

快速构建普适服务及其应用

摘要

普适计算 (Ubiquitous computing 或 pervasive computing), 又称普存计算、普及计算, 是一个强调和环境融为一体的计算概念, 而计算机本身则从人们的视线裡消失。在普适计算的模式下, 人们能够在任何时间、任何地点、以任何方式进行信息的获取与处理。本文通过自己动手构建一个普适服务应用场景来进一步了解普适计算以及展望普适计算未来的前景。实验借助了传统的个人计算机、平板电脑、网络摄像机和网络继电器等设备, 使用了近两年发展火热的 Android 操作系统, 构建了一个应用场景, 在其中用户可以使用平板电脑进行拍照签到、查看实验室历史签到数据以及完成远程监控的功能。该应用不仅能够在平板电脑上运行, 还可以在搭载了 Android 操作系统的手机上运行, 用户无需其他多余的设备, 即可完成许多复杂的操作。该应用还可以针对不同的事件作出不同的响应, 如有人签到时发送微博提醒, 当所有人都离开实验室时自动关闭灯光等。从该应用场景我们可以看到普适计算的未来, 那就是各种设备都变得十分智能, 而人们不必通过计算机才能获得服务, 使用一个小小的手机甚至是做一些动作, 说一些话就能获取想要的服务与信息。

关键词: 普适计算, 应用, Android, 智能

BUILDING OF PERVASIVE SERVICES AND APPLICATIONS

ABSTRACT

Pervasive computing (also known as ubiquitous computing) is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. More formally, ubiquitous computing is defined as "machines that fit the human environment instead of forcing humans to enter theirs." In this work, a pervasive application scenario is built to get a better understanding of pervasive computing and look into the future of pervasive computing and its applications. In this work, a traditional PC, an Android tablet, a WLAN camera and an internet relay is used to build a pervasive application scenario with the help of the popular Android operating system. Using this application on the tablet, one can check-in by taking a picture of himself by the tablet, view the previous check-in data, monitor the lab remotely and turn the light in the lab on/off remotely. The application will also make intelligences to different incidents, such as post a tweet after receiving a check-in record, turn off the light if all the people left the lab and so on. From this application we could see the future of the pervasive computing, the machines are becoming more and more intelligent, people will no longer need a traditional PC to get what they want, they could use their phones or even just make some gestures or just say something.

Key words: pervasive computing, applications, Android, intelligence

目 录

第一章 绪论.....	1
第二章 应用场景的简介和描述.....	2
2.1 实验设备.....	2
2.2 基本功能描述.....	2
2.2.1 签到功能.....	2
2.2.2 查看历史签到数据功能.....	2
2.2.3 远程监控功能.....	2
2.3 应用整体架构及描述.....	2
2.4 本章小结.....	3
第三章 应用具体实现步骤.....	4
3.1 服务器上 ftp 服务器的开启.....	4
3.2 网络摄像机的设置.....	5
3.3 网络继电器的配置.....	7
3.4 数据库的配置.....	7
3.5 平板电脑与服务器的通信协议.....	8
3.5.1 请求人员名单.....	8
3.5.2 添加新成员.....	8
3.5.3 移除一位成员.....	8
3.5.4 签到信息.....	8
3.5.5 获取签到数据.....	9
3.5.6 获取实验室监控图像.....	9
3.5.7 开灯.....	9
3.5.8 关灯.....	9
3.6 服务器端的配置与实现.....	9
3.6.1 配置.....	9
3.6.2 运行流程.....	9
3.6.3 实现.....	10
3.7 平板电脑端图形界面的设计与实现.....	10
3.7.1 总体界面设计.....	10
3.7.2 总体界面实现.....	11
3.7.3 签到页面设计与实现.....	12
3.7.4 查看历史数据页面的设计与实现.....	13
3.7.5 远程监控页面的设计与实现.....	14
3.7.6 设置界面的设计与实现.....	14
3.7.7 其他界面.....	15
3.8 平板电脑端的配置与功能实现.....	16
3.9 本章小结.....	17
第四章 遇到的问题和相应的解决方法或思路.....	18

4.1 人脸识别的实现.....	18
4.2 签到数据的存储.....	19
4.3 签到界面显示图片时发生内存溢出的问题.....	19
4.4 发送微博功能的实现.....	20
4.5 安全性的保证.....	21
4.6 如何控制更多设备.....	21
4.7 将服务器端移植到 Plug Computer 或云上的可能性.....	21
4.8 中文乱码的解决.....	21
4.9 多线程的控制.....	22
4.10 重复人员管理.....	23
4.11 Android 开发中的一些问题.....	23
4.11.1 SDK 版本.....	23
4.11.2 Activity.....	24
4.11.3 ViewPager、Fragment、ActionBar 标签.....	25
4.11.4 Handler.....	25
4.12 签到和离开的判断以及后续操作问题.....	26
4.13 平板电脑与服务器通信时签到时间的传递.....	26
4.14 本章小结.....	27
第五章 结论.....	29
谢辞.....	31

第一章 绪论

普适计算 (Ubiquitous computing 或 pervasive computing), 又称普存计算、普及计算, 最开始是由 Mark Weiser 于 1991 年所提出的一种简单的人机互动概念, 定义是一种将信息处理流程整合进每一天中的所有事物 (object) 和活动 (activities) 当中的后台式模型 (post-desktop model), 主要观念在于把计算机嵌入现实生活并且可以与现实生活融为一体的假设, 以达到在生活中无所不在的运用计算机的资源的目的, 还可以将计算机引入移动设备和同步系统, 并让人们甚至没有意识到这些设备的活动。普适计算是一个强调和环境融为一体的计算概念, 而计算机本身则从人们的视线里消失。在普适计算的模式下, 人们能够在任何时间、任何地点、以任何方式进行信息的获取与处理。

本文将描述在实验室中构建一个普适服务的应用场景的过程, 以此来对普适计算有更深入的了解, 并且对普适计算的前景进行展望。该应用场景描述了一个实验室的签到系统和一个远程监控系统, 实验室的成员每天来到实验室时可以使用实验室内的平板电脑或自己的手机拍摄一张自己的照片, 完成签到; 当老师不在实验室时, 可以使用平板电脑或手机远程监控实验室的情况, 并且可以远程控制实验室内灯光的打开和关闭; 系统本身也很智能, 在有人签到或有系统无法识别的人员试图签到时, 系统会发送一条微博提醒, 在实验室内的人员全部离开时, 系统会自动关闭实验室内灯。

在本文所描述的应用场景中, 服务器封装了数据库、网络摄像机和网络继电器等的通信和编程接口, 为平板电脑等其他设备提供了人员列表、签到数据和远程监控的三大类服务。其中人员列表服务中包含获取人员列表、添加新成员、移除成员三个具体服务, 签到数据服务中包括发送签到数据和查看签到数据两个具体服务, 远程监控服务中包括远程监控实验室图像以及开灯和关灯三个具体的服务。除了自己实现的服务器所提供的这八个服务之外, 该应用还使用了 face.com 所提供的免费人脸识别的服务。在平板电脑端的应用中, 通过和用户进行交互, 并使用上述的几个服务, 完成签到、查看数据和远程监控等功能。

用户签到的数据存储在服务端端的数据库中, 服务器端的程序使用普通的 JAVA SE 实现, 使用 JDBC 技术对数据库进行查询和修改的操作。网络继电器的输出可以通过向其发送特定的 TCP 数据包进行控制, 这一操作是由服务器与网络继电器建立 socket 连接后向其输出流中写入相应的控制数据来完成的。除了人脸识别的服务是平板电脑直接与 face.com 的服务器进行通信外, 其他的操作都是平板电脑通过调用服务器端的服务来完成的, 后期可以考虑将人脸识别的服务也集成到服务器端。

通过在本课题中对构建普适服务的研究和搭建普适应用场景的实践, 希望能够对普适计算能够有更加具体和深刻的理解, 增强自身的研究和开发技术能力, 并对未来普适计算在人们生活中的应用进行展望。

第二章 应用场景的简介和描述

2.1 实验设备

- (1) 个人计算机一台（装有 Java 运行环境，开启 mysql、ftp 服务）
- (2) Android 平板电脑一台（艾诺 NOVO7 elf 精灵，系统版本 Android4.0.3）
- (3) 网络摄像机一台（TP-Link TL-SC3130G，带有定时传送图像至指定 ftp 功能）
- (4) 网络继电器一个（深圳市精锐达电子科技有限公司 4 路常开输出网络远程控制多路电源开关）
- (5) 灯泡若干

2.2 基本功能描述

2.2.1 签到功能

用户在每天来到实验室和离开实验室时，使用平板电脑的摄像头拍摄自己的面部图像，拍摄完成后系统会对所拍摄的图像进行人脸识别操作，如果用户成功被识别出，则向服务器发送数据，在数据库中记录用户的本次签到或离开操作。如果用户无法被正确识别出，可以在管理员监督下进行机器学习和训练的操作。

2.2.2 查看历史签到数据功能

用户可以在平板电脑或服务器中查看历史签到数据。

2.2.3 远程监控功能

实验室中有一台网络摄像机实时监控实验室，在平板电脑上可以获取网络摄像机所捕捉到的画面，并可以在平板电脑上通过开灯和关灯的两个按钮来控制实验室内灯的开关。

2.3 应用整体架构及描述

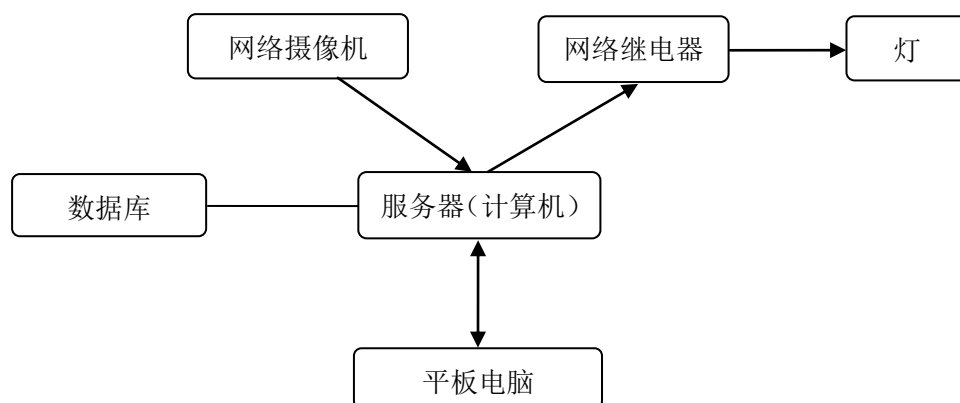


图 2-1 应用整体架构图

在服务器上开启 ftp 服务，将网络摄像机设置为每分钟捕捉图像并传送至服务器；

签到数据记录在服务器的数据库中；

平板电脑上的程序启动时先向服务器请求名单（可以在平板电脑上添加或移除人员），有人签到时，先对照片进行人脸识别（使用了 face.com 提供的人脸识别 API），若识别为名单中的人员，则向服务器发送该人员的签到信息，服务器对数据库进行更新；

在平板电脑上查看签到数据时，需向服务器请求数据，数据将以日期分组的列表形式显示；

使用远程监控功能时，平板电脑每分钟向服务器请求一次图片（网络摄像机上传至服务器的图片）并显示在屏幕上；

用户点击平板电脑上的开灯或关灯按钮时，平板电脑向服务器发送开灯或关灯的请求，服务器发送相应指令至网络继电器，完成开灯或关灯。

2.4 本章小结

本章主要介绍了本次构建普适应用场景的实验中所用到的实验设备、应用的一些基本功能及描述，以及应用的整体架构和各个组件之间的关系，并且对基本功能和各个实验设备之前的联系进行了简要的描述。

第三章 应用具体实现步骤

本章主要讨论应用的具体实现步骤，包括各个设备的配置，以及服务器端和平板电脑端程序的配置、设计与实现。要完成远程监控的功能，首先需要对网络摄像机进行配置。要配置网络摄像机将图片上传到服务器上，那么首先需要在服务器上开启 FTP 服务，然后对网络摄像机进行设置，定时将图片上传至服务器。要实现对灯的开关控制，需要对网络继电器的 IP 地址等进行设置，以便服务器向其发送指令。签到数据使用数据库进行存储，那么就需要在服务器上安装并运行 MySQL 数据库服务

在这些设备和服务配置完成后，就要开始服务器端的程序和平板电脑端的程序设计工作。而平板电脑端的应用因为有图形界面，于是需要讨论平板电脑端应用的图形界面的设计，然后针对界面的布局和用户操作对应用的功能进行实现。

3.1 服务器上 ftp 服务器的开启

测试时使用的服务器运行的操作系统是 Windows 7，因此使用了 FileZilla Server 这款软件建立 FTP 服务器。

安装后运行软件，首先点击菜单栏的 Edit 按钮，在下拉菜单中点击 Users 按钮，弹出 Users 设置窗口，如图 3-1 所示；

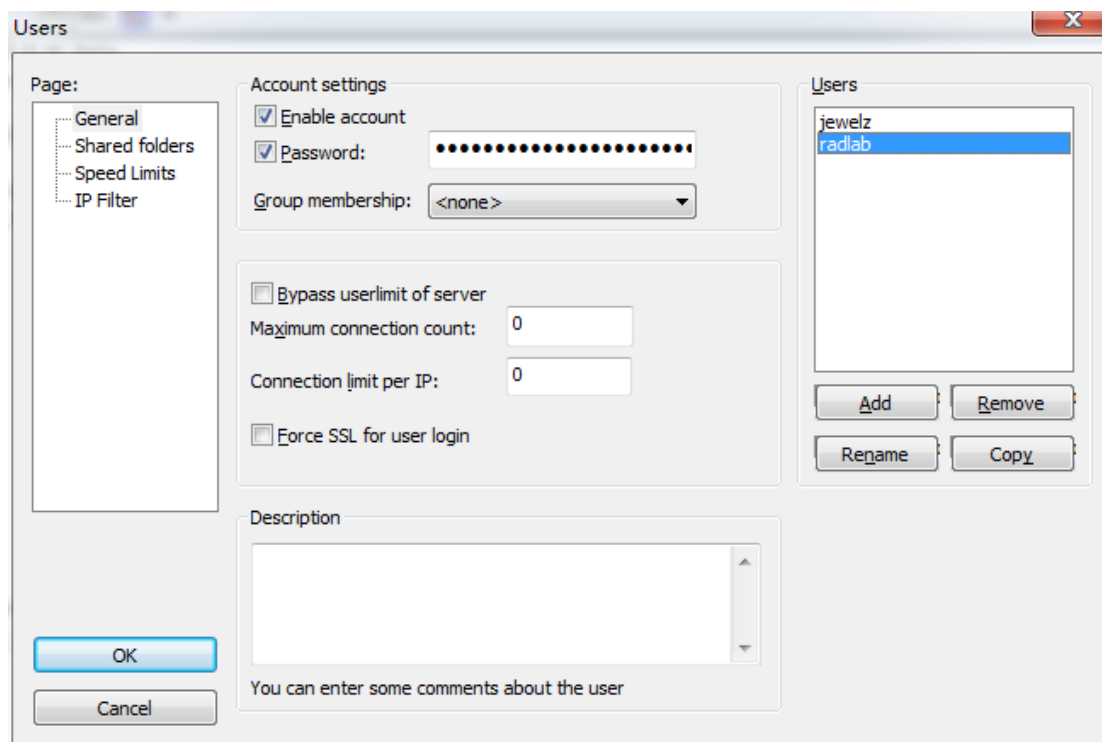


图 3-1 Filezilla Server 设置界面 1

点击右侧的 Add 按钮，添加一个新用户，在新弹出的窗口中输入用户名后点击 OK 按钮确认，如有必要可以为该用户添加密码；

点击窗口左侧的 Shared folders，在右侧选择刚才添加的用户，在中间选择一个文件夹作为该用户登录后所在的根目录，将权限设置为不可读可写可删除文件（该用户应是网络摄

