入模式

548.Door\_mode = False   # 门禁模式

549.Normal\_mode = True  # 正常模式

550.

551.fpioa = FPIOA()

552.fm.register(16, fm.fpioa.GPIOHS0)

553.key\_gpio = GPIO(GPIO.GPIOHS0, GPIO.IN)

554.fm.register(24, fm.fpioa.GPIOHS1)

555.key\_gpio1 = GPIO(GPIO.GPIOHS1, GPIO.IN, GPIO.PULL\_UP)

556.fm.register(25, fm.fpioa.GPIOHS2)

557.key\_gpio2 = GPIO(GPIO.GPIOHS2, GPIO.IN, GPIO.PULL\_UP)

558.#fm.register(20, fm.fpioa.GPIOHS3)

559.#key\_gpio3 = GPIO(GPIO.GPIOHS3, GPIO.IN, GPIO.PULL\_UP)

560.

561.last\_key\_state=1

562.key\_pressed=0

563.last\_key\_state1=1

564.key\_pressed1=0

565.last\_key\_state2=1

566.key\_pressed2=0

567.#last\_key\_state3=1

568.#key\_pressed3=0

569.

570.def check\_key(): # 按键检测函数，用于在循环中检测按键是否按下，下降沿有效

571.    global last\_key\_state

572.    global key\_pressed

573.    global last\_key\_state1

574.    global key\_pressed1

575.    global last\_key\_state2

576.    global key\_pressed2

577.    global LuRu\_mode

578.    global Door\_mode

579.    global Normal\_mode

580.    #global last\_key\_state3

581.    #global key\_pressed3

582.    #global RFID\_mode

583.

584.    val=key\_gpio.value()

585.    val1=key\_gpio1.value()

586.    val2=key\_gpio2.value()

587.    #val3=key\_gpio3.value()

588.    if last\_key\_state == 1 and val == 0:

589.        LuRu\_mode = False

590.        Door\_mode = False

591.        Normal\_mode = True

592.        #RFID\_mode = False

593.        #lcd.clear()

594.    if last\_key\_state1 == 1 and val1 == 0:

595.        LuRu\_mode = True

596.        Door\_mode = False

597.        Normal\_mode = False

598.        #RFID\_mode = False

599.        #lcd.clear()

600.    if last\_key\_state2 == 1 and val2 == 0:

601.        LuRu\_mode = False

602.        Door\_mode = True

603.        Normal\_mode = False

604.        #RFID\_mode = False

605.        #lcd.clear()

606.    #if last\_key\_state3 == 1 and val3 == 0:

607.        #LuRu\_mode = False

608.        #Door\_mode = False

609.        #Normal\_mode = False

610.        #RFID\_mode = True

611.

612.    last\_key\_state = val

613.    last\_key\_state1 = val1

614.    last\_key\_state2 = val2

615.    #last\_key\_state3 = val3

616.

617.#####################配置看门狗#####################

618.def on\_wdt(self):

619.    self.feed()

620.wdt0 = WDT(id=1, timeout=5000, callback=on\_wdt, context={})

621.

622.###################################################################

623.##                            主循环                              ##

624.###################################################################

625.

626.# 设置各种flag

627.no\_people = 0 # 是否无人

628.no\_flag = 0

629.delline = [] # 将要删除的行数

630.check\_num = 0

631.shibie\_num = 0

632.uart.read()

633.

634.# 设置人脸对比置信度

635.ACCURACY = 75

636.# 录入模式flag

637.luru\_flag = 0

638.# 查找模式flag

639.find\_flag = 0

640.

641.feature = ''

642.max\_score = 0

643.index = 0

644.while (1):

645.    # 程序首先调用了check\_key()函数，该函数可能用于检测设备是否已经授权使用相关的模型。

646.    check\_key()

647.    # 调用了sensor.snapshot()函数获取一张图像。

648.    img = sensor.snapshot()

649.    # 使用kpu.run\_yolo2()函数对输入的图像进行人脸检测，返回一个包含检测结果的列表code。

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

651.    max\_score = 0

652.    # 如果code不为空，即检测到人脸，程序会遍历code中的每个检测结果，计算人脸框的面积，并选择面积最大的人脸进行处理。

653.    if code:

654.        t = 0

655.        max\_face = 0

656.        totalRes = len(code)

657.        area = []

