中山大学数据科学与计算机学院

物联网技术实验报告

二维码解码模块的设计与实现

小组成员: (姓名 学号)

(姓名 学号)

(姓名 学号)

(姓名 学号)

## 指导教师： 胡建国

目录

[一、 设计概要 3](#_Toc498291121)

[二、 设计原理 3](#_Toc498291122)

[1. 定位功能 3](#_Toc498291123)

[2. 温度探测 3](#_Toc498291124)

[3. 身份识别 3](#_Toc498291125)

[三、 功能实现及效果展示 4](#_Toc498291126)

[1. 定位功能的实现 4](#_Toc498291127)

[1) 定位方式的选择 4](#_Toc498291128)

[2) 定位的具体实现 4](#_Toc498291129)

[3) 定位界面的效果 8](#_Toc498291130)

[2. 温度检测的实现 8](#_Toc498291131)

[1) 温度数据来源 8](#_Toc498291132)

[2) 温度的获取 8](#_Toc498291133)

[3) 温度的获取的效果 9](#_Toc498291134)

[3. 身份识别部分 10](#_Toc498291135)

[1) 身份识别的选择 10](#_Toc498291136)

[2) 二维码扫描的实现 10](#_Toc498291137)

[3) 二维码扫描的结果展示 12](#_Toc498291138)

[四、 项目总结与展望 13](#_Toc498291139)

[1. 总结 13](#_Toc498291140)

[2. 展望 13](#_Toc498291141)

## 设计概要

本项目意在开发一项手机应用端，集合位置定位，环境温度探测与二维码解码识别身份三大功能。

其中，手机位置定位功能能够识别当前位置并将其在地图上显示出来；环境温度探测功能能够测出手机当前所处环境中的温度，并在屏幕上显示；二维码解码功能能够在扫描二维码后，对其进行解码并读取其中信息。

## 设计原理

1. **定位功能**

当前主流的定位功能为GPS定位，因为目前市面上的手机基本都带有GPS的功能，且GPS精度高，所以选用GPS定位。

GPS（Global Positioning System）即全球定位系统，是由美国建立的一个卫星导航定位系统，利用该系统，用户可以在全球范围内实现全天候、连续、实时的三维导航定位和测速；另外，利用该系统，用户还能够进行高精度的时间传递和高精度的精密定位。

项目中使用百度地图SDK进行定位，百度地图的LBS（Location Based Service，基于位置的服务）的三大目标是：你在哪里；你和谁在一起；附近有什么资源，很符合我们项目中的要求。

百度智能定位服务支持GPS、WiFi、基站融合定位，其定位精度会更高。其优势在于：离线定位技术利用百度大数据分析挖掘能力，分析用户常驻点信息，在保证定位精准度的基础之上，提供离线定位能力，利用离线定位能力，不仅定位速度进一步提升，同时也完美解决无网络环境下的位置获取难题；室内高精度定位，大型商超内寻找店铺、停车场内反向寻车等业务的发展基础都是室内高精度定位。利用三角定位技术、增强WiFi指纹模型技术、地磁技术、蓝牙技术等， 提供精度1-3m的室内高精度定位服务。

百度地图Android定位SDK提供GPS、基站、WiFi、地磁、蓝牙、传感器等多种定位方式，适用于室内、室外多种定位场景，具有出色的定位性能：定位精度高、覆盖率广、网络定位请求流量小、定位速度快。

