软件详细设计说明书（例）

=

学校名称	西南大学
研究院	广东顺德工业创新研究院
指导老师	梁铭炬
组长	王成育
组员	金龙 田新明 叶昊晖 刘佳浩
日期	2022.3.31

目录

1、引言

1.1 编写目的

随着人们对美好生活的追求以及人工智能的大力发展和信息技术的快速更新迭代，智能家居慢慢地走进了大家的生活，“万物互联”的概念也深入人心，人们希望自己能够更简单更方便的控制家里的家居：在下班回家的路上提前调好空调的温度、下雨天远程关闭好门窗、躺在床上手动调整灯的状态、实时获取包括天气、温度等信息在内的每日生活信息.....基于此背景，本团队旨在开发一个小度家庭机器人以满足市民们的相关需求，提升用户们的生活质量。本文主要基于客户对该产品的需求和产品需要具备的功能和条件出发，为小度智能家庭机器人初期的规划奠定需求分析的基础。

1.2 范围

说明：

a. 待开发的软件系统的名称：SmartScreen 智能家庭机器人
b. 项目的任务提出者：梁铭炬
c. 开发者：（首字母排行）：金龙 刘佳浩 田新明 王成育 叶昊辉
d. 用户以及将运行该项软件的单位 ：广东顺德创新设计研究院

1.3 参考资料

开发过程之中使用的资料：

a. 资料，包括所要用到的软件开发标准：消息队列遥测传输协议，泛化协议，Alink 开发协议，
b. 本文件中各处引用的资料文献：

[1] 张照杰. 基于 ArcGISEngine 的多源数据拓扑检查研究 [J] . 北京测绘，2014，28(4):45－50
[2] 徐明钦. 基于 UML 的学生信息管理系统的开发与研究 [J] . 数字技术与应用，2012(2):142-144.
[3] 文潇.基于工业物联网的机器人测试系统与远程监控平台的研究[D].芜湖:安徽工程大学, 2016.
[4] 吴永琢.智能物联机器人系统的研究[J].制造业自动化, 2012, 34(7) :135-138
[5] 刘翼.B 公司智能家用机器人项目可行性分析[D].长沙:湖南大学, 2016.

2、总体设计

2.1 需求规定

对功能的规定：

该软件运行平台为 IPC-M10R800-A3399C 六核工业级行业平板

软件支持系统为 Android

物联网平台为阿里云

2.2 运行环境

开发工具：

- 平台: MacOs ,Windows 2010,Linux
- 开发工具：java, Intellij idea ,android studio ,sdk

测试环境：

MacOs ,Windows 2010,Linux

2.3 基本设计功能和设计模块

由于软件是为家庭设计，秉持着老少皆宜的观念，软件设计的界面简洁且直观，可以适应包括小孩儿、老人在内的所有年龄段的人使用，同时添加的语音控制模块，可以近一步降低软件的使用难度并增添了一定的趣味性。软件预设计了六个主体部分，来实现软件手动或语音智能控制家居：

2.3.1 股票模块

通过该模块用户可以每天实时获取自己家庭所买的股票信息，不需要专门打开股票软件，在吃早饭的闲暇时候可以更快更有效率地查看自己想要了解的股票信息。

2.3.2 油价模块

油价信息对于有车一族来说是十分重要的，通过该模块用户可以实时获取该地区油价的变化，以此来判断自己上班所要采用的交通方式。

2.3.3 天气预测模块

通过该模块用户可以实时获取本地的天气状况、温度以及 PM2.5 数值，根据这些信息来进行穿衣的选择以及出门是否需要带雨具。

2.3.4 人脸识别模块

该模块主要用于此软件的注册和登录功能，如今的人们需要注册各类的软件，那么多的账号和密码经常让人弄混，由于该软件只适用于用户本身的家庭，只需要把家庭成员的信息录入就行，通过人脸识别模块可以更有效率地进行软件的注册和登录功能，避免了由于账号和密码忘记所引起的一系列繁琐的操作。尤其对于一些年龄大的老人和年幼的孩子来说，这个功能可以让这些特殊人群更方便地操作此软件，做到更贴近家庭的设计理念。

