Autel SDK 开发环境集成

1. gradle引入SDK资源包(目前只支持Android Studio开发环境，暂不支持Eclipse)

在app工程下的gradle文件中配置路径：

```
api(name: "autel-sdk-release_V2.0.2", ext: "aar") {
        exclude module: 'okio'
        exclude module: 'okhttp'
    }
implementation 'com.tencent.mars:mars-xlog:1.2.5'
```

示例如图

```
67
68  dependencies {
69      implementation fileTree(include: ['*.jar'], dir: 'libs')
70      androidTestImplementation('com.android.support.test.espresso:espresso-core:2.2.2', {
71          exclude group: 'com.android.support', module: 'support-annotations'
72      })
73      implementation 'androidx.appcompat:appcompat:1.0.0-alpha1'
74
75      implementation 'androidx.appcompat:appcompat:1.2.0'
76      implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
77
78      implementation 'com.google.android.gms:play-services-maps:10.2.0'
79      implementation 'androidx.multidex:multidex:2.0.1'
80      implementation 'io.reactivex.rxjava2:rxjava:2.1.2'
81      implementation 'io.reactivex.rxjava2:rxandroid:2.0.1'
82      testImplementation 'junit:junit:4.12'
83      implementation('com.squareup.okhttp3:okhttp:3.8.1')
84      api(name: "autel-sdk-release_V2.0.2", ext: "aar") {
85          exclude module: 'okio'
86          exclude module: 'okhttp'
87      }
88      implementation 'com.tencent.mars:mars-xlog:1.2.5'
89  }
90
```

引入仓库后，在开发模块的gradle文件中建立依赖

2. 初始化SDK

在需要使用SDK功能前，调用init函数初始化相关SDK功能：

```
/**
        * 初始化SDK，通过网络验证APPKey的有效性
        */
       String appKey = "<SDK license should be input>";
       AutelSdkConfig config = new AutelSdkConfig.AutelSdkConfigBuilder()
```

```
                .setAppKey(appKey)
                .setPostOnUi(true)
                .create();
        AutelConfigManager.instance().init(this);
        AutelBaseApplication.setAppContext(this);

        Autel.init(this, config, new CallbackWithNoParam() {
            @Override
            public void onSuccess() {
                Log.v(TAG, "checkAppKeyValidate onSuccess");
            }

            @Override
            public void onFailure(AutelError error) {
                Log.v(TAG, "checkAppKeyValidate " + error.getDescription());
            }
        });
        NetWorkProxyJni.setType(0);//使用基站连接时设置0，使用图传直连时设置为1
        com.autel.log.AutelLog.init(BuildConfig.DEBUG,
AutelDirPathUtils.getLogCatPath(),
                AutelDirPathUtils.getLogCatPath(), 5);//日志存储 采用xlog
```

Autel.init()函数的第一个参数是会被**长期持有**的Context对象，为避免**内存泄露**，建议使用Application的Context对象

例如在自定义Application中初始化SDK服务：

```
public class TestApplication extends Application {
    private final String TAG = getClass().getSimpleName();
    private BaseProduct currentProduct;

    public void onCreate() {
        super.onCreate();
        Thread.setDefaultUncaughtExceptionHandler(new EHandle(Thread.getDefaultUncaughtExceptionHandler()));
        /**
         * 初始化SDK，通过网络验证APPKey的有效性
         */
        String appKey = "<SDK license should be input>";
        AutelSdkConfig config = new AutelSdkConfig.AutelSdkConfigBuilder()
                .setAppKey(appKey)
                .setPostOnUi(true)
                .create();
        Autel.init(this, config, new CallbackWithNoParam() {
            @Override
            public void onSuccess() { Log.v(TAG, "checkAppKeyValidate onSuccess"); }

            @Override
            public void onFailure(AutelError error) {
                Log.v(TAG, "checkAppKeyValidate " + error.getDescription());
            }
        });
    }
```

NOTE: 在整个应用结束Autel SDK服务的使用后（一般在退出APP时），调用 Autel.destroy()来结束Autel SDK相关资源的调用

NOTE: appKey 为开发者在Autel开发者平台申请的应用关联钥匙

3.SDK 功能接口调用

SDK提供以下模块的功能服务：Album（相册）、Battery（电池）、Camera（相机）、DSP（图传）、FlyController（飞行控制器）、Gimbal（云台）、Mission（任务）、RemoteController（遥控器）、Codec(视频解码)

