西安邮电大学
毕业设计（论文）

题目：基于树莓派的人脸识别系统的开发与应用

学院：自动化学院

专业：自动化

班级：自动 1403 班

学生姓名：彭大富

学号：06141083

导师姓名：何鹏举/马翔 职称：教授/助理工程师

起止时间：2017 年 12 月 5 日 至 2018 年 6 月 10 日

毕业设计（论文）声明书

本人所提交的毕业论文《基于树莓派的人脸识别系统的开发与应用》是本人在指导教师指导下独立研究、写作的成果，论文中所引用他人的文献、数据、图件、资料均已明确标注；对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明并表示感谢。

本人完全理解《西安邮电大学本科毕业设计（论文）管理办法》的各项规定并自愿遵守。

本人深知本声明书的法律责任，违规后果由本人承担。

论文作者签名：

日期：　　年　月　日

西安邮电大学本科毕业设计(论文)选题审批表

<table>
<tr><td>申 报 人</td><td>何鹏举
/马翔</td><td>职　　称</td><td>教授/助理工程师</td><td>学　　院</td><td colspan="3">自动化学院</td></tr>
<tr><td>题目名称</td><td colspan="7">基于树莓派的人脸识别系统的开发与应用</td></tr>
<tr><td>题目来源</td><td>科研</td><td colspan="2"></td><td>教学</td><td></td><td>其它</td><td>√</td></tr>
<tr><td>题目类型</td><td>硬件设计</td><td>√</td><td>软件设计</td><td></td><td>论文</td><td></td><td>艺术作品</td><td></td></tr>
<tr><td>题目性质</td><td colspan="2">应用研究</td><td colspan="2">√</td><td colspan="2">理论研究</td><td></td></tr>
<tr><td>题目简述</td><td colspan="7">当前考勤方式存在代打卡、不能及时统计考勤数据和考勤参数设置不够灵活等问题.因此基于树莓派实现人脸识别技术，不但能够降低硬件成本,并且能将识别信息与门禁或考勤等系统相结合,实现人员智能化管理。</td></tr>
<tr><td>对学生知识与能力要求</td><td colspan="7">掌握微机原理与接口技术及单片机原理及应用,理解嵌入式系统的基本组织结构与工作原理，具有一定的模拟电路技术基础及网络通信原理,熟悉常用传感器原理及使用方法,具有嵌入式软件设计能力及解决实际问题的动手能力。</td></tr>
<tr><td>具体任务以及预期目标</td><td colspan="7">设计基于树莓派的人脸识别系统， 通过摄像头抓取人脸图像，通过 OpenCV 或云服务实现人脸识别，按照系统逻辑，实现考勤， 门禁或与其他系统进行联动。</td></tr>
<tr><td>时间进度</td><td colspan="7">2017 年 12 月 05 日—2017 年 12 月 10 日 选取毕设题目
2017 年 12 月 11 日—2018 年 01 月 06 日 查阅资料，撰写提交开题报告
2018 年 01 月 07 日—2018 年 03 月 04 日 确定系统架构设计方案
2018 年 03 月 05 日—2018 年 03 月 31 日 系统软件程序设计
2018 年 04 月 01 日—2018 年 04 月 15 日 软硬件的联合调试
2018 年 04 月 16 日—2018 年 04 月 31 日 系统功能完善
2018 年 05 月 01 日—2018 年 05 月 25 日 撰写毕业设计论文
2018 年 05 月 26 日—2018 年 06 月 01 日 修改、装订论文
2018 年 06 月 02 日—2018 年 06 月 10 日 准备毕业答辩</td></tr>
<tr><td>系（教研室）主任签字</td><td colspan="3">年　月　日</td><td>主管院长签字</td><td colspan="3">年　月　日</td></tr>
</table>

西安邮电大学本科毕业设计（论文）开题报告

学生姓名	彭大富	学号	06141083	专业班级	自动 1403 班
指导教师	何鹏举/马翔	题目	基于树莓派的人脸识别系统的开发与应用		

选题目的（为什么选该课题）

人工智能（AI）是未来发展的趋势，而 AI 又分好多种类，比如机器学习，计算机视觉，等等。自动化专业的我对这方面也稍有涉及，也想要更加深入的去了解一下这个方面的东西。

前期基础（已学课程、掌握的工具，资料积累、软硬件条件等）

1、网络基础：了解网络通信的有关协议
2、Python：学习 AI 的主流编程语言
3、OpenCV：进行人脸检测
4、腾讯优图开发 SDK（Python-SDK）：提供人脸识别
5、树莓派：开发系统（软件环境、硬件设备）

要研究和解决的问题（做什么）

通过使用树莓派采集人脸照片，进行刷脸签到，记录在考勤系统中，Web 端使用 PC 端浏览器调用摄像头刷脸登录查看自身考勤记录。

工作思路和方案（怎么做）

1. 树莓派安装 usb 摄像头
2. 控制树莓派使用 usb 摄像头拍照，并使用文件 IO 将照片保存
3. 使用 OpenCV 检测出人脸，将人脸照片重新设置尺寸，仅保存人脸照片，命名 face.jpg
4. 将 face.jpg 通过优图 Python-SDK 传至人脸库进行人脸预测
5. 预测结果进行阈值筛选，签到成功
6. 签到记录使用 Web 端展示

指导教师意见

签字：　　年　月　日

西安邮电大学毕业设计（论文）成绩评定表

<table>
<tr><td>学生姓名</td><td>彭大富</td><td>性别</td><td>男</td><td>学号</td><td>06141083</td><td>专业
班级</td><td>自动 1403 班</td></tr>
<tr><td>课题名称</td><td colspan="7">基于树莓派的人脸识别系统的开发与应用</td></tr>
<tr><td>指导
教师
意见</td><td colspan="7">评分（百分制）：　　指导教师(签字)：_____　　年　月　日</td></tr>
<tr><td>评阅
教师
意见</td><td colspan="7">评分（百分制）：　　评阅教师(签字)：_____　　年　月　日</td></tr>
<tr><td>验收
小组
意见</td><td colspan="7">评分（百分制）：　　验收教师(签字)：_____　　年　月　日</td></tr>
<tr><td>答辩
小组
意见</td><td colspan="7">评分（百分制）：　　答辩小组组长(签字)：_____　　年　月　日</td></tr>
<tr><td>评分比例</td><td colspan="7">指导教师评分　(%) 评阅教师评分　(%)　验收小组评分　(%)　答辩小组评分　(%)</td></tr>
<tr><td>学生总评
成绩</td><td colspan="2">百分制成绩</td><td></td><td colspan="2">等级制成绩</td><td colspan="2"></td></tr>
<tr><td>答辩委员
会意见</td><td colspan="7">毕业论文(设计)最终成绩(等级)：

学院答辩委员会主任(签字、学院盖章)：　　年　月　日</td></tr>
</table>

目　录

摘　要

众所周知，无论人们在什么企业上班，不管是弹性上班制还是按时按点上班制度，都是需要进行签到的；无论学生们在什么学校上课，老师也是会在课堂上进行点名。因此，本课题设计基于树莓派的人脸识别考勤系统，拟提供多元考勤方式中一种较为新颖的方式。

系统是基于树莓派开发，分为 Web 部分和硬件部分。Web 部分采用了当前较为流行的 Python 语言，并使用其 Web 框架 Django，完成了一个考勤记录的查看、补签、审批等。硬件部分使用 OpenCV 进行配合树莓派的摄像头进行人脸检测，并把一帧一帧的图片中人脸部分画出一个框，将人脸包裹，当按键按下时，OpenCV 将当前帧的图片保存，后台用腾讯优图的 Python-SDK 库将人脸识别，返回人脸信息，然后进行签到逻辑。

本课题是基于树莓派的人脸识别签到系统，在人们日常生活中具有很强的实用性。由于树莓派较为轻便，从而使得整个系统的使用方便快捷。

关键字： 人脸识别；树莓派；考勤系统

ABSTRACT

As we all know, no matter what companies are working in, whether they are working on flexible schedules or on time, they all need to sign in. No matter what school students are in, the teachers will also be named in class. Therefore, this project is designed based on the Raspberry Pi face recognition attendance system, which is intended to provide a more novel way of multi-attendance.

The system is based on the Raspberry Pi development board and is divided into the Web section and the hardware section. The Web part adopts the current popular python language and uses its Web framework Django to complete the review, supplement, and approval of an attendance record. The hardware uses OpenCV to cooperate with the Raspberry Pi camera for face detection and draws a frame of the face part of the frame-by-frame picture to wrap the face. When the button is pressed, OpenCV will take a picture of the current frame. Save, background recognition using Tencent python-SDK library to face recognition, return face information, and then check-in logic.

Based on the Raspberry Pi-based face recognition sign-in system, because of the lightness of the Raspberry Pi, the entire system is easy to use and has great practicality.

Key words： Face recognition; Raspberry Pi; Attendance system

第一章　绪论

1.1 课题背景

人们总能遇到这样的窘境：当你去上班的时候你才发现你忘记带工卡，或者你的工卡突然刷不，那么你将无法签到，同时还要面对上级的询问；当你上课你没有带一卡通，或者你的一卡通掉了还没有补办，那你不得不去找老师，说明你的情况。那么，遇到这种情况应该寻求一个怎样的解决方案呢？如果人们能什么都不带，仅适用自己的特征来代替不就好了？基于树莓派的人脸识别考勤系统就是基于这个出发点而提出的课题，本课题的根本出发点就是为了解决当人们不依靠工卡等其它卡片设备进行考勤时，可以使用人脸识别来签到，这样的话，就算人们忘记带卡，也可以很方便的进行签到。

基于树莓派设计的人脸识别签到系统，主要是使用 Python 去配合 OpenCV 和腾讯优图子公司的人脸识别开发库 Python-SDK 实现的，做到就算没有带一卡通之类的信息证明卡片也能进行正常签到。那么，要实现这么一个功能，课题便要求要解决人脸识别、签到逻辑、签到查询和查看、补签、审批等问题。本课题采用的主控板是树莓派开发板，树莓派不仅是一个 Linux 服务器，同时树莓派也提供了实用的 GPIO 编程功能，而且还具有高性能、低功耗的特点，这样的话 Web 服务可以直接部署在树莓派中，硬件方面的东西也可以直接由树莓派控制。

本课题在分析各种考勤的情况下，主要解决忘记带卡、代打卡、考勤参数设置不灵活的问题。

1.2 课题任务

本课题主要是研究基于树莓派的人脸识别考勤系统，主要设计实现人脸识别和考勤相关的内容。设计使用 OpenCV 计算机视觉函数库，配合树莓派摄像头进行视屏流的采集分析，检测是否有人脸存在。当用户按下按键的使用，将用户的人脸传给 Python-SDK 进行识别，得到识别结果进行签到逻辑判断。

本课题的设计增加提示模块：当签到成功时亮一个绿灯，当签到失败的时候亮一个红灯。

为解决想要查看考勤记录的情况，需要有一个网站能查看使用者的考勤记录，网站上理应还有补签的功能，这样就不用手写说明书。也要设计审批的功能，当用户补签时，可选择通过或者拒绝。所有的 Web 端都是由 Python 语言和 Django 框架完成后台逻辑，页面设计由 HTML 以及 CSS 实现。