像机，功能是向服务器上传图片，因此需要可写权限而不需要可读权限，为避免冗余图片，可以令每次上传的图片有相同的文件名，于是服务器任何时刻只会有一张图片，每次平板电脑向服务器请求图片时服务器只需将这张图片传送过去，要实现覆盖功能，便需要为该用户赋予可删除文件的权限），如图 3-2 所示。设置完毕后点击 OK 保存。

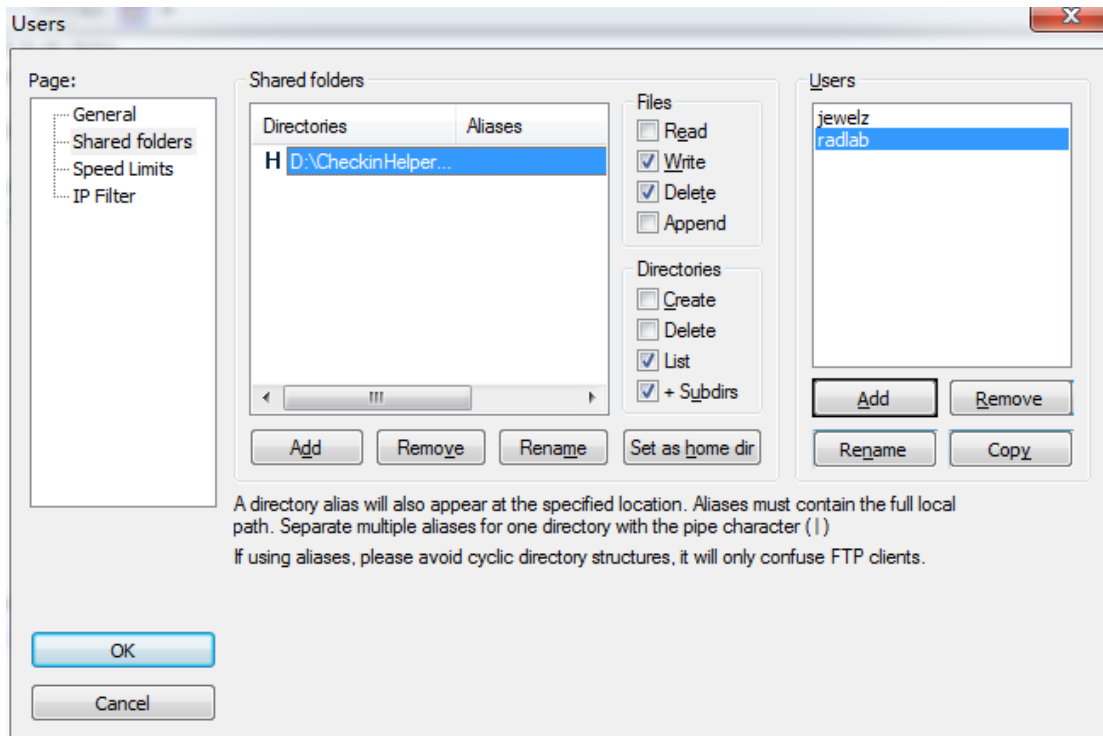


图 3-2 Filezilla Server 设置界面 2

3.2 网络摄像机的设置

首次使用时先使用有线网络连接，将实验室路由器连出的以太网线接入网络摄像机，插上电源后等待电源指示灯亮起；

安装并运行产品附带的光盘内的 Intelligent IP Installer 软件，搜索网络摄像机的 IP 地址，如图 3-3 所示；



图 3-3 Intelligent IP Installer 软件界面

在电脑上使用浏览器访问刚才得到的 IP 地址，默认用户名和密码均为 admin，登录后可以设置无线网络；

进入设定、进阶设定、FTP 客户端、一般设定，设置 FTP 服务器信息，填入服务器的 IP 地址及刚才添加的用户名和密码，点击确认按钮保存，如图 3-4 所示；

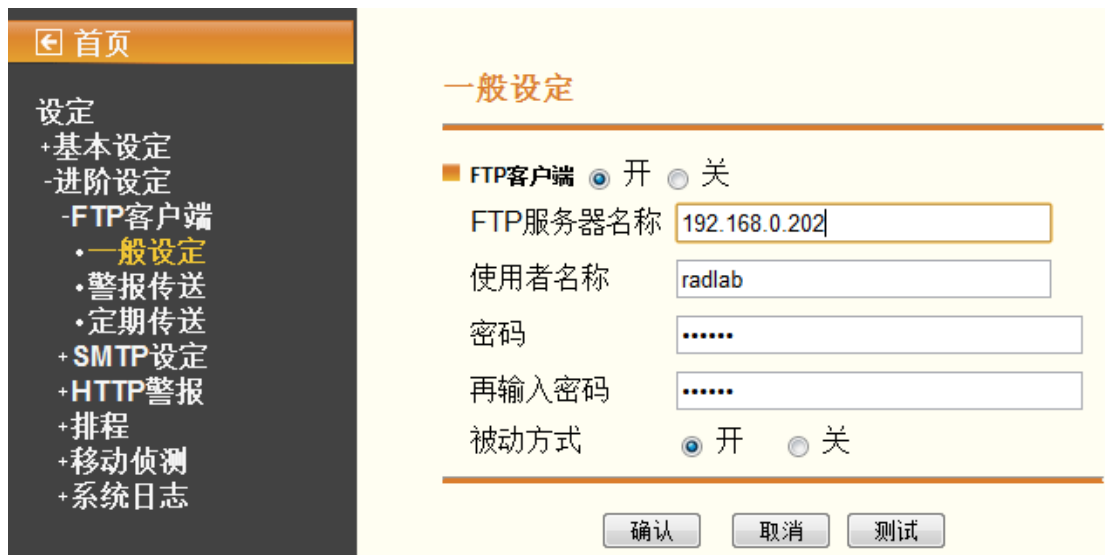


图 3-4 网络摄像机设置界面 1

进入定期传送设置页面，设置远程输出文件名，设置传送时间间隔，点击确认按钮保存，如图 3-5 所示。

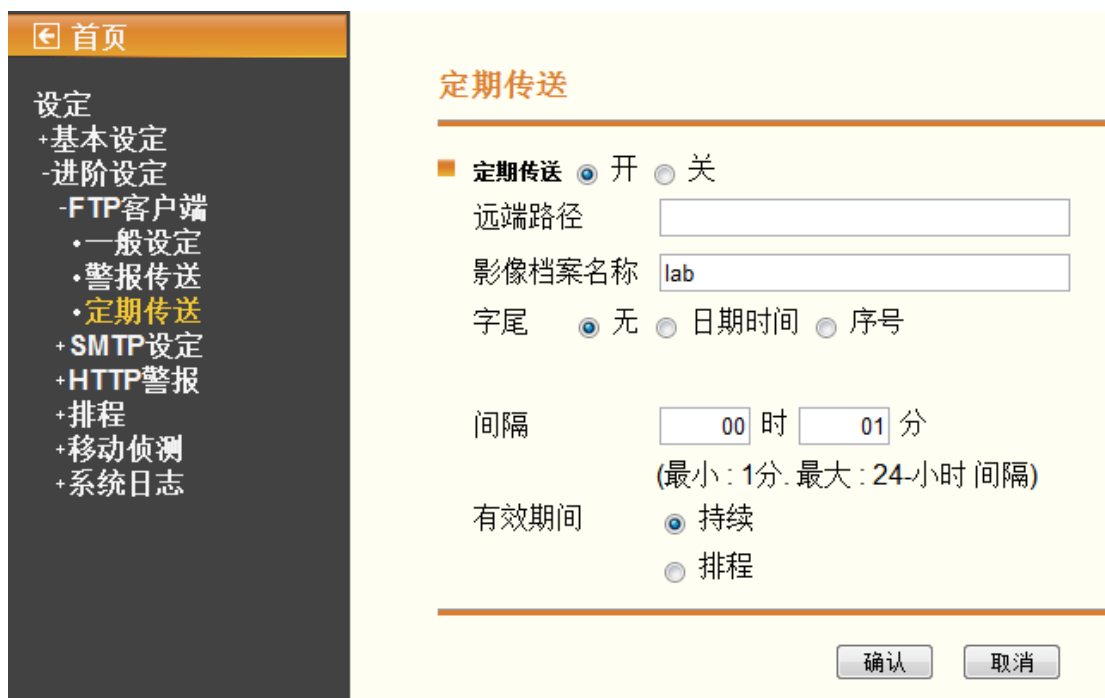


图 3-5 网络摄像机设置界面 2

3.3 网络继电器的配置

实验使用的网络继电器是深圳市精锐达电子科技有限公司 4 路常开输出网络远程控制多路电源开关（其他信息可访问 <http://pxwko.cnelc.com/ShowProduct.asp?ProductId=100800518> 查看）。

使用时首先需要设置继电器的 IP 地址，继电器上有三个跳线“ADDR1”、“ADDR0”和“SET”，当“SET”设置为 1 时使用默认 IP 地址 192.168.X.250，当“SET”设置为 0 时使用内部 IP 地址，可以通过发送数据包设置，实验室中的路由器使用 192.168.0.x 网段，因此将“SET”跳线设置为 1 使用默认 IP 地址；“ADDR1”和“ADDR0”用于设置默认 IP 地址中的 X 位，

“ADDR1”和“ADDR0”跳线的值与 X 的值的对应关系如表 3-1 所示。

表 3-1 网络继电器 IP 地址设置对应表

ADDR1	ADDR0	X
0	0	0
0	1	1
1	0	2
1	1	3

由于实验室的路由器使用 192.168.0.x 网段，因此将“ADDR1”跳线和“ADDR0”跳线都设置为 0。该继电器默认的通信端口号是 2000，因此服务器需要向继电器发送数据包时，只需先与继电器建立 socket 连接，即与 192.168.0.250:2000 建立 socket 连接，然后向其发送数据。

实验中只需用到发送开和关的命令到继电器出口的功能，因此只需了解设置继电器出口状态的命令。

命令格式：0x03,index_lsb,index_hsb,0x00,0x05,0x00,0x00,0x08,output,0x00,0x00,0x00

Index_lsb,index_hsb 为命令序号，从 0 开始计数；

Output 为输出开关的位变量；

举例：打开第 2 位，其他关闭：

0x3,0x25,0x0,0x0,0x5,0x0,0x0,0x8,0x2,0x0,0x0,0x0。

实验中将两个灯泡分别接在了继电器第 3 位和第 4 位的输出口，因此要完成开灯的功能，需要发送的数据为：

0x3,0x25,0x0,0x0,0x5,0x0,0x0,0x8,0x0C,0x0,0x0,0x0

要完成关灯功能，需要发送的数据为：

0x3,0x25,0x0,0x0,0x5,0x0,0x0,0x8,0x0,0x0,0x0,0x0。

3.4 数据库的配置

在服务器计算机上安装 MySQL Server，并使用命令行登入；

新建名为 radlab 的数据库：

CREATE DATABASE radlab;

连接 radlab 数据库：

CONNECT radlab;

新建名为 radlab 的表用于存储签到数据：

```
CREATE TABLE radlab
(
  cdate    DATE(yymmdd),
  name     VARCHAR(20),
  intime   VARCHAR(8),
  outtime  VARCHAR(8) DEFAULT '00:00:00',
  PRIMARY KEY (cdate, name)
);
```

该表包含了签到日期 (cdate)、签到者 id (name)、签到时间 (in)、离开时间 (out) 四栏，其中签到日期和签到者 id 组成了这张表的组合键，这就限制了每天每个人至多只会有一条记录。某个人某天第一次签到时，会在数据库中加入一条记录，记录了当前的日期、签到人 id 和签到时间，而离开时间则使用了默认值 00: 00: 00，这是为了防止有人离开时忘记 check-out 而使记录中出现 null 导致不方便的情况。当该人当天不是第一次签到时（即数据库中已存在当天该签到者 id 的记录），就将本次签到时间作为离开时间，并修改数据库中当天该签到者的签到记录，并且只会修改离开时间这一栏。这样一来，若某人某天没有签到，则数据库中没有记录；若某人某天只签到一次，则数据库中会将这次签到的时间作为签到时间，而离开时间会使用默认值 00: 00: 00；若某人某天签到次数多于或等于两次，则数据库中会将该人第一次签到的时间记录为签到时间，而将最后一次签到的时间记录为离开时间。

3.5 平板电脑与服务器的通信协议

平板电脑与服务器使用网络套接字 socket 进行连接，在服务器上执行一个 ServerSocket 的循环，每次平板电脑需要发送数据时与服务器建立一个 socket 连接，目前平板电脑与服务器的通信共有 8 条指令：

3.5.1 请求人员名单

命令：namelist

描述：平板电脑通过 socket 发送字符串“namelist”向服务器请求人员名单和管理员密码（使用 MD5 加密，用于权限管理），服务器收到该指令时首先传输管理员密码，然后逐一传输人员名单，每个人员信息一行，形式为[id] [name]，如“zs 张三”。人员名单以每个人的信息一行，[id] [name]的形式作为一个文件存储在服务器上。

3.5.2 添加新成员

命令：add [id] [name]

示例：add zs 张三

描述：平板电脑通过 socket 向服务器依次发送“add”、新成员 id、新成员姓名，中间以空格隔开，服务器收到该指令时，对人员名单进行修改，并对存储人员信息的文件进行写回操作。

3.5.3 移除一位成员

命令：remove [id]

示例：remove zs

描述：平板电脑通过 socket 向服务器依次发送“remove”、成员 id，中间以空格隔开，服务器收到该指令时，对人员名单进行修改，并对存储人员信息的文件进行写回操作。

3.5.4 签到信息

命令：check [time] [id1] [id2] ...

