**艾米机器人SDK使用说明**

**目录**

[1. 集成SDK 4](#_Toc7063)

[2. 开启远程控制服务 5](#_Toc27057)

[3. 获取地图列表 6](#_Toc29853)

[4. 开启导航 6](#_Toc32286)

[5. 停止导航 8](#_Toc1282)

[6. 获取导航状态 9](#_Toc24462)

[7. 导航到标记界面 10](#_Toc13806)

[8. 获取标记列表 11](#_Toc9650)

[9. 导航到标记点 12](#_Toc20205)

[10. 取消导航 13](#_Toc13843)

[11. 机器人控制API 14](#_Toc13005)

[12. 导航控制API 15](#_Toc22029)

[13. 系统Action 16](#_Toc20530)

[14. 打开/关闭唤醒的方法(只适用A1 A2 2.0版本) 17](#_Toc25988)

[15. 监听唤醒broadcast 17](#_Toc19451)

[16. 第三方开发语音app时注意事项 18](#_Toc20541)

[17. 1.0系统版本(已停止维护)启用/禁用系统语音的方法 18](#_Toc23477)

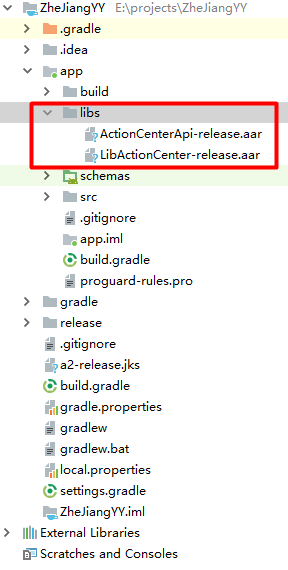
[18. 电量广播及获得电量的方法 19](#_Toc29443)

版本修改记录：

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修改说明 | 修改日期 | 修改人 |
| V1.0 | 新建Amy导航SDK使用说明文档，包括开启导航，停止导航，取消导航，获取地图列表，获取标记列表，导航到标记点等； | 2018-06-21 | mudeyu |
| V2.0 | 添加机器人控制API，导航控制API； | 2018-08-02 | mudeyu |
| V3.0 | 添加系统Action，包括音频，视频，语音播报，语音识别，视频呼叫等； | 2018-08-30 | mudeyu |
| V3.1 | 修改集成sdk方法；调整语音播报api参数；调整语音识别api参数 | 2018-11-28 | Lizz |
| V3.2 | 加入打开/关闭唤醒的方法 | 2018-12-10 | Lizz |
| V3.3 | 监听唤醒broadcast  1.0版本 启用/禁用系统语音的方法 | 2019-01-23 | Lizz |
| V3.4 | 第三方开发语音app时注意事项 | 2019-02-18 | Lizz |
| V3.5 | 加入电量广播及获取电量的方法 | 2019-03-06 | Lizz |

## 集成SDK

1. 将库文件aar拷贝到libs 目录下



1. build.gradle 导入aar



1. AndroidManifest.xml配置如下，其中**your.company.url**请修改为您所在公司的域名



1. 在App-->attachBaseContext方法中添加如下初始化代码，其中**your.company.url**与AndroidManifest.xml中设定的一致



## 开启远程控制服务

先启动远程控制服务，确认版本号大于等于 1.10.2；

查看版本：远程控制->更多设置->版本



打开AmySDKProject进入主界面；如下图所示：



## 获取地图列表

点击刷新地图列表获取地图列表；

代码示例：

robotNavActionApi.getMapList(new ActionResultCallback<MapListEntity>() {

@Override

public void onActionSuccess(final MapListEntity mapListEntity) {

// mapListEntity 地图列表

}

@Override

public void onActionFailed(int errCode, String errInfo, Throwable throwable) {

showToast("获取地图列表错误 " + errCode + ", " + errInfo);

}

});

## 开启导航

点击开启默认导航或者点击地图，开启导航；

使用默认导航地图开启导航，需要先设置默认导航地图；

开启成功返回 ActionCode.ACTION\_OK = 0；

开启失败返回错误码：

//ok

public static final int CODE\_SUCCESS = ActionCode.ACTION\_OK;

//错误

public static final int CODE\_ERROR = 1000;

//无效参数

public static final int CODE\_INVALID\_PARAMS = 1001;

//开启导航失败，一般是工控失败

public static final int CODE\_ERROR\_START\_NAV\_FAILED = 2000;

//无效默认地图id错误，用户没有设置默认导航地图导致

public static final int CODE\_ERROR\_INVALID\_DEFAULT\_MAP\_ID = 2001;