2.3.5 智能家居模块

该模块主要包含对窗帘、台灯以及空调的智能控制，其中每个控制又提供了两种选择，一种是在软件上面进行按钮的操控，另一种是通过语音识别功能来对家居进行控制。

窗帘智能控制：通过该模块用户可以根据软件上面的天气信息以及对阳光的需求来控制窗帘的推拉，同时软件控制提供了远程操控的功能，即使身在远门也可以实时根据用户的需求来对窗帘进行控制，语音控制的加入让用户可以更近一步地简化对家居的控制。

台灯智能控制：通过该模块用户可以根据自己的需求来改变灯光的亮度，同时可以远程查看台灯是否关闭，以防出门仓促忘记台灯的关闭，做到更加地环保。

空调的智能控制：通过该模块，用户可以在软件远程控制空调的开关，在回家之前调节到用户自己想要的温度，同时智能语音控制的加入可以让用户根据自己身体反馈实时进行调整，尤其在深夜，不需要起身找遥控器或者平板来控制，做到了更加地安全和便利，更好地提升用户的体验。

2.3.6 语音识别模块

该模块的加入让该软件更加的智能化和人性化，给智能家具模块提供了更加智能化的控制。通过该模块，用户可以和软件进行更加人性化的交互，尤其对于一些不熟悉智能设备的老年人和认知能力不足的孩子来说，该模块可以让这些特殊人群无障碍进行操作和控制。同时语音模块的加入给用户增添了一定的乐趣，能够实现更好的人机交互。

2.4 对性能的要求：

1.软件做到不崩溃，不死机，不长时间卡顿，功能稳定实现，延迟低，安全
2.操作方式：做到点触实现和语音唤醒
3.数据管理能力：用户注册信息不丢失，完整实现手机-服务器-设备访问功能。
4.故障处理要求：由于事故导致关机、断电后重启保证软件正常、功能完善。

3、人脸注册模块

3.1 人脸检测

3.1.1 接口能力

- 人脸检测：检测图片中的人脸并标记出位置信息;
- 人脸关键点：展示人脸的核心关键点信息，及 150 个关键点信息。
- 人脸属性值：展示人脸属性信息，如年龄、性别等。
- 人脸质量信息：返回人脸各部分的遮挡、光照、模糊、完整度、置信度等信息。

3.1.2 质量检测

如果需要判断一张图片中的人脸，是否符合后续识别或者对比的条件，可以使用此接口,在请求时在face_field参数中请求quality。基于返回结果quality中，以下字段及对应阈值，进行质量检测的判断，以保证人脸质量符合后续业务操作要求。

以下指标数据如下图所示：

指标	字段与解释	推荐数值界限
遮挡范围	**occlusion**，取值范围[0~1]，0为无遮挡，1是完全遮挡 含有多个具体子字段，表示脸部多个部位 通常用作判断头发、墨镜、口罩等遮挡	left_eye : 0.6, #左眼被遮挡的阈值 right_eye : 0.6, #右眼被遮挡的阈值 nose : 0.7, #鼻子被遮挡的阈值 mouth : 0.7, #嘴巴被遮挡的阈值 left_cheek : 0.8, #左脸颊被遮挡的阈值 right_cheek : 0.8, #右脸颊被遮挡的阈值 chin_contour : 0.6, #下巴被遮挡阈值
模糊度范围	**blur**，取值范围[0~1]，0是最清晰，1是最模糊	小于0.7
光照范围	**illumination**，取值范围[0~255] 脸部光照的灰度值，0表示光照不好 以及对应客户端SDK中，YUV的Y分量	大于40
姿态角度	**Pitch**：三维旋转之俯仰角度[-90(上), 90(下)] **Roll**：平面内旋转角[-180(逆时针), 180(顺时针)] **Yaw**：三维旋转之左右旋转角[-90(左), 90(右)]	分别小于20度
人脸完整度	**completeness**（0或1），0为人脸溢出图像边界， 1为人脸都在图像边界内	视业务逻辑判断
人脸大小	人脸部分的大小 建议长宽像素值范围：80*80~200*200	人脸部分不小于**100*100**像素