1.3 前人成果-研究现状

计算机技术用于识别一个人使用其特有的标识，比如人脸信息，识别过程的过程就叫做人脸识别。人脸识别在计算机技术研究领域广为人知，并且涉及生物

识别技术。生物体的这种生物学特性（通常特定于人）是识别生物体重要特性。目前人脸识别技术已经相当成熟，并且有许多相关的 SDK 来帮助不涉及人脸识别算法学习的开发者来快速的使用与人脸识别相关的服务。

1.4 论文结构

本论文各章节安排如下：

第一章：介绍各课题的背景；

第二章：介绍整个系统中所涉及的技术；

第三章：简述所有的功能设计方案与架构；

第四章：介绍本系统中设计的数据库；

第五章：介绍基于树莓派的人脸识别考勤中的考勤功能；

第六章：描述网页端考勤记录查看及程序设计；

第七章：描述完成系统过程中遇到的问题及收获；

第八章：描述未来的系统功能。

第二章　技术简介

2.1 Python & Django

2.1.1 Python 简介

如果说当前那种语言最流行，无疑还是 Sun Microsystems 公司于 1995 年 5 月推出的高级程序设计语言 Java，但是要说在人工智能领域中哪个语言最流行，那一定是 Python。

Python 是一种上手十分容易的编程语言，很是适合初学者进修。Python 除了相对 C 等高级语言运算速度慢一点以外，几乎再没有任何缺点。开发人员几乎可以用 Python 做任何事情，因为 Python 具有非常丰富的第三方库，比如想要使用矩阵运算时可以导入 Numpy 库，想要进行深度学习的运用可以用机器学习库，比如 Sktlearn、TensorFlow 等等。

Python 同时也支持面向对象编程，这意味着开发人员的代码重复利用率更高，程序更容易维护。

2.1.2 Django 简介

Django 是一个完全由 Python 编写的框架，主要用于支持网站开发，它的首要目标就是帮助开发者快捷的开始一个 Web 应用。Django 支持 MVC 架构，这使得模块之间耦合度更低，代码冗余率更低。Django 还具有防止跨站访问的功能，这使得用 Django 搭建的 Web 应用更加安全。

2.2 OpenCV for Python

2.2.1 OpenCV 简介

OpenCV 是基于 BSD 许可证的跨平台计算机视觉库，可以在诸如 Linux，Windows，Android，Mac OS 等操作系统上运行。

OpenCV 有许多版本，包括 C ++，Java，Python 和其他版本。该项目使用 Python 版本的 OpenCV-Python。

OpenCV – Python 是 OpenCV 在 Python 中使用的 API 接口。它同时具有 OpenCV C ++ API 和 Python 语言两种功能。 Python 语言运行速度比 C ++语言慢，但 Python 语言具有简单性、短期性和快速学习的特点。同时，Python 对调用 C ++开发的组件也非常有用，因此开发人员可以使用 C ++来实现一些高性能的需求功能。通过这种方式，开发人员可以快速运行类似于 C ++的代码，并使用 Python 语言开发软件功能。因此，OpenCV 实现 Python 接口。与此同时，OpenCV - Python 还实现了 Numpy 库的接口规范，这对于在 Python 中使用 Numpy 非常有用。例如，开发人员可以将 Numpy 的数据结构传输到 OpenCV，同样的

也可以将 OpenCV 数据结构传输到 Numpy，还可以使用 SciPY 和 Matplotlib 共同开发。由于这些工具的使用更广泛，OpenCV - Python 是开发视觉原型和视觉实验的绝佳工具。

2.2.2 Raspberry Pi 搭建 Python-OpenCV

本课题没有采用 Python2.7 的版本，而是使用 Python3.5 的版本。尽管现在仍然有许多第三方库对 Python3 的支持不是很好，但是 Python3.5 的优点已经足以弥补这些缺憾。

Raspberry Pi 安装 OpenCV 采用的是编译源码进行安装，这是一个非常艰巨并且也非常耗时的操作。

首先更新树莓派，确保树莓派的更新库是最新的，运行如图 2.1 的命令：

```
$ sudo apt-get update && sudo apt-get upgrade
```

图 2.1　更新树莓派软件

此时，树莓派的软件已经更新到最新状态。由于本系统采用源码安装，所以需要为树莓派安装编译环境。在终端中执行图 2.2 的命令。

```
$ sudo apt-get install build-essential cmake pkg-config
```

图 2.2　安装编译环境

OpenCV 是做图像处理的，所以需要为 OpenCV 安装一些图片 I/O 包和视频支持包，这些库允许开发者能灵活的使用 OpenCV，运行图 2.3 命令。

```
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

图 2.3　安装图像和视频 I/O 包

OpenCV 内置”highgui” 库，这个库是为了在显示屏中显示一个图像并组建一个 GUI，编译”highgui”库，需要安装 GTK 开发库，执行图 2.4 命令。

```
$ sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

图 2.4　安装图像显示支持

OpenCV 中有许多矩阵相关的库，可以安装一些额外的依赖来进行优化。执行图 2.5 命令。

```
$ sudo apt-get install libatlas-base-dev gfortran
```

图 2.5　优化 OpenCV 的库

当环境搭建完成，现在可以下载 OpenCV 源码，本课题采用的是 3.3.0 的版本，可以从 github 中将源码下载下来，同时为了完整安装 OpenCV，也应该下载 opencv_contrib 存储库。下载解压代码如图 2.6 所示。

```
$ cd ~
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
$ unzip opencv.zip
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archiv
e/3.3.0.zip
$ unzip opencv_contrib.zip
```

图 2.6　下载 OpenCV 源码

当源码下载完成后，则可以进行源码的编译安装。在 OpenCV 目录创建一个新的文件夹，在进行编译之前需要运行 cmake 命令进行环境依赖的检测，如图 2.7 所示。

```
$ cd ~/opencv-3.3.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
    -D BUILD_EXAMPLES=ON ..
```

图 2.7　编译 OpenCV 源码

环境依赖检测成功，没有问题，则使用图 2.8 所示命令完成编译和安装。

```
$ make -j4 && make install
```

图 2.8　编译 OpenCV 源码

接下来就是长达五六个小时的等待。

2.3 腾讯优图人脸识别开发库——Python-SDK

腾讯优图人脸识别开发库可以帮助一个非深度学习研究者快速开始一些人脸识别项目。

2.3.1 准备步骤

1、下载 Python-SDK，输入图 2.9 所示命令。

```
root@pg1314:~# git clone https://github.com/Tencent-YouTu/Python_sdk.git
```

图 2.9　下载 Python-SDK

2、在优图主页注册并获取到 appid、secret_id、secret_key、userid，然后在 Python_SDK 下面的 sample.py 填写获得的验证信息，如图 2.10 所示。

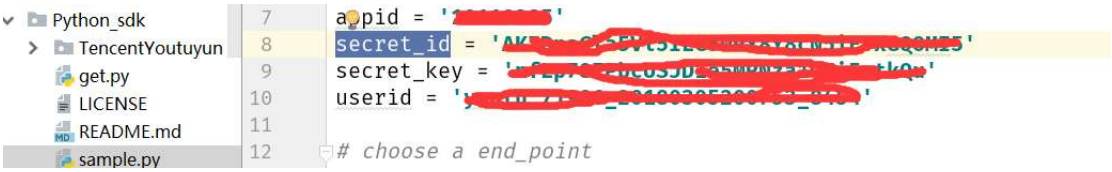

图 2.10　填写验证

3、在 Python_SDK 下面新建一个文件，并编写代码如图 2.11 所示。

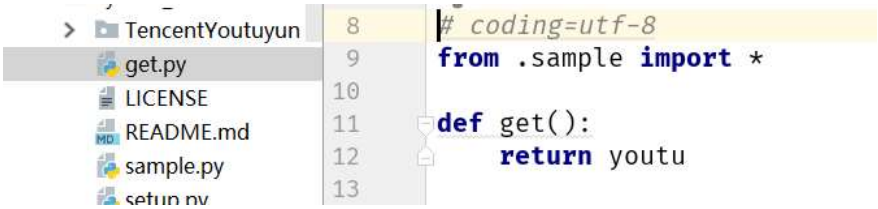

图 2.11　编写 get.py 文件

4、在开发人员想要使用 Python_SDK 的文件中，只要导入 get.py 就能使用”youtu”实例进行人脸识别等操作。

2.3.2 常用 API

1、人脸识别接口命令如图 2.12 所示。

```
FaceIdentify(self, group_id, image_path, data_type = 0)
```

图 2.12　人脸识别 api

参数中的 group_id 是指人脸组别 id，本系统可以为不同属性的组别同时命名不同的 id，比如自动 1403 班可以为一个组，每个班都可以作为单独的一个组别，在人脸识别中，指定特定的组别，系统只会在这个组别中进行人脸匹配，而不是整个人脸库。

2、增加人脸接口命令如图 2.13 所示。

```
AddFace(self, person_id, images, tag='', data_type = 0)
```

图 2.13　增加人脸的 api

参数列表中的 person_id 是指每一个人脸的 id 信息，本系统使用的是学生的学号作为 person_id；images 是人脸照片的 url 或者是人脸照片路径，URL 其实就是一个照片文件打开后进过 base64 编码产生的编码，所以在 Python-SDK 内部，如果开发者传进去的是路径，内部程序也是使用 Python 的 open 方法将照片打开，并使用 base64 编码将其编码后，再进行后续的操作。参数中 data_type 可以选择为 0 或 1，代表 images 参数是路径或者 URL。

3、增加个体接口命令如图 2.14 所示。

```
NewPerson(self, person_id, image_path, group_ids, person_name= '',
tag='', data_type = 0)
```

图 2.14　增加个体的 api

在进行添加一个人脸之前，需要新建一个个体，这个有属于的组别，有本身的名字，个体建好之后才能在这个个体中添加自己的人脸信息，每个人可以添加 10 张人脸，但是不能添加相似度过高的人脸，为了提高人脸识别准确率，最好添加 10 张不同场景的人脸。

2.4 Sqlite

SQLite 是一种进程携带库，实现一个自给自足的、不依靠服务器的、完全不用配置的、支持原子性等事务性的 SQL 数据库存储引擎。SQLite 的代码是完全开源的，因此任何人可以用对其拥有于任何目的操作，当然也可以作为商业或私人用途。SQLite 是世界上部署最广泛的数据库，其应用程序数量超过了官方可以计算的数量，其中包括几个备受瞩目的项目。SQLite 也是一种基于嵌入式 SQL 数据库存储引擎。与大多数其他诸如 MYSQL、SQLServer、Oracle 等数据库不同，SQLite 并没有自身独立的服务器进程。SQLite 是直接读取和写入普通磁盘文件的，其中包含多个表，索引，触发器和视图的完整 SQL 数据库存在于单个磁盘文件中。数据库文件由于其特使的格式所以是可以是跨平台的使用的- 您当然可以使用 32 位和 64 位系统之间进行数据库的操作或者在大端和小端体系结构之间进行自由的复制数据库。这些特殊的功能让 SQLite 成为非常热门的选择。

同时，Sqlite 支持标准 SQL 语句。

第三章　设计方案

3.1 基于树莓派的人脸识别签到系统架构

人脸识别签到系统包括签到结果展示部分和硬件签到部分。签到结果展示又分为个人签到记录的查看、全部人员签到记录查看（仅管理员有权限）、异常打卡补签以及管理员审批等部分。硬件部分则涉及签到、定时更新数据（未签到的人加进数据库）、以及一开始的数据初始化。系统架构如图 3.1 所示。