//该地图没有标记点，无法导航

public static final int CODE\_ERROR\_MAP\_MARK\_POINTS\_IS\_EMPTY = 2002;

//标记列表中没有该标记点，无法导航

public static final int CODE\_ERROR\_MAP\_MARK\_POINT\_NOT\_EXIST = 2003;

//导航失败

public static final int CODE\_ERROR\_NAV\_FAILED = 2004;

//导航放弃

public static final int CODE\_ERROR\_NAV\_GIVEUP = 2005;

//导航丢失

public static final int CODE\_ERROR\_NAV\_LOST = 2006;

//导航超时

public static final int CODE\_ERROR\_NAV\_TIME\_OUT = 2007;

//导航取消

public static final int CODE\_ERROR\_NAV\_CANCEL = 2008;

//导航停止

public static final int CODE\_ERROR\_NAV\_STOP = 2009;

**开启导航代码示例：**

int ret = robotNavActionApi.startNav(createStartNavCallBack());

if(ret == ActionCode.ACTION\_OK) {

showLoading("正在开启导航...");

} else {

showToast("错误 " + ret);

}

private ActionNotifyCallBack createStartNavCallBack() {

startNavCallBack = new ActionNotifyCallBack() {

@Override

public void onRecvActionNotify(ActionRequest notifyRequest,

String srcURI, String notifyAction, final int notifyCode,

final String notifyInfo, String notifyParams) {

LogUtils.d(TAG, "onRecvActionNotify srcURI=" + srcURI + ", notifyAction=" + notifyAction

+ ", notifyCode=" + notifyCode + ", notifyInfo=" + notifyInfo

+ ", notifyParams=" + notifyParams);

if (notifyCode == ActionCode.ACTION\_OK) {

//成功

mHandler.post(new Runnable() {

@Override

public void run() {

showToast("开启导航成功");

hideLoading();

startActivity(new Intent(getActivity(), NavMarkPointListActivity.class));

}

});

} else {

//错误，错误码：notifyCode, 错误信息：notifyInfo

mHandler.post(new Runnable() {

@Override

public void run() {

showToast("开启导航失败 " + notifyCode + ", " + notifyInfo);

}

});

}

}

};

return startNavCallBack;

}

## 停止导航

点击停止导航，执行停止导航；

**代码示例：**

robotNavActionApi.stopNav(new ActionNotifyCallBack() {

@Override

public void onRecvActionNotify(ActionRequest notifyRequest,

String srcURI, String notifyAction, final int notifyCode,

final String notifyInfo, String notifyParams) {

LogUtils.d(TAG, "onRecvActionNotify srcURI=" + srcURI + ", notifyAction=" + notifyAction

+ ", notifyCode=" + notifyCode + ", notifyInfo=" + notifyInfo

+ ", notifyParams=" + notifyParams);

if (notifyCode == ActionCode.ACTION\_OK) {

//成功

mHandler.post(new Runnable() {

@Override

public void run() {

hideLoading();

showToast("停止导航成功");

}

});

} else {

//错误，错误码：notifyCode, 错误信息：notifyInfo

mHandler.post(new Runnable() {

@Override

public void run() {

hideLoading();

showToast("停止导航失败 " + notifyCode + ", " + notifyInfo);

}

});

}

}

});

## 获取导航状态

private void getNavState() {

ViewUtils.setText(tvNavState, "正在获取导航状态");

robotNavActionApi.getNavState(new ActionNotifyCallBack() {

@Override

public void onRecvActionNotify(ActionRequest notifyRequest,

String srcURI, String notifyAction, final int notifyCode,

final String notifyInfo, final String notifyParams) {

if (notifyCode == ActionCode.ACTION\_OK) {

//成功, notifyParams

mHandler.post(new Runnable() {

@Override

public void run() {

ViewUtils.setText(tvNavState, notifyParams);

}

});

} else {

//错误，错误码：notifyCode, 错误信息：notifyInfo

mHandler.post(new Runnable() {

@Override

public void run() {

ViewUtils.setText(tvNavState, "err=" + notifyCode + ", " + notifyInfo);

}

});

}

}

});

}

## 导航到标记界面

开启导航，自动跳转到导航到标记界面



## 获取标记列表

点击获取标记列表，获取地图对应的标记列表；

可以选择根据标记坐标导航，根据标记名称导航；

代码示例：

private void getMarkPointList() {

robotNavActionApi.getMarkPointList(null, new ActionResultCallback<List<MarkPointEntity>>() {