3.1.3 调用方式

向 API 服务地址使用 POST 发送请求，必须在 URL 中带上参数 access_token，可通过后台的 API Key 和 Secret Key 生成

3.2 人脸搜索

3.2.1 能力介绍

- 人脸搜索：也称为 1：N 识别，在指定人脸集合中，找到最相似的人脸；

3.2.2 调用方式

向 API 服务地址使用 POST 发送请求，必须在 URL 中带上参数 access_token，可通过后台的 API Key 和 Secret Key 生成

3.3 人脸库管理

3.3.1 能力介绍

人脸库管理相关接口，要完成 1：N 或者 M：N 识别，首先需要构建一个人脸库，用于存放所有人脸特征，相关接口如下：

- 更新人脸库中指定用户下的人脸信息
- 人脸删除：删除指定用户的某张人脸
- 用户信息查询：查询人脸库中某个用户的详细信息
- 获取用户人脸列表：获取某个用户组中的全部人脸列表
- 获取用户列表：查询指定用户组中的用户列表
- 复制用户：将指定用户复制到另外的人脸组
- 删除用户：删除指定用户
- 创建用户组：创建一个新的用户组
- 删除用户组：删除指定用户组
- 组列表查询：查询人脸库中用户组的列表

3.3.2 人脸库结构

人脸库、用户组、用户、用户下的人脸层级关系如下所示：

```
|- 人脸库(appid)
   |- 用户组一 (group_id)
      |- 用户01 (uid)
         |- 人脸 (faceid)
      |- 用户02 (uid)
         |- 人脸 (faceid)
         |- 人脸 (faceid)
         ....
       ....
   |- 用户组二 (group_id)
   |- 用户组三 (group_id)
   ....
```

3.3.3 关于人脸库的设置限制

- 每个 appid 对应一个人脸库，且不同 appid 之间，人脸库互不相通；
  - 每个人脸库下，可以创建多个用户组，用户组（group）数量没有限制；
  - 每个用户组（group）下，可添加无限个 user_id，无限张人脸（注：为了保证查询速度，单个 group 中的人脸容量上限建议为 80 万）；
  - 每个用户（user_id）所能注册的最大人脸数量 20；

