

瞳步底盘导航 SDK 使用说明

说明：调用的任何方法请在子线程中使用，避免受网络环境影响造成请求底盘超时引起的 ANR 异常；建图必须在充电桩上开机并开始建图，否则将不能正常回去充电；保存的地图文件名字格式为.stcm 格式文件。

SlamManager 类

`static SlamManager getInstance()`

获取 SlamManager 类的实例。

`int connect()`

连接底盘，使用默认的地址跟端口号连接，推荐使用这种方法连接，结果状态码请对照 SlamCode 类，连接成功时返回 0。

`int connect(String speed)`

speed: 速度值

连接底盘

`int connect(String ip, int port)`

ip: ip 地址

port: 端口号

连接底盘

`int connect(String ip, int port, String speed)`

ip: ip 地址

port: 端口号

speed: 速度值

连接底盘

`boolean isConnected()`

是否连接到底盘

`void disconnect()`

断开底盘连接，请在应用退出后调用

`void setOnSlamExceptionListener(OnSlamExceptionListener listener)`

listener: 回调

slam 异常监听

`void setOnActionListener(OnActionListener listener)`

listener: 回调

slam 动作监听

`String getDeviceId()`

获取设备 ID

String getSlamVersion()

获取 slam 的底盘版本

String getSDKVersion()

获取 slam 的固件版本

NetBean getNet()

获取 slam 的连接信息

void requestNetAsync(OnResultListener<NetBean> listener)

listener: 回调

获取 slam 的连接信息（异步方法）

boolean setSpeed(String speed)

speed: 速度值，范围（0-0.7]

设置导航速度

void setSpeedAsync(String speed, OnResultListener<Boolean> listener)

speed: 速度值，范围（0-0.7]

listener: 回调

设置导航速度（异步方法）

String getSpeed()

获取当前导航速度

void requestSpeedAsync(OnResultListener<String> listener)

listener: 回调

获取当前导航速度（异步方法）

void setRotateSpeed(String angularSpeedValue)

angularSpeedValue: 角速度值，范围[0.05-2.0]

设置旋转的角速度

**void setRotateSpeedAsync(String angularSpeedValue,
OnResultListener<Boolean> listener)**

angularSpeedValue: 角速度值，范围[0.05-2.0]

listener: 回调

设置旋转的角速度（异步方法）

String getRotateSpeed()

获取旋转的角速度

void requestRotateSpeedAsync(OnResultListener<String> listener)

listener: 回调

获取旋转的角速度（异步方法）

boolean setPose(Pose pose)

pose: 姿态

设置机器人当前的姿态

Pose getPose()

获取当前机器人的姿态

boolean setHomePose(Pose pose)

pose: 位置

重新设置充电桩位置

void setHomePoseAsync(float chassisRadius, OnResultListener<Boolean> listener)

chassisRadius: 底盘半径，单位：米，以激光雷达为中心算底盘半径

listener: 回调

重新设置充电桩位置（异步方法），默认开机位置即充电桩位置，无需设置，只有充电桩位置发生变化时，需要重置时调用该方法

Pose getHomePose()

获取充电桩位置，如果找不到充电桩则返回 null（开机没有在充电桩充着电开机，再去对桩充电，则获取不到值）

LaserScan getLaserScan()

获取机器人扫描的地图区域

boolean isDirectCharge()

是否在直充

boolean isBatteryCharging()

机器人是否在充电

boolean isDockingStatus()

机器人是否在充电桩上（如果在充电桩上但是没有电也是返回不在充电桩上的状态）

boolean isHasPiles()

地图上是否有充电桩（现实环境有无充电桩无法判断）

int getBatteryPercentage()

获取电量（总共 100， 0<电量<=100）

IMoveAction getCurrentAction()

获取当前的动作，获取当前动作状态值需要此 action

ActionStatus getStatus()

获取当前的运动状态

boolean isSystemEmergencyStop()

是否按下急停按钮

boolean isSystemBrakeStop()

是否按下刹车按钮

boolean isSystemStop()

是否按下急停、刹车按钮

SleepMode getSleepMode()

获取激光头的状态

int getLocalizationQuality()

获取定位质量

boolean isControlBusDisconnect()

是否通信中断

HealthInfo getRobotHealthInfo()