示例: `check 1234 zs ls`

描述: 平板电脑通过 `socket` 连接向服务器一次发送“`check`”, 签到时间(长整型变量, 值为签到时间距离 1970 年 1 月 1 日的毫秒数)和签到者 `id` (至少有一个, 多个 `id` 中间以空格隔开), 服务器收到该指令时, 对每个签到者进行数据库操作, 若该签到者当天已经签到过, 则将该签到者当天离开时间修改为本次签到时间, 若该签到者当天没有签到过, 加入一条签到记录, 日期和签到时间通过毫秒数进行计算。

3.5.5 获取签到数据

命令: `SELECT` 语句

示例: `SELECT * FROM radlab`

描述: 平板电脑通过 `socket` 连接向服务器发送一条字符串, 内容为一条 `sql` 语句(必须是 `select` 语句, 可以在其中进行筛选、排序、分组等操作), 服务器收到该指令后执行这条 `sql` 语句, 将结果按照每条结果一行, 每条结果每一栏间使用空格隔开的形式返回给平板电脑用于显示和处理。

3.5.6 获取实验室监控图像

命令: `labimage`

描述: 平板电脑通过 `socket` 连接向服务器发送“`labimage`”, 服务器收到该指令后, 将网络摄像机传输至服务器的图片通过 `socket` 连接发送给平板电脑, 用于显示。

3.5.7 开灯

命令: `light on`

描述: 平板电脑通过 `socket` 连接向服务器发送“`light on`”, 服务器收到该指令后, 向网络继电器发送开灯的控制信息, 完成开灯操作。

3.5.8 关灯

命令: `light off`

描述: 平板电脑通过 `socket` 连接向服务器发送“`light off`”, 服务器收到该指令后, 向网络继电器发送关灯的控制信息, 完成关灯操作。

3.6 服务器端的配置与实现

3.6.1 配置

服务器端有三个配置/数据文件: 人员名单列表文件、实验室图片文件、网络继电器配置文件。人员名单列表文件记录了已经记录的实验室人员列表, 文件内容为每个人的信息一行, 每行内的格式为 `id name`, 例如:

`zs 张三`

`ls 李四`

.....

由于中文的特殊性, 需要为每个人分配一个只有英文字母的 `id`, 便于存储和管理, 如用于数据库存储, 用于人脸识别标识不同的人等。实验室图片文件是由网络摄像机经由 `FTP` 服务器上传至服务器端的, 每次上传的图片文件都相同, 因此每次上传新的图片都会覆盖前一张图片, 这样一来既能避免很多图片文件占用过多的磁盘空间, 又能方便服务器端代码的编写, 只需要一个文件名即可读取文件。网络继电器配置文件中保存了网络继电器的 `IP` 地址和端口号, 中间使用空格隔开。

3.6.2 运行流程

服务器开始运行时, 首先读取网络继电器配置文件和人员名单列表文件。因为每个人的 `id` 都不相同, 人员名单列表文件的内容被储存进一个哈希表, 哈希表的键是每个人的 `id`,

而哈希表的值是该 id 对应人员的姓名（可以重复），这样又能很好的区分同名人员。使用哈希表不仅便于快速查找某个 id 对应的人员的姓名，也能在新加入成员和删除成员时快速地判断这个 id 是否已经存在。

读取配置文件完毕后，服务器端启动一个 `ServerSocket` 的循环，接受其他客户端的连接。服务器接收 8 种指令，上面已经讨论过。

3.6.3 实现

服务器端功能比较简单，主要是读取文件和根据客户端发送的不同指令完成数据库操作和向网络继电器发送数据包的操作。

由于人员名单的文件中有中文字符，读取文件时需要注意字符集的问题，否则读取的文件会出现乱码。除此之外，因为需要将名单发送给客户端，因此也要注意服务器和客户端通信时也要注意字符集的问题，否则也会出现乱码的现象。

服务器启动时将人员名单读入哈希表，这样可以快速查询人员列表，避免经常进行文件读取操作。但是需要注意的是，每次对人员列表进行修改，即添加新成员和删除成员时只是对人员列表的哈希表进行操作，需要将改动写回文件，否则改动无法保存。

3.7 平板电脑端图形界面的设计与实现

3.7.1 总体界面设计

根据整个项目的功能，平板电脑端的界面总共可以分为三块：用于签到的界面、用于显示和查看历史签到记录的界面以及显示实验室图片用于远程监控的界面，再加上一个用于修改服务器的 IP 地址和端口号的设置界面，总共需要 4 个界面来完成。因为这四个界面之间没有什么从属关系，都是同一级别的，因此可以使用标签页的方式进行显示，如下图共有四个标签页，分别是“CHECK-IN/OUT”（对应签到界面）、“STATISTICS”（对应查看历史签到记录界面）、“LAB”（对应远程监控界面）以及“SETTINGS”（对应设置界面），点击某个标签，下面的内容就会被替换为相应的标签页对应的界面。每个标签页右上角会有一些菜单项方便用户进行操作如签到、刷新数据、开灯关灯等。最后，为了提升用户体验，标签页被做成了可滑动的，即用户可以像浏览图片那样在屏幕上左右滑动来在不同的标签页之间进行切换。

在 `android.support.v4.view` 包中提供了一个 `ViewPager` 类，可以让用户在很多页的数据间左右滑动来切换数据，开发者需要实现一个 `PagerAdapter` 来将需要切换的页面加进去。需要注意的是 `ViewPager` 这个类是 Android 支持包（`Support Package`）所提供的，目前正处于早期的设计和开发中，所提供的 API 有可能会在今后的兼容包的更新中发生变化，到时候编译新版本的程序时需要修改程序的代码以使用新的 API。

`ViewPager` 经常同 `Fragment` 类配合在一起使用，其中 `Fragment` 是一种很方便的提供和管理每一个页面的生命周期的方法。因此我们可以将需要实现的四个页面使用 4 个 `Fragment` 类来实现，然后加入到 `ViewPager` 中，用户就可以进行左右滑动切换页面的操作了。

只有左右滑动的界面，而没有导航界面的话用户就会十分困惑，有时候并不知道自己当前正处在哪一个页面，那么就需要使用 `ActionBar` 的标签来同时进行导航操作。每一个 `ActionBar` 的标签对应一个上面所实现的 `Fragment`，点击某一个标签时页面就切换到对应的 `Fragment`，当用户左右滑动切换到其他的页面时，`ActionBar` 的焦点也相应的移动到被切换到的 `Fragment` 所对应的标签上。另外每个 `Fragment` 需要接受用户的一些操作，可以使用添加菜单项的方式加入到 `ActionBar` 中去，显示在标签列表的右侧，用户点击了某个菜单项后进行相应的操作。

3.7.2 总体界面实现

在程序的 Activity 类（本应用只有一个 Activity，其他界面均由 Fragment 类来实现，在一个 Activity 中切换显示）中新建一个继承 FragmentPagerAdapter 的类 TabsAdapter 用于向 ViewPager 中添加 Fragment，同时 TabsAdapter 还需要实现 ActionBar.TabListener 接口和 ViewPager.OnPageChangeListener 接口以便在用户点击标签时切换页面以及在用户左右滑动切换页面时切换标签的焦点：

```
public static class TabsAdapter extends FragmentPagerAdapter implements  
    ActionBar.TabListener, ViewPager.OnPageChangeListener
```

其中比较重要的几个方法有：

addTab 方法，用于在 Activity 的 onCreate 方法中向 ViewPager 加入 Fragment；

onPageSelected 方法，在用户滑动切换到某一个页面后调用，此时需要将 ActionBar 的标签焦点切换到对应的标签：

```
public void onPageSelected(int position) {  
    mActionBar.setSelectedNavigationItem(position);  
};
```

onTabSelected 方法，在用户点击某个标签后调用，此时需要将页面切换到用户点击的标签所对应的页面：

```
public void onTabSelected(Tab tab, FragmentTransaction ft) {  
    Object tag = tab.getTag();  
    for (int i = 0; i < mTabs.size(); i++) {  
        if (mTabs.get(i) == tag) {  
            mViewPager.setCurrentItem(i);  
        }  
    }  
}
```

3.7.3 签到页面设计与实现

设计:

签到页面主界面是一张图片，显示的是上一次用户签到或训练时所拍摄的图片，如图 3-6 所示。

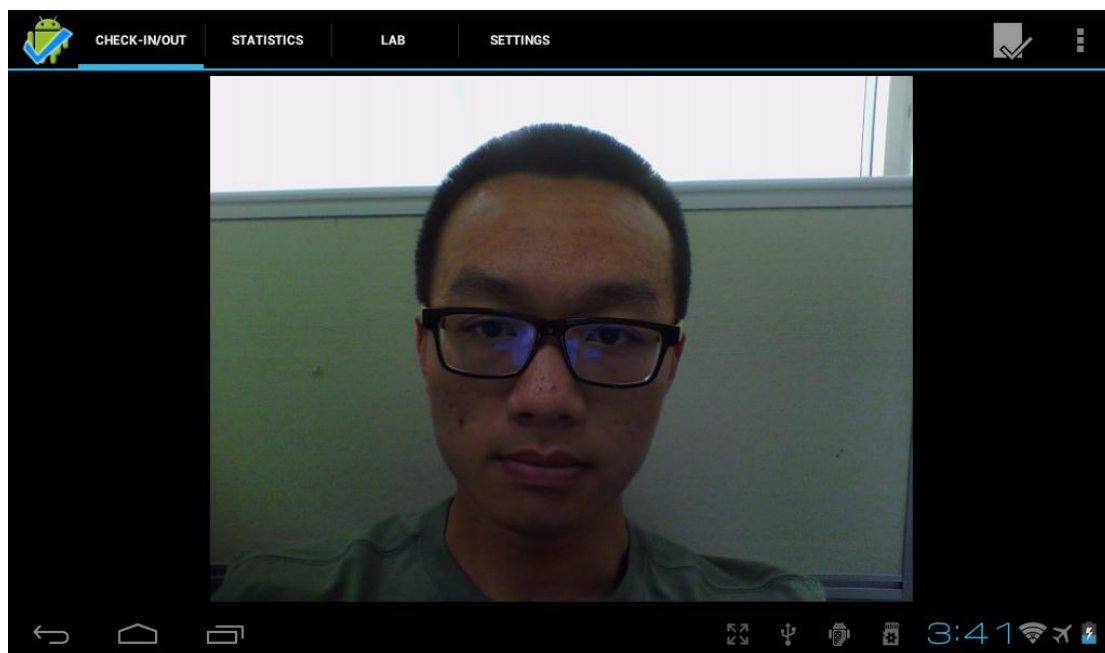


图 3-6 签到界面

签到界面右上角的 ActionBar 上有四个菜单项，其中签到菜单项总是显示，剩下的三个菜单项训练、添加新成员、移除成员处于隐藏状态，用户点击签到菜单项旁边的三个点时会显示出来，如图 3-7 所示。

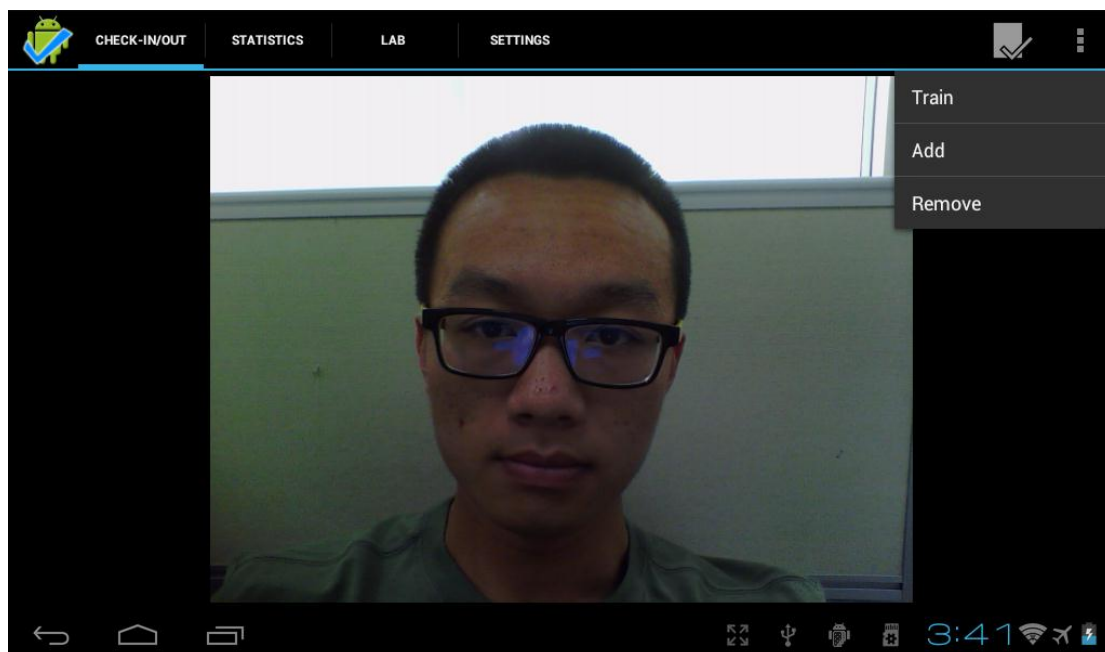


图 3-7 签到界面中被隐藏的按钮

实现:

在程序启动时和用户每次拍照后，读取平板电脑外部存储根目录下的 temp 目录中的 temp.jpg 文件，使用其替换签到界面上的图片内容。

向 ActionBar 添加四个 MenuItem，并设置签到菜单项总是显示，其他三个菜单项隐藏，并指定每个菜单项被点击后的监听器。

3.7.4 查看历史数据页面的设计与实现

设计：

主界面是包含了所有历史签到数据的列表，可以上下滑动，列表按照日期分组，右上角有一个刷新按钮，点击后会向服务器请求最新的签到数据，用于更新屏幕上的签到数据列表，如图 3-8 所示。

