RobotNavigationSDK集成说明

1 环境说明

- 操作系统：Android 8.1
- 开发软件：Android Studio 4.2+
- JDK： 1.8
- SDK版本：编译和目标SDK版本为28

2 RobotNavigationSDK功能说明

RobotNavigationSDK提供的功能是与导航功能的对接，通过调用SDK提供的接口，实现对机器人导航功能的控制。SDK提供的接口主要包括：导航、巡航、定位、切换地图等功能。地图的创建和编辑功能是在机器人导航组件中完成的。

注意，RobotNavigationSDK只能对具有导航功能的机器人使用。

3 RobotNavigationSDK集成步骤

3.1 安装或更新串口、导航组件

打开“系统设置”，然后点击“更新应用及组件”选项，在打开的页面中，检查“串口组件”和“导航组件”是否是最新版本。如果“串口组件”和“导航组件”有更新，则需要将其更新到最新版本。

3.2 获取jar包

在集成jar包之前，首先需要获取到RobotNavigationSDK-*.jar。jar包可以在提供的压缩包的libs目录获取。

3.3 集成jar包

将RobotNavigationSDK-*.jar放到项目的libs目录下，然后在项目的build.gradle文件中添加配置，如下：

```
dependencies {
  //...
  api files('libs/RobotNavigationSDK-1.3.0.jar')
}
```

在调用SDK接口的时候，很多数据的返回都是异步的，在RobotNavigationSDK中是借助EventBus返回机器人的导航信息，所以还需要在项目中集成EventBus等依赖。需要在项目的build.gradle文件中添加的依赖如下：

```
dependencies {
  //...
  api 'org.greenrobot:eventbus:3.1.1'
  api 'com.google.code.gson:gson:2.9.0'
}
```

3.4 连接导航

在调用导航相关的接口之前，需要使用PadBotNavigationClient实例来连接导航，如下：

```
//连接导航
PadBotNavigationClient.getInstance().connect(getApplicationContext());
```

3.5 初始化导航

导航连接成功以后，会通过EventBus抛出OnNavigationClientConnectedEvent事件，在监听OnNavigationClientConnectedEvent事件的方法中，初始化导航。

```
/**
 * Receive navigation connection success event
 * 接收导航连接成功事件
 * @param event Navigation connection success event
 *              导航连接成功事件
 */
@Subscribe(threadMode = ThreadMode.MAIN)
public void onEvent(OnNavigationClientConnectedEvent event) {
    try {
        //初始化导航
        PadBotNavigationClient.getInstance().initNavigation();

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

3.6 新建地图和添加目标点

在控制机器人导航之前，需要先新建地图，并在地图上添加目标点。创建地图和添加目标点的操作，都是在导航组件中完成的。可以通过如下代码打开导航组件：

```
//打开导航组件
PadBotNavigationClient.getInstance().startNavigationApp(Activity);
```

3.7 控制机器人导航

在控制机器人前往指定的目标点之前，需要先获取到地图信息，在地图信息中获取到目标点的信息，然后调用导航接口前往目标点。

1、首先获取到地图的列表

```
IndoorMapVoListResult indoorMapVoListResult =
PadBotNavigationClient.getInstance().getMapInfo();
if (null != indoorMapVoListResult) {
    //获取地图列表
    List<IndoorMapVo> indoorMapVoList = indoorMapVoListResult.getMapVoList();
}
```

2、通过地图列表，获取机器人当前的地图

```
IndoorMapVo currentIndoorMap = null;
//遍历地图列表，找到当前使用的地图
for (IndoorMapVo indoorMapVo : indoorMapVoList) {
    if (indoorMapVo.isDefaultUse()) {
        currentIndoorMap = indoorMapVo;
        break;
    }
}
```

3、在地图中获取目标点列表的信息

```
//获取目标点列表
List<RobotTargetPointVo> targetPointVoList = currentIndoorMap.getTargetPointVoList();
```

4、获取某个目标点，调用导航接口前往目标点

```
RobotTargetPointVo targetPointVo = targetPointVoList.get(0);
//导航去目标点
PadBotNavigationClient.getInstance().startNavigateByPointId(targetPointVo.getId());
```

3.8 接收机器人返回的信息

1、首先需要注册EventBus，这样才能接收到EventBus抛出的事件，如下：

```
//注册EventBus
EventBusUtils.register(this);
```

2、然后在方法上添加EventBus注解，就可以接收到指定的事件对象，例如：

```
/**
 * 接收到导航信息事件
 * @param event 导航信息事件对象
 */
@Subscribe(threadMode = ThreadMode.MAIN)
public void onEvent(OnNavigateInfoChangedEvent event) {
    try {
        NavigateInfo navigateInfo = event.getNavigateInfo();
        NavigateStatus navigateStatus = navigateInfo.getNavigateStatus();
        RobotTargetPointVo targetPointVo = navigateInfo.getTargetPoint();
        NavigateStepType stepType = navigateInfo.getNavigateStepType();
        ActionStatus actionStatus =
PadBotNavigationClient.getInstance().getActionStatus();

        //如果当前的活动是导航去目标点
        if (actionStatus == ActionStatus.NAVIGATING) {
            //开始导航去新的目标点
            if (NavigateStatus.NEW_TARGET == navigateStatus) {
                if (null != targetPointVo) {
                    showToast("Navigate to:" + targetPointVo.getName());
                }
            }
            //如果到达最终目标点
            else if (NavigateStatus.FINISHED == navigateStatus) {
                if (null != targetPointVo) {
                    showToast("Arrived:" + targetPointVo.getName());
                }
            }
         }
    } catch (RemoteException e) {
        e.printStackTrace();
```

```
    }
}
```