获取机器健康信息

`void clearRobotHealthInfoAsync(OnResultListener<Boolean> listener)`

listener: 回调

清除健康信息（异步方法）

`boolean isPathPlanSuccess()`

是否路径规划成功（只有路径规划成功后才可以导航）

`Map getMap()`

获取地图

`IMoveAction getRemainingAction()`

获取当前剩余的动作

`ActionStatus getRemainingActionStatus()`

获取当前剩余的动作状态

`Path getRemainingMilestones()`

获取剩余里程

`Path getRemainingPath()`

获取剩余路径

```
void setMapUpdateAsync(boolean isUpdate, OnResultListener<Boolean> listener)
```

isUpdate: 是否更新

listener: 回调结果

设置地图更新（异步方法）

```
boolean isMapUpdate()
```

地图是否在更新

```
IMoveAction recoverLocationByDefault()
```

重定位，重定位的时候默认自动转圈（需要自己监听机器人的定位状态）

```
IMoveAction recoverLocationByDefault(boolean isRotate)
```

isRotate: 重定位的时候是否旋转

listener: 回调结果

重定位

```
void recoverLocationByDefault(boolean isRotate, OnResultListener<Boolean> listener)
```

isRotate: 重定位的时候是否旋转

listener: 回调结果

默认重定位（异步方法），重定位的时候默认自动转圈（在充电则不旋转），如果在充电桩则默认以局部方式重定位

```
void recoverLocationByCustom(RectF area, boolean isRotate)
```

area: 重定位区域

isRotate: 重定位的时候是否旋转

自定义重定位，重定位的时候默认自动转圈（会返回重定位结果）

```
void recoverLocationByCustom(RectF area, boolean isRotate,  
OnResultListener<Boolean> listener)
```

area: 重定位区域

isRotate: 重定位的时候是否旋转

listener: 回调结果

自定义重定位（异步方法）（会返回重定位结果）

```
void moveTo(Location location, float yaw, OnNavigateListener listener)
```

location: 位置点

yaw: 位置点的角度

listener: 回调结果

机器人导航到某个地方（异步方法）（默认尝试时间为 0，遇障绕行，障碍物提醒时间为 0）

```
void moveTo(Location location, float yaw, long tryTimeMillis,  
OnNavigateListener listener)
```

location: 位置点

yaw: 位置点的角度

tryTimeMillis: 尝试时间（单位：毫秒）

listener: 回调结果

机器人导航到某个地方（异步方法）（默认遇障绕行，障碍物提醒时间为 0）

```
void moveTo(Location location, MoveOption option, float yaw, long  
tryTimeMillis, boolean isObstacleTriggerStop, OnNavigateListener  
listener)
```

location: 位置点

option: 导航参数

yaw: 位置点的角度

tryTimeMillis: 尝试时间（单位：毫秒）

isObstacleTriggerStop: 是否遇障暂停

listener: 回调结果

机器人导航到某个地方（异步方法）（默认障碍物提醒时间为 0）

```
void moveTo(Location location, MoveOption option, float yaw, long  
tryTimeMillis, boolean isObstacleTriggerStop, long  
remindIntervalTimeMillis, OnNavigateListener listener)
```

location: 位置点

option: 导航参数

yaw: 位置点的角度

tryTimeMillis: 尝试时间（单位：毫秒）（当遇障绕行时，为 0 则默认尝试 3 次；当遇障暂停时，为 0 则一直等待直到障碍物移除。当尝试时间到后则返回失败）

isObstacleTriggerStop: 是否遇障暂停（遇障暂停时直到障碍物移除后再继续行走）

remindIntervalTimeMillis: 提醒间隔时间（单位：毫秒）（为 0 则默认每隔 10s 回调一次）

listener: 回调结果

机器人导航到某个地方（异步方法）（自动避障）

void moveTo(LocationBean bean, OnNavigateListener listener)

bean: 位置点

listener: 回调结果

机器人导航到某个地方（异步方法）（默认尝试时间为 0，遇障绕行，障碍物提醒时间为 0）

void moveTo(LocationBean bean, MoveOption option, long tryTimeMillis, OnNavigateListener listener)

bean: 位置点

option: 导航参数

tryTimeMillis: 尝试时间（单位：毫秒）

listener: 回调结果

机器人导航到某个地方（异步方法）（默认遇障绕行，障碍物提醒时间为 0）

```
void moveTo(LocationBean bean, MoveOption option, long tryTimeMillis,  
boolean isObstacleTriggerStop, OnNavigateListener listener)
```