@Override

public void onActionSuccess(final List<MarkPointEntity> markPointEntities) {

LogUtils.e(TAG, "getMarkPointList ok");

mHandler.post(new Runnable() {

@Override

public void run() {

if (isFinishing()) {

return;

}

showToast("获取标记列表成功");

onGetMarkPointList(markPointEntities);

}

});

}

@Override

public void onActionFailed(int errCode, String errInfo, Throwable throwable) {

LogUtils.e(TAG, "getMarkPointList failed errCode=" + errCode + ", errInfo=" + errInfo + ", " + throwable);

showToast("获取标记列表失败 " + errCode + ", " + errInfo);

}

});

}

## 导航到标记点

private void navToPoint(final MarkPointEntity markPointEntity) {

tvState.setText("正在导航到: " + markPointEntity.text);

if (navToPointCallBack != null) {

//取消前次监听

navToPointCallBack.cancel();

}

navToPointCallBack = new ActionNotifyCallBack() {

@Override

public void onRecvActionNotify(ActionRequest notifyRequest,

String srcURI, String notifyAction, final int notifyCode,

final String notifyInfo, final String notifyParams) {

if (notifyCode == ActionCode.ACTION\_OK) {

//成功, notifyParams

mHandler.post(new Runnable() {

@Override

public void run() {

String info = "导航到达: " + markPointEntity.text;

showToast(info);

tvState.setText(info);

}

});

} else {

//错误，错误码：notifyCode, 错误信息：notifyInfo

mHandler.post(new Runnable() {

@Override

public void run() {

String info = "导航错误 " + notifyCode + ", " + notifyInfo;

showToast(info);

tvState.setText(info);

}

});

}

}

};

LogUtils.d(TAG, "navToPoint isNavPointByPos =" + isNavPointByPos + ", markPointEntity=" + JSON.toJSONString(markPointEntity));

if (isNavPointByPos) {

robotNavActionApi.navToPointByPostion(markPointEntity.realX, markPointEntity.realY, markPointEntity.realAngle, navToPointCallBack);

} else {

robotNavActionApi.navToPointByName(markPointEntity.text, navToPointCallBack);

}

}

## 取消导航

代码示例：

private void cancelNav() {

robotNavActionApi.cancelNav(new ActionNotifyCallBack() {

@Override

public void onRecvActionNotify(ActionRequest notifyRequest,

String srcURI, String notifyAction, final int notifyCode,

final String notifyInfo, String notifyParams) {

LogUtils.d(TAG, "onRecvActionNotify srcURI=" + srcURI + ", notifyAction=" + notifyAction

+ ", notifyCode=" + notifyCode + ", notifyInfo=" + notifyInfo

+ ", notifyParams=" + notifyParams);

if (notifyCode == ActionCode.ACTION\_OK) {

//成功

mHandler.post(new Runnable() {

@Override

public void run() {

showToast("取消导航成功");

}

});

} else {

//错误，错误码：notifyCode, 错误信息：notifyInfo

mHandler.post(new Runnable() {

@Override

public void run() {

showToast("取消导航失败 " + notifyCode + ", " + notifyInfo);

}

});

}

}

});

}

## 机器人控制API

1. 初始化API实例

RobotActionCenterApi api = new RobotActionCenterApiImp();

1. 运动

void moveForward();

void turnLeft();

void turnRight();

void moveBack();

1. 停止移动

void stopWalking();

void turnRound();

1. 跳舞

void robotDance();

void robotStopDance();

1. 跟随

void robotFollow();

void stopFollow();

1. 停止全部，停止导航，唱歌跳舞等等

void stopAll();

1. 头部

void turnHeadRight();

void turnHeadLeft();

void turnHeadUp();

void turnHeadDown();

void turnHeadReset();

void headUpDown();

void headLeftRight();

1. 灯环

void lightNormal();

void lightTalking();

void lightThinking();

void lightListening();

void lightSinging();

1. 启动默认任务

void taskStartDefault(int startIndex, ActionNotifyCallBack callBack);

1. 启动任务根据名称

void taskStartByName(String taskName, int startIndex, ActionNotifyCallBack callBack);

1. 启动任务根据id

void taskStartById(String taskId, int startIndex, ActionNotifyCallBack callBack);

1. 停止任务

void taskStop();

1. 暂停任务

void taskPause();

## 导航控制API

RobotNavActionApiImp api = new RobotNavActionApiImp();

1. 开启导航

int startNav(ActionNotifyCallBack callBack);

int startNavByMapId(String mapId, ActionNotifyCallBack callBack);