用户通过产品连接的监听回调，获取到与APP连接的飞行器的产品类对象BaseProduct，进而获取相关模块的接口：

```
Autel.setProductConnectListener(new ProductConnectListener() {
          @Override
          public void productConnected(BaseProduct product) {

          }

          @Override
          public void productDisconnected() {

          }
      });
```

通过向下转型成具体的产品对象，从而获取相关产品的模块接口，例如

```
  BaseProduct product;

  ...

  switch (product.getType()) {
          case DRAGONFISH:
          case DRAGONFISH_7_5_VTOL:
          case DRAGONFISH_15_VTOL:
              CruiserBattery battery = (CruiserBattery) product.getBattery();
              break;
      }
```

相机模块

目前可用相机类型为R12、XB015，获取相机服务的接口时，需要通过CameraManager监听相机状态，当相机连接成功时会返回当前的相机类型，但是需要向下手动转型

```
BaseProduct product;

...

autelCameraManager = product.getCameraManager();
```

```
  autelCameraManager.setCameraChangeListener(new CallbackWithTwoParams<CameraProduct,
AutelBaseCamera>() {
            @Override
            public void onSuccess(final CameraProduct data1, final AutelBaseCamera
data2) {
                switch (data1) {
                    switch (data1) {
                    case XT708:
                    case XT710:
                    case XT711:
                    case XT713:
                    case XT714:
                    case XT715:
                    case XT717:
                        changePage(CameraDFFragment.class);
                        break;
                    default:

                }

            }

            @Override
            public void onFailure(AutelError error) {

            }
        });
```

SDK调用函数参数范围的查询方式，示例如下

```
  AutelBaseCamera baseCamera;

    ...


  AutelMultiCamera autelMultiCamera = (AutelMultiCamera)baseCamera;
  MultiParameterRangeManager rangeManager =
autelMultiCamera.getParameterRangeManager();
  PhotoAspectRatio[] arRange = rangeManager.getPhotoAspectRatio();
```

当前支持的查询参数有：VideoResolutionAndFps、AspectRatio、WhiteBalanceType、ISO、ExposureMode等；

注意

VideoResolutionAndFps具体的参数范围依赖于相机当前的视频标准VideoStandard，调用示例如下

```
  AutelMultiCamera autelMultiCamera;
  ...
```

```
  autelMultiCamera.setInfoListener(new CallbackWithOneParam<CameraInfo>() {
                    @Override
                    public void onSuccess(CameraInfo state) {
                        logOut("setInfoListener  :" + state);
                    }

                    @Override
                    public void onFailure(AutelError error) {
                        logOut("setInfoListener  description  " +
error.getDescription());
                    }
                });
  ...

MultiParameterRangeManager rangeManager = autelMultiCamera.getParameterRangeManager();
```

视频解码模块

视频解码提供了两种获取解码数据方式，一种是提供自定义View直接显示飞行器相机实时传送的数据，只需要引用com.autel.sdk.widget.AutelCodecView即可；

```
  //xml实例化
<RelativeLayout
        android:id="@+id/content_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/white"
        android:visibility="visible">

    </RelativeLayout>
    private AutelPlayerView codecView;
    private AutelPlayer mAutelPlayer;

    private AutelPlayerView createAutelCodecView() {
        AutelPlayerView codecView = new AutelPlayerView(this);
        FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
        );
        codecView.setLayoutParams(params);
        return codecView;
    }
    private void addVideo(){
            AutelPlayerManager.getInstance().init(CodecActivity.this, false);
            codecView = createAutelCodecView();
            content_layout.addView(codecView);
            mAutelPlayer = new AutelPlayer(0);
            mAutelPlayer.addVideoView(codecView);
            AutelPlayerManager.getInstance().addAutelPlayer(mAutelPlayer);
```

```
            mAutelPlayer.startPlayer();
    }

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addVideo();
    }
    public void onDestroy() {
        super.onDestroy();
        stopPlayer();
    }

    private void stopPlayer(){
        AutelLog.debug_i("initCodec","stopPlayer  codecBase ");
        if(null != mAutelPlayer){
            mAutelPlayer.removeVideoView();
            AutelPlayerManager.getInstance().removeAutelPlayer(mAutelPlayer);
            mAutelPlayer.stopPlayer();
            mAutelPlayer.releasePlayer();

        }
    }
```

或者使用视频解码服务提供的数据监听接口获取数据

```
  BaseProduct product;
  ...
  AutelCodec codec = product.getCodec();
  codec.setCodecListener(new AutelCodecListener() {
            @Override
            public void onFrameStream(final boolean valid, byte[] videoBuffer, final
int size, final long pts) {
            }

            @Override
            public void onCanceled() {
            }

            @Override
            public void onFailure(final AutelError error) {
            }
        }, null);
```