658.        # 如果检测到的人脸数大于1，则只处理面积最大的人脸。

659.        if totalRes > 1:  #多张人脸时，挑选面积最大的，一般即最前面的

660.            for i in code: # 迭代坐标框  多张人脸

661.                area.append(i.w()\*i.h())

662.            for j in range(len(area)):

663.                if max\_face < area[j]:

664.                    max\_face = area[j]

665.                    t = j    #保存最大脸面积下标

666.            totalRes = 1

667.            del area

668.        if totalRes == 1:  #只有一张脸

669.            i = code[t]

670.            # 首先使用img.draw\_rectangle()函数在原图像上绘制人脸框

671.            # Cut face and resize to 128x128

672.            a = img.draw\_rectangle(i.rect())

673.            # 然后使用img.cut()函数对原图像进行裁剪，得到人脸图像。

674.            face\_cut = img.cut(i.x(), i.y(), i.w(), i.h())

675.            # 使用img.resize()函数将人脸图像缩放为128x128大小

676.            face\_cut\_128 = face\_cut.resize(128, 128)

677.            # 使用img.pix\_to\_ai()函数将图像转换为KPU能够处理的格式。

678.            a = face\_cut\_128.pix\_to\_ai()

679.            # a = img.draw\_image(face\_cut\_128, (0,0))

680.            # Landmark for face 5 points

681.            # 程序使用kpu.forward()函数对人脸图像进行特征点检测，得到人脸的五个关键点坐标。

682.            fmap = kpu.forward(task\_ld, face\_cut\_128)

683.            plist = fmap[:]

684.            le = (i.x() + int(plist[0] \* i.w() - 10), i.y() + int(plist[1] \* i.h()))

685.            re = (i.x() + int(plist[2] \* i.w()), i.y() + int(plist[3] \* i.h()))

686.            nose = (i.x() + int(plist[4] \* i.w()), i.y() + int(plist[5] \* i.h()))

687.            lm = (i.x() + int(plist[6] \* i.w()), i.y() + int(plist[7] \* i.h()))

688.            rm = (i.x() + int(plist[8] \* i.w()), i.y() + int(plist[9] \* i.h()))

689.            # 使用img.draw\_circle()函数在原图像上绘制出这些关键点的位置。

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

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

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

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

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

695.            # align face to standard position

696.            src\_point = [le, re, nose, lm, rm]

697.            # 程序使用这些关键点的坐标计算人脸的仿射变换矩阵，并使用image.get\_affine\_transform()函数计算变换矩阵。

698.            T = image.get\_affine\_transform(src\_point, dst\_point)

699.            # 使用image.warp\_affine\_ai()函数将人脸图像进行对齐，得到标准的人脸图像。

700.            a = image.warp\_affine\_ai(img, img\_face, T)

701.            # 然后，程序使用img.ai\_to\_pix()函数将对齐后的图像转换为原图像能够显示的格式。

702.            a = img\_face.ai\_to\_pix()

703.            # a = img.draw\_image(img\_face, (128,0))

704.            del (face\_cut\_128)

705.            # calculate face feature vector

706.            # 程序使用kpu.forward()函数对对齐后的人脸图像进行特征提取

707.            fmap = kpu.forward(task\_fe, img\_face)

708.            # 使用kpu.face\_encode()函数将特征向量进行编码

709.            feature = kpu.face\_encode(fmap[:])

710.            reg\_flag = False

711.            scores = []

712.            # 程序遍历预先录入的人脸特征列表

713.            for j in range(len(record\_ftrs)):

714.                # 使用kpu.face\_compare()函数计算当前人脸特征向量与列表中每个特征向量的相似度得分。

715.                score = kpu.face\_compare(record\_ftrs[j], feature)

716.                scores.append(score)

717.            max\_score = 0

718.            index = 0

719.            # 最终，程序选择得分最高的特征向量，并返回其在列表中的索引。这个索引可以用于识别当前人脸是否为已知人脸。

720.            for k in range(len(scores)):

721.                if max\_score < scores[k]:

722.                    max\_score = scores[k]

723.                    index = k

724.    # 执行录入任务

725.    if luru\_flag == 1:

726.        # 如果检测到当前画面中存在人脸，且识别率超过预设值

727.        if max\_score > ACCURACY:

728.            # 人脸采集次数清零

729.            check\_num = 0

730.            # 在屏幕上显示提示信息“Face Exist”