1. 停止导航

int stopNav(ActionNotifyCallBack callBack);

1. 取消导航

int cancelNav(ActionNotifyCallBack callBack);

1. 导航到某个点

int navToPointByName(String pointName, ActionNotifyCallBack callBack);

1. 根据坐标导航到某个点

int navToPointByPostion(float x, float y, float z, ActionNotifyCallBack callBack);

1. 获取导航状态

int getNavState(ActionNotifyCallBack callBack);

1. 回冲

int backDock(ActionNotifyCallBack callBack);

1. 取消回冲

int cancelBackDock(ActionNotifyCallBack callBack);

## 系统Action

系统Action，包括音频，视频，语音播报，语音识别，视频呼叫等等；

SystemActionCenterApiImp api = new SystemActionCenterApiImp();

1. 播放音频

int playMusic(String url, ActionNotifyCallBack callBack);

void resumeMusic();

void pauseMusic();

void stopMusic();

1. 播放视频

int playVideo(String videoUrl, String title, boolean isMute, ActionNotifyCallBack callBack);

void resumeVideo();

void pauseVideo();

void stopVideo();

1. 语音播报

int speakStart(String words, int speakmode, ActionNotifyCallBack callBack);

void speakStop();

1. 语音识别

int speechRecognitionStart(ActionNotifyCallBack callBack);

void speechRecognitionStop();

1. 图片轮播

int showPicsStart(List<String> imageList);

void showPicsStop();

1. 视频聊天

/\*\*

\* 呼叫好友

\* @param acountId

\* @param name

\* @param type

\*/

void callTo(String acountId, String name, int type);

/\*\*

\* 呼叫好友

\* @param params

\*/

void callTo(SystemCallParams params);

/\*\*

\* 跳转到好友列表

\*/

void jumpFriendList();

## 打开/关闭唤醒的方法(只适用A1 A2 2.0版本)

艾米桌面切换为对话桌面或者笑脸桌面时(艾米ASR开启)，麦克风被艾米语音桌面占用，第三方APP需要录音时，需要调用麦克风唤醒关闭的方法。

打开:

Settings.System.*putInt*(getContext().getContentResolver(), **"is\_enble\_microphone"**,1);

关闭:

Settings.System.*putInt*(getContext().getContentResolver(), **"is\_enble\_microphone"**,0);

注意：A. 调用关闭方法后，最好延迟2s再调用录音功能，以便唤醒功能关闭完成

艾米桌面切换为APP应用列表桌面时(艾米ASR关闭)，麦克风未被占用，第三方APP无需处理开关。

## 监听唤醒broadcast

对机器人讲：“你好，艾米”时，系统会发送com.amyrobotics.wakeup广播，第三方app监听此广播即可。

艾米桌面切换为对话桌面或者笑脸桌面(艾米ASR开启)时，艾米桌面收到唤醒广播时会接管语音识别。如果第三方APP在前台，需要监听唤醒broadcast，重新调用接管艾米语音ASR的方法，并调用abortBroadcast() 中断此广播。同时设置APP android:priority 大于101。

<receiver

android:name=".receiver.WakeUpReceiver"

