NavKit Qt

Getting Started Guide

Version 1.02

Address

TeleCommunication Systems, Inc.
6A Liberty
Aliso Viejo, Ca 92656 USA

(949) 453-1646
www.telecomsys.com

Contents

1 Overview

1.1 Introduction

This *Getting Start Guide* provides basic information to create applications using the NavKit library from the Location Toolkit (LTK) on PC and embedded systems. After reading this guide, you should be able to create a sample application that uses the LTK libraries to do navigation.

1.2 Assumptions

This document assumes that you are familiar with Qt and C++ development. Before getting started, you should install the Qt libraries on your computer. If you are working on the Windows platform, you will also need to install MinGW as the compiler environment. Because this SDK makes use of OpenGL to render the map, make sure that the OpenGL libraries are correctly installed and Qt's OpenGL support is enabled. We recommend that you install and use QtCreator to facilitate faster development.

For development environment setup and required Qt libraries, please refer to:

LTK Qt Linux Dev Environment Setup.pdf

For runtime environment setup and required Qt libraries, please refer to:

LTK Qt Linux Run Environment Setup.pdf

We strongly recommend you to read through these documents before getting started.

2 Getting Started

2.1 Import the NavKit Sample Project and Run

2.1.1 Unzip the Package

Unzip the release package ltk_<version>.zip. The unzipped folder structure is:

```
<release_package_folder>
+----assets
+----bin
|    +----mapkit3d_sample
|    +----navkit_sample
+----docs
+----library
|    +----include
|    +----lib
|    +----locationtoolkit.pro
+----samples
|    +----mapkit3d_sample
|    +----navkit_sample
```

2.1.2 Importing the NavKit Sample Project

1. Open Qt creator from the start menu
2. Select “Open Project” in the Qt creator welcome page.
3. Choose the project file from release_package_folder/samples/navkit_sample/project/

2.1.3 Compile the Project

1. On Windows
    a. In the Qt creator “Edit” page, right click on navkitsample
    b. Choose “build” in the popup menu.
    c. Once the compile finishes, the output file navkit_sample.exe can be found at:

    release_package_folder/samples/navkit_sample/project/debug

    or

    release_package_folder/samples/navkit_sample/project/release

2. On Ubuntu Linux
    a. First ensure the Qt development environment is properly installed and working.
    b. In this project folder release_package_folder/samples/navkit_sample/project/
    c. Run these commands in sequence.
        i. qmake
        ii. make
    d. Once the compile finishes, the output file navkit_sample.exe can be found at:

    release_package_folder/samples/navkit_sample/project/release

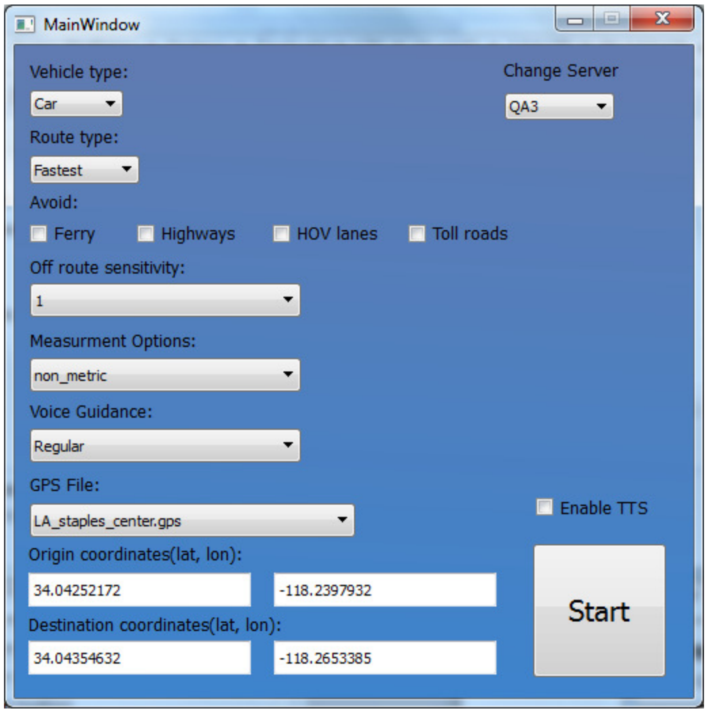
2.1.4 Run the NavKit Sample Application

1. On Windows
    a. Copy the output file navkit_sample.exe into this folder.

        release_package_folder/bin/samples/navkit_sample
    b. Double click the file navkit_sample.exe