3.3.4 调用方式

向 API 服务地址使用 POST 发送请求，必须在 URL 中带上参数 access_token，可通过后台的 API Key 和 Secret Key 生成

3.4 核心代码

3.4.1 点击注册人脸

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.activity_main);
        user_name = this.findViewById(R.id.user_name);
        View v = this.findViewById(R.id.tv_face_login);
        v.setOnClickListener(this);
        v = this.findViewById(R.id.tv_face_register);
        v.setOnClickListener(this);
        v = this.findViewById(R.id.tv_login);
        v.setOnClickListener(this);
        v = this.findViewById(R.id.enter_btn);
        v.setOnClickListener(this);
    }
```

3.4.2 人脸注册验证

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
 {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE_OP && resultCode == RESPONSE_CODE_RE
GISTER_SUCCESS) {
```

```
4.              Toast.makeText(this, "人脸注册成功!
    ", Toast.LENGTH_SHORT).show();
5.          }else if (requestCode == REQUEST_CODE_OP && resultCode == RESPONSE_C
    ODE_REGISTER_FAIL) {
6.              Toast.makeText(this, "人脸注册失败!
    ", Toast.LENGTH_SHORT).show();
7.          }else if (requestCode == REQUEST_CODE_OP && resultCode == RESPONSE_C
    ODE_LOGIN_SUCCESS) {
8.              String name = data.getStringExtra("name");
9.              setLoginok(name);
10.             Toast.makeText(this, "人脸登录成功!
    ---"+name, Toast.LENGTH_SHORT).show();
11.         }else if (requestCode == REQUEST_CODE_OP && resultCode == RESPONSE_C
    ODE_LOGIN_FAIL) {
12.             Toast.makeText(this, "人脸登录失败!
    ", Toast.LENGTH_SHORT).show();
13.             login_status = false;
14.         }else if (requestCode == REQUEST_CODE_LOGIN && resultCode == RESPONS
    E_CODE_LOGIN_SUCCESS) {
15.             String name = data.getStringExtra("name");
16.             setLoginok(name);
17. //              Toast.makeText(this, "账号密码登录成功!
    ---"+name, Toast.LENGTH_SHORT).show();
18.         }else
19.         if (requestCode == REQUEST_CODE_IMAGE_OP && resultCode == RESULT_OK)
     {
20.             Uri mPath = data.getData();
21.             String file = getPath(mPath);
22.             Bitmap bmp = Application.decodeImage(file);
23.             if (bmp == null || bmp.getWidth() <= 0 || bmp.getHeight() <= 0 )
     {
24.                 Log.e(TAG, "error");
25.             } else {
26.                 Log.i(TAG, "bmp [" + bmp.getWidth() + "," + bmp.getHeight());

27.             }
28.             startRegister(bmp, file);
29.         } else if (requestCode == REQUEST_CODE_OP) {
30.             Log.i(TAG, "RESULT =" + resultCode);
31.             if (data == null) {
32.                 return;
33.             }
34.             Bundle bundle = data.getExtras();
35.             String path = bundle.getString("imagePath");
```

```
36.             Log.i(TAG, "path="+path);
37.         } else if (requestCode == REQUEST_CODE_IMAGE_CAMERA && resultCode ==
    RESULT_OK) {
38.             Uri mPath = ((Application)(MainActivity.this.getApplicationConte
    xt())).getCaptureImage();
39.             String file = getPath(mPath);
40.             Bitmap bmp = Application.decodeImage(file);
41.             startRegister(bmp, file);
42.         }
43.     }
```

4、语音控制模块

4.1 下载安装

- 从 maven 服务器下载最新版本 SDK，下载 Demo 源码 ZIP 包。

```
<dependency>
        <groupId>com.alibaba.nls</groupId>
        <artifactId>nls-sdk-transcriber</artifactId>
        <version>2.2.0</version>
</dependency>
```

Demo 解压后，在 pom 目录运行 mvn package，会在 target 目录生成可执行 JAR：nls-example-transcriber-2.0.0-jar-with-dependencies.jar，将 JAR 包拷贝到目标服务器，用于快速验证及压测服务。

- 服务验证。

运行代码，并按提示提供相应参数。运行后在命令执行目录生成 logs/nls.log。

- 服务压测。

运行如下代码，并按提示提供相应参数。其中阿里云服务 url 参数为：wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1

4.2 关键接口

- NlsClient：语音处理客户端，利用该客户端可以进行一句话识别、实时语音识别和语音合成的语音处理任务。该客户端为线程安全，建议全局仅创建一个实例。
- SpeechTranscriber：实时语音识别类，通过该接口设置请求参数，发送请求及声音数据。非线程安全。
- SpeechTranscriberListener：实时语音识别结果监听类，监听识别结果。非线程安全。