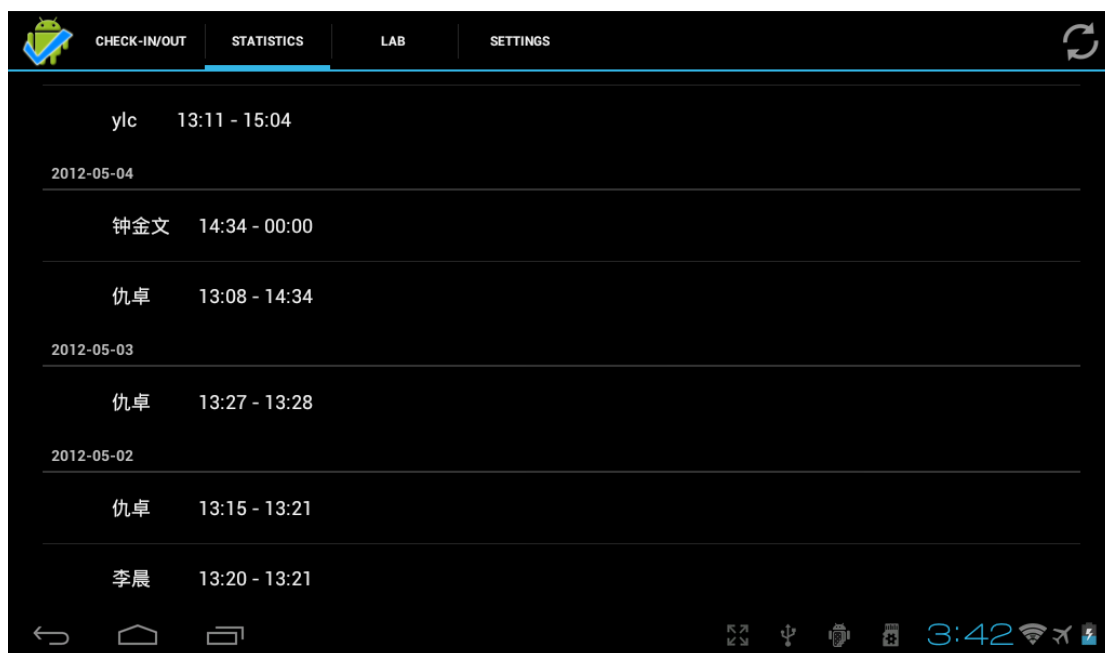


图 3-8 查看历史数据页面

实现：

该页面的 Fragment 继承了 PreferenceFragment，PreferenceFragment 本身就是一个列表，并可以方便地利用 Preference 等控件动态地添加和修改列表的内容，因为列表使用签到日期进行分组，因此使用签到日期作为 PreferenceCategory，当天的签到数据都作为 Preference 加入到这个 PreferenceCategory 中。除此之外 Preference 有默认的 UI 样式，跟系统的 UI 风格比较统一。

当刷新按钮被用户点击后，平板电脑向服务器请求签到数据，接收到服务器返回的签到数据后，清空列表中的数据，对每一行数据进行解析，构建新的数据列表。首先从一行数据中得到签到日期、签到人员 id、签到时间和离开时间 4 个数据，然后程序根据签到日期查找屏幕上是否存在当天的 PreferenceCategory，如果已经存在，那么直接构造一个新的 Preference 并加入这个 PreferenceCategory 中，如果不存在对应该签到日期的 PreferenceCategory，则新建一个 PreferenceCategory，将这个 PreferenceCategory 的 key(用于查找该 PreferenceCategory) 和 title(用于显示给用户看) 都设置为签到日期的字符串（形如“2012-05-14”），然后构造新的 Preference 加入这个 PreferenceCategory 中。

3.7.5 远程监控页面的设计与实现

设计：

远程监控页面同签到页面十分类似，主界面是一张图片，显示的是由网络摄像机所拍摄并上传至服务器的图像，右上角有 ON 和 OFF 两个按钮，分别代表开灯和关灯的操作，如图 3-9 所示。

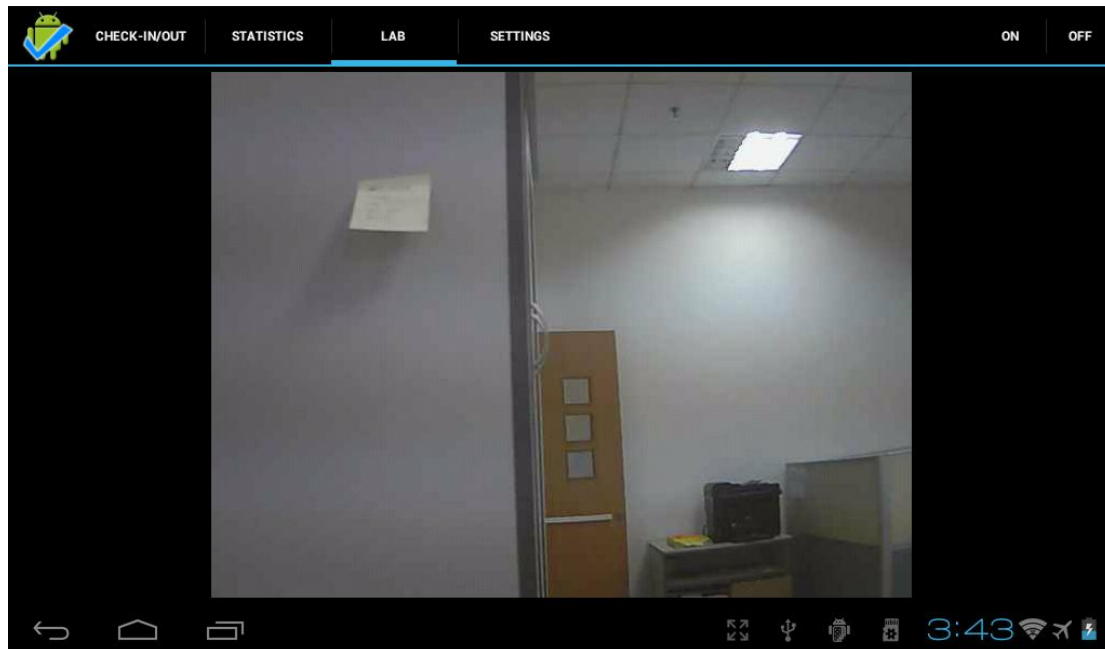


图 3-9 远程监控页面

实现：

平板电脑端每分钟向服务器请求一次图片，并显示在屏幕上。当用户点击 ON 按钮或 OFF 按钮时，平板电脑端向服务器发送相应的开灯和关灯的请求，然后服务器向网络继电器发送相应的开灯和关灯的指令，完成开灯和关灯的操作。