731.            a = img.draw\_string(200,0, b'Face Exist', color=(255,0,0),scale=1.6,mono\_space=1)  #提示人脸已存在

732.            # 在屏幕上显示图像

733.            a = lcd.display(img)

734.            # 继续循环，等待下一张图像的检测和处理

735.            continue

736.        # 如果检测到人脸特征

737.        if code:

738.            # 记录当前采集的人脸数量，每检测到一次人脸就加1

739.            check\_num = check\_num + 1     #检测到一次人脸则加 1

740.            # 在屏幕上显示当前采样进度

741.            a = img.draw\_string(5,40, b'%d'%check\_num, color=(0,255,0),scale=1.4,mono\_space=1) #显示采样进度

742.            # 间隔录入，每采集6次人脸特征就将当前特征加入到临时特征值列表中

743.            if check\_num % 6 == 0 and check\_num != 0:

744.                record\_ftrtemp.append(feature)   #加入当前人脸特征

745.            # 如果已采集到18次人脸特征，表示录入结束

746.            if check\_num == 18:

747.            # 录入成功标志位设为1

748.                save\_success = 1

749.                try:

750.                    # 尝试打开SD卡上的faceinfo.txt文件，并以追加模式写入数据

751.                    with open("/sd/faceinfo.txt", "a") as f:

752.                        for i in range(len(record\_ftrtemp)):  #循环遍历临时特征值列表，写入SD卡

753.                            f.write(stu\_num+'#'+str(record\_ftrtemp[i]))

754.                            # 每个特征值占一行

755.                            f.write("\n")

756.                        # 关闭文件

757.                        f.close()

758.                except Exception:

759.                    save\_success = 0  #表示保存到SD卡失败了

760.                    pass

761.                # 如果保存到SD卡成功，则将当前特征添加到已知特征列表中，并将编号添加到names列表中；同时清空临时特征值列表和采集次数，显示录入成功的提示信息，并进行蜂鸣器声音提示；最后退出录入任务并回到正常模式。

762.                if save\_success == 1:

763.                    #将当前特征添加到已知特征列表，编号添加到names

764.                    for i in record\_ftrtemp:

765.                        record\_ftrs.append(i)

766.                        names.append(stu\_num)

767.                    record\_ftrtemp.clear()  #清空临时特征值列表

768.                    # 在屏幕上显示录入成功的提示信息

769.                    a = img.draw\_string(0,5, b'Success!', color=(255,0,0),scale=1.4,mono\_space=1)  #录入成功

770.                    # 在屏幕上显示图像

771.                    a = lcd.display(img)

772.

773.                    #蜂鸣器声音提示

774.                    beep.enable()

775.                    beep.freq(1000)

776.                    time.sleep(2)

777.                    beep.disable()

778.                    luru\_flag = 0  # 退出录入任务

779.                    # 回到正常模式

780.                    LuRu\_mode = False   # 录入模式

781.                    Door\_mode = False   # 门禁模式

782.                    Normal\_mode = True  # 正常模式

783.                #如果保存到SD卡失败，则按键次数清零，check\_num清零，编号清空

784.                else:

785.                    a = img.draw\_string(0,5, b'Fail!', color=(255,0,0),scale=1.4,mono\_space=1)  #录入失败

786.                    #蜂鸣器声音提示

787.                    beep.enable()

788.                    beep.freq(600)

789.                    time.sleep(2)

790.                    beep.disable()

791.                    luru\_flag = 0  # 退出录入任务

792.                    # 回到正常模式

793.                    LuRu\_mode = False   # 录入模式

794.                    Door\_mode = False   # 门禁模式

795.                    Normal\_mode = True  # 正常模式

796.        # 删除code变量，释放内存空间。

797.        del code

798.

799.    # 有人的时候LCD显示摄像头的拍摄画面

800.    if people\_find.value() == 1:

801.        no\_flag = 1

802.

803.

804.

805.        # 接收蓝牙数据

806.        # 垃圾回收

807.        gc.collect()

808.        # 读取蓝牙数据

809.        text = uart.read()

810.        if text != None and len(text) >= 2: #如果读取到了数据，且大于等于2

811.            # 临时开门禁

812.            # 如果蓝牙数据中包含 'open'

813.            if 'open' in text:

814.                print("--------蓝牙开门--------")

815.                # 舵机旋转

816.                Servo(SS,90)

817.                time.sleep(2)

818.                Servo(SS,0)

819.

