

**智能药箱**

**-----私人的用药管理助手**

**技术分析报告**

**团队名称： 随缘队**

**项目名称： 3D的你**

**队 长： 梁惠欣**

**团队成员： 梁惠欣，李永琪，叶蕴盈，董亭，邵明山**

**指导老师： 戴鸿君**

**学 校： 山东大学**

**撰 写 人： 梁惠欣**

**时 间： 2015年9月17日**

**目录**

**一、设备端关键技术实现**···········································1

1、rfid··························································1

2、pyqt5·························································1

3、扫码模块······················································1

**二、Server关键技术实现**·········································1

1、分词··························································2

2、匹配··························································2

1. 情感分析·····················································2

4、学习··························································2

5、专业模块·····················································2

**三、Android关键技术实现**········································3

**一、设备端关键技术实现**

1. **rfid模块**

射频标签是产品电子代码（EPC）的物理载体，附着于可跟踪的物品上，可全球流通 并对其进行识别和读写。RFID（Radio Frequency Identification）技术作为构建“物联网” 的关键技术近年来受到人们的关注。RFID 技术 是一种自动识别技术，应用分布在身份证件和门禁控制、供应 链和库存跟踪、汽车收费、防盗、生产控制、资产管理。

本项目的rfid模块采用了高频循环探测模块。

该模块通过ttl串口转usb插口与树莓派相连，并通过其传递数据。根据协议发送命令后，该模块会尝试一次读取范围内所有标签。本项目通过对标签与药品进行绑定，从而令rfid的循环读取能够不断检测药箱中所有药品的现状。

1. **pyqt5**

Python自带的gui不能满足于我们项目的需要，在本次编写中我们采用了pyqt5，众所周知qt是这个星球最强大的gui库之一，pyqt是qt的python版本。通过pyqt编写的gui界面，远比默认界面要美观方便。

1. **扫码模块**

扫码添加药品在设备端通过扫码模块来实现，通过修改协议设置，我们将其设置为虚拟键盘输入，以此降低UTAR通信的不稳定性。同时通过pyqt的事件槽方法，将输入事件作为信号，使得输入条形码极为简单。

**二、Server关键技术实现**

1、 分词

使用lucene搜索引擎的IKAnalyzer2.0.2分词库来进行中文分词，将难以处理的适应症信息转换为比较突出的病症标签，并对无意义词、近义词进行整理，存入用户对应的数据库中。

1. 数据挖掘

用户在使用过程中会积累下大量的标签，通过获取每个标签频率和时间，使用推荐算法，就可以整理出用户最关心或是当前最急需的病症的信息，将对应的文章推送给用户。

3、 用户数据库

通过独特的数据结构整理用户的各种信息，来完善整个软件的用户对应机制，实现药箱与用户一对一，甚至多对一的良好体验。

**三、Android端关键技术实现**

1、文章列表的滑動頁面----------------官方PagerAdapter框架運用

2、可滑動按鈕--------------------------------結合手勢監聽的實現

3、掃碼頁面的實現---------------調用攝像頭，結合網絡jar包的運用

（1）攝像頭：

private void initCamera(SurfaceHolder surfaceHolder) {

try {

CameraManager.get().openDriver(surfaceHolder);

} catch (IOException ioe) {

return;

} catch (RuntimeException e) {

return;

}

if (handler == null) {

handler = new CaptureActivityHandler(QRScanActivity.this, decodeFormats, characterSet);

}

}

（2）掃碼分析

：

import com.google.zxing.BarcodeFormat;

private static final Pattern COMMA\_PATTERN = Pattern.compile(",");

static final Vector<BarcodeFormat> PRODUCT\_FORMATS;

static final Vector<BarcodeFormat> ONE\_D\_FORMATS;

static final Vector<BarcodeFormat> QR\_CODE\_FORMATS;

static final Vector<BarcodeFormat> DATA\_MATRIX\_FORMATS;

static {

PRODUCT\_FORMATS = new Vector<BarcodeFormat>(5);

PRODUCT\_FORMATS.add(BarcodeFormat.UPC\_A);

PRODUCT\_FORMATS.add(BarcodeFormat.UPC\_E);

PRODUCT\_FORMATS.add(BarcodeFormat.EAN\_13);

PRODUCT\_FORMATS.add(BarcodeFormat.EAN\_8);

PRODUCT\_FORMATS.add(BarcodeFormat.RSS\_14);

ONE\_D\_FORMATS = new Vector<BarcodeFormat>(PRODUCT\_FORMATS.size() + 4);

ONE\_D\_FORMATS.addAll(PRODUCT\_FORMATS);

ONE\_D\_FORMATS.add(BarcodeFormat.CODE\_39);

ONE\_D\_FORMATS.add(BarcodeFormat.CODE\_93);

ONE\_D\_FORMATS.add(BarcodeFormat.CODE\_128);

ONE\_D\_FORMATS.add(BarcodeFormat.ITF);

QR\_CODE\_FORMATS = new Vector<BarcodeFormat>(1);

QR\_CODE\_FORMATS.add(BarcodeFormat.QR\_CODE);

DATA\_MATRIX\_FORMATS = new Vector<BarcodeFormat>(1);

DATA\_MATRIX\_FORMATS.add(BarcodeFormat.DATA\_MATRIX);

}

private DecodeFormatManager() {}

static Vector<BarcodeFormat> parseDecodeFormats(Intent intent) {

List<String> scanFormats = null;

String scanFormatsString = intent.getStringExtra(Intents.Scan.SCAN\_FORMATS);

if (scanFormatsString != null) {

scanFormats = Arrays.asList(COMMA\_PATTERN.split(scanFormatsString));

}

return parseDecodeFormats(scanFormats, intent.getStringExtra(Intents.Scan.MODE));

}

static Vector<BarcodeFormat> parseDecodeFormats(Uri inputUri) {

List<String> formats = inputUri.getQueryParameters(Intents.Scan.SCAN\_FORMATS);

if (formats != null && formats.size() == 1 && formats.get(0) != null){

formats = Arrays.asList(COMMA\_PATTERN.split(formats.get(0)));

}

return parseDecodeFormats(formats, inputUri.getQueryParameter(Intents.Scan.MODE));

}

private static Vector<BarcodeFormat> parseDecodeFormats(Iterable<String> scanFormats,

String decodeMode) {

if (scanFormats != null) {

Vector<BarcodeFormat> formats = new Vector<BarcodeFormat>();

try {

for (String format : scanFormats) {

formats.add(BarcodeFormat.valueOf(format));

}

return formats;

} catch (IllegalArgumentException iae) {

// ignore it then

}

}

if (decodeMode != null) {

if (Intents.Scan.PRODUCT\_MODE.equals(decodeMode)) {

return PRODUCT\_FORMATS;

}

if (Intents.Scan.QR\_CODE\_MODE.equals(decodeMode)) {

return QR\_CODE\_FORMATS;

}

if (Intents.Scan.DATA\_MATRIX\_MODE.equals(decodeMode)) {

return DATA\_MATRIX\_FORMATS;

}

if (Intents.Scan.ONE\_D\_MODE.equals(decodeMode)) {

return ONE\_D\_FORMATS;

}

}

return null;

}