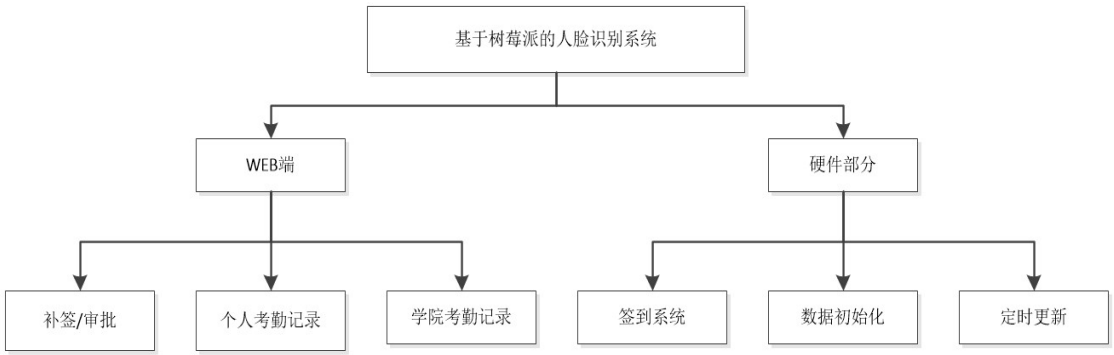

图 3.1　人脸识别系统架构图

从整个系统来看，主要分为两大部分，然后再细分许多功能，每个功能有独立的逻辑处理，功能与功能之间的通信只能通过预留的 API 接口。这样做的目的是为了让代码能够一次一次的被调用，而不用编写重复的代码。

3.2 人脸识别考勤系统的功能流程

在系统运行之前，首先要做的是将一些数据进行初始化，比如学生信息的导入、课表的导入，然后才能启动 Web 服务、考勤程序和定时数据更新程序。本课题采用三个进程来启动这些程序，三个进程互不干扰的执行，每个进程只负责自己的任务，考勤程序只负责对刷脸的学生进行考勤统计，定时数据更新则是在几个特点的时间扫描数据库，看看有没有旷课的学生，而 Web 服务则只是负责考勤记录的查看与异常考勤的补签流的操作。

尽管每个进程只负责自己的任务，但是同处于一个系统下，进程之间还是有通信的。

考勤程序会将考勤结果持久化到数据库中，只有插入操作；定时更新程序会从数据库中查询出没有哪些应该进行签到但是没有签到的学生，将他们的考勤状态设为旷课，定时更新程序有查询插入操作；而 Web 端，也是从数据库中查询考勤记录，并展示在页面中，补签等操作也是与数据库打交道，即 Web 端涉及数据库的查改增操作。所以这几个进程的能够进行通信，因为它们完全依赖于使

用同一个数据库。当然，本系统并不需要担心由多进程引发的安全问题，这得益于 SQLite 的数据库操作是原子操作。人脸考勤系统的整体功能逻辑如图 3.2 所示。

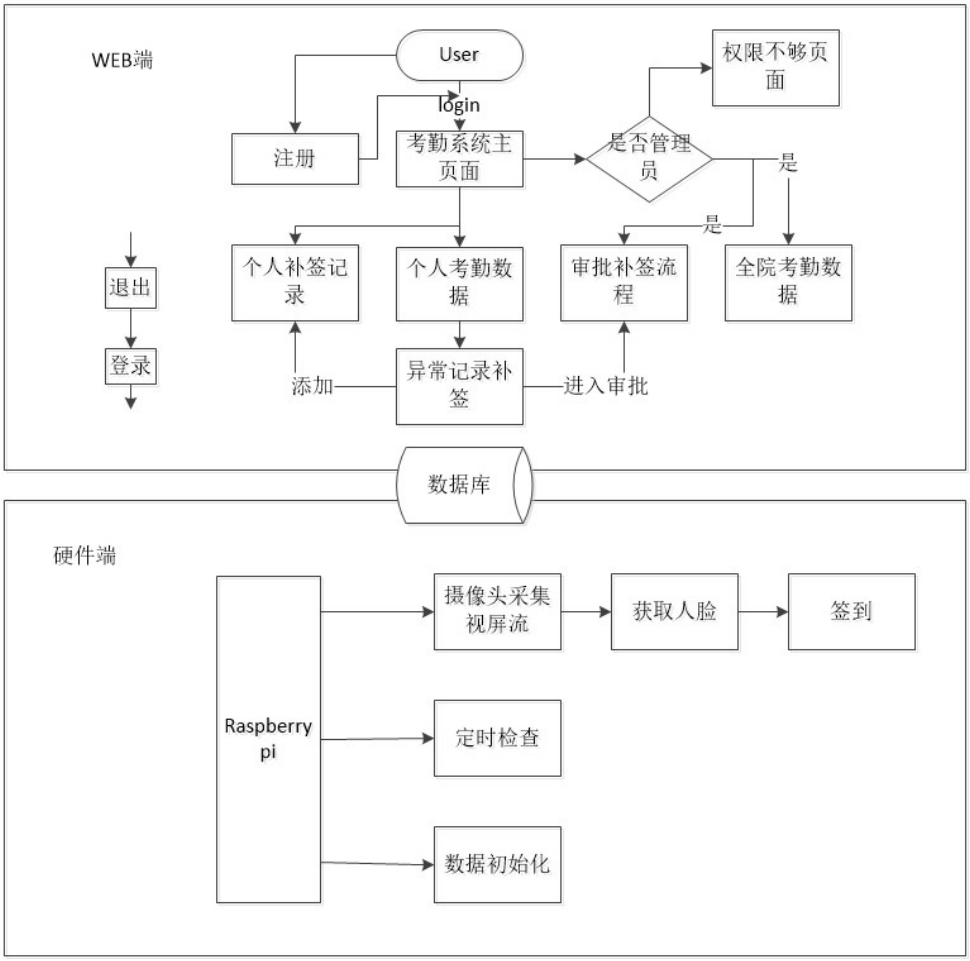

图 3.2　人脸识别考勤系统的功能逻辑

三个进程执行三个程序是可行的，首先定时更新函数，它是利用定时器做的，只有在设定的时间才会执行更新函数，其他时候仅仅只是做一个无限循环，既没有连接到数据库也没有进行网络资源的占用，所以对 CPU 的消耗是可观的；签到功能程序，仅仅是占用了 Raspberry Pi 的视频设备，而且为了节省网络资源，系统还设计了按键，只有按键按下的时候，才截取一帧照片，进行签到逻辑的运算；而 Web 端的设计将所有网页中的静态资源比如 JavaScript 文件、CSS 文件和图片等文件都放在了 Django 设置的静态资源文件夹下，这使得在客户端访问的时候，除了第一次访问需要下载静态资源，后面的访问将使用缓存，很大程度地减轻服务器的压力，使访问速度得以提高。

3.3 设计原则

本课题设计的第一原则为可行性原则：所有的设计都是具有理论基础的，都是经过大量调试验证的，只有确定可行，才会继续下一步的操作。这是整个系统的开发中最主要、最基本的原则。

分块设计原则：本课题采用切割划分出许多不同的功能块，在确保第一原则的前提下，划分出尽可能详细的，功能单一的块，这样能使得代码的可重用率提升。

低能耗原则：对于连接资源，在逻辑结束后，一定要释放，避免浪费资源。系统采用的硬件设备也是功耗非常低的设备。

第四章　数据库设计

4.1 数据库设计 EER 模型

本课题一共设计了四张表存在 sqlite 数据库中，包括了 T_XUPT_PERSON（学生信息表）、T_XUPT_TIMETABLE（课表）、T_XUPT_ATTEND（签到表）、T_XUPT_REATTEND（补签表）。表的设计遵循数据表设计的三大范式，数据表的 EER 模型图见图 4.1 所示。

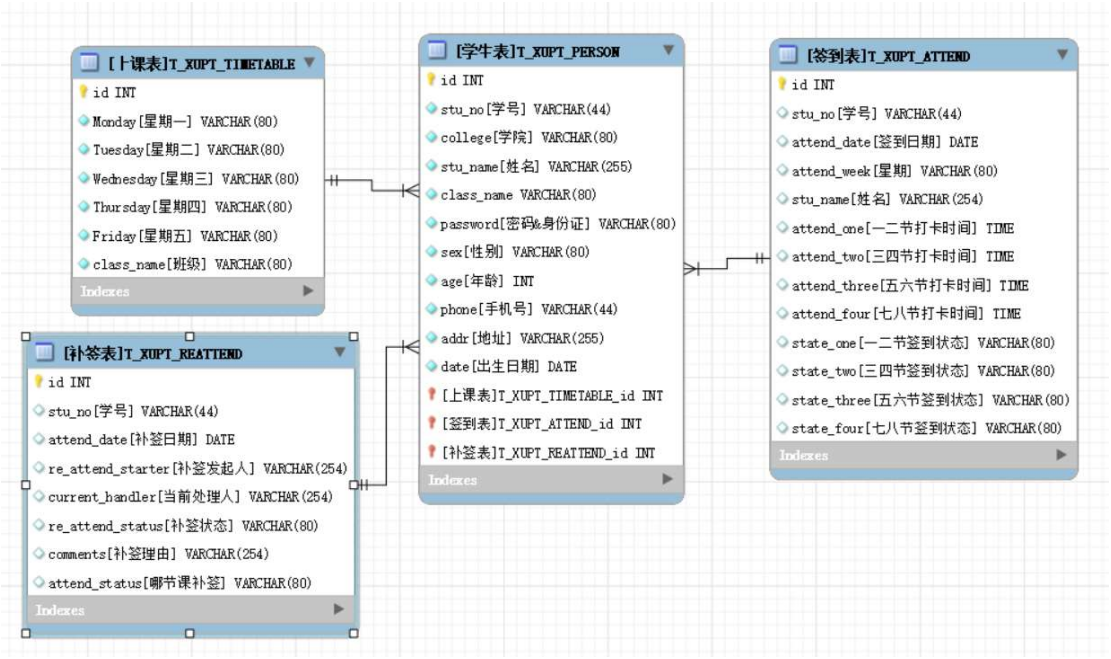

图 4.1　人脸识别考勤系统 EER 模型图设计

这就是整个人脸识别考勤系统的 EER 模型图设计，接下来几个小节将详细描述每个表的具体设计与实现。

4.2 学生表的设计

4.2.1 总体说明

学生表在数据库中的表名是 T_XUPT_PERSON，一共包含 11 个字段，通过这个表可以定义每个学生在数据库中存储的基本信息。

学生表在数据初始化时候插入，数据由管理员定义在 excel 表格中，只有初始化时才能增加学生，在表中有信息的学生才能进行注册人脸和刷新签到操作。

在 Web 端的登录模块、签到模块、注册人脸模块都涉及到此表。

4.2.2 表结构说明

表结构设计见表 4.1。

表 4.1　学生表（T_XUPT_PERSON）

字段名称	数据类型	是否可空	字段描述
id	INTEGER	N	主键字段，自增长
stu_no	VARCHAR(80)	N	学生的学号，同时也是账号
stu_name	VARCHAR(255)	N	学生的姓名
college	VARCHAR(80)	N	学生所在的院系
class_name	VARCHAR(80)	N	学生的班级，与课表 class_name 字段关联
password	VARCHAR(80)	N	学生的登录密码（身份证）
sex	VARCHAR(80)	N	学生的性别
age	INTEGER	N	学生的年龄
phone	VARCHAR(44)	N	学生的手机号
addr	VARCHAR(255)	N	学生的地址所在
date	date	N	学生的出生日期

