Phantom 语音控制 APP 设计文档

版本 2.0

修订日期 2015.01.08

版本	日期	说明	人员
1.0	2014.12.27	首次创建文档	evan
2.0	2015.01.08	增加天线自动追踪功能	evan

一．开发环境

OS: Mac OS X 10.10.1

IDE:ADT 23.0.2.1259578

Java Version: 1.8.0_25

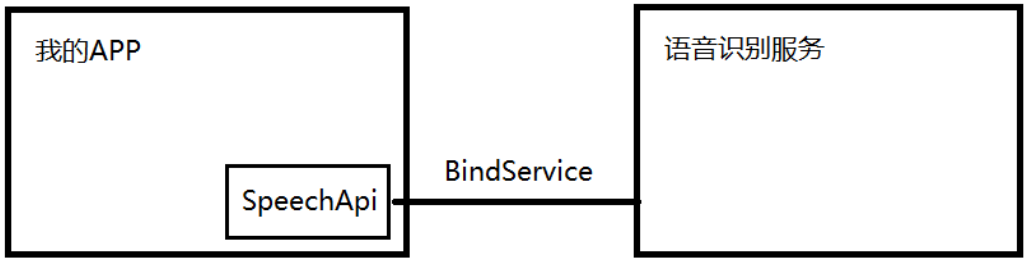
二．语音识别方案选择

方案一：Google 语音识别服务；方案二：讯飞语音识别 SDK

Android 系统自带 Google 语音识别服务，但是 Google 语音识别对中文的支持不是很理想，由于 GFW，在线识别不稳定。不支持离线识别。

讯飞语音识别对中文的支持很好，支持在线识别和离线识别，在线识别完全免费，离线识别分为免费版和收费版，二者无功能上的差异，区别在于免费版需要讯飞语音服务的支持，而收费版本则可将语音识别服务集成到用户 APP 中。

由于手机在连接上 Phantom 的 Wifi 后无法接入互联网，所以只能采用离线识别的方案，所以我们最终采用讯飞免费的离线语音识别方案。中国大部分手机厂商如华为、小米、魅族都在其系统上内置了讯飞语音服务，对于其他没有内置讯飞语音服务的手机，我们会在用户安装 APP 时同时为用户安装语音识别服务。我们使用免费版的语音服务方式如下图所示：



三．天线自动追踪系统方案选择

Arduino 是一款便捷灵活、方便上手的开源电子原型平台，它作为全球最流行的开源硬件，也是一个优秀的硬件开发平台，更是硬件开发的趋势。Arduino 简单的开发方式使得开发者更关注创意与实现，更快的完成自己的项目开发，大大节约了学习的成本，缩短了开发的周期。