同时百度地图Android的优势还在于其拥有完善的开发文档，对于学习过Android开发的人很容易通过文档上手。

1. **温度探测**

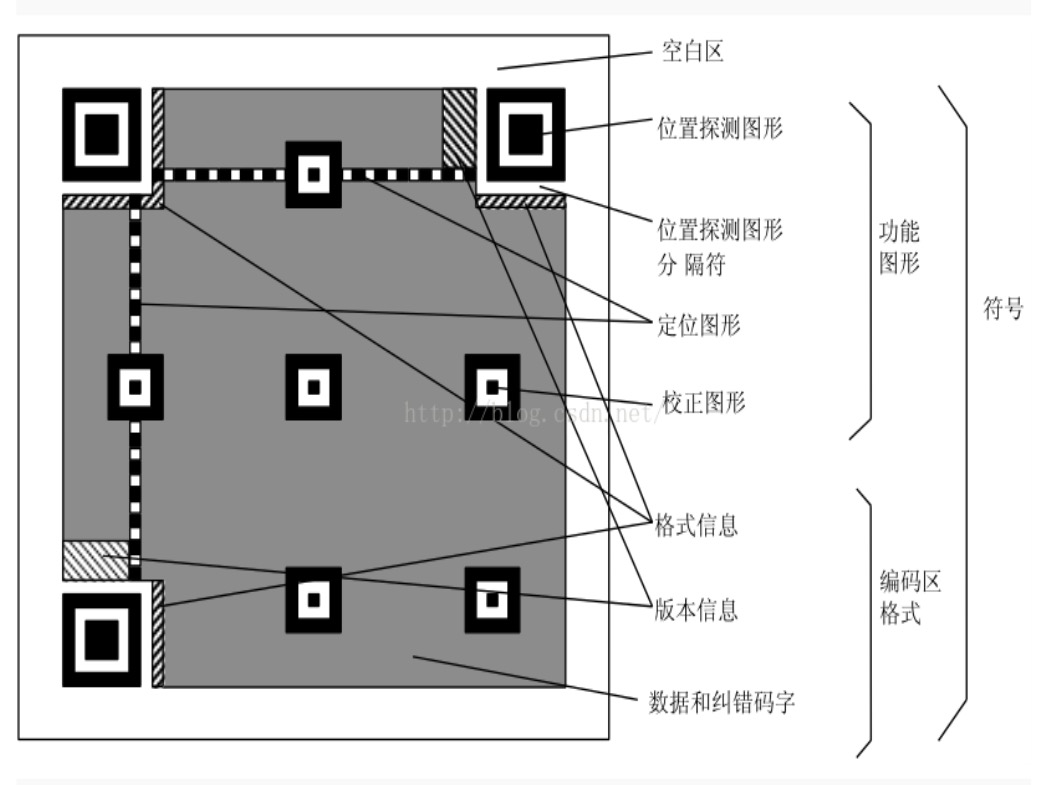
在温度探测功能的实现上，因为手机基本没有配置环境温度传感器，所以只能认为在非过度使用的情况下，手机电池温度近似于所处环境的温度，因此，利用手机中电池自带的温度传感器来实现该功能,但是其也有缺陷检测的只是手机周边的温度，净空中温度无法被检测到，例如手机放在热源旁只会显示热源的温度，其温度检测的范围导致温度很有局限性。不过碍于手机的局限性，只能使用电池温度。

1. **身份识别**

前手机中较为流行的身份识别技术是以指纹识别为主，最近也在兴起人脸识别和虹膜识别，但是这几种识别技术都需要有专门的硬件，同时还要涉及到专门的API以及系统权限问题。所以我们项目中选择二维码识别。

二维码是用某种特定的几何图案按照一定的规律在平面上分布的黑白相间的图形来记录数据符号信息的，其能够在横向和纵向两个方位同时表达信息，因此能在很小的面积内表达大量的信息，与条形码相比，其优势在于节省空间。在代码编制上，二维码巧妙的利用构成计算机内部逻辑基础的0/1比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图像输入设备或光电扫描设备自动识读以实现信息自动处理。

下图为一个二维码的基本结构。



位置探测图形、位置探测图形分隔符、定位图形：用于二维码的定位，对每个QR来说，位置都是固定存在的，只是大小规格有所差异；

校正图形：确定规格，校正图形的数量和位置也就确定了；格式信息：表示二维码的纠错级别，分为L、M、Q、H；

版本信息：即二维码的规格，QR码符号共有40种规格的矩阵；

数据和纠错码字：实际保存的二维码信息，和纠错码字（用于修正二维码损坏带来的错误）。

在本次实现二维码解码功能的过程中，主要利用了Zxing。Zxing是一个开源的，用于Java实现的多种格式的1D/2D条码图像处理库，它包含了联系到其他语言的接口，而且可以实现使用手机的内置摄像头完成条形码和二维码的扫描与解码。目前为止，其支持的格式有、代码128、QR码，满足了我们项目的需求。

## 功能实现及效果展示

1. **定位功能的实现**
   1. **定位方式的选择**
      * 通过手机内部的GPS进行定位：GPS定位的优势在于可以离线定位，其定位精度更高为（1-3M）。
      * 通过连接网络使用IP进行定位：使用上网是产生的ip地址进行定位，这种方式针对没有GPS模块的产品，其缺点是精度较差（350m）。