3.7.6 设置界面的设计与实现

设计：

设置界面比较简单，有两个设置项：服务器的 IP 地址和端口号，如图 3-10 所示。

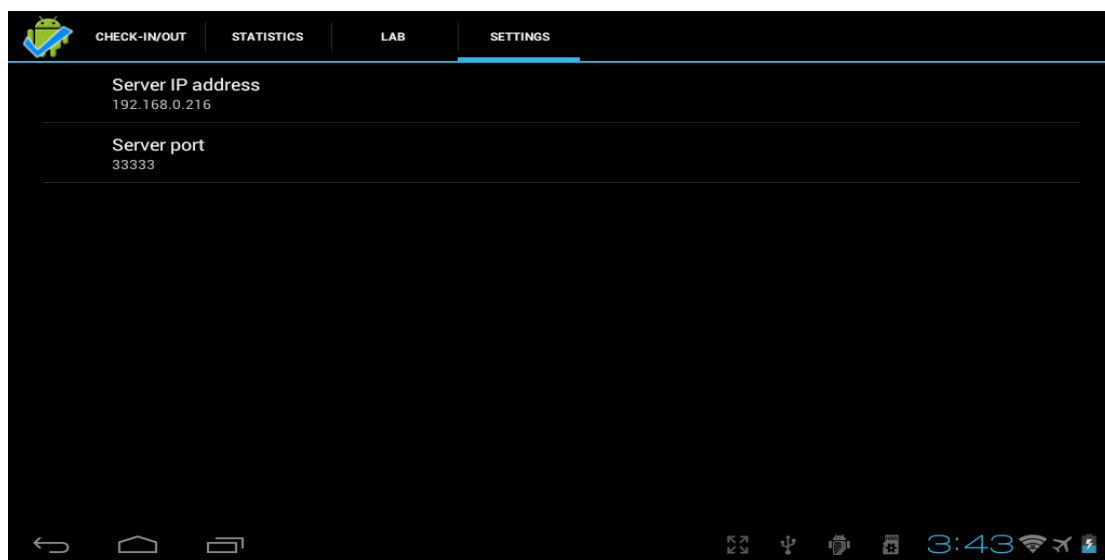


图 3-10 设置界面

实现：

设置界面也是使用 PreferenceFragment 实现，在其中添加了两个 EditTextPreference。

3.7.7 其他界面

用户签到后识别成功，弹出欢迎界面，如图 3-11 所示。

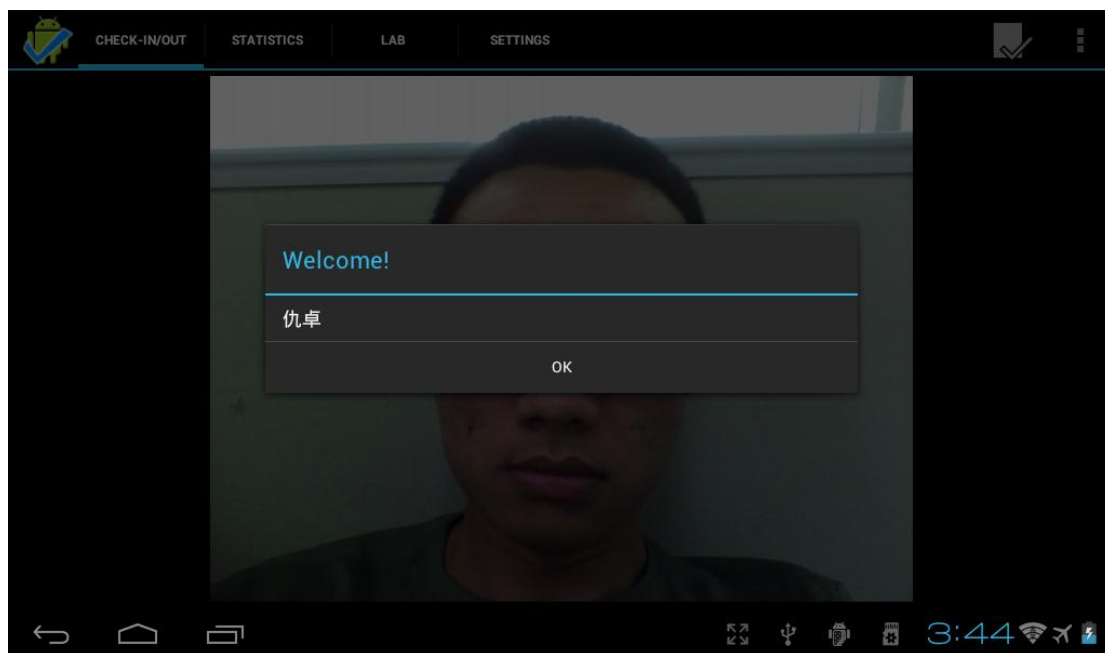


图 3-11 欢迎界面

用户点击训练按钮拍照完毕后或用户点击移除成员按钮后，弹出对话框要求用户输入要训练的人员 id 或要移除的人员 id 以及管理员密码，如图 3-12 所示。

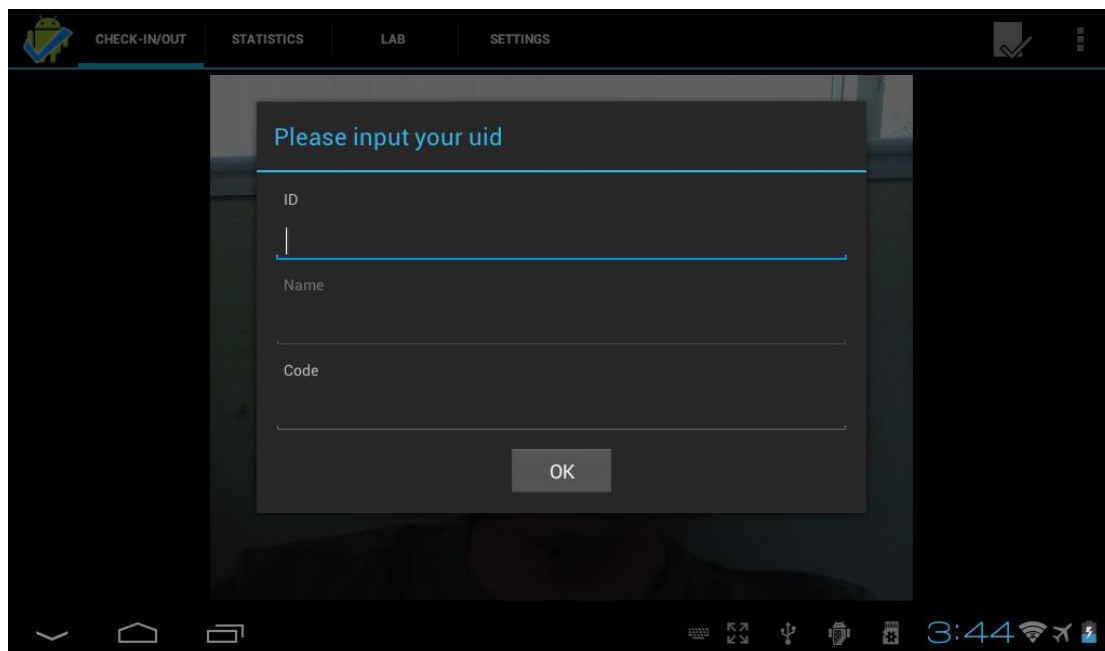


图 3-12 对话框 1

用户点击添加成员按钮后，弹出对话框要求用户输入新加入的成员的 id、新加入的成员的姓名以及管理员密码，如图 3-13 所示。

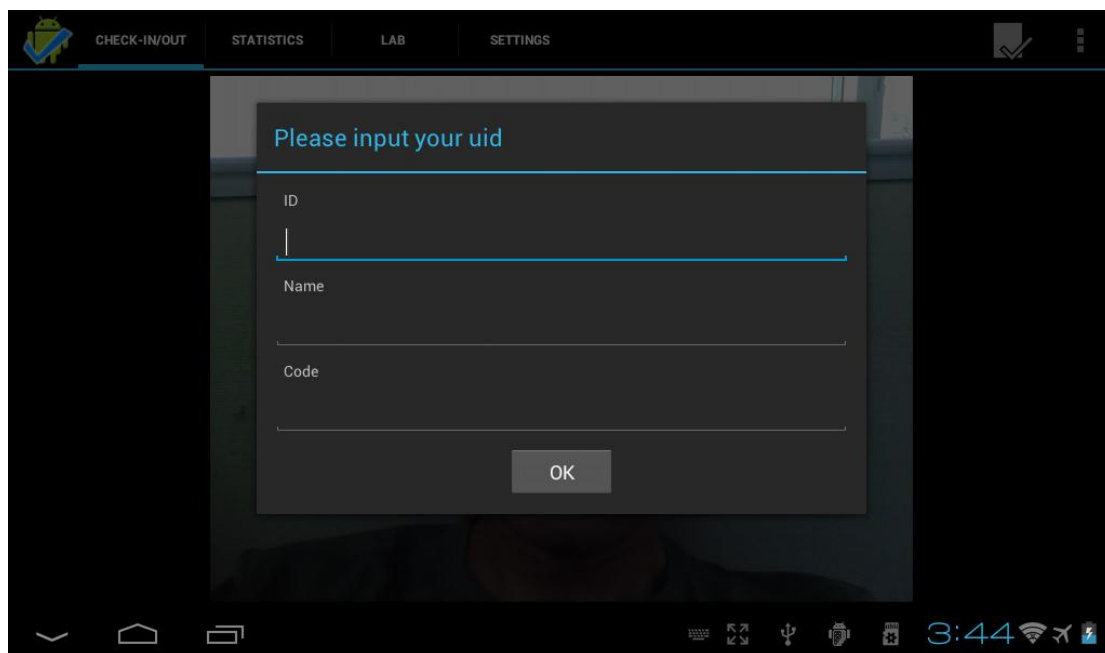


图 3-13 对话框 2

3.8 平板电脑端的配置与功能实现

平板电脑端需要存储的数据包括：服务器的 IP 地址和端口号、上一次拍照（签到或训练）的图片以及从服务器端获得的实验室的图像。

服务器的 IP 地址和端口号使用 Android 系统自带的 SharedPreference 存储方式进行存储，该存储方式非常适合存储此类配置信息，并且可以和 Android 的 Preference 控件进行配合，方便实用。

上一次拍照的图像保存在平板电脑外部存储（SD 卡）根目录中的 temp 文件夹中，图片文件名为 temp_ori.jpg。由于直接使用拍摄的原图进行显示和人脸识别时因为原图比较大，会造成性能的降低，增加运行时间，用户在操作时会有卡顿的感觉，因此另外保存了一张对原图进行压缩后的图片，图片名为 temp.jpg，同样保存在平板电脑外部存储根目录的 temp 文件夹中，大小降低为原图的 10%，在不失去所需主要图像信息的前提下大大缩短了运行和响应时间，增加性能。

从服务器端获取到的实验室内图片保存在平板电脑外部存储根目录中的 temp 文件夹中，文件名为 lab.jpg。每次获取到新的图片后，会将旧的图片覆盖，避免产生大量无关数据，占用存储空间。

平板电脑端程序在初始化时，首先从程序存储的 SharedPreference（可以在程序的设置标签页中进行修改）中读取服务器的 IP 地址和端口号。在平板电脑上启动程序后，可以看到程序有 4 个标签页，首先显示的是第一个标签页，签到标签页。签到标签页在初始化时，首先向服务器发送“namelist”指令请求人员列表和管理员密码，并将人员列表存储在哈希表中，哈希表的键是人员 id，哈希表的值是该人员的姓名，存储方式同服务器端的人员列表哈希表一致。然后程序从平板电脑外部存储根目录的 temp 文件夹中读取上次拍摄的照片，并显示在屏幕上。签到标签页上方有四个按钮，签到按钮、训练按钮（隐藏）、添加新成员按钮（隐藏）和移除一位成员按钮（隐藏）。

用户点击签到按钮时，程序调用 `startActivityForResult` 方法调用系统的照相机程序，并传入了本次操作类型（整型，有 0 和 1 两个参数，分别对应于签到和训练两个操作）和输出文件的路径参数等，只要用户确认了拍摄的照片，系统的照相机程序就会将图片保存到指定的路径，即平板电脑外部存储根目录的 `temp` 文件夹中，文件名为 `temp_ori.jpg`。被 `startActivityForResult` 调用的程序，会给调用者返回一些信息，如执行结果以及结果对应于哪一个操作等。程序在 `onActivityResult` 方法中接收这些返回信息，如果结果是执行正常则将新拍摄的图片压缩后显示在屏幕上，如果结果对应于签到的操作，那么程序会读取压缩过后的图片，并对其进行人脸识别的操作，如果顺利识别出某个或某几个成员，则弹出对话框提示识别成功，并在对话框中显示被识别出的成员的姓名，同时程序向服务器发送这些人在此时签到的信息，然后将图片上传至微博，并加入一些描述信息，如某某某刚刚签到等。如果没有能够识别出成员列表中的成员，则弹出对话框提示识别失败，可以取消操作或进行训练操作。

用户点击训练按钮时，程序调用 `startActivityForResult` 方法调用系统的照相机程序，传入表示训练的操作参数和输出文件路径。拍摄完成后，程序在 `onActivityResult` 方法中接收返回信息，如果结果是执行正常则将新拍摄的图片压缩后显示在屏幕上，如果结果对应于训练的操作，则在屏幕上显示一个对话框，要求用户输入用户 `id`（需要被训练的用户 `id`，即刚才照相的用户的 `id`）和管理员密码（避免被随意训练，造成结果错误），点击确认后开始对图片进行人脸检测和训练的操作。若无法识别出人脸则会弹出错误提示的对话框，当连续出现三次错误（无论是签到还是训练）时，程序会发送一条微博提醒有陌生人，并且将图片上传至微博。

用户点击添加新用户按钮时，会弹出一个对话框，要求用户输入希望添加的新用户的 `id` 和新用户的姓名以及管理员密码（权限管理），输入完毕后，平板电脑会向服务器发送添加新用户的请求，并将新用户的 `id` 和姓名发送至服务器。该操作执行完毕后，平板电脑会向服务器再次请求人员列表名单，以更新本地的人员列表名单。

用户点击移除一位成员按钮时，会弹出一个对话框，要求用户输入希望移除的用户的 `id` 和管理员密码，输入完毕后，平板电脑会向服务器发送删除用户的请求，并发送要删除的用户的 `id` 至服务器。该操作执行完毕后，平板电脑会向服务器再次请求人员列表名单，以更新本地的人员列表名单。

3.9 本章小结

本章主要描述了构建普适应用场景的过程及该应用的实现方法，本章详细描述了 FTP 服务器、网络摄像机以及网络继电器的配置，描述了数据库的配置以及签到数据在数据库中的存储方式，即每人每天至多会有一条签到记录，这条签到记录包含了签到日期、签到人员 `id`、签到时间以及离开时间 4 个数据，如果离开时忘记签到那么离开时间将会使用默认值替代，多次签到时离开时间会使用最后一次签到的时间。本章还描述了平板电脑端的程序与服务器进行通信时的 8 个动作：请求人员名单、添加新成员、移除一位成员、签到信息、获取签到数据、获取实验室监控图像、开灯、关灯。本章 3.8 节讨论了服务器端的配置、运行流程和实现。除此之外，本章着重讨论了平板电脑端界面的设计与实现和平板电脑端功能的实现。平板电脑端为用户展现了一个标签页式的界面，用户可以通过点击标签切换不同的页面，出了这种传统的方式，用户还可以通过左右滑动来切换页面，这种方式更加适合触摸操作，提升了用户体验。

第四章 遇到的问题和相应的解决方法或思路

4.1 人脸识别的实现

由于本课题主要研究内容是构建普适服务,并且之前对机器学习、模式识别领域了解不多,因此无法独立完成人脸识别的算法,即使能够实现,效果也必定无法令人满意,于是需要寻找一个开源的人脸识别项目或提供人脸识别 API 的服务。首先尝试了几个在互联网上搜索到的用 Java 实现的人脸识别的开源项目,有一些因为系统配置和依赖软件包等问题效果十分不理想或根本无法使用,还有一些对于照片的拍摄要求太高,都无法令人满意。后来找到了 face.com,这个网站提供免费的人脸识别的可编程接口,使用时程序将图片上传至该网站的服务器进行处理,处理结束后服务器会将结果返回给客户端,经测试该网站的人脸识别准确率很高,而且能够识别出一张照片中的多个人,功能十分强大。客户端与服务器的通信是通过 json 传递数据,但是网站提供了封装好的 Java 软件开发包,只需将这些源文件加入平板电脑端的程序源代码中,加入所依赖的软件包,然后在自己的程序中调用相关的接口即可使用,无需考虑同网站服务器通信时的具体数据,十分方便易用。

要使用 face.com 所提供的人脸识别服务,首先需要在该网站注册一个开发帐号,注册好后新建一个应用程序(免费用户可以建立 5 个应用程序),这个应用程序会有唯一的 API 公钥和密钥,在代码中使用这两个公钥和密钥,服务器就可以知道是哪一个应用程序正在与服务器交换数据。然后需要建立一个新的命名空间,因为本项目是实验室签到系统,因此新建一个名为“radlab”的命名空间,如果一个成员的 id 是“zs”,那么服务器接收到的 id 就是“zs@radlab”,这样不同命名空间中就可以有相同 id 的人员。

要能够识别出一个人,服务器首先需要经过训练,即给服务器一张图片,告诉服务器这张图片里的人是某某某,经过这样的训练以后,再给服务器一张图片,服务器就可以计算出这张图片上的人是刚才训练的某某某的概率是多少。在训练时,首先需要从平板电脑拍摄一张照片,然后将这张图片压缩后(减小照片大小,缩短处理时间)上传至服务器,请求服务器检测照片之中的人脸。训练时拍照要注意镜头中不要出现其他人脸,否则可能导致训练了错误的信息。服务器将检测的人脸信息返回给平板电脑时,平板电脑端得到检测出的人脸的 id(每个人脸有不同的 id),然后向服务器发送请求将这个人脸的 id 保存在某个人员的 id(如“zs@radlab”)的名下,之后再向服务器发送请求对该人员进行训练,服务器就会对该人员 id 名下所保存的所有人脸进行学习和训练。

在签到时,就需要直接向服务器请求识别的服务。用户在平板电脑上点击签到的按钮后,拍摄一张照片,然后平板电脑会将这张照片压缩,然后发送到服务器,向服务器请求人脸识别的服务。服务器首先会检测照片上的人脸,然后对每一个人脸进行识别操作,计算出这张人脸是每一个已经训练的人员的概率,然后将这些结果返回给平板电脑。在平板电脑端接收到服务器返回的数据后,我们只需也对每一张脸进行处理,找到这张脸中概率最大的人员 id,如果这个概率比较大,那么我们就认为这张脸是这个人。对所有人连处理完后,把每张脸对应的人员 id 存储在一个列表中,用于后续的用户提示和签到操作。

4.2 签到数据的存储

签到数据使用 MySQL 数据库存储在服务器上。所有的签到数据都存储在一张表里，表名为 radlab，共有四栏数据，分别是签到日期 (cdate)、签到人员 id (name)、签到时间 (intime) 和离开时间 (离开时间)。签到日期使用数据库的 DATE 类型存储，签到人员 id 使用字符串存储，签到时间和离开时间使用定长字符串存储，格式为 “xx:xx:xx”，记录了小时、分钟和秒数。

其中签到日期和签到人员 id 共同组成这张表的组合键，因此保证了每人每天在数据库中最多只有一条记录。离开时间有默认值 00:00:00，这是因为当用户某天第一次签到之前，数据库中没有当天该人员的签到记录，当该用户签到后，会向数据库中添加一条记录，而签到时间保存在 intime 这一栏数据中，此时离开时间这一栏是没有数据的，因此为防止出现不可预计的错误，为离开时间设置了默认值，如果用户忘记离开时签到，那么离开时间就是默认值 00:00:00。当某人某天签到多于或等于两次签到时，服务器会发现数据库中已经存在了当天该人员的签到记录，于是将使用本次签到的时间覆盖原记录中的离开时间，即多次签到会不断修改离开时间，而不会改变签到时间。

4.3 签到界面显示图片时发生内存溢出的问题

在用户签到或训练拍照后，程序需要将刚才拍摄的照片显示在屏幕上，具体操作是调用 BitmapFactory 的 decodeFile 方法，传入参数为图片文件的路径，得到一个 Bitmap 对象，如果直接将这个 Bitmap 对象显示在屏幕上，由于拍摄的照片分辨率比平板电脑的分辨率高很多，经常会报出 OutOfMemory 内存溢出的错误，于是需要使用缩略图的方式避免发生错误。主要是通过 BitmapFactory.Options 类来实现。

Options 类中有一个属性 inJustDecodeBounds，如果该值设置为 true，那么将不返回实际的 Bitmap，也不为其分配内存空间，但是允许查询图片的信息，这样就可以避免内存溢出。可以通过 Options.outHeight 和 Options.outWidth 获取图片的原始高度和图片的原始宽度。然后比较图片原始高度除以当前显示高度和图片原始宽度除以当前显示宽度两个值，取较大的一个值为缩放参数，将 Options.inSampleSize 设置为这个缩放参数，然后将 Options.inJustDecodeBounds 的值设置为 false，再次调用 BitmapFactory.decodeFile 方法，将返回一个缩放为合适显示大小的缩放后的 Bitmap，然后将这个 Bitmap 对象显示在屏幕上，就不会再出现内存溢出的错误。

另外，可以将缩放后的 Bitmap 另存为一张图片，下次程序启动时可以直接使用这张缩放后的图片，而无需再次进行缩放操作。除此之外，因为人脸识别的操作需要将图片上传至 face.com 的服务器，缩放后的图片的大小减小了很多，而并没有丢失人脸的细节，可谓是一箭双雕。