4.3 SDK 关键接口

- initialize：初始化 SDK。

```
/**
     * 初始化 SDK，SDK 为单例，请先释放后再次进行初始化。请勿在 UI 线程调用，可能会引起阻塞。
     * @param callback：事件监听回调，参见下文具体回调。
     * @param parameters：初始化参数，参见接口说明。
```

```
        * @param level：log 打印级别，值越小打印越多。
        * @param save_log：是否保存 log 为文件，存储目录为 parameter 中的 debug_path 字段值。
        * @return：参见错误码。
        */
    public synchronized int initialize(final INativeNuiCallback callback,
                                       String parameters,
                                       final Constants.LogLevel level,
                                       final boolean save_log)

λ   onNuiAudioStateChanged：根据音频状态进行录音功能的开关。
/**
        * 当 start/stop/cancel 等接口调用时，SDK 通过此回调通知 App 进行录音的开关操作。
        * @param state：录音需要的状态（打开/关闭）。
        */
void onNuiAudioStateChanged(AudioState state);

λ   onNuiNeedAudioData：在回调中提供音频数据。

/**
        * 当开始识别时，此回调被连续调用，App 需要在回调中进行语音数据填充。
        * @param buffer：用于给用户填充语音的存储区。
        * @param len：需要填充语音的字节数。
        * @return：实际填充的字节数。
        */
int onNuiNeedAudioData(byte[] buffer, int len);

λ   onNuiEventCallback：SDK 事件回调。
/**
        * SDK 主要事件回调
        * @param event：回调事件，参见如下事件列表。
        * @param resultCode：参见错误码，在出现 EVENT_ASR_ERROR 事件时有效。
        * @param arg2：保留参数。
        * @param kwsResult：语音唤醒功能。
        * @param asrResult：语音识别结果。
        */
void onNuiEventCallback(NuiEvent event, final int resultCode, final int arg2, KwsResult kwsResult, AsrResult asrResult);

λ   onNuiAudioRMSChanged：音频能量值回调。
/**
        * 音频能量值回调
        * @param val: 音频数据能量值回调，范围-160 至 0，一般用于 UI 展示语音动效
        */
```

```
public void onNuiAudioRMSChanged(float val);

λ   set_params：以 JSON 格式设置 SDK 参数。
/**
     * 以 JSON 格式设置参数
     * @param params：参见接口说明。
     * @return：参见错误码。
     */
public synchronized int setParams(String params)

λ   startDialog：开始识别。

/**
     * 开始识别
     * @param vad_mode：多种模式，对于识别场景，使用 P2T。
     * @return：参见错误码。
     */
public synchronized int startDialog(VadMode vad_mode, String dialog_params)

λ   stopDialog：结束识别。
/**
     * 结束识别，调用该接口后，服务端将返回最终识别结果并结束任务。
     * @return：参见错误码。
     */
public synchronized int stopDailog()
λ   release：释放 SDK。
/**
     * 释放 SDK 资源
     * @return：参见错误码。
     */
public synchronized int release()

λ   NUI SDK 初始化
//拷贝资源
CommonUtils.copyAssetsData(this);
//SDK 初始化
int ret = NativeNui.GetInstance().initialize(this, genInitParams(path,path2), C
onstants.LogLevel.LOG_LEVEL_VERBOSE, true);

λ   其中，genInitParams 生成为 String JSON 字符串，包含资源目录和用户信息。其中用户信息
包含如下字段。
private String genInitParams(String workpath, String debugpath) {
        String str = "";
        try{
```

```
                JSONObject object;
                object.put("app_key","");
                object.put("token","");
                object.put("device_id",Utils.getDeviceId());
                object.put("url","wss://nls-gateway.cn-shanghai.aliyuncs.com:443/ws
/v1");
                object.put("workspace", workpath);
                object.put("debug_path",debugpath);
                str = object.toString();
            } catch (JSONException e) {
                e.printStackTrace();
            }
      return str;
  }
  λ    以 JSON 字符串形式进行设置。
  private String genParams() {
      String params = "";
      try {
          JSONObject nls_config = new JSONObject();
          nls_config.put("enable_intermediate_result", true);
          JSONObject parameters = new JSONObject();
          parameters.put("nls_config", nls_config);
          //选择实时语音识别服务
          parameters.put("service_type", Constants.kServiceTypeSpeechTranscriber);
          params = parameters.toString();
      } catch (JSONException e) {
          e.printStackTrace();
      }
      return params;
  }
  NativeNui.GetInstance().setParams(genParams());

  通过 startDialog 接口开启监听。
  NativeNui.GetInstance().startDialog(Constants.VadMode.TYPE_P2T, genDialogParams
());

   回调处理
  public void onNuiAudioStateChanged(Constants.AudioState state) {
            Log.i(TAG, "onNuiAudioStateChanged");
            if (state == Constants.AudioState.STATE_OPEN) {
                Log.i(TAG, "audio recorder start");
                mAudioRecorder.startRecording();
            } else if (state == Constants.AudioState.STATE_CLOSE) {
                Log.i(TAG, "audio recorder close");
```

```
            mAudioRecorder.release();
        } else if (state == Constants.AudioState.STATE_PAUSE) {
            Log.i(TAG, "audio recorder pause");
            mAudioRecorder.stop();
        }
    }

    λ   onNuiNeedAudioData：录音数据回调，在该回调中填充录音数据。
    public int onNuiNeedAudioData(byte[] buffer, int len) {
        int ret = 0;
        if (mAudioRecorder.getState() != AudioRecord.STATE_INITIALIZED) {
            Log.e(TAG, "audio recorder not init");
            return -1;
        }
        ret = mAudioRecorder.read(buffer, 0, len);
        return ret;
    }
```