820.            # 删除全部用户命令

821.            # 如果蓝牙数据中包含 'erase'

822.            if 'erase' in text:

823.                print("--------删除全部用户--------")

824.                # 清空名称列表、特征值列表和人脸信息文件

825.                names.clear()

826.                record\_ftrs.clear()

827.                # 打开文件

828.                file\_new = open('/sd/faceinfo.txt', 'w')

829.                # 写入空字符串

830.                file\_new.write(''.join(''))

831.                # 关闭文件

832.                file\_new.close()

833.                # 清除图像

834.                img.clear()

835.                # 绘制矩形

836.                img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

837.                # 绘制字符串

838.                img.draw\_string(110, 112, "delete all", color=(255, 255, 255), scale=1.5, mono\_space=0)

839.                # 在LCD屏幕上显示图像

840.                lcd.display(img)

841.                # 蜂鸣器声音提示

842.                beep.enable()

843.                beep.freq(1000)

844.                time.sleep(2)

845.                beep.disable()

846.                a = img.clear()

847.

848.            # 删除用户命令

849.            # 如果蓝牙数据中包含 'delete'

850.            if 'delete' in text:

851.                print("--------删除用户--------")

852.                delnum = text[3:].decode('utf-8')  #截取出删除的编号

853.                print('删除的编号: ', delnum)

854.                print(delnum)

855.                del text

856.                k=-1

857.                # 打开人脸信息文件

858.                with open('/sd/faceinfo.txt', 'r') as f:

859.                    while(1):

860.                    # 逐行读取文件内容

861.                        thisline=f.readline()

862.                        if not thisline:

863.                            # 如果文件已读完

864.                            break

865.                        # 计数器加1

866.                        k=k+1

867.                        # 判断是否找到要删除的编号

868.                        result = delnum in thisline

869.                        # 如果找到了第一个匹配项

870.                        if result==True and find\_flag == 0:

871.                            # 标记为找到

872.                            find\_flag=1

873.                            print('ok')

874.                        # 如果找到了多个匹配项

875.                        if result==True and find\_flag == 1:

876.                            # 记录要删除的行数

877.                            delline.append(k)

878.                        # 如果匹配项已结束

879.                        if result==False and find\_flag == 1:

880.                            # 标记为未找到

881.                            find\_flag = 0

882.                            break

883.                    # 关闭文件

884.                    f.close()

885.                # 如果SD卡中有此人，那么就删除，

886.                if delline:

887.                    # 按行读入，删除最后一行

888.                    # 打开文件

889.                    file\_old = open('/sd/faceinfo.txt', 'r')

890.                    # 逐行读取文件内容

891.                    lines = [i for i in file\_old]

892.                    # 删除指定行数

893.                    del lines[int(delline[0]):int(delline[-1])+1]

894.                    # 关闭文件

895.                    file\_old.close()

896.                    del file\_old

897.                    # 清空要删除的行数列表

898.                    delline.clear()

899.                    # 再覆盖写入

900.                    # 打开文件

901.                    file\_new = open('/sd/faceinfo.txt', 'w')

902.                    # 将修改后的内容写入文件

903.                    file\_new.write(''.join(lines))

904.                    # 关闭文件

905.                    file\_new.close()

906.                    del lines

907.                    ##删除姓名列表和特征值列表中的数据

908.                    #print('之前',names)

909.                    i = 0

910.                    # 清空名称列表

911.                    temp\_num = ''

912.                    names = []

913.                    # 清空特征值列表

914.                    record\_ftrs = []

915.                    # 垃圾回收

916.                    gc.collect()

917.                    try:

918.                        # 打开人脸信息文件

919.                        with open("/sd/faceinfo.txt", "r") as f:

920.                            while(1):

921.                                # 垃圾回收

922.                                gc.collect()

923.                                # 逐行读取文件内容

924.                                lin = f.readline()

925.                                # 如果文件已读完

926.                                if not lin:

927.                                    break

928.                                stunum = lin[0:lin.index('#')]    #获取编号

929.                                lin = lin[lin.index('#')+1:]      #截取除了编号以外的字符串