APP中我们选择前者进行定位，因为现在市面大部分手机都带有GPS模块，所以我们可以使用GPS获得精确的位置信息进行定位。

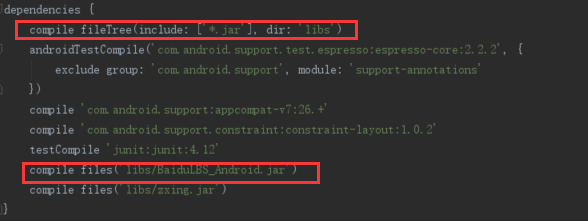
* 1. **定位的具体实现**

此处我们使用Android手机进行定位实现，因为Android开源所有我们有很多的开发文档可以使用，同时我们很容易使用已有的SDK进行定位。

我们此处选用百度开发的SDK进行定位，开发工具使用Google原生的Android studio。

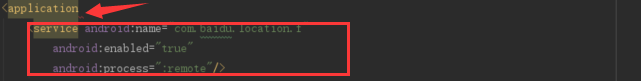
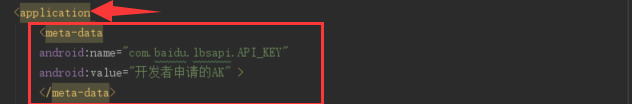
1. 首先我们从百度官方获取SDK包并挂载到我们的Android studio。

* 配置build.gradle文件使得我们项目中存在百度的SDK。



* 配置AndroidManifest.xml，AndroidManifest.xml是我们在安卓系统中获取权限的一个配置文件，同时我们也需要对我们挂载的百度API进行鉴权。此处配置权限我们才可以获得使用手机GPS模块的权利。

在Application标签中增加鉴权代码和声明service组件



然后我们申请权限，添加如下代码

|  |
| --- |
| <manifest xmlns:android=<http://schemas.android.com/apk/res/android>  package="cn.chonor.sen">  <!-- SDK2.1新增获取用户位置信息 -->  <uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS" />  <uses-permission android:name="com.android.launcher.permission.READ\_SETTINGS" />  <uses-permission android:name="android.permission.WAKE\_LOCK"/>  <!-- SDK1.5需要android.permission.GET\_TASKS权限判断本程序是否为当前运行的应用? -->  <uses-permission android:name="android.permission.GET\_TASKS" />  <uses-permission android:name="android.permission.WRITE\_SETTINGS" />  <!-- 这个权限用于进行网络定位-->  <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  <!-- 这个权限用于访问GPS定位-->  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  <!-- 用于访问wifi网络信息，wifi信息会用于进行网络定位-->  <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>  <!-- 获取运营商信息，用于支持提供运营商信息相关的接口-->  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>  <!-- 这个权限用于获取wifi的获取权限，wifi信息会用来进行网络定位-->  <uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"/>  <!-- 用于读取手机当前的状态-->  <uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>  <!-- 写入扩展存储，向扩展卡写入数据，用于写入离线定位数据-->  <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>  <!-- 访问网络，网络定位需要上网-->  <uses-permission android:name="android.permission.INTERNET" />  <!-- SD卡读取权限，用户写入离线定位数据-->  <uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"/>  <!--允许应用读取低级别的系统日志文件 -->  <uses-permission android:name="android.permission.READ\_LOGS"/>  <!-- 定位所需的权限 -->  <uses-permission android:name="android.permission.VIBRATE" /> |

1. 获取所在位置的经纬度，并显示在地图上。

通过参数配置，可选择定位模式、可设定返回经纬度坐标类型、可设定是单次定位还是连续定位。定位SDK所提供的定位模式包括三种：高精度、低功耗和仅用设备定位。开发者请根据自己的实际使用需求进行选择。定位SDK能够返回三种坐标类型的经纬度（国内），分别是gcj02（国测局坐标）、bd09（百度墨卡托坐标）和bd09ll（百度经纬度坐标）。此处因为我们需要在带百度地图上顺带把位置信息显示出来所以我们选择bd09ll（百度经纬度坐标）