bean: 位置点

option: 导航参数

tryTimeMillis: 尝试时间（单位：毫秒）

isObstacleTriggerStop: 是否遇障暂停

listener: 回调结果

机器人导航到某个地方（异步方法）（默认障碍物提醒时间为 0）

```
void moveTo(LocationBean bean, MoveOption option, long tryTimeMillis,  
boolean isObstacleTriggerStop, long remindIntervalTimeMillis,  
OnNavigateListener listener)
```

bean: 位置点

option: 导航参数

tryTimeMillis: 尝试时间（单位：毫秒）（当遇障绕行时，为 0 则默认尝试 3 次；当遇障暂停时，为 0 则一直等待直到障碍物移除。当尝试时间到后则返回失败）

isObstacleTriggerStop: 是否遇障暂停（遇障暂停时直到障碍物移除后再继续行走）

remindIntervalTimeMillis: 提醒间隔时间（单位：毫秒）（为 0 则默认每隔 10s 回调一次）

listener: 回调结果

机器人导航到某个地方（异步方法）（自动避障）

```
void moveToRelocate(LocationBean bean, long tryTimeMillis, long  
remindIntervalTimeMillis, OnNavigateListener listener)
```

bean: 位置点

tryTimeMillis: 尝试时间（单位：毫秒）

remindIntervalTimeMillis: 提醒间隔时间（单位：毫秒）

listener: 回调结果

机器人导航到某个地方后并主动重定位（异步方法）（自动避障，默认遇障绕行）

```
void moveToRelocate(LocationBean bean, MoveOption option, long  
tryTimeMillis, long remindIntervalTimeMillis, OnNavigateListener  
listener)
```

bean: 位置点

option: 导航参数

tryTimeMillis: 尝试时间（单位：毫秒）（为 0 则默认尝试 3 次）

remindIntervalTimeMillis: 提醒间隔时间（单位：毫秒）（为 0 则默认每隔 10s 回调一次）

listener: 回调结果

机器人导航到某个地方后并主动重定位（异步方法）（自动避障，默认遇障绕行）

```
void moveToDistance(float distance, OnResultListener<Boolean>  
listener)
```

distance: 移动距离（单位：米）

listener: 回调结果

以导航的形式向前移动指定的距离（异步方法）（自动避障，默认精确到点，到点旋转）

```
void moveToDistance(float distance, boolean isEndRotate,  
OnResultListener<Boolean> listener)
```

distance: 移动距离（单位：米）

isEndRotate: 到达后是否旋转

listener: 回调结果

以导航的形式向前移动指定的距离（异步方法）（自动避障，默认精确到点）

```
void moveToDistance(float distance, boolean isEndRotate, MoveOption  
option, OnResultListener<Boolean> listener)
```

distance: 移动距离（单位：米）

isEndRotate: 到达后是否旋转

option: 导航参数

listener: 回调结果

以导航的形式向前移动指定的距离（异步方法）（自动避障）

```
void goHome(OnChargeListener listener)
```

listener: 回调结果

回程充电（异步方法）（自动避障）

```
void goHome(float pileDistanceX, float offsetY, OnChargeListener  
listener)
```

pileDistanceX: 距离充电桩前多少米开始检测对桩（单位：米）

offsetY: 对桩偏移值（单位：米）

listener: 回调结果

回程充电（异步方法）（自动避障，默认尝试时间为 0，遇障绕行，障碍物提醒时间为 0）

```
void goHome(float pileDistanceX, float offsetY, long tryTimeMillis,  
OnChargeListener listener)
```

pileDistanceX: 距离充电桩前多少米开始检测对桩（单位：米）

offsetY: 对桩偏移值（单位：米）

tryTimeMillis: 尝试时间（单位：毫秒）

listener: 回调结果

回程充电（异步方法）（自动避障，默认遇障绕行，障碍物提醒时间为 0）

```
void goHome(float pileDistanceX, float offsetY, long tryTimeMillis,  
boolean isObstacleTriggerStop, long remindIntervalTimeMillis,  
OnChargeListener listener)
```

pileDistanceX: 距离充电桩前多少米开始检测对桩（单位：米）

offsetY: 对桩偏移值（单位：米），当对桩偏差超过该值时，则重新对桩，为 0 则不检测偏移

tryTimeMillis: 尝试时间（单位：毫秒）（当遇障绕行时，为 0 则默认尝试 3 次；当遇障暂停时，为 0 则一直等待直到障碍物移除。当尝试时间到后则返回失