在保存缩放后的图片时，首先建立一个 FileOutputStream 准备向其写入数据，然后将缩放后的 Bitmap 压缩后写入文件：

```
FileOutputStream out = new FileOutputStream("temp.jpg");  
bmp.compress(Bitmap.CompressFormat.JPEG, 80, out);  
fout.close();
```

注意 Bitmap 的 compress 方法的第二个参数，这个参数的意义是压缩图片的品质，范围是 0 到 100，经测试使用参数 80 后得到的输出文件大小仅仅是使用参数 100 后得到的输出文件的大小的 10%~20%，而图片内容则看不出明显的差别，这就相当于在进行人脸识别时需要上传的数据变为了仅有原来的 10%~20%，而且基本不会影响识别操作，大大地降低了

程序的响应速度，提升了性能和用户体验。

4.4 发送微博功能的实现

希望实现有人签到或离开时发送一条微博，并上传当时拍摄的照片的功能，因此需要使用新浪微博官方提供的程序接口。要实现发送微博的功能首先需要验证和登录帐号，新浪微博官方提供了两种授权方式 OAuth 接口和 OAuth2.0 接口，其中 OAuth2.0 授权接口使用后有时间限制，即只有授权成功后的一小段时间内可以使用，超过时间后就需要再次授权，而 OAuth 授权则可以做到一次授权，永久使用。虽然 OAuth2.0 的安全性更高，但是经常需要授权十分不方便，于是选择使用 OAuth 授权接口。

除了直接使用新浪微博提供的使用 json 传递数据的接口外，新浪微博还提供了多种语言的封装好的软件开发包，进入新浪微博 Java SDK 的下载地址，其中有两个开发包可供下载：weibo4j-oauth2 和 Weibo4J，其中 weibo4j-oauth2 使用的是 OAuth2.0 授权方式而 Weibo4J 使用的是 OAuth 授权方式，因此选择使用 Weibo4J 开发包。

首先在新浪微博上注册一个帐号，然后创建一个应用程序，记录该应用程序的公钥和密钥。下载 Weibo4J 开发包，将下载好的压缩包解压后，将解压好的项目文件导入 eclipse 中。打开 weibo4j 包中的 Weibo.java 文件，将 Weibo 类的 CONSUMER_KEY 成员和 CONSUMER_SECRET 成员的值分别修改为创建应用程序时获得的 App Key 和 App Secret，然后在 eclipse 中运行 weibo4j.examples 包中的 OauthUpdate.java。运行成功后程序会调用浏览器打开一个网页，要求登录微博和授权，登录后会得到一个 PIN 码，将该 PIN 码输入 eclipse 的控制台，并按回车确认，控制台会输出本次程序所获得的 Access token 和 Access token secret，有了这两个值，以后就可以无需授权而发送微博了。

在 weibo4j.examples 包中新建一个 MyWeibo.java 文件仿照 OauthUpdate 类中的方法编写一个 update 方法，输入参数为要发送的微博内容，但是现在无需用户每次获取和输入 PIN 码，新建了一个 Weibo 类的对象后，直接调用该 Weibo 对象的 setToken 方法，传入的参数就是刚才控制台输出的 Access token 和 Access token secret 这两个字符串。然后调用该 Weibo 对象的 updateStatus 方法，传入参数是一个字符串，即想要发送的微博。至此基本的发送微博功能已经实现。

因为发送微博的同时要上传图片，给 MyWeibo 类再添加一个 upload 的方法，输入参数为要发送的微博内容和要上传的图片路径。新建一个 Weibo 对象后，调用该对象的 setToken 方法，设置 Access token 和 Access token secret，然后调用该 Weibo 对象的 uploadStatus 方法，传入参数为要发送的微博内容和要上传的文件（File 类型）。这样发送微博的同时上传图片的功能就实现了。

发送微博时还可以带有位置信息，于是可以给程序中所发送的微博加上位置信息，即实验室的位置。在地图上大致确定实验室所处的经纬度，在 MyWeibo 类的 upload 方法中，调用 uploadStatus 方法时，增加两个参数：纬度和经度，即调用 uploadStatus 方法时传入了四个参数，依次是要发送的微博（String 类型）、要上传的图片（File 类型）、纬度（Double 类型）和经度（Double 类型）。

这样一来，就实现了发送带有位置信息的微博，同时上传图片的功能。在有人签到或离开时，可以发送一条微博提示“某某某来到/离开了实验室，现在实验室里有某某某和某某某”，同时将签到时所拍摄的图片上传。当程序检测到连续三次识别失败时，程序发送一条微博提醒无法识别，注意陌生人，同时上传第三次失败时所拍摄的照片。

4.5 安全性的保证

因为在平板电脑端可以进行训练、添加新成员和删除一位成员的操作，如果不采取一些措施保证安全性的话，任何人都可以任意训练，随便修改成员列表，造成灾难性的问题。因此在进行训练、添加新成员和删除一位成员等重要操作时，会要求用户输入一个管理员密码，即用户需要在管理员的监督下完成这些重要的操作。仅仅是输入密码还不够，因为黑客可能会监听服务器或客户端，截取他们所发送的数据，这样密码就被暴露了。因此在服务器向平板电脑发送管理员密码时，并不是发送明文的密码，而是使用某种加密算法加密后的密码，而在平板电脑端，当用户进行重要操作输入管理员密码时，平板电脑会使用同样的加密算法对用户所输入的密码进行加密，然后将加密后的内容同服务器传输过来的内容进行对比，依次来验证管理员授权。

4.6 如何控制更多设备

现阶段的实验中使用了一个继电器控制两个 20V 的灯泡，如果要实际控制实验室的日光灯，只需使用或增加一块更大电压的继电器作为开关。使用继电器实验开关的功能同样可以用于空调、饮水机等电器，可以实现当所有人都离开了实验室时，平板电脑向服务器发送关闭电器的请求，服务器向继电器发送相应的指令，从而关闭实验室的日光灯、空调、饮水机等，无需人为操作。如果使用了可调亮度的灯，则可以利用平板电脑的光线传感器，配合一个可调节电压继电器实现根据室内亮度的大小调节灯的亮度的功能。

4.7 将服务器端移植到 Plug Computer 或云上的可能性

现阶段服务器是运行在一台个人计算机上，操作系统为 Windows 7 X64，后期可以考虑将服务器端移植到实验室的服务器上，或是其他设备上。因为服务器端使用 Java 语言进行编写，签到数据使用 MySQL 数据库进行存储，鉴于 Java 语言和 MySQL 数据库的跨平台特性，可以很方便地将整个服务器端移植到其他的平台和操作系统上。

Plug Computer 是一种小巧的便携性计算机，只要将其插在插座上 Plug Computer 就会自动开机运行，多数运行 Linux 操作系统。Plug Computer 具有低功耗和廉价的特点，非常适合在家庭或办公室中作为小型服务器进行使用。后期可以将服务器端的部分运行在 Plug Computer 上，同平板电脑端进行通信，并对网络继电器发出指令。但是 Plug Computer 存储空间较小，因此可以把数据库存储在实验室长期运行的服务器上，Plug Computer 访问数据库时只需将数据库地址中的“localhost”改为实验室服务器的域名或 IP 地址即可，对于平板电脑端来说没有任何区别。

最近一段时间在计算机科学领域云计算可谓是一大热门研究方向，本系统同样可以借助云计算的思想。将服务器运行在云上，比如多台 Plug Computer 构成的云，平板电脑同云进行通信，获取服务，而云端则可以有多台设备可以提供服务，若某一台设备发生故障，则可以利用其它设备继续提供服务，这样更增加了服务器的可靠性，而对于平板电脑端来说，无需做太多的修改，因为接口并没有发生变化。

4.8 中文乱码的解决

因为人员名字可能是中文，因此在人员名单列表文件的读写时以及平板电脑端和服务器端进行通信时需要注意字符集的问题，否则中文字符会出现乱码现象。为了统一和方便，都

使用了通用的 UTF-8 编码方式。

在读取人员名单列表文件时，使用如下的方法指定使用 UTF-8 编码进行读取：

```
BufferedReader reader = new BufferedReader(  
    new InputStreamReader(new FileInputStream("filename"), "UTF-8")  
);
```

在写入人员名单列表文件时，使用如下的方法指定使用 UTF-8 编码进行写入：

```
PrintWriter writer = new PrintWriter(  
    new BufferedWriter(  
        new OutputStreamWriter(new FileOutputStream("filename"), "UTF-8")  
    )  
);
```

当平板电脑端同服务器端建立了一个 socket 连接之后，双方对数据的读写都使用 UTF-8 编码：

```
BufferedReader reader = new BufferedReader(  
    new InputStreamReader(socket.getInputStream(), "UTF-8")  
);  
PrintWriter writer = new PrintWriter(  
    new OutputStreamWriter(  
        new BufferedOutputStream(socket.getOutputStream()), "UTF-8")  
);
```

4.9 多线程的控制

平板电脑端的界面中有一个标签页是显示网络摄像机上传至服务器的所拍摄的实验室内的图片，因为网络摄像机被设置为每分钟向服务器上传一次图片（可设置的最小间隔就是 1 分钟），因此在平板电脑端最快只需每一分钟向服务器请求一次图片，而且当用户从该标签页切换到其他的标签页或者返回主页最小化了程序或关闭了程序时，就没有必要继续每分钟向服务器请求一次图片。因此需要实现一个要显示图片时每分钟向服务器请求一次图片，不需要显示图片时就暂停向服务器请求图片的功能。

因为网络环境有可能比较复杂，有时网络传输延迟较高时，发送一张图片会花费很多时间，如果将同服务器进行网络通信的操作都写在平板电脑端的 UI 主线程里的话，在等待网络操作完成时很容易发生程序无响应的情况，因此必须使用多线程实现网络操作，即使网络操作需要花费很长时间，也不会导致平板电脑端的程序发生用户操作无响应的情况。

首先想到的是使用 Java 中常用的 Thread 类或 Runnable 接口来实现，在 run 方法中运行一个循环，每次循环开始时进行网络操作，结束后调用 sleep 方法休眠 1 分钟，然后进入下一次循环。当用户切换到其他的标签页时，调用网络操作的线程的 interrupt 方法暂停线程，当用户切换回显示实验室图片的标签页时，再次调用网络操作的线程的 start 方法，恢复运行。在实际测试过程中，这种方法并没有达到预期的效果，在用户切换到其他标签页后确实不会再进行网络操作，但是当用户重新切换到显示实验室图片的标签页后，平板电脑端就会不停地向服务器请求图片，具体原因没有找到。

Thread 类是 Java 中本来就有的类，因为 Android 操作系统的虚拟机同标准的 Java 虚拟机有一些不同，可能实现上有一些区别，因此还需找到适合 Android 操作系统的解决方案。Android 操作系统的软件开发包中提供了一个 Handler 类，可以用于实现多线程。在用于显

示实验室图片的 `Fragment` 类中添加一个 `Handler` 类成员变量 `handler`，因为并没有在新的线程（`Thread`）类中新建这个 `handler` 而是在主线程中新建了这个 `handler`，`handler` 是属于主线程即 UI 线程的，因此 `handler` 可以对程序的界面元素进行操作，如使用新获得的图片替换界面上就的图片。`Handler` 类有一个 `postDelayed` 方法，接受参数是一个 `Runnable` 类型变量和时间，意义是在这么多时间后执行这个 `Runnable` 类变量中的 `run` 函数。现在就十分简单了，实现一个继承 `Runnable` 接口的类，在它的 `run` 方法中首先进行网络操作，向服务器请求网络摄像机所上传的实验室图片，将其保存在平板电脑外部存储根目录的 `temp` 文件夹中，文件名为 `lab.jpg`，然后使用刚刚获取的图片替换界面上原来的图片，最后调用 `handler` 的 `postDelayed` 方法，参数依次是这个 `Runnable` 类的变量本身和代表 1 分钟的时间数字，这样在第一次执行该 `Runnable` 变量的 `run` 方法之后就会每过一分钟再次执行。在 `Fragment` 的 `onResume` 方法（新建该 `Fragment` 和恢复运行时都要调用）中调用 `handler` 的 `post` 方法，传入参数是刚才新建的 `Runnable` 类变量，这样在 `Fragment` 新建时就会进行一次网络操作，之后每过一分钟执行一次。在 `Fragment` 的 `onPause` 方法中调用 `handler` 的 `removeCallbacks` 方法，传入参数仍然是之前新建的 `Runnable` 类变量，该方法会清空 `handler` 里所有等待运行的 `Runnable` 变量，于是就不会进行网络操作，直到用户切换回显示实验室图片的标签页后调用 `onResume` 方法时才继续运行，达到预期功能要求。

4.10 重复人员管理

在测试的过程中发现了一些开始设计时没有考虑到的问题，比如如果有同一个人训练了两个 `id`，而在人员列表中只有一个 `id`（因为每个人无法对应两个 `id`）那么在人脸识别的时候选取概率最大的 `id` 的时候就有可能出现错误，比如概率最大的那个 `id` 刚好是不在人员列表中的 `id`，就会发生错误，本来是识别正确但是程序判断出现识别错误。这种情况是可能发生的，比如一个人先被加入了人员名单，之后被移除了，后来再次被加入人员名单，如果第二次加入人员名单时使用的 `id` 跟第一次不同，就会出现这样的情况。这是因为人员名单只是存储在服务器上，并没有存储在人脸识别服务提供者 `face.com` 网站那里，`face.com` 所提供的 API 中并没有包含删除某个具体的人的接口，因此此人第一次训练的数据一直都在 `face.com` 的数据库中。目前的解决办法主要有两个，一是尽量避免同一个人多次加入时使用不同的 `id`，二是如果发生了这种情况，只好清空 `face.com` 中所有已经训练的数据，然后对所有人进行重新训练。

4.11 Android 开发中的一些问题

4.11.1 SDK 版本

众所周知，版本分裂问题是 Android 的一大问题，因为 Android 的开放性，世界上有无数各式各样的设备运行着 Android 操作系统，谷歌公司每半年发布一次 Android 的重大版本更新，但是各个厂商和公司却无法在第一时间为如此多的设备做好新版本的适配，这就导致了设备现在依然运行着出厂时的系统，因此世界上的 Android 设备运行着各种版本的 Android 操作系统，这对开发者来说是一个十分令人头疼的问题：使用新版本的 SDK 程序会更加漂亮，有了新的 API 开发起来更容易，并且能实现一些以前无法实现的功能，但是使用了新版本的 SDK 和新的 API 就会使程序无法在低版本的 Android 设备上运行，而世界上绝大部分 Android 设备并没有运行着最新版本的 Android 操作系统。