|  |
| --- |
| private void initView() {  mMapView = (MapView) findViewById(R.id.*main\_bdmap*);  mBaiduMap = mMapView.getMap();  poisLL = (ListView) findViewById(R.id.*main\_pois*);  //定义地图状态  MapStatus mMapStatus = new MapStatus.Builder().zoom(18).build();  MapStatusUpdate mMapStatusUpdate = MapStatusUpdateFactory.*newMapStatus*(mMapStatus);  //改变地图状态  mBaiduMap.setMapStatus(mMapStatusUpdate);  //地图状态改变相关监听  mBaiduMap.setOnMapStatusChangeListener(this);  //开启定位图层  mBaiduMap.setMyLocationEnabled(true);  //定位图层显示方式  mCurrentMode = MyLocationConfiguration.LocationMode.*NORMAL*;  */\*\*  \* 设置定位图层配置信息，只有先允许定位图层后设置定位图层配置信息才会生效  \* customMarker用户自定义定位图标  \* enableDirection是否允许显示方向信息  \* locationMode定位图层显示方式  \*/* mBaiduMap.setMyLocationConfigeration(new MyLocationConfiguration(mCurrentMode, true, null));  //初始化定位  mLocClient = new LocationClient(this);  //注册定位监听  mLocClient.registerLocationListener(this);  //定位选项  LocationClientOption option = new LocationClientOption();  */\*\*  \* coorType - 取值有3个：  \* 返回国测局经纬度坐标系：gcj02  \* 返回百度墨卡托坐标系 ：bd09  \* 返回百度经纬度坐标系 ：bd09ll  \*/* option.setCoorType("bd09ll");  //设置是否需要地址信息，默认为无地址  option.setIsNeedAddress(true);  //设置是否需要返回位置语义化信息，可以在BDLocation.getLocationDescribe()中得到数据，ex:"在天安门附近"， 可以用作地址信息的补充  option.setIsNeedLocationDescribe(true);  //设置是否需要返回位置POI信息，可以在BDLocation.getPoiList()中得到数据  option.setIsNeedLocationPoiList(true);  */\*\*  \* 设置定位模式  \* Battery\_Saving  \* 低功耗模式  \* Device\_Sensors  \* 仅设备(Gps)模式  \* Hight\_Accuracy  \* 高精度模式  \*/* option.setLocationMode(LocationClientOption.LocationMode.*Hight\_Accuracy*);  //设置是否打开gps进行定位  option.setOpenGps(true);  //设置扫描间隔，单位是毫秒 当<1000(1s)时，定时定位无效  option.setScanSpan(1000);  //设置 LocationClientOption  mLocClient.setLocOption(option);  //开始定位  mLocClient.start(); }  public void onReceiveLocation(BDLocation bdLocation) {  //如果bdLocation为空或mapView销毁后不再处理新数据接收的位置  if (bdLocation == null || mBaiduMap == null) {  return;  }  //定位数据  MyLocationData data = new MyLocationData.Builder()  //定位精度bdLocation.getRadius()  .accuracy(bdLocation.getRadius())  //此处设置开发者获取到的方向信息，顺时针0-360  .direction(bdLocation.getDirection())  //经度  .latitude(bdLocation.getLatitude())  //纬度  .longitude(bdLocation.getLongitude())  //构建  .build();  //设置定位数据  mBaiduMap.setMyLocationData(data);  //是否是第一次定位  if (isFirstLoc) {  isFirstLoc = false;  LatLng ll = new LatLng(bdLocation.getLatitude(), bdLocation.getLongitude());  MapStatusUpdate msu = MapStatusUpdateFactory.*newLatLngZoom*(ll, 18);  mBaiduMap.animateMapStatus(msu);  }  //获取坐标，待会用于POI信息点与定位的距离  locationLatLng = new LatLng(bdLocation.getLatitude(), bdLocation.getLongitude());  //获取城市，待会用于POISearch  city = bdLocation.getCity();  //创建GeoCoder实例对象  geoCoder = GeoCoder.*newInstance*();  //发起反地理编码请求(经纬度->地址信息)  ReverseGeoCodeOption reverseGeoCodeOption = new ReverseGeoCodeOption();  //设置反地理编码位置坐标  reverseGeoCodeOption.location(new LatLng(bdLocation.getLatitude(), bdLocation.getLongitude()));  geoCoder.reverseGeoCode(reverseGeoCodeOption);  //设置查询结果监听者  geoCoder.setOnGetGeoCodeResultListener(this); } |

1. 同时我们考虑到定位可能不够准确，所以我们要能够提供一些周边位置以供选择，上部分代码中已经获取了反地理编码所以我们只需要利用这个反地理编码去请求周围位置，上部分代码中已经请求了所以我们只需要接收并填充即可。

|  |
| --- |
| //反地理编码查询结果回调函数  @Override  public void onGetReverseGeoCodeResult(ReverseGeoCodeResult reverseGeoCodeResult) {  List<PoiInfo> poiInfos = reverseGeoCodeResult.getPoiList();  PoiAdapter poiAdapter = new PoiAdapter(Positioning.this, poiInfos);  poisLL.setAdapter(poiAdapter); } |

