How to implement map building and navigation

Preface

Since NAOqi 2.9.1, SoftBank Robotics has provided a map building tool and navigation framework to assist users in indoor navigation development.

Creating maps and points with the map builder tool

Installing the map builder

The Pepper map builder is an application built on the QSDK API to assist developers in creating indoor maps efficiently. It is supported in NAOqi 2.9.1 and above.

The map building tool is currently in internal testing, please contact technical support through the hotline 400-639-1680, Softbank Robotics WeChat public number, support@softbankrobotics.com.cn to submit a test application, which will then be used as the default installation application for robots in mainland China.

Preliminary Preparation

- Since the scanning recognition range of the laser sensor is about 3-5 metres, the map building site should not be too empty, and there needs to be a clear identifier within 3 metres.
- Choose the time when people move the least, remove the temporary obstacles, and tidy up the items placed for a long period of time.

Create a map

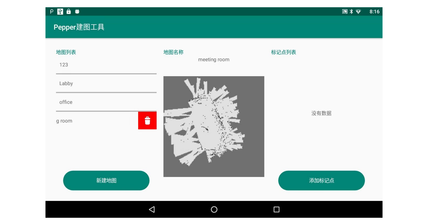
Place Pepper at any map building start point (we suggest using the actual walking start point as the map building start point), tap the New Map button in the picture above or tap Pepper's head to start creating a map.

Make sure there is no one within 3 metres of Pepper, and after waiting for Pepper to finish positioning in place, follow the voice prompts to approach Pepper from behind.

Push Pepper to walk throughout the map building site (during pushing, the pace should be as far away from Pepper as possible to minimise noise points in the map creation)

Tap the Finish building button or tap Pepper's head to finish, then follow the prompts to enter the map name to finish building the map.

Delete map



Press and hold the map you want to delete and swipe left, click the delete button to delete the map.

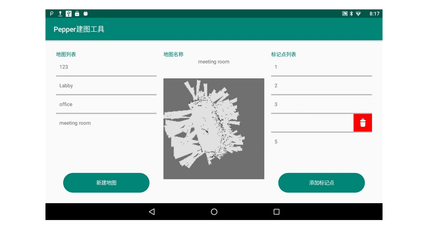
Save Marker

Select the map you want to add a marker to and push Pepper to the location you want to mark after positioning the marker on the map.

Click the Add Marker button and enter the name of the marker to save the marker.

Note: In the actual walking process, the distance between two adjacent marker points should not be more than 6 metres, if the distance is bigger, you can build more auxiliary points in the middle (Pepper can not perform any action when it reaches that point).

Modify marker points



Press and hold the marker you want to delete and swipe left, then click the delete button to delete the marker.

Implementing navigation logic using Navigation framework

Navigation framework is based on the navigation framework supported by Softbank Pepper robot since NAOqi 2.9.1 and above.

Navigation framework provides a simple navigation interface oriented to business needs, encapsulates the complex logic of using QSDK API to implement navigation, and allows users to focus more on business development. It is recommended that beginners use the Navigation framework to implement the navigation logic, and if they have more advanced customisation needs, they can also use the Navigation API in the QSDK API directly.

Functions

- Get the list of maps created by the map builder.
- Get the list of marker points of the corresponding map.
- Define marker order
- Automatic navigation
- Event callback handling during navigation

Introducing dependency packages

First, in the Android project, add the maven repository:

```
maven {
url 'http://maven.softbankrobotics.com.cn/releases'
}
```

Add the following dependencies to the **build.gradle** file of your Android project:

```
implementation 'cn.softbankrobotics.transam:navigation:1.0.6'
```

Implementing navigation logic in an application

Broad steps for implementing Pepper's navigation logic

- Create the Pepper robot application, see link for details
- Get the map to be navigated
- Load pre-set marker points
- Sort the marker points according to business needs
- Automatically navigate according to the set order of marker points

Navigation Framework API Introduction

- Create the RobotMapService service, note that the qiContext is obtained from the onRobotFocusGained(QiContext qiContext) function.

``` java @Override public void onRobotFocusGained(QiContext qiContext) { RobotMapService robotMapService = new RobotMapServiceBuild().setContext(this, qiContext).build(); }

```
* Get all the current map names
```

java List maps = robotMapService.listRobotMapName();

```
* Load selected map data and get marker points corresponding to the map.
```

java // Get the specified map String mapName = ; RobotMap robotMap = robotMapService.getLocalRobotMap(mapName); //Get the marker points in the map. // Get the markers in the map, the key is the name of the marker entered when building the map, it is recommended to check if the marker exists in this step. Map positions = robotMap.getPositions(); // Get the positions in the map.

```
* Creating the NavigationService service
```

java NavigationService navigationService = new NavigationServiceBuild().setRobotMap(robotMap).setContext(qiContext).build();

```
* According to the business needs of the marker points sorted into an array of names and converted
```

java String[] positionNames = ...; List positions = navigationService.getPositions(positionNames);

```
* Adding a listener event for the NavigationService service.
```

java navigationService.setListener(new NavigationListener() { /**

- @param curerentPosition current marker point
- @param nextPosition the next marker point
- @param errorMessage error message
- @param errorType Error type */ void onError(Position curerentPosition, Position nextPosition, String errorMessage, Enum errorType) { // Any error that occurs in the navigation will be returned through this function. }

/**

- @param position the current marker point
- @return Whether to start navigation from this marker point or not */ boolean leaveFromPosition(Position position) { // Leave the marker. }

/**

- @param isArrived Whether or not this marker point is arrived at
- @param position Arrived at this marker
- @param errorType Returns the error type if this marker was not reached.
- @return Whether to proceed to the next marker. */ boolean arrivedAtPostion(boolean isArrived, Position position, Enum errorType) { // Arrived at the postion. } });

```
* Turn on navigation based on the Position list, and move the points sequentially
```

java navigationService.startByPositon(positions);

```
* Stop navigation
```

java navigationService.stop();

```
### Notes

As the current tablet for Pepper is the Android 6.0 (API 23) platform, the user needs to authorise

When using the navigation framework, it needs to be added to the AndroidManifest.xml file:
```

xml

```
Also add runtime authorisation to the app (easypermissions 3rd party library is recommended):
```

groovy implementation 'pub.devrel:easypermissions:1.0.1' ```
```