2. On Ubuntu Linux
    a. Run following command in sequence.
        i. sudo apt-get install alsa-oss
        ii. ./ aoss ./navkit_sample

The following window will display on successful execution:



Vehicle type: Car

Route type: Fastest

Avoid: Ferry, Highways, HOV lanes, Toll roads

Off route sensitivity: 1

Measurment Options: non_metric

Voice Guidance: Regular

GPS File: LA_staples_center.gps

Change Server: QA3

Enable TTS

Origin coordinates(lat, lon): 34.04252172, -118.2397932

Destination coordinates(lat, lon): 34.04354632, -118.2653385

2.2 Build your own application using the LTK SDK

2.2.1 Create a Project

For simplicity, start with the QtCreator wizard. For example, create a Qt application with the mainwindow wizard, and then modify the pro file to include the LTK SDK into your project:

a. Define an environment variable named "LTK_ROOT" that contains the library path of the LTK SDK, and then include locationtoolkit.pro in your own project file. This will look similar to the following:

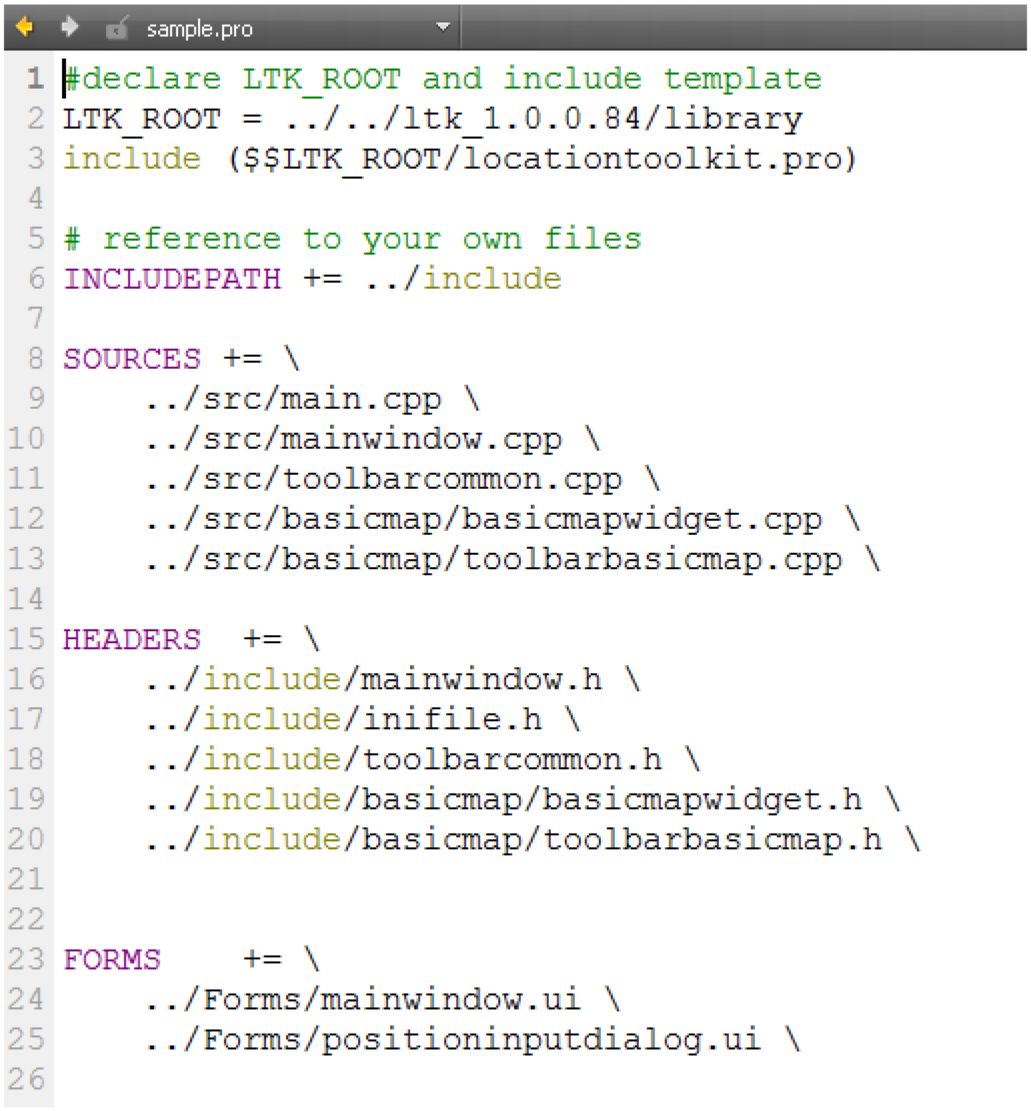
```
LTK_ROOT = some path…/ltk_1.0.0.84/library
include ($$LTK_ROOT/locationtoolkit.pro)
```