1. 获取位置代码的其他部分，此部分主要是地图界面的构建不包括上述的定位以及位置选择，以及位置列表项的填充，此部分不在展示，见代码附件。
   1. **定位界面的效果**

|  |  |
| --- | --- |
|  |  |
| 定位模式 | 出现偏差可进行修改 |

1. **温度检测的实现**
   1. **温度数据来源**

首先因为大部分手机都没有配置额外的温度传感器，所以我们只能通过近似的方法来获得环境温度。现在手机电池中因为锂电池的控制需要一般都有电池温度传感器，我们可以使用电池温度作为一个近似的环境温度，但因为电池在冲放电的时候回发热所以这个温度是不准确的。在手机使用频率较低的情况下我们可以近似的认为电池温度就是周围的环境温度。

* 1. **温度的获取**
     + Android手机中电池的信息是以系统广播的方式进行向外的传递，所以我们可以直接接受这个系统广播来获取温度信息。
     + 首先注册这个广播接收者

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  //注册电池信息广播接收器  registerReceiver(mBatInfoReceiver, new IntentFilter(Intent.*ACTION\_BATTERY\_CHANGED*));  init();//界面按钮初始化  Check(); } |

* + - 重写接收函数

|  |
| --- |
| private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver() {  public void onReceive(Context context, Intent intent) {  String action = intent.getAction();  if (Intent.*ACTION\_BATTERY\_CHANGED*.equals(action)) {   BatteryT = intent.getIntExtra("temperature", 0); //电池温度 单位是0.1度  textView.setText("当前温度:" + String.*valueOf*(BatteryT/10);  }  } }; |

* + - 在关闭界面是销毁广播接收器

|  |
| --- |
| @Override  protected void onDestroy() {  super.onDestroy();  unregisterReceiver(mBatInfoReceiver); } |

* 1. **温度的获取的效果**

|  |  |
| --- | --- |
|  |  |
| 使用时温度 | 静止10分钟情况下温度 |

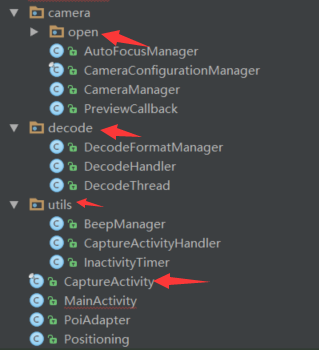
1. **身份识别部分**
   1. **身份识别的选择**

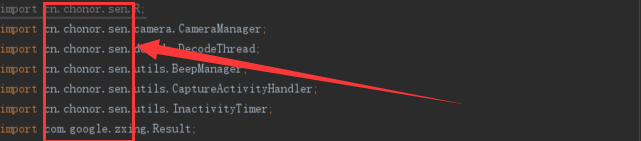
当前手机中较为流行的身份识别技术是以指纹识别为主，最近也在兴起人脸识别和虹膜识别，但是这几种识别技术都需要有专门的硬件，同时还要涉及到专门的API以及系统权限问题。所以我们此处的身份识别使用日常较为简单的二维码模式，通过给每个人一个特定的二维码实现识别。