930.                                stu\_name = lin[0:lin.index('#')]  #获取姓名

931.                                names.append(stunum+'#'+stu\_name) #追加到姓名列表

932.                                lin = lin[lin.index('#')+1:]      #截取人脸特征

933.                                record\_ftrs.append(eval(lin))     #向人脸特征列表中添加已存特征

934.                    except:

935.                        pass

936.                    # 垃圾回收

937.                    gc.collect()

938.                    # 清除图像

939.                    img.clear()

940.                    # 绘制矩形

941.                    img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

942.                    # 绘制字符串

943.                    img.draw\_string(110, 112, "delete %s"%delnum, color=(255, 255, 255), scale=1.5, mono\_space=0)

944.                    # 在LCD屏幕上显示图像

945.                    lcd.display(img)

946.                    #蜂鸣器声音提示

947.                    beep.enable()

948.                    beep.freq(1000)

949.                    time.sleep(2)

950.                    beep.disable()

951.                    a = img.clear()

952.

953.                    # 回到正常模式

954.                    LuRu\_mode = False   # 录入模式

955.                    Door\_mode = False   # 门禁模式

956.                    Normal\_mode = True  # 正常模式

957.                else:

958.                    a = img.clear()

959.                    # 绘制矩形

960.                    a = img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

961.                    # 绘制字符串

962.                    a = img.draw\_string(118, 112, "No people!", color=(255, 255, 255), scale=1.5, mono\_space=0)

963.                    # 在LCD屏幕上显示图像

964.                    a = lcd.display(img)

965.                    #蜂鸣器声音提示

966.                    beep.enable()

967.                    beep.freq(600)

968.                    time.sleep(2)

969.                    beep.disable()

970.                    a = img.clear()

971.                    # 回到正常模式

972.                    LuRu\_mode = False   # 录入模式

973.                    Door\_mode = False   # 门禁模式

974.                    Normal\_mode = True  # 正常模式

975.

976.        # 检测按键

977.        # 录入模式按键

978.        if LuRu\_mode:

979.            try:

980.                # 在屏幕上显示“录入模式”字样

981.                a = img.draw\_string(0,0, b'Register Mode', color=(0,255,0),scale=1.6,mono\_space=1)

982.            except:

983.                pass

984.            # 判断蓝牙数据是否读取到并且长度大于等于2

985.            if text != None and len(text) >= 2: #如果读取到了数据，且大于等于2

986.                # 判断是否收到“register”命令，如果是则进行人脸注册

987.                if 'register' in text:

988.                    print("--------人脸注册--------")

989.                    stu\_num = text[2:].decode('utf-8')  # 截取出编号

990.                    print('编号: ', stu\_num)

991.

992.                    # 判断该编号是否已被录入

993.                    # 如果该编号已经被录入，显示“ID Exist!”字样，并在屏幕上显示红色矩形区域

994.                    if stu\_num in names:

995.                        a = img.clear()

996.                        a = img.draw\_rectangle((90, 85, 140, 70), fill=True, color=(0, 0, 255))

997.                        a = img.draw\_string(118, 112, "ID Exist!", color=(255, 255, 255), scale=1.5, mono\_space=0)

998.                        a = lcd.display(img)

999.                        # 蜂鸣器声音提示

1000.                        beep.enable()

1001.                        beep.freq(600)

1002.                        time.sleep(2)

1003.                        beep.disable()

1004.                        # 回到正常模式

1005.                        LuRu\_mode = False   # 录入模式

1006.                        Door\_mode = False   # 门禁模式

1007.                        Normal\_mode = True  # 正常模式

1008.                    # 如果编号没有被录入，进行下一步录入

1009.                    else:

1010.                        # 设置录入标志位为1

1011.                        luru\_flag = 1

1012.

1013.        # 门禁模式按键

1014.        if Door\_mode:

1015.        # 录入标志位设置为0

1016.            luru\_flag = 0

1017.            # 校验次数设置为0

1018.            check\_num = 0

1019.            # 在屏幕上显示“门禁模式”字样

1020.            a = img.draw\_string(0,0, b'Entrance Mode', color=(0,255,0),scale=1.6,mono\_space=1)

1021.            # ================================== RFID ================================

1022.

1023.            # time.sleep(2)

1024.            # from micropython import const

1025.            ################### config ###################

1026.            CS\_NUM = const(18)

1027.            SPI\_FREQ\_KHZ = const(600)

1028.            SPI\_SCK = const(19)

1029.            SPI\_MOSI = const(8)

1030.            SPI\_MISO = const(15)

1031.