4.2.3 建表语句

学生表建表 SQL 语句如图 4.2 所示。

```
CREATE TABLE "T_XUPT_PERSON" (
  "id"         INTEGER      NOT NULL PRIMARY KEY AUTOINCREMENT,
  "stu_no"     VARCHAR(44)  NOT NULL,
  "stu_name"   VARCHAR(255) NOT NULL,
  "college"    VARCHAR(80)  NOT NULL,
  "class_name" VARCHAR(80)  NOT NULL,
  "password"   VARCHAR(80)  NOT NULL,
  "sex"        VARCHAR(80)  NOT NULL,
  "age"        INTEGER      NOT NULL,
  "phone"      VARCHAR(44)  NOT NULL,
  "addr"       VARCHAR(255) NOT NULL,
  "date"       DATE         NOT NULL
)
```

图 4.2　创建 T_XUPT_PERSON

4.3 签到表的设计

4.3.1 总体说明

签到表表名为 T_XUPT_ATTEND，设计 13 个字段，这个表定义所有学生的考勤时间与状态。

签到表涉及到的模块有：刷脸签到、考勤记录查看、补签通过、定时数据更新模块。

4.3.2 表结构说明

签到表的设计参照西安邮电大学一天八节课的情况，八节课又分为早上一二节、早上三四节、下午五六节、下午七八节，所以设计 8 个字段分别表示这四个不同时间段的考勤时间和考勤状态，考勤状态分为 4 种。表结构设计见表 4.2 所示。

表 4.2　签到表（T_XUPT_ATTEND）

字段名称	数据类型	是否可空	字段描述
id	INTEGER	N	主键字段，自增长
stu_no	VARCHAR(80)	Y	学生的学号，同时也是账号
stu_name	VARCHAR(254)	Y	学生的姓名
attend_date	date	Y	考勤当天日期
attend_week	VARCHAR(80)	Y	考勤当天的星期
attend_one	time	Y	一二节课打卡时间
attend_two	time	Y	三四节课打卡时间
attend_three	time	Y	五六节课打卡时间
attend_four	time	Y	七八节课打卡时间
state_one	VARCHAR(80)	Y	一二节考勤状态
state_two	VARCHAR(80)	Y	三四节考勤状态
state_three	VARCHAR(80)	Y	五六节考勤状态
state_four	VARCHAR(80)	Y	七八节考勤状态

四种考勤状态见图 4.3 所示。

```
status = (
    (0, '迟到'),
    (1, '旷课'),
    (2, '正常'),
    (3, '已补签'),
)
```

图 4.3　四种签到状态

4.3.3 建表语句

建表语句见图 4.4 所示。

```
CREATE TABLE "T_XUPT_ATTEND" (
  "id"           INTEGER      NOT NULL PRIMARY KEY AUTOINCREMENT,
  "stu_no"       VARCHAR(44)  NULL,
  "attend_date"  DATE         NULL,
  "attend_week"  VARCHAR(80)  NULL,
  "stu_name"     VARCHAR(254) NULL,
  "attend_one"   TIME         NULL,
  "attend_two"   TIME         NULL,
  "attend_three" TIME         NULL,
  "attend_four"  TIME         NULL,
  "state_one"    VARCHAR(80)  NULL,
  "state_two"    VARCHAR(80)  NULL,
  "state_three"  VARCHAR(80)  NULL,
  "state_four"   VARCHAR(80)  NULL
)
```

图 4.4　创建 T_XUPT_ATTEND

采用的无意义的主键列，仅仅是为了方便索引，其它字段允许为空，意味着增大了容错率。

4.4 上课表的设计

4.4.1 总体说明

上课表名为 T_XUPT_TIMETABLE，共设计 7 个字段，此表定义每个班级一周的上课安排。

上课表涉及到的模块有：刷脸签到、定时数据更新模块。

4.4.2 表结构说明

上课表（T_XUPT_TIMETABLE）的设计很简单，使用了无意义的自增属性作为 id，然后有不可重复的班级字段、周一到周五这几个字段。表结构设计见表 4.3 所示。

表 4.3　上课表（T_XUPT_TIMETABLE）

字段名称	数据类型	是否可空	字段描述
id	INTEGER	N	主键字段，自增长
Monday	VARCHAR(80)	N	
Tuesday	VARCHAR(80)	N	8 长度的 01 字符串，每两位为
Wednesday	VARCHAR(80)	N	一组，两个 1 代表有课，两个 0
Thursday	VARCHAR(80)	N	表示没有课
Friday	VARCHAR(80)	N	
class_name	VARCHAR(80)	N	班级名称，唯一

4.4.3 建表语句

建表语句如图 4.5。

```
CREATE TABLE "T_XUPT_TIMETABLE" (
  "id"          INTEGER     NOT NULL PRIMARY KEY AUTOINCREMENT,
  "Monday"      VARCHAR(80) NOT NULL,
  "Tuesday"     VARCHAR(80) NOT NULL,
  "Wednesday"   VARCHAR(80) NOT NULL,
  "Thursday"    VARCHAR(80) NOT NULL,
  "Friday"      VARCHAR(80) NOT NULL,
  "class_name" VARCHAR(80) NOT NULL UNIQUE
)
```

图 4.5　创建 T_XUPT_TIMETABLE

4.5 补签表的设计

4.5.1 总体说明

补签表（T_XUPT_REATTEND）共有 8 个字段，其设计是由于有些学生可能迟到、或者忘记打卡而导致的打卡异常，用来记录补签的表。

补签表设计到的模块：补签审批、补签记录查看。

4.5.2 表结构说明

补签表表结构见表 4.4 所示。

表 4.4　补签表（T_XUPT_REATTEND）

字段名称	数据类型	是否可空	字段描述
id	INTEGER	N	主键字段，自增长
stu_no	VARCHAR(44)	Y	学生学号
attend_date	date	Y	需要补签哪天
re_attend_starter	VARCHAR(254)	Y	谁发起的补签
current_handler	VARCHAR(254)	Y	当前处理补签的人
re_attend_status	VARCHAR(80)	Y	当前补签流程的状态
comments	VARCHAR(254)	Y	补签申请的理由
attend_status	VARCHAR(80)	Y	对哪节课进行补签

补签流程的状态说明见表 4.5 所示。

表 4.5　补签流程状态对应表

re_attend_status	说明
1	处理中
2	审批通过
3	审批不通过

对哪节课进行补签的对应关系见表 4.6 所示。

表 4.6　补签节数对应表

re_attend_status	说明
0	一二节课
1	三四节课
2	五六节课
3	七八节课

4.5.3 建表语句

建表语句见图 4.6。

```
CREATE TABLE "T_XUPT_REATTEND" (
  "id"                INTEGER      NOT NULL PRIMARY KEY AUTOINCREMENT,
  "stu_no"            VARCHAR(44)  NULL,
  "attend_date"       DATE         NULL,
  "re_attend_starter" VARCHAR(254) NULL,
  "current_handler"   VARCHAR(254) NULL,
  "re_attend_status"  VARCHAR(80)  NULL,
  "comments"          VARCHAR(254) NULL,
  "attend_status"     VARCHAR(80)  NULL
)
```

图 4.6　创建 T_XUPT_REATTEND

4.6 数据库设计总结

表的设计是按照范式法则并结合实际情况进行设计的，每个字段都有必要存在而不是赘余，数据库 EER 模型的设计经历多次的修改重建，阅读了大量的数据库文献，保证数据表的优化。

第五章　硬件设计与实现

5.1 硬件设计架构

在硬件部分，本课题设计了三个功能点：数据初始化、签到、定时的数据库更新，程序的主要入口在签到功能里面，签到功能程序开始，创建了两个进程来启动数据初始化和定时更新，其中数据初始化执行完毕后进程就结束，定时更新则在无限循环，设计架构图见图 5.1 所示。

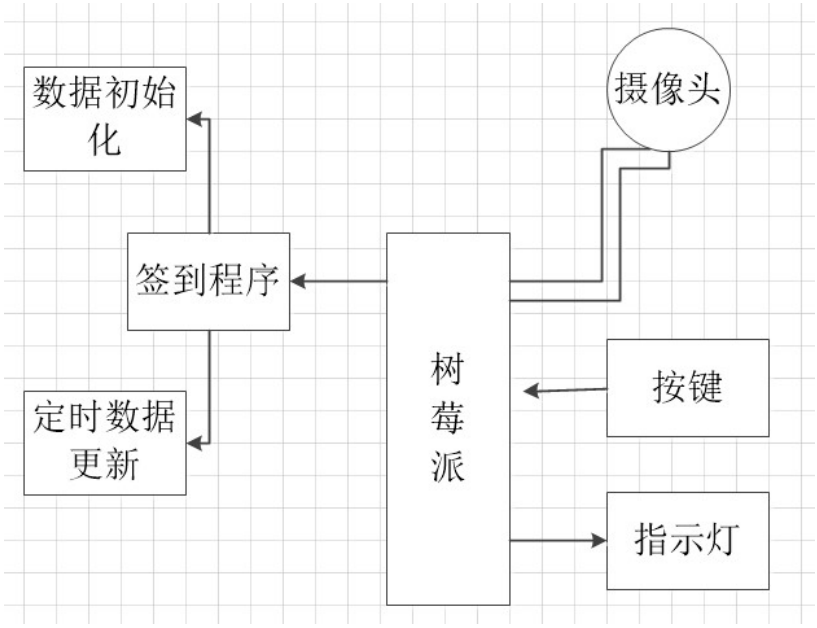

图 5.1　硬件部分架构

指示灯模块是为了显示一个签到是否成功的模块。

5.2 数据初始化

本课题的设计要求管理员做一个 excel 表格，表格里面要有学生的信息，这些信息要求和数据库中的 T_XUPT_PERSON 所规定的一致，放在 sheet0 中，模板如图 5.2。在 sheet1 中要求管理员填写每个班级的课表，有课填写 1，否则填写 0，模板见图 5.3 所示。

人员导入模板									
学号	学生姓名	学院	班级	密码	性别	年龄	手机号	地址	出生日期
123456	张三	自动化学院	自动xxxx班	123456789123456000	男	-5873	1234567891	北京	1996/6/1

图 5.2　人员导入模板

	星期/朝夕	周一	周二	周三	周四	周五
自动1403班课表	上午	1	0	0	0	1
		1	0	0	0	1
		0	1	1	0	1
		0	1	1	0	1
	下午	0	0	1	0	1
		0	0	1	0	1
		1	0	0	0	1
		1	0	0	0	1
	星期/朝夕	周一	周二	周三	周四	周五

图 5.3　课表模板

当签到程序运行时，会开启一个进程，在这个进程中调用初始化数据程序。数据初始化通过 Python 的 xlrd 库来完成，xlrd 是 Python 读取 excel 表格的一个库。数据初始化流程图根据图 5.4 所示。