注意

AutelCodecView提供了两个功能接口：pause()、resume()

AutelCodec提供了AutelPlayerManager.getInstance().addCodecListeners();设置监听的同时开启视频解码，实时返回视频解码信息

```
AutelPlayerManager.getInstance().addCodecListeners(TAG, 0, new
OnRenderFrameInfoListener() {
    @Override
    public void onRenderFrameTimestamp(long l) {

    }
    /**
     * Invokes this method when the render view size is updated.
     *
     * @param width  the width of the render view.
     * @param height the height of the render view.
     */
    @Override
    public void onRenderFrameSizeChanged(int i, int i1) {

    }
    /**
     * The H264 video stream data.
     *
     * @param videoBuffer the video buffer data.
     * @param isIFrame    true if I Frame, otherwise false.
     * @param size        the data size.
     * @param pts         the present leftTime stamp.
     */
    @Override
    public void onFrameStream(byte[] bytes, boolean b, int i, long l, int i1) {

    }
    /**
     * The YUV video stream data.
     *
     * @param videoBuffer the video buffer data.
     * @param isIFrame    true if I Frame, otherwise false.
     * @param mInfo
     * @param isIFrame
     * @param width
     * @param height
     * @param formatType
     */
    @Override
    public void onFrameStream(ByteBuffer byteBuffer, MediaCodec.BufferInfo bufferInfo,
boolean b, int i, int i1, int i2) {

    }
});
```

任务模块

任务模块目前支持三种任务：WaypointMission(航点任务、多边形、矩形任务）所有任务由任务管理器执行相关操作

任务管理器的具体操作有：prepare(准备)、start(开始)

航点任务为例，调用航线算法库生成绘制航线相关操作代码如下：

```
  ...
            PathPlanningResult result = NativeHelper.getWaypointMissionPath(drone,
homePoint, upHomePoint, downHomePoint, waypointParams);
            int errorCode = result.getErrorCode();//是否规划任务成功，0-成功，1-失败
            double flyLength = result.getFlyLength();//航线总距离
            double flyTime = result.getFlyTime();//预计飞行总时间
            double pictNum = result.getPictNum();//预计拍照数量
            double optCourseAngle = result.getOptCourseAngle();//自动规划主航线角度时使用的
主航线角度
            List<AutelCoordinate3D> latLngList = result.getLatLngList();//整条航线所有点的
纬经高
            List<DirectionLatLng> directionLatLngList =
result.getDirectionLatLngList();//航线中箭头的纬经度
            List<DistanceModel> distanceModelList = result.getDistanceModelList();//航线
中两个航点的距离的显示位置的纬度、经度、距离
            List<AutelCoordinate3D> plusList = result.getPlusList();//两个航点间加号的纬
度、经度

            AutelLog.debug_i("NativeHelper:", "flyTime = " + flyTime
                    + ", flyLength = " + flyLength + ", picNum = " + pictNum
```

```
                    + ",errorCode = " + errorCode);
            Toast.makeText(this, "testWaypoint result -> " + errorCode,
Toast.LENGTH_SHORT).show();
```

生成任务文件实例：

```
 File myDir = new File(FileUtils.getMissionFilePath());
 ....
            //返回0表示成功，返回非0表示失败
            int res = NativeHelper.writeMissionFile(filePath, missionType,
                    droneLocation, homeLocation,
                    launchLocation, landingLocation,
                    avoidPosition, UAVTurnRad,
                    UAVFlyVel, UserFPKIsDef,
                    UserFlyPathA, WidthSid,
                    OverlapSid, WidthHead,
                    OverlapHead, UAVFlyAlt,
                    waypointLen, waypointParamList,
                    poiPointLen, poiParamList, linkPoints, isEnableTopographyFollow ? 1
: 0);
            AutelLog.d("NativeHelper", " writeMissionFile result -> " + res);
            Toast.makeText(this, "writeMissionFile result -> " + res,
Toast.LENGTH_SHORT).show();
```

使用MissionManager来准备环绕任务mOrbitMission

```
  BaseProduct product;
  ...
  MissionManager myMissonManager = ((XStarAircraft) product).getMissionManager();
  myMissonManager.prepareMission(mOrbitMission, new
CallbackWithOneParamProgress<Boolean>() {...});
```