本实验所使用的平板电脑所运行的 Android 版本为 4.0.3，是目前较新的 Android 版本，又因为使用新版的 SDK 所提供的一些新的类和 API 可以实现更加美观和方便的 Android 程

序，本次实验中使用了 API level 15（对应于 Android 4.0 版本的 SDK）的 SDK 进行平板电脑端程序的开发。

4.11.2 Activity

在 Android 开发中 Activity 就是一个程序的灵魂，Activity 是程序的界面，是用户与程序交互的途径，在开始学习 Android 应用开发时首先就应该了解 Activity 的生命周期，如图 4-1 所示。

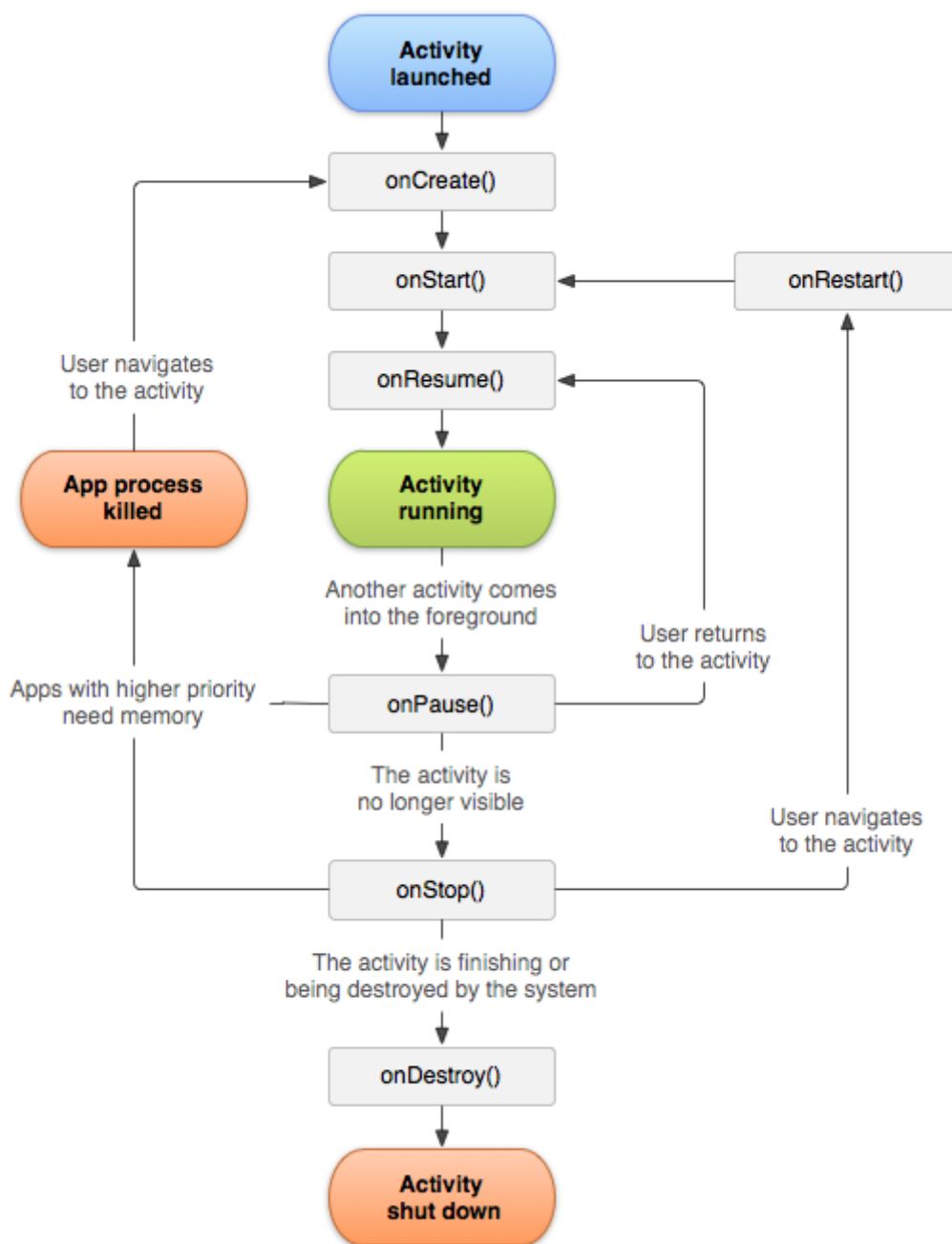


图 4-1 Activity 的生命周期^[1]

在一个应用被用户启动后，会有一个主 Activity 启动。Activity 启动时首先会调用 onCreate 方法，一般需要在 Activity 首次创建和运行时进行的操作都应该写在 onCreate 方法中，如读取配置，初始化界面等。接下来会调用 onStart 方法，如果 Activity 在前台显示时有其他 Activity 占据了屏幕，该 Activity 就会在后台运行，当该 Activity 再次回到前台显示

时也会调用 `onStart` 方法，因此需要每次显示 Activity 时显示给用户不同的界面或经常刷新界面显示的操作应该写在 `onStart` 方法中。之后会调用 `onResume` 方法，`onResume` 也是在 Activity 从后台回到前台显示后会调用的方法，一些 Activity 处于后台需要暂停而处于前台显示状态需要进行的操作应当写在 `onResume` 方法中。之后 Activity 就处于了正常运行状态，直到有别的 Activity 处于了屏幕前台显示，该 Activity 在即将处于后台运行时调用 `onPause` 方法，该方法同 `onResume` 方法对应，`onPause` 方法暂停一些操作，而 `onResume` 则应该恢复这些操作。当 Activity 不可见（处于后台）之后，会调用 `onStop` 方法，此时主线程（UI 线程）的操作都被迫停止，如果使用了多线程那么其他线程的操作仍会继续进行，一些需要在程序停止时保存数据等的操作应当写在 `onStop` 方法中。当 Activity 再次回到前台显示时，会调用 `onRestart` 方法，之后会调用 `onStart` 方法、`onResume` 方法然后 Activity 就再次进入正常运行状态，等待用户交互等操作，因此一些恢复数据的操作应该写在 `onRestart` 方法中。在 Activity 处于后台调用了 `onStop` 方法之后，如果因为其他程序需要内存而运行内存不足，该 Activity 有可能被迫结束自己释放内存，那么当用户回到该 Activity 的时候就会从 Activity 开始时运行，即从调用 `onCreate` 方法开始。当 Activity 要退出时会调用 `onDestroy` 方法，因此清理临时数据等操作应该写在 `onDestroy` 方法中。

4.11.3 ViewPager、Fragment、ActionBar 标签

ActionBar 是从 Android 3.0 版本（API level 11）开始引入的一个新的类，用于在应用的 Activity 顶部的应用名称旁添加一些新的界面控件，如标签页、按钮等。目前 Android 版本的发展预示着今后的 Android 设备都不再需要实体的物理按键，因此导航等操作都需要应用自己为用户提供解决方案，这时 ActionBar 就十分有用，可以使用 ActionBar 的标签对多个 Fragment（可以看作简化版的 Activity）进行切换的操作，可以添加像之前版本的菜单键按下后弹出的菜单的操作。而 ViewPager 是一个正在开发中的类，以后版本中的 API 可能会改变，因此目前使用 ViewPager 需要添加一个 Android 支持包，该包中包含许多正在开发中的新的类和 API。ViewPager 可以方便地实现类似于图片浏览器的滑动效果，如果配合 Fragment 进行开发，则可以实现向图片浏览器一样的滑动效果，只不过被滑动的对象是不同的 Fragment，而 Fragment 同 Activity 类似，因此用户就可以方便地通过左右滑动来切换要显示的界面。再同 ActionBar 配合，在标题栏显示一个标签栏，标签项就是各个 Fragment 的名字，用户可以通过点击标签或左右滑动来切换 Fragment。每个 Fragment 还可以添加一些菜单项，显示在 ActionBar 上，便于用户同应用进行交互操作。

4.11.4 Handler

当一个应用程序启动时，Android 会首先启动一个主线程（UI 线程），主线程管理界面中的 UI 控件，进行事件分发，比如用户点击了一个按钮，主线程会将事件分发到按钮上，调用按钮的相关回调函数对本次点击事件进行响应。但是如果在主线程中进行一个比较耗时的操作，如使用网络传输很多数据，那么主线程会因为等待操作的完成而失去响应，界面出现假死现象。因此我们需要将这些耗时的操作都放在一个子线程中，很多情况下这些操作结束后程序都需要根据这些操作得到的数据来更新应用的界面控件，但是界面控件只有主线程才能操作，子线程无法操作。这时就需要 Handler 类来帮助我们，Handler 运行在主线程中，因此可以对界面控件进行操作，但是 Handler 可以同子线程通信，接受子线程所传递的数据。子线程可以用 `sendMessage` 方法向 Handler 传递带有数据的 Message 对象，也可以直接调用 Handler 的 `post` 方法执行一个动作，甚至是调用 Handler 的 `postDelayed` 方法让主线程在指定的时间间隔后执行一个操作。使用了 Handler 可以缩短程序的响应时间，避免网络状况不好时导致程序失去响应，提升用户体验。

4.12 签到和离开的判断以及后续操作问题

服务器端对于签到和离开的判断及后续操作：

当有人签到时，需要向数据库中添加或修改数据，这时就需要判断本次动作是签到动作还是离开动作。当服务器收到平板电脑端传来的签到数据时，对于本次签到的人员进行遍历，在数据库中查找是否存在当天该人员的签到记录，如果不存在则说明这是该人员当日的第一次签到操作，如果数据库中已经存在当天该人员的签到操作，则说明这是该人员的离开操作。对于当日第一次签到的人员，将在数据库中添加一条当日该人员的签到记录，对于离开操作的人员，数据库中当日该人员的签到记录的“离开时间”一栏的数据将被修改为本次平板电脑端所传输至服务器的时间。

平板电脑端对于签到和离开的判断及后续操作：

在用户使用平板电脑完成拍照并成功被识别出后，将会向服务器发送签到的数据，服务器会按照上面的方法进行判断用户是签到操作还是离开操作，对数据库做出相应的修改后，服务器会向平板电脑返回两个列表：本次操作中签到的人员列表和本次操作中离开的人员列表。服务器端的具体做法是使用 `PrintWriter` 对 `socket` 使用 `println` 方法写入两行数据，第一行数据是本次操作中签到的人员 `id` 列表，相邻的人员的 `id` 间使用空格隔开，第二行数据是本次操作中离开的人员 `id` 列表，相邻的人员的 `id` 之间使用空格隔开。

平板电脑端使用了一个哈希集（`HashSet` 类）变量 `present` 来保存当前在实验室中的所有人员列表，初始化时 `present` 是空的。当用户完成一次签到后，平板电脑端接收到服务器返回的签到的人员列表和离开的人员列表后，先把签到的人员列表中的所有 `id` 加入 `present` 中，然后将离开的人员列表中的所有 `id` 从 `present` 中移除，这样 `present` 就仍然保存着目前处在实验室中没有离开的人员列表。

在一次签到操作的最后，平板电脑端的程序会发送一条微博，内容大致为某某某刚刚来到实验室，某某某刚刚离开实验室，目前在实验室中的人有：某某某 某某。同时这条微博还会附带一张本次签到操作时用户所拍摄的图片，并且带有位置信息。

如果本次操作后所有人都离开了实验室，即程序发现 `present` 的长度等于 0 了，那么所发送的微博结尾就不再提示目前实验室中有哪些人，而是提示实验室中已经没有人，然后平板电脑会向服务器发送一个关灯的请求，防止有人离开实验室后忘记关闭实验室的灯。

4.13 平板电脑与服务器通信时签到时间的传递

服务器向签到数据库添加记录时，需要加入的数据有：签到日期、签到人员 `id`、签到时间和离开时间。因为签到时间和离开时间不能同时添加，因此每次平板电脑端向服务器发送一个签到操作的数据时，至少需要包含签到日期、签到人员 `id` 和一个签到或离开的时间。签到人员 `id` 很容易处理，只需一个字符串。但签到日期和签到时间如果按照数据库中的格式传递的话就会比较麻烦，例如本次签到操作发生在 2012 年 5 月 14 日 13 点 11 分 20 秒，那么平板电脑端需要向服务器发送签到日期“2012-05-14”和签到时间“13: 11: 20”两个字符串，总长度达到了 18 个字节，尽管这样在服务器端处理时比较方便，直接将这两个字符串连同签到人员的 `id` 加入数据库即可，但是使用网络传递了过多的数据，在网络状况不好的情况下会带来性能的降低。

实验中使用的方法是调用系统方法 `System.currentTimeMillis()` 获得当前时间（当前时间距离 1970 年 1 月 1 日的毫秒数），这个方法返回一个长整型的数据，占用 8 个字节长度，而在这 8 个字节的数据当中既包含了签到日期的信息，又包含了签到时间的信息，甚至可以精确到毫秒级别，这样一来原来需要传递 18 个字节的数据，现在只需传递 8 个字节的数据，

并且将精度从秒提高到了毫秒,虽然后者意义不大,但是多次签到后累计节省的网络流量还是十分可观的。

一次完整的签到过程中平板电脑和服务器的通信过程如下:

用户使用平板电脑拍摄照片,并成功被识别出后,平板电脑建立一个同服务器之前的 socket 连接,向服务器写入一行数据,格式为“check [time] [ids]”:

```
writer.println("check " + System.currentTimeMillis() + " " + uids);
```

其中 `System.currentTimeMillis()` 是当前的时间, `uids` 是一个字符串,里面保存了本次签到的所有人员的 id,每个 id 之前使用空格隔开,例如一个可能的数据是“check 1336973888231 zs ls”,表示在 1336973888231 这个时间, zs 和 ls 使用平板电脑进行了签到的操作。

服务器接收到这些数据后,首先使用接收到的 1336973888231 这个长整型数字得到签到日期和时间:

```
String cdate = new java.sql.Date(militime).toString();
```

```
String time = String.format("%tR", militime);
```