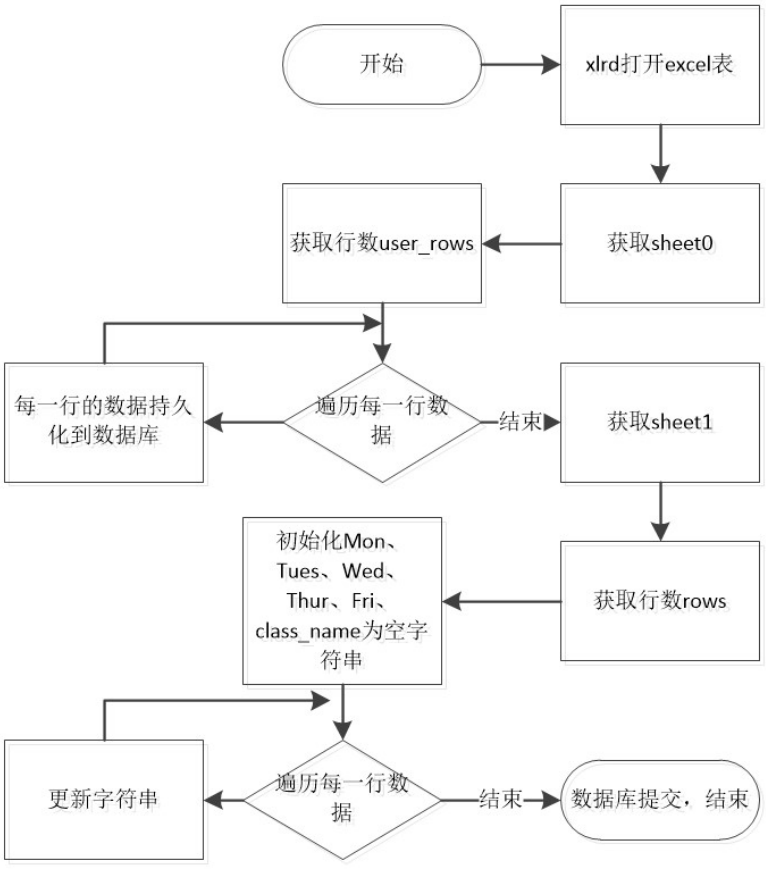

图 5.4　数据初始化流程

在流程图中，前面的循环是将学生信息插入读取到后插入到数据库中，后面的循环是遍历每列数据，因为模板里面存的是 01，所以数据库存的是一天 8 节课的信息，需要全部遍历完成才能获取到 8 位长度的 01 课表串。

5.3 定时更新数据库

定时更新数据库程序很有必要，因为在上课前 20 分钟到下课这个时间段是

可以正常刷脸签到，但是存在一些同学在这个时间段没有进行签到，那么这些人在签到表里没有记录，同时他们的状态应该属于旷课存在。

设计这个程序的目的就是排除掉签到的同学，把需要签到但是没有签到的同学将他们的考勤状态设置为旷课。

定时更新程序使用了 Python 的 schedule 模块，schedule 模块可以通过设置一个时间和指定一个函数，当进程还在运行并且时间到达时，系统就会运行那个函数。本课题一共设置了四个函数，分别会在早上一二节放学、早上三四节放学、下午五六节放学和下午七八节放学执行。选择这几个时间是有道理的，因为在放学后要是学生还没有进行签到，签到系统就会拒绝学生的签到了，那么学生一定是旷课状态。定时更新程序逻辑流程根据图 5.5 所示。

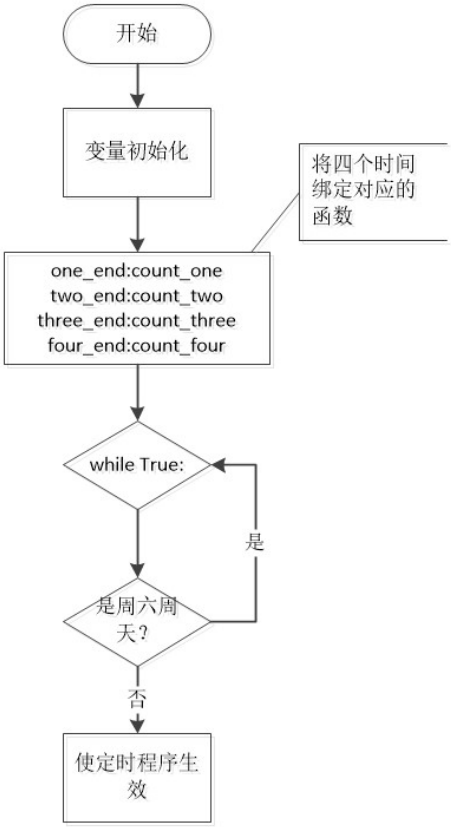

图 5.5　定时更新数据库流程

由于周六周天不需要上课，所有在程序设计时候加了判断，如果不是周六周天才使得定时器程序生效。具体逻辑业务四个 count_*决定，它们通过 schedule 与*_end 时间进行绑定。本课题对于上下课时间做出图 5.6 的规定：

```
# 早上一二节打卡开始、结束
one_start = '8:00'
one_end = '10:15'
# 早上三四节打卡开始、结束
two_start = '10:30'
two_end = '12:00'
# 下午一二节打卡开始、结束
three_start = '14:30'
three_end = '16:45'
# 下午七八节打卡开始、结束
four_start = '17:00'
four_end = '19:00'
```

图 5.6　上课时间安排

每次下课放学都会执行相应的函数。一开始根据课表获取当前时间刚下课的班级，然后对这个班级进行遍历，获取当前班级中除了已经签到过的学生的学号，再对这个学号进行遍历，然后对这些人更新数据库，如果他们一天刷脸签到，则在数据库中新增一条当前时间的考勤状态为旷课的记录，如果在前面几节课中已经签到了，则是对那条记录进行更新操作，设置当前时间的考勤状态为旷课。早上一二节统计逻辑流程图如图 5.7 所示。

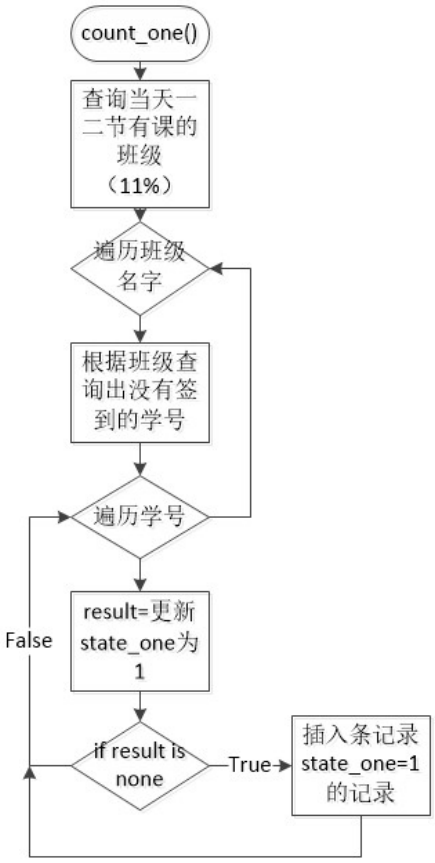

图 5.7　一二节下课数据更新

剩下三个函数的逻辑是一样的，只不过查询班级的时候模糊查询的关键字和更新或者插入的字段不一样，比如三四节课就不是'11%'，而是'__11%'，其中下划线代表占位符，一个下划线可以代表一个任意字符，一个百分号代表任意个的任意字符。

5.4 签到

5.4.1 电路连接

签到模块是整个课题最重要的部分，也是逻辑最复杂的部分。本课题采用树莓派连接一个摄像头、一个按键、和两个 led 灯完成硬件部分的操作。电路设计如图 5.8 所示。

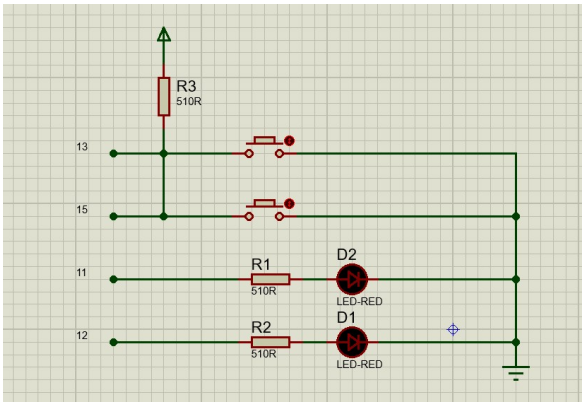

图 5.8　电路设计图

树莓派和摄像头安装见图 5.9：

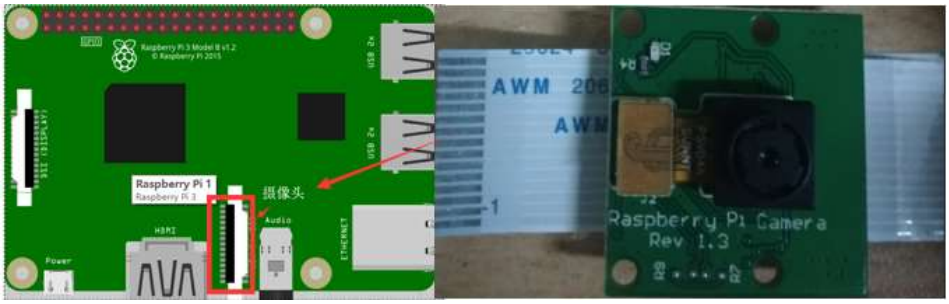

图 5.9　树莓派和摄像头安装

摄像头用来拍摄视屏流，为了避免一直进行人脸识别的请求，本课题增加了一个按键，当只有按键被按下时才进行人脸识别，如果签到成功，则绿灯会亮，否则红灯亮。

5.4.2 程序实现

程序使用 OpenCV 捕获本机摄像头设备，使用默认的人脸检测器”

haarcascade_frontalface_default.xml”文件作为检测器，腾讯优图的 Python-SDK 为人脸识别工具，整个签到功能流程见图 5.10。

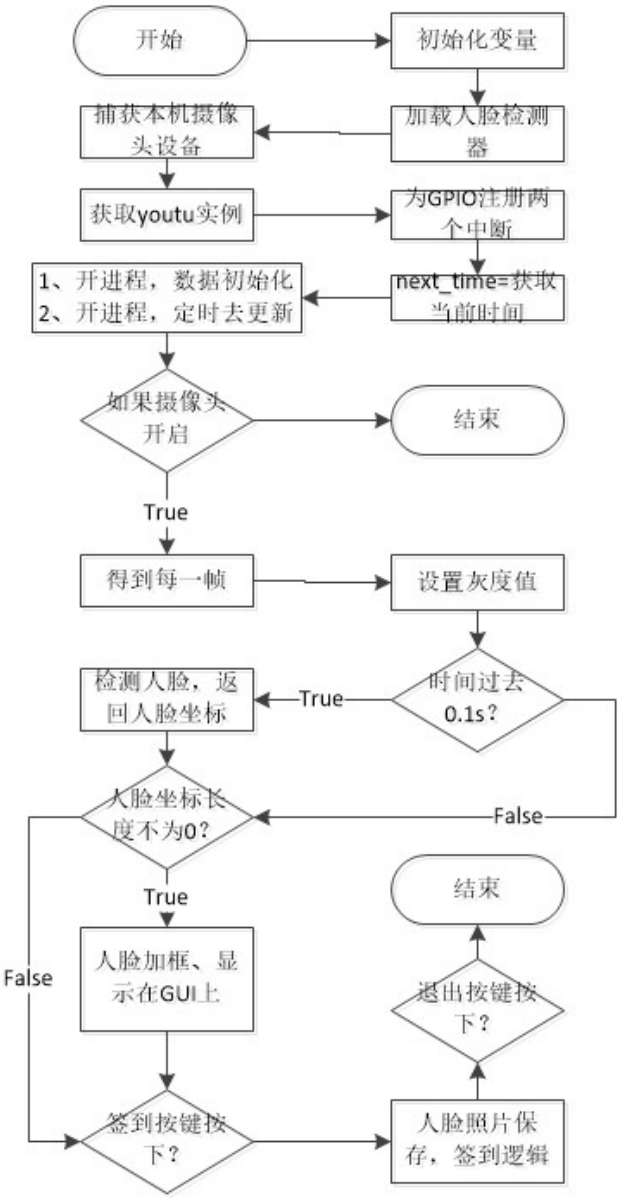

图 5.10　人脸识别签到流程

当学生按下签到按键时候，树莓派的 GPIO 中断会将一个全部变量 key 设置为 1，在主程序检测到 key 为 1 时，OpenCV 会把当前一帧照片保存，然后将这张照片的路径传到人脸识别函数中，人脸识别函数流程如图 5.11。