b. Remove the auto generated code, it will look similar to the following:

```
QT       += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
```

The locationtoolkit.pro file will already contains these lines.

After these steps, your project file should contain two parts, one is the declaration of LTK_ROOT and includes the pro file in the SDK, the other part references your own .h, .cpp and .ui files, as seen here:



```
#declare LTK_ROOT and include template
LTK_ROOT = ../../ltk_1.0.0.84/library
include ($$LTK_ROOT/locationtoolkit.pro)

# reference to your own files
INCLUDEPATH += ../include

SOURCES += \
    ../src/main.cpp \
    ../src/mainwindow.cpp \
    ../src/toolbarcommon.cpp \
    ../src/basicmap/basicmapwidget.cpp \
    ../src/basicmap/toolbarbasicmap.cpp \

HEADERS  += \
    ../include/mainwindow.h \
    ../include/inifile.h \
    ../include/toolbarcommon.h \
    ../include/basicmap/basicmapwidget.h \
    ../include/basicmap/toolbarbasicmap.h \


FORMS    += \
    ../Forms/mainwindow.ui \
    ../Forms/positioninputdialog.ui \

```

2.2.2 Create the LTK Context Object

You need to create an LTK Context object first when using this SDK. There are seven parameters needed; refer to the API documentation for details. The code will look similar to the following:

```
ltkContext = new locationtoolkit::LTKContext(credential,
                    "qt",
                    "en-US",
                    0000000000ULL,
                    domain,
                    hostname,
                    dpi );
```

2.2.3 Create and initialize a navigation session instance

Create an instance of locationtoolkit::Navigation with the required parameters; refer to the API documentation for details. The code will look similar to the following:

```
mNavigationImpl = Navigation::CreateNavigation(ltkcontext, place, routeoption, preference);
```

Connect the signals and slots you need prior to starting a navigation session. The slots need to connect with the signals provided by the navigation instance. The code will look similar to the following:

Connect the session signals; refer to the API documentation for details:

```
QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(OffRoute()),                    this,
SLOT(OnOffRoute()) );

QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(OnRoute()),                    this,
SLOT(OnRoute()) );

QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(RouteReceived( RouteRequestReason, const
QVector<QSharedPointer<RouteInformation> >&)), this, SLOT(OnRouteReceived( RouteRequestReason, const
QVector<QSharedPointer<RouteInformation> >&)) );


QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(RouteRequested(RouteRequestReason)), this,
SLOT(OnRouteRequested(RouteRequestReason)) );

QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(RouteProgress(qint32)),                    this,
SLOT(OnRouteProgress(qint32)) );

QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(RouteError(NavigateRouteError)), this,
SLOT(OnRouteError(NavigateRouteError)) );

QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(ArrivingToDestination(DestinationStreetSide )), this,
SLOT(OnArrivingToDestination(DestinationStreetSide)) );

QObject::connect( &mNavigationImpl->GetSessionSignals(), SIGNAL(RouteFinish()),                    this,
SLOT(OnRouteFinish()) );
```