1032.            #############################################

1033.

1034.            #continue\_reading = True

1035.

1036.            # 20: CS\_NUM;

1037.            fm.register(CS\_NUM, fm.fpioa.GPIOHS20, force=True)

1038.

1039.            # set gpiohs work mode to output mode

1040.            cs = GPIO(GPIO.GPIOHS20, GPIO.OUT)

1041.

1042.            spi1 = SPI(SPI.SPI\_SOFT, mode=SPI.MODE\_MASTER, baudrate=SPI\_FREQ\_KHZ \* 1000,

1043.                    polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck=SPI\_SCK, mosi=SPI\_MOSI, miso=SPI\_MISO)

1044.

1045.            # Create an object of the class MFRC522

1046.            MIFAREReader = MFRC522(spi1, cs)

1047.

1048.            # time.sleep\_ms(300)

1049.            # Scan for cards

1050.            (status, ataq) = MIFAREReader.MFRC522\_Request(MIFAREReader.PICC\_REQALL)

1051.

1052.            # If a card is found

1053.            if status == MIFAREReader.MI\_OK:

1054.                # 如果卡片被找到，打印卡片类型和UID

1055.                print("Card detected type: ",hex(ataq[0]<<8|ataq[1]))

1056.                # Get the UID of the card

1057.                (status, uid) = MIFAREReader.MFRC522\_Anticoll()

1058.

1059.                # If we have the UID, continue

1060.                if status == MIFAREReader.MI\_OK:

1061.

1062.                    # Print UID

1063.                    print("Card read UID: " +

1064.                        str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3]))

1065.

1066.                    # This is the default key of M1(S50) for authentication

1067.                    # M1卡片的默认密钥

1068.                    key = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]

1069.

1070.                    # Select the scanned tag

1071.                    # 选择扫描到的标签

1072.                    MIFAREReader.MFRC522\_SelectTag(uid)

1073.

1074.                    # Authenticate

1075.                    # 验证卡片密钥

1076.                    status = MIFAREReader.MFRC522\_Auth(

1077.                        MIFAREReader.PICC\_AUTHENT1A, 0x12, key, uid)

1078.

1079.                    # Check if authenticated

1080.                    # 检查是否验证成功

1081.                    #if status == MIFAREReader.MI\_OK:

1082.                        # 示例：定义一个包含16个元素的列表，并将第一个元素设为'Y'

1083.                        ##data = ['Y',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

1084.                        ##data = ['N',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

1085.                        ## Fill the data with 0~16

1086.                        #for x in range(0, 16):

1087.                            #data.append(x)

1088.

1089.                        ## Write the data

1090.                        # 向扇区0x12写入数据

1091.                        #print("Sector 11 will now be filled with 1~16:")

1092.                        # 使用MFRC522\_Write方法将data列表写入扇区0x12

1093.                        #status = MIFAREReader.MFRC522\_Write(0x12, data)

1094.                    # 如果验证成功，读取RFID模块中的数据

1095.                    try:

1096.                        if status == MIFAREReader.MI\_OK:

1097.                            print("start to read")

1098.                            # read the data

1099.                            # MIFAREReader.MFRC522\_Read(0x12)

1100.                            datas = MIFAREReader.MFRC522\_Read(0x12)

1101.                            #读出卡中的id号

1102.                            strnum = ' '

1103.                            strnum = chr(datas[0])+str(datas[1])+str(datas[2])+str(datas[3])+str(datas[4])

1104.                            #如果第一个字符不是B则说明卡错误

1105.                            if chr(datas[0]) == 'Y':

1106.                                led\_g.value(0)  # 绿灯亮

1107.                                Servo(SS,90)

1108.                                time.sleep(2)

1109.                                Servo(SS,0)

1110.                                led\_g.value(1)  # 绿灯灭

1111.

1112.                            if chr(datas[0]) == 'N':

1113.                                #蜂鸣器声音提示

1114.                                beep.enable()

1115.                                beep.freq(300)

1116.                                time.sleep(2)

1117.                                beep.disable()

1118.                                ##读出卡中的姓名

1119.                                #if datas[11]==0: #名字两个字

1120.                                    #strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+"'"

1121.                                #else :          #名字三个字

1122.                                    #strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+hex(datas[11])+hex(datas[12])+hex(datas[13])+"'"

1123.                                #strname = eval(strname.replace('0','\\')).decode('utf8')

1124.