android:exported="true">

<intent-filter android:priority="102"> //设置APP android:priority 大于101。

<action android:name="com.amyrobotics.wakeup" />

</intent-filter>

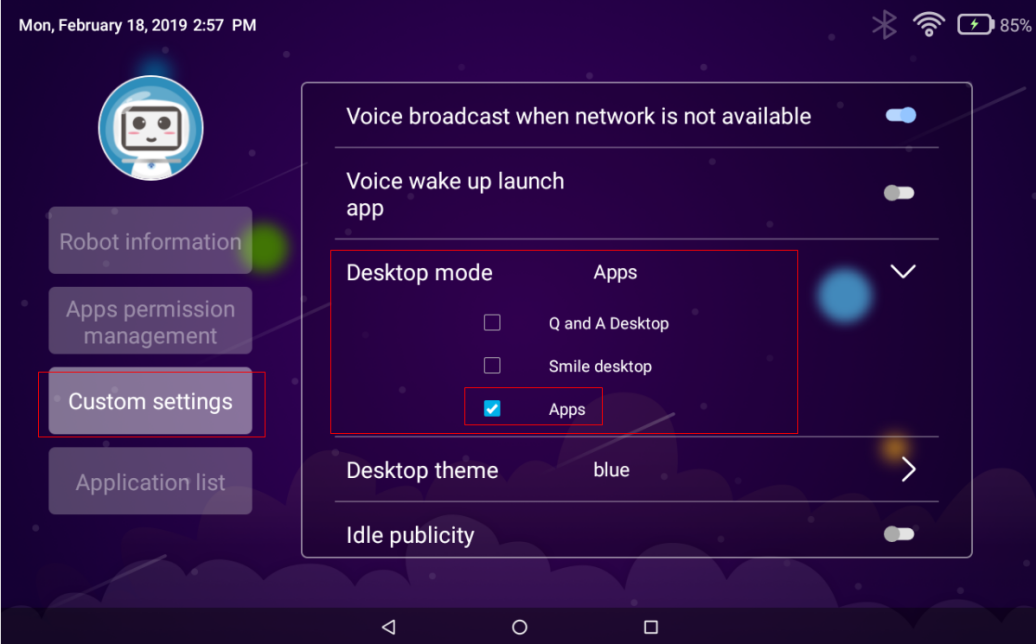
</receiver>

## 第三方开发语音app时注意事项

第三方开发语音app有两种方式：

1 艾米桌面切换为对话桌面或者笑脸桌面(艾米ASR开启)。利用艾米语音桌面的ASR功能，接管其ASR返回的文字。拿到文字后自行处理。APP需要监听唤醒broadcast，参考第15条，此时需要重新调用接管艾米语音ASR的方法。

2 艾米桌面切换为APP应用列表桌面(艾米ASR关闭)。接入自己的ASR平台处理语音识别，拿到文字后自行处理。APP需要自行监听唤醒broadcast，参考第15条。



## 1.0系统版本(已停止维护)启用/禁用系统语音的方法

依次执行下方命令，即可禁用amy voice

adb root

adb remount

adb shell

cd /system/app/VoiceProvider

mv VoiceProvider.apk VoiceProvider

cd /system/app/AmyVoiceZh

mv AmyVoiceZh.apk AmyVoiceZh

reboot

依次执行下方命令，即可启用amy voice

adb root

adb remount

adb shell

cd /system/app/VoiceProvider

mv VoiceProvider VoiceProvider.apk

cd /system/app/AmyVoiceZh

mv AmyVoiceZh AmyVoiceZh.apk

reboot

## 电量广播及获得电量的方法

远程控制中的“自动回充”设置打开时，低电时会强制中断任务，去执行充电任务。

开发者如果不希望低电回冲打断自己APP的任务，可以先将远程控制中的“自动回充”设置关闭，开发APP时自行处理低电行为即可。

1. 充电完成时，会发送如下广播**private static final** String ***CHARGE\_COMPLETE\_CAST*** = **"com.amyrobotics.battery.chargeComplete"**;
2. 低电量时，会发送如下广播(当电量小于等于15%则判定为低电量)  
   **private static final** String ***CHARGE\_LOW\_CAST*** = **"com.amyrobotics.battery.chargeLower"**;
3. 电量广播，包括电量值与充电状态  
   **private static final** String ***CHARGE\_CHANGE*** = **"com.amyrobotics.batteryChange"**;

从intent中读出电量值与充电状态参考代码

**int** level = intent.getIntExtra(**"level"**, 0);  
**int** charge = intent.getIntExtra(**"charging"**, 0);

1. 获得电量及充电状态api如下

**public static final** String ***APP\_SRC\_URI*** = **"content://com.amyrobotics.amyservice.provider"**;

*/\*\*  
 \* 获得电量Action，任何app调用此uri，可获得机器人当前电量。  
 \*/***public static final** String ***ACTION\_NAME\_GET\_POWER\_LEVEL*** = **"com.amyrobotics.amyservice.get.power.level"**;  
*/\*\*  
 \* 获得充电状态Action，任何app调用此uri，可获得机器人当前是否在充电。  
 \*/***public static final** String ***ACTION\_NAME\_GET\_CHARGE\_STATE*** = **"com.amyrobotics.amyservice.get.charge.state"**;

ActionResult actionResult = ActionCenterManager.*getInstance*().executeAction(APP\_SRC\_URI,  
 ***ACTION\_NAME\_GET\_POWER\_LEVEL***, 0, **""**, **null**);  
LogUtils.*d*(***TAG***, **"current power level: "** + actionResult.mVal);

ActionResult actionResult = ActionCenterManager.*getInstance*().executeAction(APP\_SRC\_URI,  
 ***ACTION\_NAME\_GET\_CHARGE\_STATE***, 0, **""**, **null**);  
LogUtils.*d*(***TAG***, **"current charge state: "** + actionResult.mVal);

//1: charging