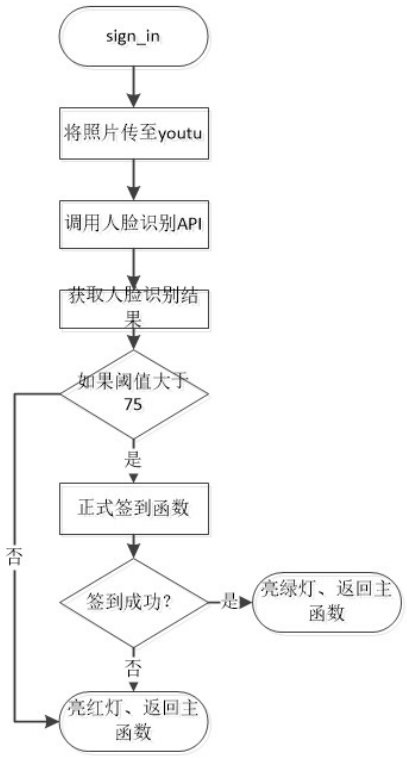

图 5.10　人脸识别流程

当获取到人脸识别结果的时候，程序还要对结果进一步的分析，只有当人脸匹配度超过 75%的时候，才能算这个识别结果是有效的，才能进入正式的签到流程，函数名称为 attend，接收识别的结果中的 id 值为参数。签到逻辑如图 5.11 所示。

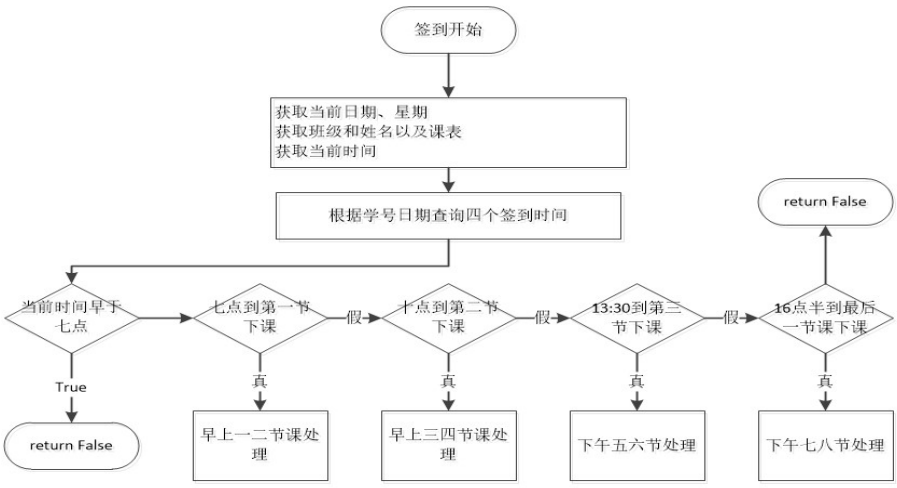

图 5.11　签到的大概逻辑

四个具体签到处理的思想是一致的：首先判断该时间段是不是一个有效的签到时间段，如果不是，则不能签到，程序返回 False，如果是，则根据学号获取到

班级名，再由班级名字得到当天的课表，判断该时间段上该学生有没有课，没有课则直接返回 False，有课才能进行签到。具体的签到处理以早上三四节课处理程序为例，逻辑流程见图 5.12。

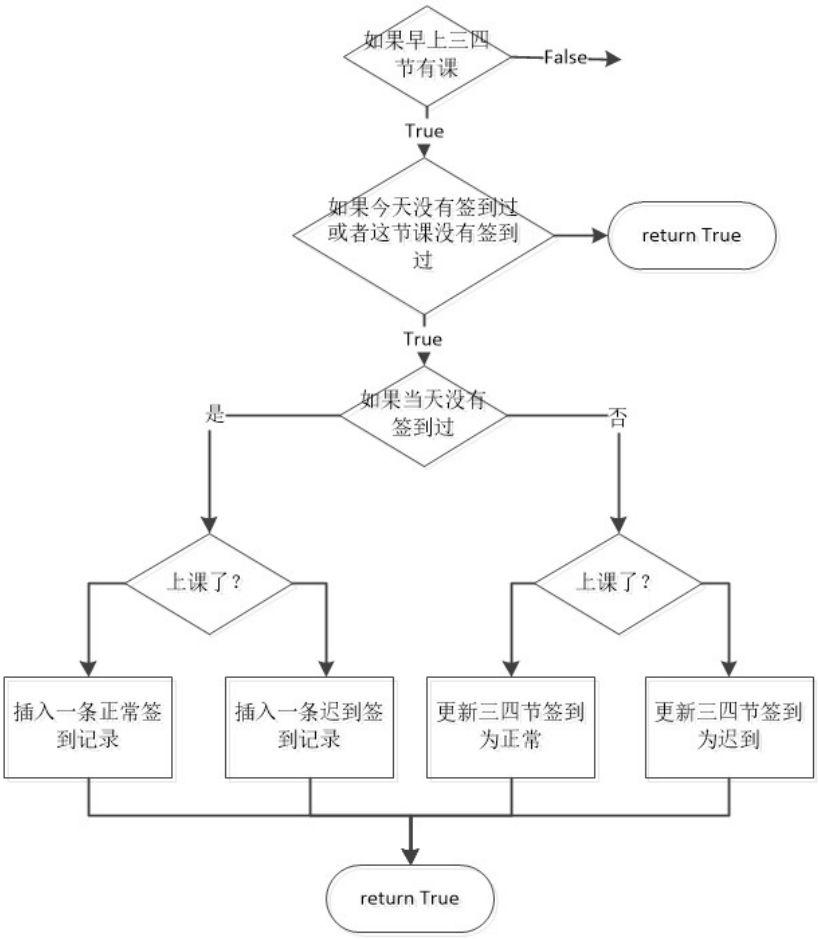

图 5.12　早上三四节签到业务

本课题设计的签到逻辑能有效地进行签到操作，当存在两节连续的课程存在时，第二节课的签到一定在第一节课下课后才能进行，而对于第一节课没有课的学生，在第二节课上课前半个小时都能进行签到。对于已经签到的学生，如果再次进行刷脸签到，虽然会提示签到成功，但是并不会更新数据库，这样可以很大程度的减少了使用数据库的连接资源。

本课题设计了两个指示灯，当签到成功时，绿灯将闪烁一次，签到失败时，红灯将闪烁一次。

5.5 硬件部分总结

本课题在硬件部分一共做了三件事情：签到、数据更新和数据初始化。本课题硬件部分用的器件比较少，只有一个树莓派主控板、一个摄像头、两个按键和两个指示定，但是内容和功能比较完善。代码的实现经过了大量的验证、调试，

并经大量的实践检验，唯一美中不足的是当网络连接断开或者是网络不太好的时候，签到系统将陷入瘫痪状态，因为在人脸识别阶段人脸识别的 SDK 必须要使用网络。签到部分已经设计完毕，接下来的章节将详细介绍对签到结果的展示部分。

第六章　Web 界面设计

6.1 Web 端概述

整个 Web 端都是基于 django 的，前端使用了 bootstrap、jQuery 等前端框架进行页面的美化，同时还借助了一些优秀的 JS 脚本来对 Web 端的功能设计进行优化。Web 端一共有 8 个页面，包括：主页面、注册页面、登录页面、个人签到记录查看页面、无权限提示页面、补签记录页面、补签审核页面和所有人的签到记录查看页面。每个页面都有自己独特的特点，也有自己的所负责的事情，但是所有的页面也都有联系。下面几个小节将重点讲述这几个页面的主要功能和具体实现。

6.2 个人考勤记录查看

6.2.1 登录

本课题在开始建立项目的时候，在 django 提供的中间件中，设计了一个登录跳转的验证，当学生没有进行登录时，如果访问除了登录页面和注册页面，访问其他页面将会被拦截，django 自动的将网页进行重定向到登录页面。

登录有两种方式，一种是使用摄像头拍照进行人脸识别登录，一种是在网页中输入账号密码登录。当学生使用的客户端具有能调用本机摄像头功能时将显示一个由摄像头拍摄的页面；当学生使用的客户端不具有调用摄像头的功能时，页面会自动切换到账号密码登录形式。

通过网页摄像头进行人脸识别登录如图 6.1 所示。

图 6.1　人脸识别登录

通过网页输入账号密码进行登录如图 6.2 所示。

图 6.2　账号密码登录

人脸识别登录在前端页面主要依靠了 Webcam.js，它提供了在网页调用本地摄像头的 API，并能够返回调用成功或者失败，成功则显示视屏拍摄效果，失败则隐藏该模块，然后让账号密码框显示，相关代码如图 6.3 所示。

```
$(document).ready(function () {
    Webcam.on('error', function (err) {
        $('#camera-div').css('display', 'none');
        $("#input-div").css('display', 'block');
    });
    Webcam.set({
        width: 300,
        height: 150
    });
    Webcam.attach('#my_camera');
});
```

图 6.3　选择登录方式相关代码

在使用调用摄像头 API 之前，它要求开发人员必须在页面中留有一个专门显示摄像头拍摄情况的块级标签，并且这个标签必须是唯一的，也就是说这个标签一定要有 id 属性。

无论是人脸是被验证登录还是账号密码登录，后台请求都是同一个，只不过参数不一样而已。使用人脸识别登录时，当学生点击登录，Webcam.js 将会把当前摄像头的这帧照片以 base64 编码保存，然后将此编码通过 Ajax 传至后台；使用账号密码时，当学生输入完账号密码点击登录时，程序将通过两个输入框的 id 属性将学生刚输入的信息保存，并通过 Ajax 发送请求将数据传到后台。后台接收到请求后，转到相应的程序进行处理，并获取前端传过来的值，并通过是否为空来判断使用者到底是使用的人脸登录还是账号密码登录来进行不同的逻辑判断。后台处理登录的流程根据图 6.4 所示。