败) isObstacleTriggerStop: 是否遇障暂停 (遇障暂停时直到障碍物移除后再继续行走)

remindIntervalTimeMillis: 提醒间隔时间 (单位: 毫秒) (为 0 则默认每隔 10s 回调一次)

listener: 回调结果

回程充电 (异步方法) (自动避障)

IMoveAction goHome()

回去充电

IMoveAction moveBy(MoveDirection direction)

direction: 方向

机器人行走 (没有避障功能)

void rotateCircle(int circle, boolean isLeft, OnResultListener<Boolean> listener)

circle: 圈数

isLeft: 是否向左转

listener: 回调结果

旋转多少圈 (异步方法) (没有避障功能, 需要地图在匹配的情况下才会执行)

void rotate(int rotateAngle, boolean isLeft, OnResultListener<Boolean> listener)

rotateAngle: 旋转角度

isLeft: 是否向左转

listener: 回调结果

旋转多少度（异步方法）（没有避障功能，需要地图在匹配的情况下才会执行）

boolean rotate(int rotateAngle, MoveDirection direction)

rotateAngle: 旋转角度

Direction: 方向

机器人旋转（没有避障功能）

void rotate(int rotateAngle, MoveDirection direction, String angularSpeedValue)

rotateAngle: 旋转角度

Direction: 方向

angularSpeedValue: 角速度（0.05-2.0）

机器人旋转（没有避障功能）

IMoveAction rotateTo(Rotation rotation)

rotation: 方向

机器人旋转（没有避障功能）

void cancelAction()

取消机器人当前所有的动作行为，包括导航、行走、旋转、重定位、回去充电等

void cancelMove()

取消机器人当前行走与旋转行为

**void publishDepthCamFrame(int sensorId, DepthCameraFrame
depthCameraFrame)**

sensorId: 传感器 id

depthCameraFrame: 深度数据

将深度摄像头的数据传递给底盘

boolean addLines(ArtifactUsage artifactUsage, List<Line> lines)

artifactUsage: 线类型

lines: 线数据

添加线

List<Line> getLines(ArtifactUsage artifactUsage)

artifactUsage: 线类型

获取线

boolean clearLines(ArtifactUsage artifactUsage)

artifactUsage: 线类型

清除线

boolean removeLineById(ArtifactUsage artifactUsage, int lineId)

artifactUsage: 线类型

lines: 线 id

移除线

**void addLineAsync(ArtifactUsage artifactUsage, Line line,
OnResultListener<Boolean> listener)**

artifactUsage: 线类型

lines: 线数据

listener: 回调

添加线（异步方法）

**void clearLinesAsync(ArtifactUsage artifactUsage,
OnResultListener<Boolean> listener)**

artifactUsage: 线类型

listener: 回调

清除线（异步方法）

**removeLineByIdAsync(ArtifactUsage artifactUsage, int lineId,
OnResultListener<Boolean> listener)**

artifactUsage: 线类型

lineId: 线 id

listener: 回调

移除线（异步方法）

`List<ImpactSensorInfo> getSensors()`

获取机器人所有的传感器信息

`HashMap<Integer, ImpactSensorValue> getSensorValues()`

获取机器人传感器信息

`List<ImpactSensorValue> getSensorValues(List<Integer> id)`

id: 传感器 id

获取机器人传感器信息

`ImpactSensorValue getSensorValue(int id)`

id: 传感器 id

获取机器人传感器信息

`ICustomerLogReceiver getCustomerLogReceiver()`

获取机器运行的 Log 信息

`void clearMapAsync(OnResultListener<Boolean> listener)`

listener: 回调结果

清除地图（异步方法）

```
void saveMapAsync(String fileFolder, String fileName,  
List<LocationBean> data, OnResultListener<Boolean> listener)
```

fileFolder: 文件目录

fileName: 文件名字 (例如: 1.stcm)

data: 地图位置点

listener: 回调结果

保存地图 (异步方法)

```
void loadMapAsync(String filePath,  
OnFinishListener<List<LocationBean>> listener)
```

filePath: 文件路径

listener: 回调结果

加载地图 (异步方法) (默认以开机原点为当前点定位, 如果在充电桩则以充电点为定位点)

```
void loadMapAsync(String filePath, Pose pose,  
OnFinishListener<List<LocationBean>> listener)
```