1125.                            # Stop

1126.                            MIFAREReader.MFRC522\_StopCrypto1()

1127.                        #else:

1128.                            #print("Authentication error")

1129.                    except Exception as e:

1130.                         beep.enable()

1131.                         beep.freq(1000)

1132.                         time.sleep(2)

1133.                         beep.disable()

1134.

1135.

1136.            try:

1137.                # 如果人脸识别分数高于设定的阈值，显示编号与分数，并开启绿灯

1138.                if max\_score > ACCURACY:

1139.                    a = img.draw\_string(i.x()+4,i.y()-20, ("%s: %2.1f" % (names[index], max\_score)), color=(0,255,0),scale=2) # 显示编号与分数

1140.                    a = lcd.display(img)

1141.                    print("--------人脸识别成功--------")

1142.                    led\_g.value(0)  # 绿灯亮

1143.                    Servo(SS,90)

1144.                    time.sleep(2)

1145.                    Servo(SS,0)

1146.                    led\_g.value(1)  # 绿灯灭

1147.                    shibie\_num = 0

1148.                elif code:

1149.                    # 如果人脸识别分数低于设定的阈值，显示分数，并计数，如果计数超过3次则表示人脸识别失败，开启红灯并发出蜂鸣器声音

1150.                    img.draw\_string(i.x()+4,i.y()-20, ("%s: %2.1f" % ("No",max\_score)), color=(0,255,0),scale=2) # 显示分数

1151.                    a = lcd.display(img)

1152.                    shibie\_num = shibie\_num + 1

1153.                    if shibie\_num > 3:

1154.                        shibie\_num = 0

1155.                        print("--------人脸识别失败--------")

1156.                        led\_r.value(0)  # 红灯亮

1157.                        beep.enable()

1158.                        beep.freq(600)

1159.                        time.sleep(2)

1160.                        beep.disable()

1161.                        led\_r.value(1)  # 红灯灭

1162.            except:

1163.                pass

1164.                # 删除变量code，释放内存空间

1165.            del code

1166.

1167.        # 正常模式按键

1168.        # 设置录入标志和检查数字为0，使用LCD显示屏显示“等待……”

1169.        if Normal\_mode:

1170.            luru\_flag = 0

1171.            check\_num = 0

1172.            a = img.draw\_string(0,0, b'Waiting......', color=(0,255,0),scale=1.6,mono\_space=1)

1173.            ## ================================== RFID ================================

1174.

1175.            ## time.sleep(2)

1176.            ## from micropython import const

1177.            #################### config ###################

1178.            #CS\_NUM = const(18)

1179.            #SPI\_FREQ\_KHZ = const(600)

1180.            #SPI\_SCK = const(19)

1181.            #SPI\_MOSI = const(8)

1182.            #SPI\_MISO = const(15)

1183.

1184.            ##############################################

1185.

1186.            ##continue\_reading = True

1187.

1188.            ## 20: CS\_NUM;

1189.            #fm.register(CS\_NUM, fm.fpioa.GPIOHS20, force=True)

1190.

1191.            ## set gpiohs work mode to output mode

1192.            #cs = GPIO(GPIO.GPIOHS20, GPIO.OUT)

1193.

1194.            #spi1 = SPI(SPI.SPI\_SOFT, mode=SPI.MODE\_MASTER, baudrate=SPI\_FREQ\_KHZ \* 1000,

1195.                    #polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck=SPI\_SCK, mosi=SPI\_MOSI, miso=SPI\_MISO)

1196.

1197.            ## Create an object of the class MFRC522

1198.            #MIFAREReader = MFRC522(spi1, cs)

1199.

1200.            ## time.sleep\_ms(300)

1201.            ## Scan for cards

1202.            #(status, ataq) = MIFAREReader.MFRC522\_Request(MIFAREReader.PICC\_REQALL)

1203.

1204.            ## If a card is found

1205.            #if status == MIFAREReader.MI\_OK:

1206.                #print("Card detected type: ",hex(ataq[0]<<8|ataq[1]))

1207.                ## Get the UID of the card

1208.                #(status, uid) = MIFAREReader.MFRC522\_Anticoll()

1209.

1210.                ## If we have the UID, continue

1211.                #if status == MIFAREReader.MI\_OK:

1212.