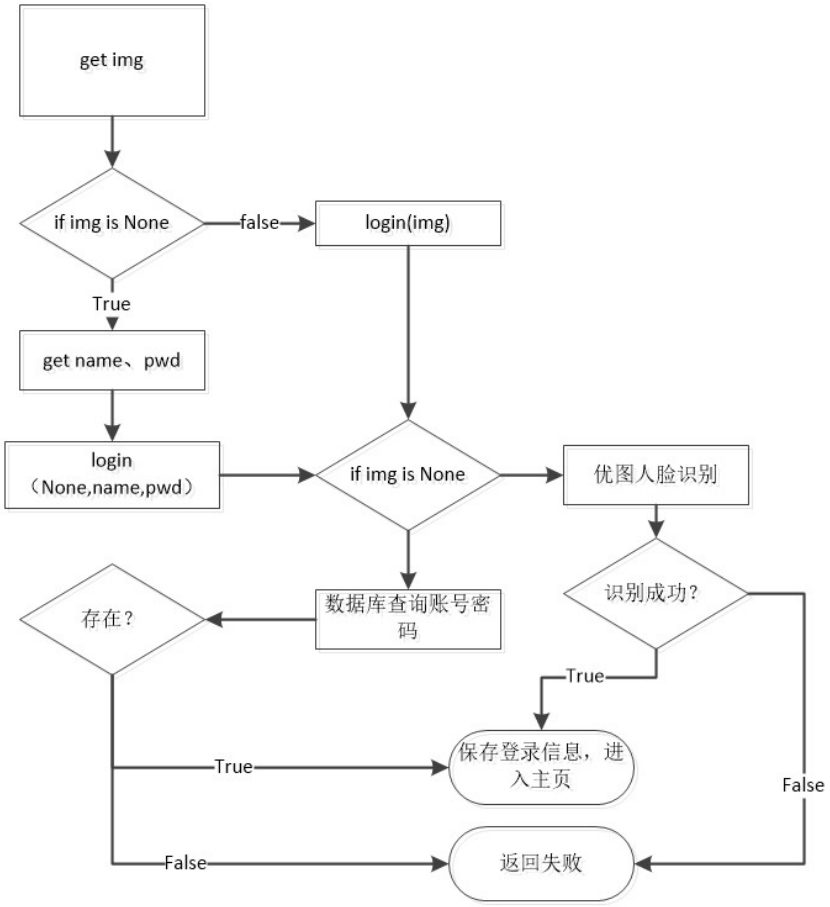

图 6.4　登录逻辑流程

在登录成功后，系统将自动重定向到主页面，登录失败则会弹出相应的提示。学生可以在登录之前进行注册。

6.2.2 注册

学生没有进行注册之前是登录不了系统的，但是本课题的注册又不是简单的注册账号密码，相对的来说更像是激活一下账号。注册功能要求学生必须要使用能调用摄像头的浏览器注册，并输入两次学号，调整好自己在网页中的成像，就可以开始注册了。然而，如果学生使用的是不能调用本机摄像头的浏览器，这将导致不能进行注册，页面上什么也不会出现。

注册的过程其实就是往人脸库增加个人的过程，因为签到部分用的是云端人脸库，如果没有注册，那么签到也将导致失败，因为签到过程中查询不到这个人脸的信息。

人脸检测多次，可提高识别准确率。

注册功能的前端部分依旧是使用 Webcam.js 作为调用摄像头的工具，后台接收到 base64 编码的图片和学号后，直接调用添加人脸 API，值得说一下的是，如

果学生第一次注册，要做的不仅是添加人脸，还要增加一个个体，毕竟人脸依赖于个体的存在。

注册成功后页面将提示注册成功，并跳转到主页中。

6.2.3 主页面

主页面是整个 Web 端的欢迎界面，在左侧“我的考勤”中，有三个功能，分别是查看自身的签到记录、自己的补签记录和代办补签功能，但是使用者不是管理员的话，代办补签功能点击将提示没有权限操作。如果使用者是管理员，在主页面左侧会多一个功能：查看所有学生的考勤记录。

6.2.4 查看个人考勤记录

学生可以查看自己的考勤记录，如图 6.5 所示。

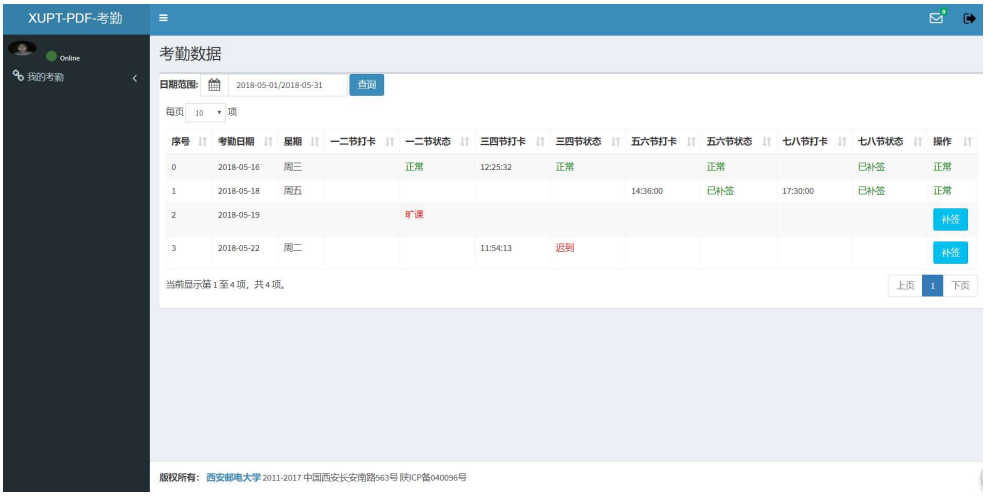

图 6.5　个人考勤记录

在存在考勤异常信息时，学生可以选择补签操作，如果考勤完全正常，则显示正常。点击补签后，客户端将弹出提示框，见图 6.6 所示。

图 6.6　补签信息提示

学生需要选择补签哪节课，如果选择的并没有存在考勤异常则会提示考勤正

常。当补签提交时，Ajax 请求将会给后台传递选择补签的日期、补签人的 id 以及选择的哪节课补签，后台收到数据将会更新数据库，提交补签流逻辑见图 6.7 所示。

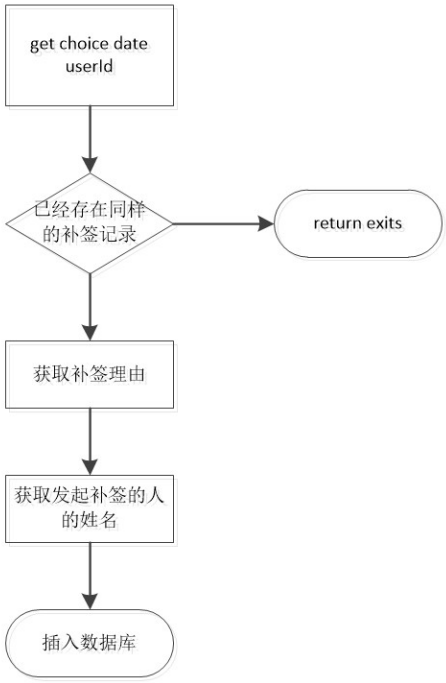

图 6.7　提交补签流

如果学生存在四次考勤都是异常的，那么学生必须要进行四次的补签提交，每次提交选择不同的考勤时间。

个人考勤记录的查看使用了 BootStrap 进行分页处理，还可以由使用者选择一页展示多少项，也可以选择日期范围的选择查询，使用者点击查询后，会将需要的显示项和两个日期范围传至后台。后台实现分页的处理，主要原理是利用了标准 SQL 的 limit 选项。部分相关代码见图 6.8 所示。

```
count = (Attend.objects.all().filter(stu_no=condition.get_id(), attend_date__range=(
    condition.get_start_date(), condition.get_end_date()))).count()
page_result = PageQueryBean()
if count > 0:
    page_result.set_total_rows(count)
    page_result.set_current_page(condition.get_current_page())
    page_result.set_page_size(condition.get_page_size())
    param = []
    sql = '''select * from main.T_XUPT_ATTEND WHERE stu_no=''' + "'" + condition.get_id() + "'"
    if condition.get_start_date() is not None and condition.get_end_date() is not None:
        sql += ' and attend_date between ' + "'" + condition.get_start_date() + "'" + ' and ' + "'" \
               + condition.get_end_date() + "' "
    sql += 'limit ' + str(condition.get_start_row()) + ',' + str(condition.get_page_size())
    print(sql)
    items = Attend.objects.raw(sql)
    page_result.set_items(items)
```

图 6.8　分页相关代码

6.3 补签、审批

6.3.1 查看个人补签记录

在考勤记录表中，如果使用者存在异常信息，并成功的提交了补签，那么这个页面将显示使用者提交的信息。如果管理员没有同意也没有拒绝使用者的补签，信息会显示审核中，通过会显示已通过，同时如果个人考勤记录中那天没有其他的异常信息，则会把补签按钮隐藏，替换成绿色的正常信息；管理员拒绝了使用者的补签，补签记录中将会显示审核未通过。如图 6.9 所示。

图 6.9　补签记录

获取补签记录的主要后台代码如图 6.10 所示。

```
def get_re_attend(id):
    return ReAttend.objects.all().filter(stu_no=id)
```

图 6.10　获取补签记录

6.3.2 补签审核

当有学生申请补签时，管理员登录系统后，右上角的邮箱将会显示数字，表示有多少条补签没有处理，如图 6.11 所示。

图 6.11　审批页面

管理员可以选择通过该学生的补签申请也可以选择拒绝该学生的补签申请，如果是拒绝，学生依旧可以重新申请补签。管理员同意该申请或者拒绝，实际上后台仅仅做了一件事，就是更新数据库中的补签状态。获取申请补签记录后台代码见图 6.12：

```
def get_re_attend_list():
    return ReAttend.objects.all().filter(re_attend_status='1')
```

图 6.12　获取申请补签列表

如果非管理员点击了代办补签功能选项卡，则会跳到显示没有权限操作的页面。

6.4 管理员查看所有的考勤记录

查看所有的学生的考勤记录，只有是管理员登录，并且是在主页的功能选项卡才能进入。管理员可以选择时间范围、班级进行查询，每条记录最后的显示如果没有异常则是绿色的正常，否则是红色的异常。

6.5 Web 设计总结

Web 端设计了学生的登录，设定学生只能通过登录考勤系统来查看自己的考勤记录，并在考勤数据异常时进行选择申请补签。

Web 端的设计经历非常多的曲折，尤其是在分页设计的时候遇到了非常多的 BUG，在经过数次的寻求老师和同学的帮助，最后终于是完成了整个 Web 端的设计。

Web 端的功能一环扣一环，并且和现实十分接近。

第七章　总结

这个课题从 2017 年 12 月份拿到手，整整用了半年时间将它完成。期间经历了各种风风雨雨，有遇到过那种大起大落的感觉。从不知所措到现在成果初现，实在是翻阅了太多的资料，查看了各种文献，并于各种论坛搜索相关的知识，才有了本课题设计的内容。

笔者觉得设计毕设的这整个过程，能学到的东西真的非常之多，不仅能接触到平时没有接触到的知识，更为重要的是能培养自身在遇到问题时候的如何给出一个解决办法。从设计开始到结束，笔者明白突然明白了一个道理古人诚不我欺：办法一定比问题多。