* 1. **二维码扫描的实现**

此处我们使用Google开发的ZXingAPI进行识别，同样的我们去Google官方的GitHub上下载ZXing，然后将其加入我们的项目中。

* + - 首先我们只需要修改导入的部分的import，箭头部分为google导入后自带的程序文件。是我们运行时候的依赖文件。





* + - 在CaptureActivity中配置二维码扫描部分的外观和扫描区域的大小

|  |
| --- |
| @Override public void onCreate(Bundle icicle) {  super.onCreate(icicle);   Window window = getWindow();  window.addFlags(WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*);  setContentView(R.layout.*activity\_capture*);  //设置拍照界面  scanPreview = (SurfaceView) findViewById(R.id.*capture\_preview*);  scanContainer = (RelativeLayout) findViewById(R.id.*capture\_container*);  scanCropView = (RelativeLayout) findViewById(R.id.*capture\_crop\_view*);  scanLine = (ImageView) findViewById(R.id.*capture\_scan\_line*);   inactivityTimer = new InactivityTimer(this);  beepManager = new BeepManager(this);  //动画参数设置  TranslateAnimation animation = new TranslateAnimation(Animation.*RELATIVE\_TO\_PARENT*, 0.0f, Animation.*RELATIVE\_TO\_PARENT*, 0.0f, Animation.*RELATIVE\_TO\_PARENT*, 0.0f, Animation.*RELATIVE\_TO\_PARENT*,  0.9f);  animation.setDuration(4500);  animation.setRepeatCount(-1);  animation.setRepeatMode(Animation.*RESTART*);  scanLine.startAnimation(animation); }  private void initCrop() {  int cameraWidth = cameraManager.getCameraResolution().y;  int cameraHeight = cameraManager.getCameraResolution().x;  */\*\* 获取布局中扫描框的位置信息 \*/* int[] location = new int[2];  scanCropView.getLocationInWindow(location);  int cropLeft = location[0];  int cropTop = location[1] - getStatusBarHeight();  int cropWidth = scanCropView.getWidth();  int cropHeight = scanCropView.getHeight();  */\*\* 获取布局容器的宽高 \*/* int containerWidth = scanContainer.getWidth();  int containerHeight = scanContainer.getHeight();  */\*\* 计算最终截取的矩形的左上角顶点x坐标 \*/* int x = cropLeft \* cameraWidth / containerWidth;  */\*\* 计算最终截取的矩形的左上角顶点y坐标 \*/* int y = cropTop \* cameraHeight / containerHeight;  */\*\* 计算最终截取的矩形的宽度 \*/* int width = cropWidth \* cameraWidth / containerWidth;  */\*\* 计算最终截取的矩形的高度 \*/* int height = cropHeight \* cameraHeight / containerHeight;  */\*\* 生成最终的截取的矩形 \*/* mCropRect = new Rect(x, y, width + x, height + y); } |

其他部分代码见附件，因为是google的官方代码没有改动。

* + - 我们在MainActivity中设置一个转跳按钮进入二维码进入google配置好的界面，然后在设置接收传回的二维码图像和二维码信息。

|  |
| --- |
| public void Check(){// 获取二维码图像信息  Bundle extras = getIntent().getExtras();  if (null != extras) {  String result = extras.getString("result");  // 获取二维码信息  mResultText.setText(result);  // 获取二维码图像  Bitmap barcode = null;  byte[] compressedBitmap = extras.getByteArray(DecodeThread.*BARCODE\_BITMAP*);  if (compressedBitmap != null) {  barcode = BitmapFactory.*decodeByteArray*(compressedBitmap, 0, compressedBitmap.length, null);  // Mutable copy:  barcode = barcode.copy(Bitmap.Config.*RGB\_565*, true);  }  mResultImage.setImageBitmap(barcode);  } } |

* 1. **二维码扫描的结果展示**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 扫描效果 | 扫描结果的读取 | 二维码扫描的结果 |

此处我们实现的只是结果的提取，如果要做认证的话需要结合信息数据库，此部分因为时间问题没有实现。

## 项目总结与展望

1. **总结**

项目中实现了定位，当前位置温度检测和二维码身份识别。定位我们使用了百度提供的API，检测温度使用了手机自带的电池温度传感器，二维码识别使用了ZXing API，这些基础功能是我们都实现并集成在一个APP中的。

项目中也有不足，比如我们获取到的周围温度因为条件的限制并不是一个真正的温度，只是一个近似值，在我们使用过手机后或者仅仅是将手机握在手里都会导致温度的上升，要解决这个问题用手机是不行的，需要额外的附加嵌入式传感器。远离其他的干扰才可以获得真的温度。项目中的身份识别也是不完美的，我们虽然得到了二维码的信息，但是二维码这个始终安全性不是很高，所以最好还是能做到调用指纹等生物识别方式。

1. **展望**

项目中的定位，未来希望可以做一个服务器端，将手机上的位置信息等间隔的传入到服务器的数据库中，然后服务器在架设一个web控制台，可以获取到几小时或者几天内用户端的位置变换情况，并把路线通过地图展示出来。

同时我们可以把温度等信息同时传输到服务器上做一个记录用来查询。

识别这个部分就是依赖服务器做一个用户数据库，然后通过我们扫描到的二维码中提取出来的信息，去做匹配，然后服务器以此作为客户端登录的凭据。同时也可以实现客户端登录后反登陆服务器进行信息的查询。