蓝牙 4.0 是目前最新蓝牙版本，是 3.0 的升级版本；有更省电、成本低、3 毫秒低延迟、超长有效连接距离等特性，是目前便携式设备通讯开发的常用方案。

综上分析我们采用 Arduino 作为天线自动追踪系统的主控器，通过蓝牙 4.0 模块与手机通讯，数据链路如下图 1 所示：

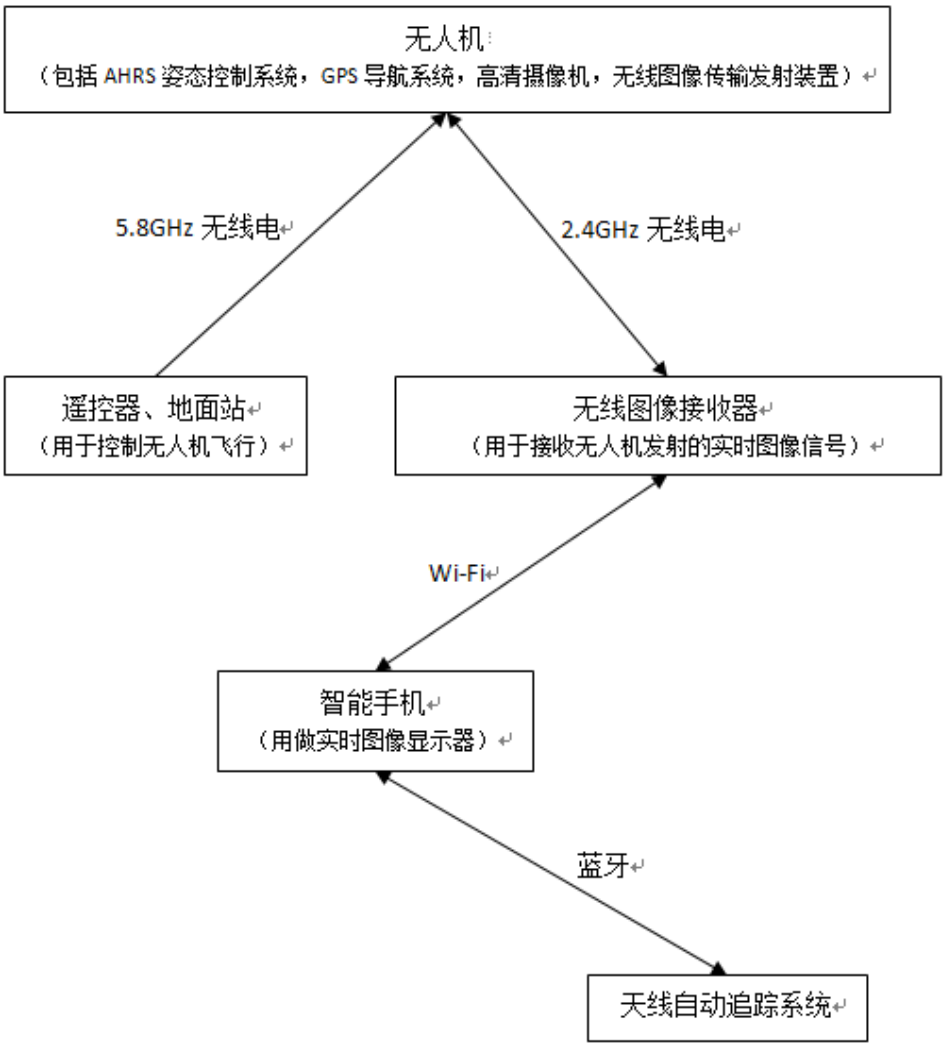


图 1 天线自动追踪系统数据链路

四．代码设计详解

代码结构如下图 2 所示：

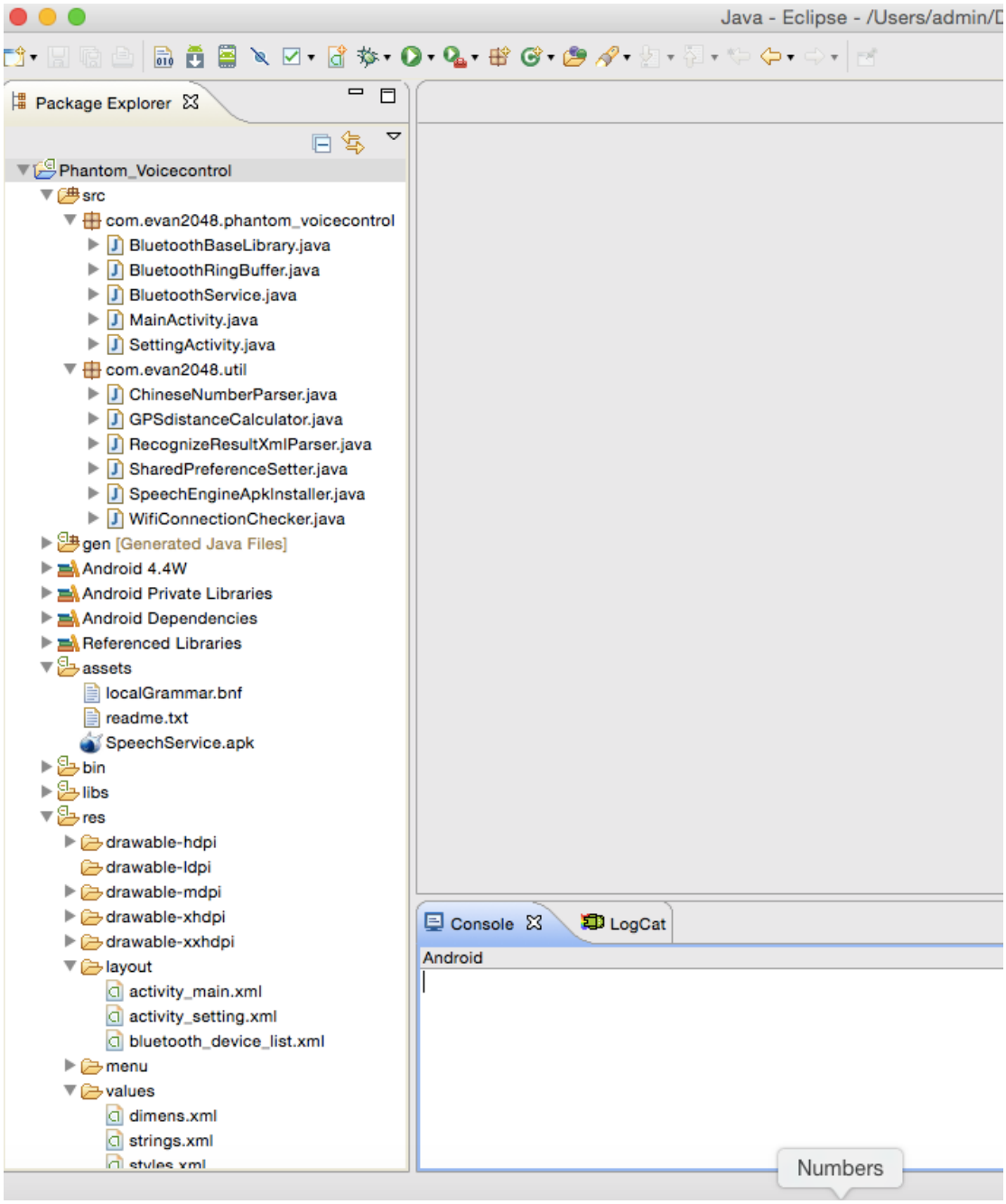


图 2 代码结构

1.src 代码详细设计

src 目录下有两个包，com.evan2048.phantom_voicecontrol 和 com.evan2048.util。主要功能实现类放在 com.evan2048.phantom_voicecontrol 内，工具类放在 com.evan2048.util。

com.evan2048.phantom_voicecontrol 下的 BluetoothBaseLibrary 类、BluetoothRingBuffer 类、BluetoothService 类为蓝牙相关类，为蓝牙连接提供支持；MainActivity 为应用主界面，大部分功能在这里实现；SettingActivity 为应用设置界面，用于设置应用参数。

com.evan2048.util 下的 ChineseNumberParser 为中文数字转为阿拉伯数字工具；GPSdistanceCalculator 为计算两个 GPS 坐标之间的距离、方位的工具；RecognizeResultXmlParser 为将语音识别引擎识别返回的 XML 文本解析器；SharedPreferenceSetter 为应用设置参数保存、修改、删除的工具。SpeechEngineApkInstaller 为语音识别引擎的安装工具；WifiConnectionChecker 为检测 Wifi 连接状态的工具。

2.assets 目录下资源说明

assets 目录里面的文件都是保持原始的文件格式，可以使用 AssetManager 以字节流的形式读取文件。我们在 assets 目录下存放 localGrammar.bnf 文件和 SpeechService.apk 文件。localGrammar.bnf 文件为自定义的语音识别命令词的语法文件；SpeechService.apk 为语音识别服务的安装文件。

3.res 目录下布局文件说明

layout 目录下 activity_main.xml 为应用主界面的布局文件；activity_setting.xml 为设置界面的布局文件；bluetooth_device_list.xml 为蓝牙设备列表对话框的布局文件。