Connect the navigation update signals; refer to the API documentation for details:

```
QObject::connect( &mNavigationImpl->GetNavigationUpdateSignals(), SIGNAL(TripRemainingTime(quint32)), this,
SLOT(OnTripRemainingTime(quint32)) );

QObject::connect( &mNavigationImpl->GetNavigationUpdateSignals(), SIGNAL(TripRemainingDistance(const qreal&)), this,
SLOT(OnTripRemainingDistance(const qreal&)) );

QObject::connect( &mNavigationImpl->GetNavigationUpdateSignals(), SIGNAL(UpdateManeuverList(const ManeuverList&)), this,
SLOT(OnUpdateManeuverList(const ManeuverList&)) );

QObject::connect( &mNavigationImpl->GetNavigationUpdateSignals(), SIGNAL(ManeuverRemainingDistance(const qreal&)), this,
SLOT(OnManeuverRemainingDistance(const qreal&)) );
```

Connect the traffic signals; refer to the API documentation for details.

```
QObject::connect( &mNavigationImpl->GetTrafficSignals(), SIGNAL(TrafficAlerted(const TrafficEvent&)), this,
SLOT(OnTrafficAlerted(const TrafficEvent&)) );

QObject::connect( &mNavigationImpl->GetTrafficSignals(), SIGNAL(TrafficChanged(const TrafficInformation&)), this,
SLOT(OnTrafficChanged(const TrafficInformation&)) );

QObject::connect( &mNavigationImpl->GetTrafficSignals(), SIGNAL(DisableTrafficAlerted()), this, SLOT(OnDisableTrafficAlerted()) );
```

For any other signals not listed here, you can follow the sample code above on how to connect with your slots.

The GPS can then be updated by calling the interface provided by the navigation instance of the navigation session. The code will look similar to the following:

```
mNavigationImpl->UpdatePosition(location);
```

When you want to stop the navigation session, just call the interface provide by the navigation instance. The code will look similar to the following:

```
mNavigationImpl->StopSession();
```

2.2.4 Copy Navigation Resource Files

After successfully compiling your project, the navigation resource directory and its files need to be copied from the NavKit sample app binary directory (bin\navkit_sample\resource) to the bin\resource directory of your newly built project.

2.2.5 Run the Navigation Application

You should now be able run your project’s binary executable.

3 NavKit Sample Source Code Key Points

3.1.1 Modify the Navigation Parameters

3.1.1.1 Create the RouteOptions Object

The RouteOptions object is created in the on_StartButton_clicked() function. For example:

```
RouteOptions* routeOption = new RouteOptions(mCurrentRouteType,mCurrentVehicleType,mCurrentAvoid);
```

The members of this object can be changed with the following functions.

3.1.1.1.1 Change Vehicle type

Use the on_comboBox_vehicleType_currentIndexChanged() function to change the vehicle type that you want:

```
mCurrentVehicleType = TM_Car;
```

3.1.1.1.2 Change Route type

Use the on_comboBox_routeType_currentIndexChanged() function to change the route type that you want:

```
mCurrentRouteType = RT_Fastest;
```

3.1.1.1.3 Change Avoid

Use the “on_checkBox_tollroads_clicked()” function to change the avoid type to “Toll roads”:

```
mCurrentAvoid = mCurrentAvoid | RA_Toll;
```

For the other three types of “Avoid”, the process flow is the same as “Toll roads”; please refer to following functions:

on_checkBox_highways_clicked()

on_checkBox_hovlanes_clicked()

on_checkBox_ferry_clicked()