filePath: 文件路径

pose: 机器人当前姿态

listener: 回调结果

加载地图 (异步方法), 加载完地图可随时对地图进行操作, 如果加载的地图有充电点并且在充电桩上充着电加载地图, 则默认主动进行局部重定位操作来匹配地图; 否则需自行匹配地图, 只有在地图匹配的情况下才能导航。

```
void requestAllLocationAsync(List<String> mapFilePath,  
OnResultListener<List<LocationBean>> listener)
```

mapFilePath: 文件路径集合

listener: 回调结果

请求所有位置点（异步方法）

```
void requestLocationAsync(String mapFilePath,  
OnResultListener<List<LocationBean>> listener)
```

mapFilePath: 地图文件路径

listener: 回调结果

请求位置点（异步方法）

```
void deleteFile(String filePath)
```

filePath: 文件路径

删除文件

```
boolean renameFile(String oldFilePath, String newFilePath)
```

oldFilePath: 将要重命名的文件路径

newFilePath: 新文件名称

重命名文件

```
void saveFile(String filePath, String content)
```

filePath: 文件路径

content: 内容

保存文件

String readFile(String filePath)

filePath: 文件路径

读取文件

List<String> getMapList(String fileDirectory, String fileSuffix)

fileDirectory: 文件目录

fileSuffix: 文件后缀名（例如：.stcm）

获取地图列表

void startMonitorObstacle(OnObstacleListener listener)

listener: 回调结果

检测障碍物（异步方法）

void startMonitorObstacle(float verticalMaxDistance, float horizontalMaxRadius, boolean isFilterUltrasonic, OnObstacleListener listener)

verticalMaxDistance: 以激光雷达为中心点，雷达前方的最大检测距离，单位：米

horizontalMaxRadius: 以激光雷达为中心点，雷达水平方向的最大检测半径，
单位：米

isFilterUltrasonic: 是否过滤超声波，过滤的话则不考虑超声波的触发

listener: 回调结果

自定义检测障碍物（异步方法）

void stopMonitorObstacle()

停止检测障碍物

**void upgradeControlPanelAsync(String filePath, OnUpgradeListener
listener)**

filePath: 文件路径

listener: 回调结果

升级底盘控制板固件（异步方法）

**void startSystemStopMonitor(boolean isMonitorBreak, boolean
isMonitorEmergency, long millis, OnSystemStopListener listener)**

isMonitorBreak: 是否检测刹车

isMonitorEmergency: 是否检测急停

millis: 检测时间频率

listener: 结果回调

检测系统急停和刹车触发情况（异步方法）

void stopSystemStopMonitor()

停止检测

void startMonitorSensor(long millis, OnSensorListener listener)

millis: 检测时间频率

listener: 结果回调

检测传感器（异步方法）

void stopMonitorSensor()

停止检测传感器

void setRecoverQualityMin(int value)

value: 定位质量值

设置导航过程中的最低定位质量，低于该值时会主动进行重定位（该方法对线程无要求）

int getRecoverQualityMin()

获取导航过程中的最低定位质量（该方法对线程无要求）

MoveOption 类参数介绍说明

参数名称	参数类型	描述
appending	boolean	用于决定SLAMWARE是清除当前任务建立新的点还是将新的点添加到已有的节点列表中。
milestone	boolean	用于决定SLAMWARE是规划路径到一系列节点还是直接前往。当这个参数为true时，机器人会将上述点视作关键点，通过路径搜索的方式前往目的地；当参数为false时，会被视作普通点，不会启用路径搜索功能。
noSmooth	boolean	暂时没有开放。
keyPoints	boolean	设置是否走虚拟轨道。
precise	boolean	机器人移动的时候精确到点。
withYaw	boolean	是否让机器人停下来的时候旋转。
yaw	float	机器人停下来的时候旋转到一定的角度。角度范围？
returnUnreachableDirectly	boolean	为true时，当机器人规划路径失败后，机器人不进行旋转重新规划。
trackWithOA	boolean	trackWithOA 为true时,机器人走虚拟轨道时候，也会进行避障，避障后继续优先走虚拟轨道.(如果不走虚拟轨道,trackWithOA 设置为true,没有作用)。
speedRatio	Double	机器人行走的速度,范围是0到1,(只用于moveBy) 。