开始任务自检

```
  public void autoCheck(final ModelType modelType) {

        if (null != mEvoFlyController) {
            new IOUiRunnable<Boolean>() {
                @Override
                protected Observable<Boolean> generateObservable() {
                    return mEvoFlyController.toRx().autoSafeCheck(modelType);
                }

                @Override
                public void onNext(@NonNull final Boolean success) {
                    super.onNext(success);
```

```
                }

                @Override
                public void onError(@NonNull Throwable e) {
                    super.onError(e);

                }
            }.execute();
        }
    }
```

查询任务自检结果

```
    private void getAutoCheckResult(ModelType modelType) {
          if (null != mEvoFlyController) {
              new IOUiRunnable<AutoSafeState>() {
                  int retryCount = 0;

                  @Override
                  protected Observable<AutoSafeState> generateObservable() {
                      return mEvoFlyController.toRx().getAutoSafeCheck(modelType);
                  }

                  @Override
                  public void onNext(@NonNull final AutoSafeState safeState) {
                      super.onNext(safeState);
                      .....

                  }

                  @Override
                  public void onError(@NonNull Throwable e) {
                      super.onError(e);
                      retryCount++;
                      if (retryCount < 3) {
                          getAutoCheckResult(modelType);
                      } else {

                      }
                  }
              }.execute();
          }
      }
```

- **任务模块相关注意事项**

1. 由于任务需要采集GPS信息，GPS模块(手机或者飞行器)采集的数据和地图工具(Google地图、高德地图)输出的数据相比较,针对**中国大陆地区**的GPS信息可能存在**坐标系偏差**，如果不做地图纠偏处理，在执行飞行任务时，**会有500米左右的位置偏差**

2. 任何任务都需要任务管理器**准备完成**后，才能有效执行，下面是创建任务实例

```
autelMission = new CruiserWaypointMission();
autelMission.missionId =  BytesUtils.getInt(UUID.randomUUID().toString().replace("-",
"").getBytes()); //任务id
autelMission.missionType = MissionType.Waypoint; //任务类型(Waypoint(航点)、RECTANGLE(矩
形)、POLYGON(多边形))
autelMission.finishedAction = CruiserWaypointFinishedAction.RETURN_HOME;
if (null != missionManager) {
            autelMission.localMissionFilePath = filePath;
            missionManager.prepareMission(autelMission, new
CallbackWithOneParamProgress<Boolean>() {
                @Override
                public void onProgress(float v) {
                    AutelLog.d(TAG, " prepareMission onProgress " + v);
                }

                @Override
                public void onSuccess(Boolean aBoolean) {
                    flyState = FlyState.Prepare;
                    AutelLog.d("prepareMission success");
                    Toast.makeText(DFWayPointActivity.this, "prepare success",
Toast.LENGTH_LONG).show();



                }

                @Override
                public void onFailure(AutelError autelError) {
                    AutelLog.d("prepareMission onFailure");
                    Toast.makeText(DFWayPointActivity.this, "prepare failed",
Toast.LENGTH_LONG).show();

                }
            });
        }
```

3. 飞行器只有执行**航点任务**时会反馈任务实时信息，针对实时信息监听接口，获取的数据对象需要手动**向下转型**

```
   missionManager.setRealTimeInfoListener(new CallbackWithOneParam<RealTimeInfo>() {
                @Override
                public void onSuccess(RealTimeInfo realTimeInfo) {
                    CruiserWaypointRealTimeInfoImpl info =
(CruiserWaypointRealTimeInfoImpl) realTimeInfo;
                    AutelLog.d("MissionRunning", "timeStamp:" + info.timeStamp +
",speed:" + info.speed + ",isArrived:" + info.isArrived +
                            ",isDirecting:" + info.isDirecting + ",waypointSequence:" +
info.waypointSequence + ",actionSequence:" + info.actionSequence +
```

```
                            ",photoCount:" + info.photoCount + ",MissionExecuteState:"
+ info.executeState + ",missionID:" + info.missionID);
                }

                @Override
                public void onFailure(AutelError autelError) {

                }
            });
```

相册模块

相册模块提供了相机媒体资源文件的元信息管理，信息包括：原图地址，小缩略图地址，大缩略图地址，创建时间，文件大小；

```
public interface MediaInfo {

    long getFileSize();

    String getFileTimeString();

    String getSmallThumbnail();

    String getLargeThumbnail();

    String getOriginalMedia();
}
```

注意

getVideoResolutionFromLocalFile和getVideoResolutionFromHttpHeader用于获取相机中的视频资源的分辨率大小，并不能适用于图片资源；
getVideoResolutionFromLocalFile输入的文件对象为从相机中下载的视频资源文件；

```
  BaseProduct product;
  ...
  AutelAlbum album = product.getAlbum();
```

结束