3.1.1.2 Create the Preferences Object

The Preferences object is created in the on_StartButton_clicked() function. For example:

```
Preferences preference;
preference.SetMeasurement(mCurrentMeasurmentOption);
preference.SetOffRouteIgnoreCount(mCurrentOffRouteSensitivity);

if(mCurrentVoiceGuide == "Regular")
{
   preference.SetLaneGuidance(QBool(false));
   preference.SetNaturalGuidance(QBool(false));
}
else if(mCurrentVoiceGuide == "Natural Guidance")
```

```
{
   preference.SetNaturalGuidance(QBool(true));
}
else if(mCurrentVoiceGuide == "Lane Guidance")
{
   preference.SetLaneGuidance(QBool(true));
}
else if(mCurrentVoiceGuide == "Natural and Lane Guidance")
{
   preference.SetLaneGuidance(QBool(true));
   preference.SetNaturalGuidance(QBool(true));
}
else
{
   preference.SetLaneGuidance(QBool(false));
   preference.SetNaturalGuidance(QBool(false));
}
```

The members of this object can be changed with the following functions:

3.1.1.2.1 Change Off Route Sensitivity

Use the on_comboBox_offRouteSens_currentIndexChanged() function to change the off route sensitivity:

```
mCurrentOffRouteSensitivity = arg1.toUInt();
```

3.1.1.2.2 Change Measurement Options

Use the on_comboBox_measurmentOption_currentIndexChanged() function to change the measurement options:

```
mCurrentMeasurmentOption = Preferences::Metric;
```

3.1.1.2.3 Change Voice Guidance

Use the on_comboBox_voiceGuid_currentIndexChanged() function to change the voice guidance:

```
mCurrentVoiceGuide = arg1;
```

3.1.1.3 Create the Place Object

The Place object is created in the "on_StartButton_clicked()" function. For example:

```
MapLocation maplocation;
Place place;
maplocation.center.latitude  = ui->lineEdit_destcoord_lat->text().toDouble();
maplocation.center.longitude = ui->lineEdit_destcoord_lon->text().toDouble();
place.SetLocation(maplocation);
```

3.1.2 Start Navigation

To start a new Navigation, you need to:

a. Use the function on_StartButton_clicked() to create all the parameters for navigation

b. Use the OnStartNavigation(LTKContext& ltkcontext, const Place& place, const RouteOptions& routeoption, const Preferences& preference) function to start navigation