尽管耗费了非常多的时间，并做了非常多的优化，但是笔者相信，基于树莓派的人脸签到系统，依旧不是一个完美的作品，这当然是因为笔者所涉及的知识面依旧太少的缘故，笔者将不断的学习，不断的在各种学习论坛中丰富自己的知识面。

目前，基于树莓派的人脸识别系统已经完成的功能，包括：

a、 自定义课表

b、 自定义系统的使用者

c、 刷脸签到

d、 刷脸登录、用户名密码登录

e、 考勤查看、补签

f、 审核考勤

笔者坚信，随着不断的学习，基于树莓派的识别签到系统一定会越来越完善，功能越来越多。

第八章　展望

随着时代的持续进步，人们考勤的方式越来越多元化，然而早期的考勤方式依旧是占比例最高的一种，但是这种刷卡式的考勤存在着诸多隐患，譬如代打卡现象严重，考勤参数设置不灵活等。基于树莓派的人脸考勤系统完美的解决了这个问题，它使用人脸识别技术，别人再也无法代刷卡了，同时自定义课表的功能使得整个系统变的灵活。所以，基于树莓派的人脸识别考勤系统具有非常好的应用前景。

笔者相信，本系统还可以有更加完美的设计，还可以增加更多实用的功能，比如可以使用网页拍照进行人脸识别并且配合地点考虑进行考勤，还可以在上课前几分钟给应该签到但是还没有签到的学生发送邮件或者短信提醒一下，还可以在每节课结束的时候给老师发微信告知哪些人没有签到或者迟到了。

这将是笔者接下来研究的方向，笔者相信，未来的基于树莓派的人脸识别考勤系统一定会更加完善，功能更加强大的！

致　谢

我首先要感谢我的论文指导老师：西安邮电大学自动化学院的马翔老师。马老师对我论文的研究方向做出了指导性的意见和推荐，在论文撰写过程中及时对我遇到的困难和疑惑给予悉心指点，提出了许多有益的改善性意见，投入了超多的心血和精力。我对马老师帮忙和关怀表示诚挚的谢意!同时，还要感谢西安邮电大学自动化专业的授课老师们和所有同学们，大家在西安邮电大学的学习中互相学习，互相帮忙，共同度过了一段完美难忘的时光。

此外，还要感谢朋友以及同学们在论文编写中带给的大力支持和帮忙，给我带来极大的启发。也要感谢参考文献中的作者们，透过他们的研究文章，使我对研究课题有了很好的出发点。

最后，谢谢论文评阅老师们的辛苦工作。衷心感谢我的家人、朋友，以及同学们，真是在他们的鼓励和支持下我才得以顺利完成此论文。

参考文献

[1] 深入浅出 MySQL[M]. 人民邮电出版社， 唐汉明， 2013.

[2] 李玉鹏，宋维，程超伟.基于树莓派的人脸识别考勤系统的开发与实现[J].单片机与嵌入式系统应用，2016，16(11):28-30+34.

[3] 秦小文，温志芳，乔维维.基于 OpenCV 的图像处理[J].电子测试，2011(07):39-41.

[4] 汤德俊. 人脸识别中图像特征提取与匹配技术研究[D].大连海事大学，2013.

[5] 苏祥林，陈文艺，闫洒洒.基于树莓派的物联网开放平台[J].电子科技，2015，28(09):35-37+41.

[6] 韩宇，张磊，吴泽民，胡磊.基于嵌入式树莓派和 OpenCV 的运动检测与跟踪系统[J].电视技术，2017，41(02):6-10.

[7] 丁忠俊.从 EER 模型到关系模型转换系统的设计与实现[J].计算机工程与应用，1997(10):3-7.

[8] 王冉阳.基于 Django 和 Python 的 Web 开发[J].电脑编程技巧与维护，2009(02):56-58.

[9] 齐金刚，李滔，李晋军.Django 框架 Web 数据查询分页技术研究[J].电子设计工程，2014，22(05):33-37.

[10] 嵩天，黄天羽，礼欣.Python 语言:程序设计课程教学改革的理想选择[J].中国大学教学，2016(02):42-47.

[11] Warren W. Gay. GPIO[M].Apress:2014-06-15.

[12] Nanlesta A. Pilgrim，Saifuddin Ahmed，Ronald H. Gray，Joseph Sekasanvu，Tom Lutalo，Fred Nalugoda，David Serwadda，Maria J. Wawer. Multiple sexual partnerships among female adolescents in rural Uganda: the effects of family structure and school attendance[J]. International Journal of Adolescent Medicine and Health，2015，27(3).

[13] Muhammad Sajjad，Mansoor Nasir，Khan Muhammad，Siraj Khan，Zahoor Jan，Arun Kumar Sangaiah，Mohamed Elhoseny，Sung Wook Baik. Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities[J]. Future Generation Computer Systems，2017.

附录 A　view.py 代码

```
import traceback

from django.shortcuts import render
from django.http import HttpResponse, HttpResponseRedirect
from sign_in_system.service import service
import base64
import os
from .models import QueryCondition
from .models import ReAttend
from .models import Person

# Create your views here.
class Controller:
    def all_list(self, request):
        range_date = request.GET['rangeDate']
        range_date = request.GET['rangeDate']
        class_name = request.GET['className']
        page_size = request.GET['pageSize']
        start = request.GET['start']
        current_page = request.GET['currentPage']
        id = request.session.get('user', None)
        condition = QueryCondition(id=id, range_date=range_date,
class_name=class_name)
        condition.set_page_size(page_size)
        condition.set_current_page(current_page)
        condition.set_start_date(range_date.split('/')[0])
        condition.set_end_date(range_date.split('/')[1])
        condition.set_start_row(start)
        # print(condition)
        result = service.get_all_list(condition)
        return HttpResponse(result)

    def all(self, request):
```

```
        ctx = {}
        id = request.session.get('user', None)
        ctx['count'] = 0
        if id == '06141083':
            ctx['count'] = service.get_msg_count()
        ctx['img'] = '../static/img/' + id + '.jpg'
        return render(request, 'all.html', ctx)

    def passIn(self, request):
        id = request.GET['id']
        msg = service.pass_re_attend(id)
        return HttpResponse(msg)

    def notPass(self, request):
        id = request.GET['id']
        msg = service.not_pass_re_attend(id)
        return HttpResponse(msg)

    def re_attend_list(self, request):
        id = request.session.get('user', None)
        ctx = {}
        ctx['img'] = '../static/img/' + id + '.jpg'
        ctx['count'] = 0
        if id == '06141083':
            ctx['count'] = service.get_msg_count()
            attend_list = service.get_re_attend_list()
            ctx['list'] = attend_list
            return render(request, 'reAttendApprove.html', ctx)
        else:
            li = service.get_user(id)
            ctx['li'] = li
            return render(request, 'no_permission.html', ctx)

    def reattend(self, request):
        ctx = {}
```

```
        id = request.session.get('user', None)
        ctx['count'] = 0
        if id == '06141083':
            ctx['count'] = service.get_msg_count()
        ctx['img'] = '../static/img/' + id + '.jpg'
        li = service.get_re_attend(id)
        ctx['li'] = li
        return render(request, 'reAttend.html', ctx)

    def create_re_attend(self, request):
        re_attend = ReAttend()
        try:
            re_attend.attend_status = request.GET['attendStatus']
            re_attend.attend_date = request.GET['attendDate']
            re_attend.stu_no = request.GET['userID']
            if ReAttend.objects.filter(attend_date=re_attend.attend_date,
attend_status=re_attend.attend_status,
                                       stu_no=re_attend.stu_no).exists():
                return HttpResponse('exits')
            re_attend.comments = request.GET['remark']
            re_attend.re_attend_starter =
Person.objects.filter(stu_no=request.GET['userID']).values('stu_name')[0][
                'stu_name']
            re_attend.current_handler = '彭大富'
            print(re_attend.attend_date)
            re_attend.save()
        except Exception:
            print(traceback.print_exc())
            return HttpResponse('error')
        return HttpResponse('ok')

    def attendList(self, request):
        range_date = request.GET['rangeDate']
        page_size = request.GET['pageSize']
        start = request.GET['start']
```

```
        current_page = request.GET['currentPage']
        id = request.session.get('user', None)
        condition = QueryCondition(id=id, range_date=range_date,
class_name="")
        condition.set_page_size(page_size)
        condition.set_current_page(current_page)
        condition.set_start_date(range_date.split('/')[0])
        condition.set_end_date(range_date.split('/')[1])
        condition.set_start_row(start)
        result = service.get_list(condition)
        print(result)
        return HttpResponse(result)

    def attend(self, request):
        ctx = {}
        id = request.session.get('user', None)
        ctx['count'] = 0
        if id == '06141083':
            ctx['count'] = service.get_msg_count()
        ctx['img'] = '../static/img/' + id + '.jpg'
        return render(request, 'attend.html', ctx)

    def home(self, request):
        ctx = {}
        id = request.session.get('user', None)
        ctx['count'] = 0
        if id == '06141083':
            ctx['count'] = service.get_msg_count()
        li = service.get_user(id)
        ctx['li'] = li
        ctx['img'] = '../static/img/' + id + '.jpg'
        return render(request, 'home.html', ctx)

    def logout(self, request):
        request.session['user'] = None
```

```
        return render(request, 'login.html')

    def login(self, request):
        return render(request, 'login.html')

    def checkUser(self, request):
        img = request.POST.get('img', None)
        if img is None:
            name = request.POST.get('username', None)
            pwd = request.POST.get('password', None)
            if name is None:
                return HttpResponse('用户名不能为空')
            id = service.login(None, name, pwd)
            if id != 0:
                request.session['user'] = name
                return HttpResponse('true')
            else:
                return HttpResponse('登录失败，账号或者密码错误')
        img_ = base64.b64decode(img)
        with open("./media/1.jpg", "wb") as fout:
            fout.write(img_)
        img_path = os.path.abspath("./media/1.jpg")
        id = service.login(img_path)
        if id != 0:
            with open('./static/img/' + id + '.jpg', 'wb') as f:
                f.write(img_)
            request.session['user'] = id
            return HttpResponse('true')
        else:
            return HttpResponse('登录失败，您可能不是我们的同学，又或
许光线不好')

    def add(self, request):
        return render(request, 'signup.html')
```

```
def add_face(self, request):
    img = request.POST['img']
    stu_no = request.POST['stu_no']
    msg = service.add(img, stu_no)
    if msg == 'OK':
        with open('./static/img/' + stu_no + '.jpg', 'wb') as f:
            f.write(base64.b64decode(img))
    return HttpResponse(msg)
```

附录 B　urls.py 代码

```
from django.contrib import admin
from django.urls import path
from sign_in_system import tests
from sign_in_system.views import Controller

ct = Controller()

urlpatterns = [
    path('admin/', admin.site.urls),
    path('s/', tests.s),
    path('q/', tests.q),
    path('show/', tests.show),
    path('check_login/', ct.checkUser),
    path('login', ct.login),
    path('logout/', ct.logout),
    path('add', ct.add),
    path('add_face', ct.add_face),
    path('home', ct.home),
    path('attend', ct.attend),
    path('attend/attendList', ct.attendList),
    path('reAttend.do', ct.create_re_attend),
    path('reAttend', ct.reattend),
    path('reAttend/list', ct.re_attend_list),
    path('reAttend/pass', ct.passIn),
    path('reAttend/notPass', ct.notPass),
    path('all', ct.all),
    path('all/list', ct.all_list),
]
```