OnActionListener 接口类

`void onActionReason(String actionName, String reason)`

actionName: 动作名称

reason: 原因

执行动作时回调，若失败可以查看失败原因

OnSlamExceptionListener 接口类

`void onSlamException(Exception e)`

e: 异常原因

slam 异常回调

OnFinishListener<T>泛型接口类

`void onFinish(T data)`

结果回调

`void onError()`

异常回调

OnResultListener<T>泛型接口类

`void onResult(T data)`

结果回调

OnChargeListener 接口类

`void onChargeStartTry()`

充电开始尝试

`void onChargeObstacleRemind()`

充电遇到障碍物时的提醒（障碍物一直存在时默认每隔 10s 回调一次）

`void onChargeRelocateBegin();`

充电重定位开始，当充电过程中，主动进行重定位时回调

```
void onChargeRelocateEnd(boolean isRelocateSuccess);
```

isRelocateSuccess: 重定位结果

充电重定位结束，只有执行重定位后才会回调

```
void onChargeResult(boolean isChargeSuccess)
```

isChargeSuccess: 是否成功

充电结果回调

OnNavigateListener 接口类

```
void onNavigateStartTry()
```

导航开始尝试

```
void onNavigateObstacleRemind()
```

导航遇到障碍物时的提醒（障碍物一直存在时默认每隔 10s 回调一次）

```
void onNavigateRelocateBegin();
```

导航重定位开始，当导航过程中，主动进行重定位时回调

```
void onNavigateRelocateEnd(boolean isRelocateSuccess);
```

isRelocateSuccess: 重定位结果

导航重定位结束，只有执行重定位后才会回调

`void onNavigateResult(boolean isNavigateSuccess)`

isNavigateSuccess: 是否成功

导航结果回调

OnObstacleListener 类

`void onObstacle(boolean isTrigger)`

isChargeSuccess: 是否触发

检测障碍物的回调（当障碍物一直存在时只会回调一次）

OnUpgradeListener 类

`void onUpgradeProgress(int progress)`

progress: 升级进度

升级进度

`void onUpgradeResult(boolean isSuccess)`

isSuccess: 回调结果

升级结果

OnSystemStopListener 接口类

`void onSystemStop(boolean isTrigger)`

isTrigger: 是否触发

急停、刹车结果回调

OnSensorListener 接口类

```
void onSensorTrigger(SensorType sensorType, int id)
```

sensorType: 传感器类型

id: 传感器 id

传感器触发的回调

SlamCode 类

```
public static final int SUCCESS = 0;
```

连接成功

```
public static final int PARAMETER_INVALID = 10000;
```

参数无效

```
public static final int AUTHENTICATION_FAILED = 10001;
```

底盘认证失败

```
public static final int ERROR = 10002;
```

异常

```
public static final int STATUS_SENSOR_OPEN = 0;
```

传感器状态默认打开

```
public static final int STATUS_SENSOR_ALL_CLOSE = 1;
```

传感器状态全部关闭

```
public static final int STATUS_SENSOR_SONAR_CLOSE = 2;
```

超声波状态关闭

```
public static final int STATUS_SENSOR_CLIFF_CLOSE = 3;
```

防跌落状态关闭

```
public static final int STATUS_SENSOR BUMPER_CLOSE = 4;
```

防碰撞状态关闭

```
public static final int TYPE_IDLE = 0;
```

无类型

```
public static final int TYPE_MARK_LOCATION = 1;
```

标记位置

```
public static final int TYPE_RELOCATION = 2;
```

重定位

```
public static final int TYPE_ARRIVE_NOT_ROTATE = 3;
```

到点不旋转

```
public static final int CODE_MOTOR_ERROR = 33882112;
```


电机异常

```
public static final int CODE_BATTERY_COMMUNICATE_ERROR = 33882113;
```

电池通信异常

LocationBean 实体类

```
private String mapName;  
private String locationNumber;  
private String locationNameChina;  
private String locationNameEnglish;  
private String content;  
private float x;  
private float y;  
private float yaw;  
private int type;  
private int sensorStatus;  
private float startX;  
private float startY;  
private float endX;  
private float endY;  
private long time;
```

NetBean 实体类

```
private String mode;  
private String ssid;  
private String ip;
```

地图界面实现使用 MapView 类 (`com.tobot.slam.view.MapView`)