其中 `militime` 表示接收到的 1336973888231 这个长整型数字,第一句代码用于获得签到日期,使用这个时间构造一个 `java.sql.Date` 类,然后调用它的 `toString` 方法,返回值就是“2012-05-14”这样格式的一个字符串。第二句代码用于获得签到时间,它将 `militime` 格式化为“13: 11: 20”这样形式的字符串。有了签到日期和签到时间,再对 `uids` 中的每一个人员 id 进行遍历,查看数据库中是否存在 `uids` 其中每个人员当天的签到记录,如果不存在就新增一条记录,签到日期是刚才得到的 `cdate`,签到时间是刚才得到的 `time`,同时将这个人员 id 加入一个表示本次签到人员的列表 `inlist` 中,如果存在就将该记录的离开时间修改为刚才得到的 `time`,并将这个人员 id 加入一个表示本次离开人员的列表 `outlist` 中。签到人员列表 `inlist` 和离开人员列表 `outlist` 分别表示本次操作中签到的人员和离开的人员,类型都是字符串,列表内每个人员 id 之间都使用空格隔开。在服务器端完成对数据库的操作之后,会将 `inlist` 和 `outlist` 返回给平板电脑,然后平板电脑根据服务器所返回的数据进行发送微博和关灯等的提醒操作。

4.14 本章小结

本章对应用的构建过程中遇到的一些重点、难点和问题进行了详细的描述和讨论,并对以及解决的问题详细解释了解决方法,对于无法实现或来不及实现的方面提供了解决思路,可供参考。首先,最重要的功能就是人脸识别的实现,由于实验的重点在于构建普适应用而非人脸识别的研究,因此没有足够的精力自己完成人脸识别算法的实现,需要寻找其他的开源项目或免费的人脸识别服务,在尝试了一些开源项目后, `face.com` 所提供的免费人脸识别服务最终脱颖而出,不仅识别正确率高,而且由于使用了云计算的技术,庞大的训练集对于服务器集群来说不在话下,响应时间很短。其次就是 `Android` 开发中的一些问题,要学习 `Android` 平台开发首先需要学习的就是 `Activity` 类,它是一个 `Android` 应用的门面,是多数时候程序与用户进行交互操作的唯一途径,在了解了 `Activity` 的生命周期后,才能更好的开发 `Android` 应用,将正确的代码放入正确的方法中。然后就是 `Android` 应用的界面问题,为了更加直观地显示以及能够另用户更加方便地操作,在实验中使用了 `ViewPager` 加 `Fragment` 加 `ActionBar` 这些较新版本的 `Android` 软件开发包所提供的类和 `API` 实现了一个能够让用户在屏幕上左右滑动在不同的界面直接切换的界面。除此之外的另一个重要功能就是发送新浪微博的实现,通过阅读新浪微博的官方文档以及新网微博所提供的 `Java` 开发包源代码,顺利实现出了发送一条带有位置信息的微博的功能,同时这条微博还能够附带一张图片。本章还讨论了一些其他的问题,比如为了安全性的问题,使用了一个管理员密码,在用户进行重

要操作时要求用户输入该管理员密码，并且在服务器和平板电脑的通信过程中，管理员密码并不是以明文进行传输，增加了安全性；比如中文乱码的问题，通过在读写文件、读写 socket 连接的输入输出流时都指定使用 UTF-8 编码，有效解决了中文字符乱码的问题；还有拍摄的图片过大，直接显示在应用中会发生内存溢出的问题，通过对图片进行压缩，不仅避免出现了内存溢出的问题，还在人脸识别过程中缩短了响应时间，提升了整个应用的性能。

第五章 结论

本次实验借助了传统计算机、网络摄像机、新型的平板电脑等设备，构建出了一个普适服务的应用场景——智能签到和远程监控系统。该系统可以让用户通过拍摄一张自己的照片来进行签到，无需输入很多信息，大大简化了现在签到系统的使用复杂度；该系统还包含了一个远程监控功能，通过 Android 端的应用，用户可以查看到实验室内网络摄像机所拍摄到的图片，除此之外用户还能够通过 Android 设备对实验室内的灯进行开关控制；该系统还能够对产生的事件作出智能的响应，当用户成功签到时，系统会发送一条微博提醒该用户签到，并且可以显示出当前实验室内的人员列表；当用户所拍摄的照片无法被正确识别时，会提示用户是否要进行训练，以不断完善训练集，提高以后的识别概率；当用户签到时连续出现错误时，系统会怀疑有陌生人在使用该系统，并将此人所拍摄的图片上传至新浪微博，提醒管理员和其他人员；当实验室内所有人员均已离开实验室后，系统可以自动关闭实验室内的灯，避免浪费。

通过这一系统，可以看到普适计算是一个涉及范围很广的课题，包括分布式计算、移动计算、人机交互、人工智能、嵌入式系统、感知网络以及信息融合等多方面的技术的融合。通过这一系统，我们也可以看到普适计算未来的一些前景和发展方向，那就是计算机会嵌入到环境或日常工具中去，人们能够更加自然地和计算机交互，而计算设备则会变得越来越智能，它们可以感知周围环境的变化，根据环境的变化作出自动的基于用户需要或设定的行为。比如手机能够在用户开会时感知用户正在开会，于是自动将情景模式改为静音模式，并且能够自动给来电人员回复“主人正在开会”之类的信息；又比如当用户进入一家商店，通过商店门口的摄像头计算机可以识别出该用户的一些历史购物信息，猜测这位用户可能感兴趣的商品，直接为其推荐。这意味着在普适计算的时代，用户不必为了使用计算机而去寻找一台计算机，无论走到哪里，无论什么时间，用户都可以从环境中获得所需要的服务和信息。

参考文献

- [1] Google Inc.: Activity | Android developers. [OL]. [2012-05-09].
<http://developer.android.com/reference/android/app/Activity.html>.
- [2] 蔡海滨. 普适计算中智能服务选择算法研究[D]. 东华大学 2008
- [3] 王苗苗. 面向普适计算的无线传感器网络中间件研究[D]. 中国科学技术大学 2008
- [4] 郭亚军. 普适计算安全的关键技术研究[D]. 华中科技大学 2006
- [5] 刘志强. 普适计算环境下基于场景的可配置服务研究[D]. 上海交通大学 2007
- [6] 蒲芳. 普适计算中位置感知服务的研究[D]. 东华大学 2008
- [7] 周文嘉,黄林鹏,陈俊清. 普适环境中的动态更新模型[J]. 微电子学与计算机. 2011(08)
- [8] 郭亚军,李蓉. 普适计算的访问控制研究[J]. 华中师范大学学报(自然科学版). 2006(04)
- [9] 李芬. 下一代计算模式:普适计算[J]. 硅谷. 2011(15)
- [10] 曾宪权,裴洪文. 普适计算技术研究综述[J]. 计算机时代. 2007(02)

谢辞

思源湖畔，菁菁礼堂，这是我对交大的第一印象，而转眼间四年已经过去。这四年，是我人生中最重要的一年，我有幸接触到了这么多不仅能够向我传授知识，还能指导我的人生道路的良师。在进入大学前我只是个什么都不懂的孩子，而如今我已经规划好了未来的人生之路，因此我首先要向尊敬的母校和老师们表达深深的谢意！

这篇论文中所进行的研究和开发工作都是在黄林鹏老师的指导下完成的，从一年前的暑假实习开始，黄老师就一直不辞辛劳地同我一起讨论和研究，在毕业设计的开始阶段，对我提出了很多的建议和意见，我才得以顺利地完成本次毕业设计。除此之外，黄老师还经常经常传授一些经验，对我今后的科研和工作道路有很大的帮助。在此，请允许我向尊敬的黄林鹏老师表示真挚的谢意！

在实验室的这段时间里，实验室的各位师兄和师姐也经常帮助和指导我，对于实验室中的设备都耐心地向我讲解，对于研究中的问题也是指导我找出解决方案，还将自己的经验传授给我，对我今后的研究生生活有很大帮助，在此我要对各位师兄师姐说一声谢谢。

我还要感谢一下我的父母，没有我的父母就没有今天的我，感谢你们对我 21 年的养育之恩，在我做错事时及时纠正我，让我能够顺利走上研究生之路，感谢你们对我的教育和包容。

最后我还要感谢一下我的朋友们和同学们，感谢你们在我开心时陪我一起开心，在我郁闷时帮我开导。这里特别感谢一下远在新加坡的陈同学，感谢你在我大学的最后一个学期为我带来的快乐和动力！

BUILDING OF PERVASIVE SERVICES AND APPLICATIONS

Pervasive computing (also known as ubiquitous computing) is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. More formally, ubiquitous computing is defined as "machines that fit the human environment instead of forcing humans to enter theirs." Pervasive computing is a concept which includes a lot of research fields, such as distributed computing, mobile computing, human-machine interaction, artificial intelligence, embedded systems, cognitive network and information integration. In the world of pervasive computing, all the computers are embedded into environment and everyday objects, people are going to interacting with computers more naturally, even without knowing where the computers are. Machines are becoming more and more intelligent, they can perceive the changes of the environment and can react based on user's needs or execute some predefined actions according to the changes of the environment. For example, the cellphone will "know" that the user is at a meeting when the user is at a meeting, and the cellphone will automatically change itself to silent mode, maybe even reply to others who call the user with a message saying "I am at a meeting now, will call back soon!"

In this work, a pervasive application scenario is built to get a better understanding of pervasive computing and look into the future of pervasive computing and its applications. In this work, a traditional PC, an Android tablet, a WLAN camera and an internet relay is used to build a pervasive application scenario with the help of the popular Android operating system. The application is called "Check-in helper", it has three major functions. The first function is the check-in/check-out function. Using this application on the Android tablet, one can check-in to the lab by taking a picture of him through the camera of the Android tablet. The second function is to view the previous check-in data of the lab. The last function is to monitor the lab and control the lights in the lab remotely. The following paragraphs will describe the architecture of the whole system and the details of the three major functions.

In this system, there are 4 main parts, the Android tablet, the personal computer, the WLAN camera and the internet relay with some light bulbs. The user executes all the actions on the Android tablet, and the tablet will send the operation codes to the server (the personal computer), the server will communicate with the camera or the internet relay if necessary. The data of users' check-in records is stored in the MySQL database in the server, when someone checked in, the tablet will send the information of this check-in record to the server, and the server will modify the data of the database to complete the check-in action. A FTP server needs to be established on the server, and the WLAN camera is configured to send the captured picture of the lab to the FTP server every minute (this is a built-in function of the camera, and the smallest interval is one minute). When the user wants to see the picture of the lab, the tablet will ask for the picture and the server will send the latest picture that the camera uploaded to the tablet. When the user executes the "turning on the lights" operation and the "turning off the lights" operation, the tablet will send the

codes to the server accordingly, and the server will send proper control codes to the internet relay to turn on the lights or turn off the lights.

First function, check-in/check-out function, is the core function of the whole system. In the center of the check-in view, there is a picture which is the last picture taken by the users, it shows the last person that checks in to the lab or checks out of the lab. At the top of the view, there are four buttons, check-in/out, train, add and remove. The check-in/out button is always showed and other three buttons is hidden, and they will show when the user click menu button or the overflow menu button beside the check-in/out button. When the user wants to check in to the lab, he should click the check-in button at the top, and the application will invoke the default camera app of the Android system to take a picture. After taking a picture of the user's face, the user should click the OK button and the tablet will process the human faces recognition. In this work, the human faces recognition is realized using the free face recognition API provided by face.com. In the process of face recognition, the tablet will upload the picture to the server of face.com and face.com will processed with the picture and return the result data to the tablet, if the confidence of the face in the picture being someone is high, for example, higher than 80, than we will assume this face is belong to this person, and the tablet will show some welcome information and send the data of this check-in record, such as data, name, time to the server. If the user cannot be recognized, the tablet will show a pop up window, the user can choose to use the picture just taken to train or cancel to check-in again.

If the user is newly added or cannot be recognized by the system, we need to train this person with his pictures. The user should click the train button on the top of the view, and the application will invoke the default camera app of the system to take a picture. Then the user should take a picture of him; notice that a good picture with clear face image and great light condition will be very helpful. After taking and confirming a picture, the tablet will show the user a pop up dialog window, in which the user should fill his id and the administrator password. Because train is a important operation, improper operation will do harm to the face database and have a bad impact on the accuracy of face recognition, therefor an administrator password is necessary, the user could only do these vital operations under the monitoring of the administrator. When the user entered his id and the administrator password and pressed the OK button, the tablet will send this picture and the user's id to the server of face.com to train.

Users can also modify the member list of the lab by operating on the Android tablet, and obviously this operation should also be done under supervision of the administrator. If the user want to add a member, he should click the add button on the top of the view on the tablet, and tablet will show up a pop up dialog window, in which the user should enter the new member's id, new member's name and the administrator password. After entering the information and pressed the OK button, the tablet will send the information of the new member to the server. When the server receives the information of the new member, the server will see if there is already a user with this new member's id, if not, server will add this new member to member list, which is a file stored on the server. If the user want to remove someone from the member list of the lab, he should click the remove button at the top of the view on the tablet, and the tablet will show up a pop up dialog window, in which the user should enter the member id of the member whom he want to remove and the administrator password. After the user complete entering the information and press the OK button, the tablet will send the information to the server. When the server receives this information, it will remove this member from the member list.

The second function is to view the previous check-in data of the lab. When the user switches to the statistics view, the tablet will ask for the check-in data from the server. When the server receives the request from the tablet, it will query data from database and reorganize the data to String format and send the data to the tablet. After the tablet receives the data, it will show all the data on the screen, grouped by the date. It is just a simple way to show the data, and in the future the way users viewing the data should be changed to improve the user experience.

The third function is to monitor the lab and control the lights in the lab remotely. The WLAN camera used in this work has a built-in function, continuously sending captured pictures to a specific FTP server after a specific time. Therefore we set up a FTP server on the server, and configure the camera properly to send a picture to the server every one minute (the smallest time interval is one minute). While on the other hand, the tablet request the picture of the lab from the server every one minute, and show the picture on the screen. At the top of the view, there are two buttons, "on" and "off", the user could use these two buttons to turn on or turn off the lights in the lab. If the user clicks one of the two buttons, the tablet will send the operation code to the server, when the server receives the operation code, it will send the binary code to the internet relay according to the operation code the tablet sent, and the internet relay will control the lights.

The application will also make some intelligent responses to different incidents. If someone successfully checked in or checked out, the application will automatically post a sina weibo to a specific account, with the picture uploaded, showing that this person has checked in or checked out. If all the people had left the lab, the application will find that there is nobody in the lab, it will automatically turn off the lights of the lab.

From this application we could see the future of the pervasive computing, the machines are becoming more and more intelligent, people will no longer need a traditional PC to get what they want, and they could use their phones or even just make some gestures or just say something.