For example:

```
mNavigationImpl = Navigation::CreateNavigation(ltkcontext, place, routeoption, preference);

LocationProvider& locProvider = LocationProvider::GetInstance( mLocationConfiguration );

locProvider.StartReceivingFixes( static_cast<LocationListener&>(*this) );

QObject::connect(&mNavigationImpl->GetSessionSignals(),SIGNAL(OffRoute()),this,SLOT(OnOffRoute()));

QObject::connect(&mNavigationImpl->GetSessionSignals(),SIGNAL(OnRoute()),this,SLOT(OnRoute()));

QObject::connect(&mNavigationImpl-
>GetSessionSignals(),SIGNAL(RouteReceived(RouteRequestReason,constQVector<QSharedPointer<RouteInformation>>&)),thi
s,SLOT(OnRouteReceived(RouteRequestReason,constQVector<QSharedPointer<RouteInformation>>&)));

QObject::connect(&mNavigationImpl-
>GetSessionSignals(),SIGNAL(RouteRequested(RouteRequestReason)),this,SLOT(OnRouteRequested(RouteRequestReason)));

QObject::connect(&mNavigationImpl-
>GetSessionSignals(),SIGNAL(RouteProgress(qint32)),this,SLOT(OnRouteProgress(qint32)));

QObject::connect(&mNavigationImpl-
>GetSessionSignals(),SIGNAL(RouteError(NavigateRouteError)),this,SLOT(OnRouteError(NavigateRouteError)));

QObject::connect(&mNavigationImpl-
>GetSessionSignals(),SIGNAL(ArrivingToDestination(DestinationStreetSide)),this,SLOT(OnArrivingToDestination(DestinationStr
eetSide)));

QObject::connect(&mNavigationImpl->GetSessionSignals(),SIGNAL(RouteFinish()),this,SLOT(OnRouteFinish()));

QObject::connect(&mNavigationImpl-
>GetNavigationUpdateSignals(),SIGNAL(TripRemainingTime(quint32)),this,SLOT(OnTripRemainingTime(quint32)));

QObject::connect(&mNavigationImpl-
>GetNavigationUpdateSignals(),SIGNAL(TripRemainingDistance(constqreal&)),this,SLOT(OnTripRemainingDistance(constqreal
&)));

QObject::connect(&mNavigationImpl-
>GetNavigationUpdateSignals(),SIGNAL(UpdateManeuverList(constManeuverList&)),this,SLOT(OnUpdateManeuverList(const
ManeuverList&)));

QObject::connect(&mNavigationImpl-
>GetNavigationUpdateSignals(),SIGNAL(ManeuverRemainingDistance(constqreal&)),this,SLOT(OnManeuverRemainingDistanc
e(constqreal&)));
```

```
QObject::connect(&mNavigationImpl-
>GetTrafficSignals(),SIGNAL(TrafficAlerted(constTrafficEvent&)),this,SLOT(OnTrafficAlerted(constTrafficEvent&)));



QObject::connect(&mNavigationImpl-
>GetTrafficSignals(),SIGNAL(TrafficChanged(constTrafficInformation&)),this,SLOT(OnTrafficChanged(constTrafficInformation&)
));

QObject::connect(&mNavigationImpl-
>GetTrafficSignals(),SIGNAL(DisableTrafficAlerted()),this,SLOT(OnDisableTrafficAlerted()));

QObject::connect(&mNavigationImpl-
>GetAnnouncementSignals(),SIGNAL(Announce(constAnnouncement&)),this,SLOT(OnAnnounce(constAnnouncement&)));
```

3.1.3 Show a popup window when navigation starts

Use the NavigationSession::OnRouteRequested() callback function to show a “Start Navigation” or “Recalculating” or “Detour” popup window when navigation starts:

```
switch(reason)
{
   case RRR_Calculate:
      emit ShowWidget(2);
      break;
   case RRR_Recalculate:
      emit ShowWidget(3);
      break;
   case RRR_RouteSelector:
      break;
   case RRR_Detour:
      emit ShowWidget(4);
      break;
   default:
      break;
 }
```

3.1.4 Show a popup window when a route error occurs

Use the NavigationSession::OnRouteError() callback function to show any error messages received from the server:

```
emit ShowMessageBox(errorText, 2);
OnStopNavigation();
```

3.1.5 Update Maneuver List

Use the NavigationSession::OnUpdateManeuverList() callback function to update the maneuver list in navigation:

```
emit UpdateManeuverList(maneuvers);
```

3.1.6 Cancel navigation

If you want cancel the current navigation, use the `OnStopNavigation()` function. For example:

```
mNavigationImpl->StopSession();
delete mNavigationImpl;
mNavigationImpl = NULL;

LocationProvider& locProvider = LocationProvider::GetInstance( mLocationConfiguration );
locProvider.StopReceivingFixes( static_cast<LocationListener&>(*this) );
mLocationConfiguration.locationFilename = "";
```

3.1.7 Enable Text to Speech (TTS)

Use the function `OnEnableTTS()` to enable TTS before navigation begins:

```
mEnableTTS = ttsStatus;
```

Use this variable to control the voice in the `OnAnnounce()` function:

```
if(mEnableTTS)
{
   mTTSPlayer->Play(announcement);
}
```

3.1.8 Select a Maneuver to Announce

In navigation, select a maneuver from the list, the maneuver sound will be played immediately.

Use the function `on_listWidget_pressed(const QModelIndex &index)` to check for a “press” event:

```
QListWidgetItem* curItem = ui->listWidget->currentItem();
qint32 maneuverID = curItem->data(ManeuverIDRole).toInt();
```

The sound will then be played in the `OnPlayAnounce()`function

```
mNavigationImpl->Announce((qint32)index);
```