4.4 核心代码

语音唤醒

```
1.  public abstract class WakeUpUtil {
2.      /**
3.       * 唤醒的回调
4.       */
5.      public abstract void wakeUp();
6.
7.      // Log 标签
8.      private static final String TAG = "WakeUpUtil";
9.
10.     // 上下文
11.     private Context mContext;
12.     // 语音唤醒对象
13.     private VoiceWakeuper mIvw;
14.
15.     private int curThresh = 1450;
16.
17.     public WakeUpUtil(Context context) {
18.         mContext = context;
19.
20.         // 初始化唤醒对象
21.         mIvw = VoiceWakeuper.createWakeuper(context, null);
22.     }
```

```
23.
24.     /**
25.      * 获取唤醒词功能
26.      *
27.      * @return 返回文件位置
28.      */
29.     private String getResource() {
30.         final String resPath = ResourceUtil.generateResourcePath(mContext, R
    ESOURCE_TYPE.assets, "ivw/6049c765.jet");
31.         return resPath;
32.     }
33.
34.     /**
35.      * 唤醒
36.      */
37.     public void wake() {
38.         // 非空判断，防止因空指针使程序崩溃
39.         mIvw = VoiceWakeuper.getWakeuper();
40.         if (mIvw != null) {
41.             // textView.setText(resultString);
42.             // 清空参数
43.             mIvw.setParameter(SpeechConstant.PARAMS, null);
44.             // 设置唤醒资源路径
45.             mIvw.setParameter(SpeechConstant.IVW_RES_PATH, getResource());
46.             // 唤醒门限值，根据资源携带的唤醒词个数按照“id:门限;id:门限”的格式传
    入
47.             mIvw.setParameter(SpeechConstant.IVW_THRESHOLD, "0:" + curThresh)
    ;
48.             // 设置唤醒模式
49.             mIvw.setParameter(SpeechConstant.IVW_SST, "wakeup");
50.             // 设置持续进行唤醒
51.             mIvw.setParameter(SpeechConstant.KEEP_ALIVE, "1");
52.             mIvw.startListening(mWakeuperListener);
53.         } else {
54.             Toast.makeText(mContext, "唤醒未初始化
    ", Toast.LENGTH_SHORT).show();
55.         }
56. }
```