values 目录下 strings.xml 存储着所有应用界面出现的文字。

五．语音识别服务检测与安装

由于手机连接 Phantom 的 Wifi 后无法连接互联网，所以采用免费版的离线语音识别方案，不能将语音识别服务集成到 APP 中，在应用启动时，我们会检测用户手机是否存在语音识别引擎，如果没有则会弹窗提示用户进行安装，实现代码如下：

```
    //检测语音引擎状态
    private boolean checkSpeechEngineState()
    {
        if(checkSpeechServiceInstall()==false ||
SpeechUtility.getUtility(this).queryAvailableEngines()==null ||
SpeechUtility.getUtility(this).queryAvailableEngines().length<
=0)
        {
            //未检测到语音引擎
            showLOG("recognizerEngine not installed");
            //弹窗提醒用户安装语音引擎
            DialogInterface.OnClickListener
positiveButtonOnClickListener=new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface arg0, int arg1)
{
                    // TODO Auto-generated method stub
                    //先尝试本地安装,若不成功则转至网页安装

    if(SpeechEngineApkInstaller.installFromAssets(MainActivity.
this)==true)
                    {
                        showLOG("recognizerEngine installing from
local...");
                    }
                    else
                    {
                        showLOG("recognizerEngine install from local
failed");

    SpeechEngineApkInstaller.openDownloadWeb(MainActivity.this,
SPEECH_ENGINE_DOWNLOAD_URL);
                    }
                }
            };
            new
AlertDialog.Builder(MainActivity.this).setTitle(getString(R.st
ring.alertString)).setMessage(getString(R.string.speechEngineI
nstallMessageString)).setPositiveButton(getString(R.string.con
firmString),
positiveButtonOnClickListener).setNegativeButton(getString(R.s
tring.cancelString), null).show();
            return false;
```

```
        }
        else
        {
            //语音引擎正常
            showLOG("recognizerEngine installed");
            //设置申请的appid

    SpeechUtility.getUtility(this).setAppid(VOICE_API_KEY);
            isSpeechServiceInstall=true;
            return true;
        }
        }
```

六．语音识别结果解析与执行

手机麦克风采集到用户语音后通过语音识别引擎进行识别，返回 XML 形式的识别结果，下面代码实现对 XML 识别结果的解析：

```
        public void onResult(final RecognizerResult result, boolean
isLast) throws RemoteException {
            //识别结果回调(在线识别返回json，离线识别返回xml)
            String
resultCommand=RecognizeResultXmlParser.getResult(result.getRes
ultString());
            int
resultConfidence=RecognizeResultXmlParser.getResultConfidence(
result.getResultString());
            //showLOG(result.getResultString());
            //如果识别结果为空，可能是语音引擎没配置正确
            if(resultCommand.equals(""))
            {
                showLOG("recognize result empty");

    showSpeechCammandMessage((getString(R.string.recognizeResul
tEmptyString)));
                return;
            }
            if(resultConfidence>=speechCammandConfidenceBoundary)
            {
                //执行语音命令
                executeSpeechCommand(resultCommand);
                showLOG("recognize success:"+resultCommand+"
confidence:"+resultConfidence+"/"+speechCammandConfidenceBound
```

```
ary+"  executed");
            }
            else
            {
                showLOG("recognize success:"+resultCommand+"
confidence:"+resultConfidence+"/"+speechCammandConfidenceBound
ary+"  passed");
            }
            //说话结束之后继续监听
            if(isLast==true)
            {
                reStartRecognition();
            }
            }
```