1213.                    ## Print UID

1214.                    #print("Card read UID: " +

1215.                        #str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3]))

1216.

1217.                    ## This is the default key of M1(S50) for authentication

1218.                    #key = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]

1219.

1220.                    ## Select the scanned tag

1221.                    #MIFAREReader.MFRC522\_SelectTag(uid)

1222.

1223.                    ## Authenticate

1224.                    #status = MIFAREReader.MFRC522\_Auth(

1225.                        #MIFAREReader.PICC\_AUTHENT1A, 0x12, key, uid)

1226.

1227.                    ## Check if authenticated

1228.                    ##if status == MIFAREReader.MI\_OK:

1229.                        ###data = ['Y',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

1230.                        ###data = ['N',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

1231.                        ### Fill the data with 0~16

1232.                        ##for x in range(0, 16):

1233.                            ##data.append(x)

1234.

1235.                        ### Write the data

1236.                        ##print("Sector 11 will now be filled with 1~16:")

1237.                        ##status = MIFAREReader.MFRC522\_Write(0x12, data)

1238.                    #try:

1239.                        #if status == MIFAREReader.MI\_OK:

1240.                            #print("start to read")

1241.                            ## read the data

1242.                            ## MIFAREReader.MFRC522\_Read(0x12)

1243.                            #datas = MIFAREReader.MFRC522\_Read(0x12)

1244.                            ##读出卡中的id号

1245.                            #strnum = ' '

1246.                            #strnum = chr(datas[0])+str(datas[1])+str(datas[2])+str(datas[3])+str(datas[4])

1247.                            ##如果第一个字符不是B则说明卡错误

1248.                            #if chr(datas[0]) == 'Y':

1249.                                #led\_g.value(0)  # 绿灯亮

1250.                                #Servo(SS,90)

1251.                                #time.sleep(2)

1252.                                #Servo(SS,0)

1253.                                #led\_g.value(1)  # 绿灯灭

1254.

1255.                            #if chr(datas[0]) == 'N':

1256.                                ##蜂鸣器声音提示

1257.                                #beep.enable()

1258.                                #beep.freq(300)

1259.                                #time.sleep(2)

1260.                                #beep.disable()

1261.                                ###读出卡中的姓名

1262.                                ##if datas[11]==0: #名字两个字

1263.                                    ##strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+"'"

1264.                                ##else :          #名字三个字

1265.                                    ##strname = "b'"+hex(datas[5])+hex(datas[6])+hex(datas[7])+hex(datas[8])+hex(datas[9])+hex(datas[10])+hex(datas[11])+hex(datas[12])+hex(datas[13])+"'"

1266.                                ##strname = eval(strname.replace('0','\\')).decode('utf8')

1267.

1268.                            ## Stop

1269.                            #MIFAREReader.MFRC522\_StopCrypto1()

1270.                        ##else:

1271.                            ##print("Authentication error")

1272.                    #except Exception as e:

1273.                         #beep.enable()

1274.                         #beep.freq(1000)

1275.                         #time.sleep(2)

1276.                         #beep.disable()

1277.

1278.        a = lcd.display(img)

1279.

1280.    # 无人的时候LCD黑屏，休眠状态（录入的时候不能进入休眠状态）

1281.    # 如果没有人在操作，将LCD清空并显示“休眠中……”，并将标志位no\_people设置为1。

1282.    elif luru\_flag == 0:

1283.        no\_people = 1

1284.    # 如果有人在操作，则将标志位no\_people和no\_flag都置为0，表示有人正在操作，不进行休眠。

1285.    if no\_people == 1 and no\_flag == 1:

1286.        no\_people = 0

1287.        no\_flag = 0

1288.        a = img.clear()   # 清空一次LCD

1289.        # 最后，将img对象在LCD上显示出来，并回收内存。

1290.        a = img.draw\_string(40,110, b'Suspending......', color=(0,255,0),scale=2,mono\_space=1)

1291.        a = lcd.display(img)

1292.    # del img用于删除img对象占用的内存

1293.    del img

1294.    # gc.collect()用于回收未被使用的内存。

1295.    gc.collect()

1296.

1297.

1298.# a = kpu.deinit(task\_fe)

1299.# a = kpu.deinit(task\_ld)

1300.# a = kpu.deinit(task\_fd)

9.2 代码流程图



图4-3 代码流程图