语音识别:

```
1.  public abstract class BuildLocalGrammar {
2.
3.      /**
```

```
4.       * 构建语法的回调
5.       *
6.       * @param errMsg null 构造成功
7.       */
8.      public abstract void result(String errMsg, String grammarId);
9.
10.     // Log 标签
11.     private static final String TAG = "BuildLocalGrammar";
12.
13.     public static final String GRAMMAR_PATH = Environment.getExternalStorage
    Directory().getAbsolutePath() + "/msc/test";
14.     // 上下文
15.     private Context mContext;
16.     // 语音识别对象
17.     private SpeechRecognizer mAsr;
18.
19.     public BuildLocalGrammar(Context context) {
20.         mContext = context;
21.
22.         // 初始化识别对象
23.         mAsr = SpeechRecognizer.createRecognizer(context, new InitListener()
     {
24.
25.             @Override
26.             public void onInit(int code) {
27.                 Log.d(TAG, "SpeechRecognizer init() code = " + code);
28.                 if (code != ErrorCode.SUCCESS) {
29.                     result(code + "", null);
30.                     Log.d(TAG, "初始化失败,错误码：" + code);
31.                     Toast.makeText(mContext, "初始化失败,错误码：
    " + code, Toast.LENGTH_SHORT).show();
32.                 }
33.             }
34.         });
35.     }
36.
37.     ;
38.
39.     /**
40.      * 构建语法
41.      *
42.      * @return
43.      */
44.     public void buildLocalGrammar() {
```

```
45.         try {
46.             /*
47.              * TODO 如果你要在程序里维护 bnf 文件，可以在这里加上你维护的一些逻辑
48.              * 如果不嫌麻烦，要一直改 bnf 文件，这里的代码可以不用动，不过我个人不建议一直手动修改 bnf 文件
49.              * ，内容多了以后很容易出错，不好找 Bug，建议每次改之前先备份。 建议用程序维护 bnf 文件。
50.              */
51.
52.             /*
53.              * 构建语法
54.              */
55.             String mContent;// 语法、词典临时变量
56.             String mLocalGrammar = FucUtil.readFile(mContext, "call.bnf", "utf-8");
57.             mContent = new String(mLocalGrammar);
58.             mAsr.setParameter(SpeechConstant.PARAMS, null);
59.             // 设置文本编码格式
60.             mAsr.setParameter(SpeechConstant.TEXT_ENCODING, "utf-8");
61.             // 设置引擎类型
62.             mAsr.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL);
63.             // 设置语法构建路径
64.             mAsr.setParameter(ResourceUtil.GRM_BUILD_PATH, GRAMMAR_PATH);
65.             // 使用 8k 音频的时候请解开注释
66.             // mAsr.setParameter(SpeechConstant.SAMPLE_RATE, "8000");
67.             // 设置资源路径
68.             mAsr.setParameter(ResourceUtil.ASR_RES_PATH, getResourcePath());
69.             // 构建语法
70.             int ret = mAsr.buildGrammar("bnf", mContent, new GrammarListener() {
71.
72.                 @Override
73.                 public void onBuildFinish(String grammarId, SpeechError error) {
74.                     if (error == null) {
75.                         Log.d(TAG, "语法构建成功");
76.                         result(null, grammarId);
77.                     } else {
78.                         Log.d(TAG, "语法构建失败,错误码:" + error.getErrorCode());
79.                         result(error.getErrorCode() + "", grammarId);
80.                     }
```

```
                }
            });
            if (ret != ErrorCode.SUCCESS) {
                Log.d(TAG, "语法构建失败,错误码:" + ret);
                result(ret + "", null);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
}

读取声音:

public class TTSUtil {
    // 发音人
    public final static String[] COLOUD_VOICERS_VALUE = {"xiaoyan", "xiaofeng"};

    public static final String TAG = "TTSUtility";
    // 语音合成对象
     public static SpeechSynthesizer mTts;
     //上下文
     public Context mContext;

     public volatile static com.gidisoft.sdk_demo.TTS.TTSUtil instance;


     /**
      * 合成回掉监听
      */
     public static SynthesizerListener mTtsListener = new SynthesizerListener() {
         @Override
         public void onSpeakBegin() {
             Log.d(TAG, "开始播放");
         }

         @Override
         public void onBufferProgress(int percent, int beginPos, int endPos, String info) {
             // TODO 缓冲的进度
             Log.d(TAG, "缓冲 : " + percent);
         }