得到识别结果后，执行相应的语音命令，代码如下：

```
    //处理语音命令
    private void executeSpeechCommand(String cammand)
    {
        if(cammand.equals("拍照"))
        {
            DJIDrone.getDjiCamera().startTakePhoto(new
DJIExecuteResultCallback(){
                @Override
                public void onResult(DJIError mErr)
                {
                    if(mErr.errorCode==DJIError.RESULT_OK)
                    {
                        showLOG("voice take photo success");

    showSpeechCammandMessage(getString(R.string.takePhoto)+getS
tring(R.string.success));
                    }
                    else
                    {
                        showLOG("voice take photo failed");

    showSpeechCammandMessage(getString(R.string.takePhoto)+getS
tring(R.string.failed));
                    }
                }
            });
        }
        else if(cammand.equals("开始录像"))
```

```
        {
            DJIDrone.getDjiCamera().startRecord(new
DJIExecuteResultCallback(){
                @Override
                public void onResult(DJIError mErr)
                {
                   if(mErr.errorCode==DJIError.RESULT_OK)
                   {
                      showLOG("voice start recording video
success");
                      isRecordingVideo=true;
                      runOnUiThread(new Runnable() {

                         @Override
                         public void run() {
                            // TODO Auto-generated method stub

   recordingVideoButton.setBackgroundResource(R.drawable.video
_green);
                         }
                      });

   showSpeechCammandMessage(getString(R.string.startRecordingV
ideo));
                   }
                   else
                   {
                      showLOG("voice start recording video failed");

   showSpeechCammandMessage(getString(R.string.startRecordingV
ideo)+getString(R.string.failed));
                   }
                }

            });
        }
        else if(cammand.equals("停止录像"))
        {
            DJIDrone.getDjiCamera().stopRecord(new
DJIExecuteResultCallback(){
                @Override
                public void onResult(DJIError mErr)
                {
                   if(mErr.errorCode==DJIError.RESULT_OK)
```

```
                {
                    showLOG("voice stop recording video success");
                    isRecordingVideo=false;
                    runOnUiThread(new Runnable() {

                        @Override
                        public void run() {
                            // TODO Auto-generated method stub

    recordingVideoButton.setBackgroundResource(R.drawable.video
_gray);
                        }
                    });

    showSpeechCammandMessage(getString(R.string.stopRecordingVi
deo));
                }
                else
                {
                    showLOG("start recording video failed");

    showSpeechCammandMessage(getString(R.string.stopRecordingVi
deo)+getString(R.string.failed));
                }

            }

        });
    }
    else if(cammand.equals("镜头平视"))
    {
        setGimbalPitchAngle(phantom_gimbal_min_angle);

    showSpeechCammandMessage(getString(R.string.gimbalAngleHori
zontal));
    }
    else if(cammand.equals("镜头俯视"))
    {
        setGimbalPitchAngle(phantom_gimbal_max_angle);

    showSpeechCammandMessage(getString(R.string.gimbalAngleVert
ical));
    }
    else if(cammand.equals("镜头上移"))
```

```
        {
            int angle=currentGimbalPitchAngle-200;
            setGimbalPitchAngle(angle);

    showSpeechCammandMessage(getString(R.string.gimbalAngleUp))
;
        }
        else if(cammand.equals("镜头下移"))
        {
            int angle=currentGimbalPitchAngle+200;
            setGimbalPitchAngle(angle);

    showSpeechCammandMessage(getString(R.string.gimbalAngleDown
));
        }
        else if(cammand.equals("电池电量"))
        {
            //should be "电池电量xx%"

    startSynthesize(getString(R.string.battery)+getString(R.str
ing.speechPause)+remainPowerPercent+"%");

    showSpeechCammandMessage(getString(R.string.battery)+remain
PowerPercent+"%");
        }
        else if(cammand.startsWith("镜头角度"))
        {
            String cammandWithoutUnit=cammand;
            //如果带有单位度，先丢弃
            if(cammand.endsWith("度"))
            {
                cammandWithoutUnit=cammand.substring(0,
cammand.length()-1);
            }
            //提取出"十"，"二十"之类的中文
            String
chineseNumberAngle=cammandWithoutUnit.substring(4);
            int
angle=ChineseNumberParser.toIntNumber(chineseNumberAngle);
            //将角度转换为云台对应的值(0-90转换为0-1000)
            int
gimbalAngle=(int)((angle/90.0)*phantom_gimbal_max_angle);
            setGimbalPitchAngle(gimbalAngle);
```

```
    showSpeechCammandMessage(getString(R.string.gimbalAngle)+an
gle+getString(R.string.gimbalAngleUnit));
        }
        }
```

七．总结

最终 APP 能够识别“拍照”、“开始录像”、“停止录像”、“镜头平视”、“镜头俯视”、“镜头上移”、“镜头下移”、“镜头角度 xx 度(0~90)”、“电池电量”这些语音命令，语音识别成功率和响应时间能够达到预期效果。天线自动追踪系统的刷新率受 Phantom 的限制，经测试 Phantom 主控信息回传的最小周期为一秒，所以天线自动追踪系统每隔一秒钟追踪一次。能够 360 无死角的对无人机进行追踪。