```

```
122.          @Override
123.          public void onSpeakPaused() {
124.              Log.d(TAG, "暂停播放");
125.
126.          }
127.
128.          @Override
129.          public void onSpeakResumed() {
130.              Log.d(TAG, "继续播放");
131.          }
132.
133.          @Override
134.          public void onSpeakProgress(int percent, int beginPos, int endPos)
    {
135.              // TODO 说话的进度
136.              Log.d(TAG, "合成 : " + percent);
137.          }
138.
139.          @Override
140.          public void onCompleted(SpeechError error) {
141.              if (error == null) {
142.                  Log.d(TAG, "播放完成");
143.              } else if (error != null) {
144.                  Log.d(TAG, error.getPlainDescription(true));
145.              }
146.          }
147.
148.          @Override
149.          public void onEvent(int eventType, int arg1, int arg2, Bundle obj)
    {
150.
151.          }
152. };
```

5、预报天气模块

5.1 请求 URL

开发版 https://devapi.qweather.com/v7/weather/now?[请求参数]

5.2 请求参数

请求参数包括必选和可选参数，如不填写可选参数将使用其默认值，参数之间使用&进行分隔。

- key

用户认证 key，请参考如何获取你的 KEY。支持数字签名方式进行认证。例如 key=123456789ABC

- location

需要查询地区的 LocationID 或以英文逗号分隔的经度,纬度坐标(十进制)，LocationID 可通过城市搜索服务获取。例如 location=101010100 或 location=116.41,39.92

5.3 核心代码

- 创建一个 OkHttp 实例:

```
public void getWeatherDataBySeniverseApi() {
        try {
            OkHttpClient client = new OkHttpClient();//创建一个 OkHttp 实例

            Request request = new Request.Builder().url("http://api.seniverse.com/v3/weather/now.json?key=S7VPeL8VAkPGyFM9P&location=foshan&language=zh-Hans&unit=c").build();//创建 Request 对象发起请求,记得替换成你自己的 key


            Response response = client.newCall(request).execute();//创建 call 对象并调用 execute 获取返回的数据


            String responseData = response.body().string();

            System.out.println(responseData);

```

```
                parseJSONWithJSONObject(responseData);//解析 SSON 数据
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

- 用 JSONObect 解析 JSON 数据

```
    private void parseJSONWithJSONObject(String jsonData) {//用 JSONObect 解析 JSON 数据
        try {
            JSONObject jsonObject = new JSONObject(jsonData);
            JSONArray results = jsonObject.getJSONArray("results");    //得到键为 results 的 JSONArray
            JSONObject now = results.getJSONObject(0).getJSONObject("now");//得到键值为"now"的 JSONObject
            JSONObject location = results.getJSONObject(0).getJSONObject("location");   //得到键值为 location 的 JSONObject

            temperature = now.getString("temperature"); //获取温度
            area = location.getString("name");

            tianqi = now.getString("text");

        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
```

6、系统出错处理设计

6.1 出错信息

因为系统的性能可能不太优化，可能会出现一些卡顿的问题，以及调取 API 的时候会出现结果返回不及时的情况

6.2 补救措施

说明故障出现后可能采取的变通措施，包括：

a. 多项选择技术：这里会调取多个大厂商的 API 接口，这样就会返回即使一个数据接口无法即使返回，也可以有其他接口来进行补充

b. 恢复及再启动技术：说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。