国立大学法人
電気通信大学
Unique & Exciting Campus

The University of Electro-Communications

# Tutorial for SUMO and OMNeT++ interfacing by libsumo

Author: Chanarisu

Celimuge laboratory, Chofu, Tokyo, Japan

November, 2022

## Contents

## 1 Introduction

**S**imulation of **U**rban **MO**bility (SUMO) can simulate a realistic vehicular traffic. So far, four kinds of programming language can process the taffic data from SUMO, python, matlab, java and C++. Here, we mainly introduce SUMO coupling with OMNeT++ by C++ programming language. Regarding C++ language, three interfacing approaches can couple SUMO with OMNeT++, including TraCIScenarioManager, Veins and libsumo.

- TraCIScenarioManager appeared first and is an interface based on **Tra**ffic **C**ontrol **I**nterface (TraCI) TraCI command, which can control on-line in the bi-directional between OMNeT++ and SUMO. For TraCI command, more information see the website `https://sumo.dlr.de/docs/TraCI.html`.
- **Ve**hicles **i**n **n**etwork **s**imulation (Veins) was developed by Christoph Sommer and his team. Veins is as a middle-ware based on TraCI command for coupling SUMO with OMNeT++. Veins provides a suit of modules for simulating vehicular network with the mobility. Before running the veins module, sumo_launchd.py (so far, is updated as veins_launchd.py in the *veins_of_dir/bin*) needs to be started, to create a deamon for listening the request. For veins, more information see the website `https://veins.car2x.org`.
- **libsumo** is an interfacing library developed by SUMO. Due to the communication overhead running the TraCIScenarioManager, libsumo can provide a more efficient coupling without the need for socket communication.

The libsumo library provides a flexible interface between SUMO and OMNeT++. Some functionality that veins simulator could not implement can be easily achieved. Here, we implement the functions below.

1. Bus, truck, coach, taxi and e-vehicle can be mapped different nodes into OMNeT++.
2. The configured route for each vehicle is valid route and does not occur no connection or other TRACIException. In contrast, the random trips generated be the randomTrips.py (in the sumo_home_dir/tools) has not been validated.
3. All vehicle entered the map in the same time.
4. The random route for each vehicle is generated when the simulator starts.
5. Does not start veins_launchd.py to listen port.

This tutorial and related code have been updated to the github and can be downloaded from `https://github.com/nrs018/omnet_libsumo`.

# 2 Installation

The libsumo library is tested on Ubuntu and on windows and MacOS, has not been tested yet.

For the version of the software are listed below.

- Ubuntu: 20.04.2 LTS.
- OMNeT++: 5.5.1, download from https://omnetpp.org/download/.
- INET framwork: 4.2.2, download from https://inet.omnetpp.org/Download.html.
- SUMO: 1.11.0, download from https://sumo.dlr.de/docs/Downloads.php.

We recommend that installing SUMO, OMNeT++ and INET framework are manual.

This tutorial does not yet tested on the Windows and MacOS.

## 2.1 Installing SUMO

Download sumo-src-1.11.0.tar.gz package from the site https://sumo.dlr.de/releases/1.11.0/ and unzip the package to your dir.

And then run the command following.

```
> sudo apt-get install cmake python g++ libxerces-c-dev libfox-1.6-dev
```

```
> sudo apt-get install libgdal-dev libproj-dev libgl2ps-dev swig
```

```
> cd <yout_sumo_dir>
```
# *your_sumo_dir* needs to be replaced real address in your computer.

```
> export SUMO_HOME="\$PWD"
```
# SUMO_HOME can be set to be the system environment variable.

```
> mkdir build/cmake-build && cd build/cmake-build
```

```
> cmake ../..
```

```
> make -j\$(nproc)
```

After installing SUMO, check whether the following files exist in the directory *your_sumo_dir*.

|–bin
    |– ...
    |–libsumocpp.so ← This file will be linked to the OMNeT++.
    |–libtracicpp.so ← This file will be linked to the OMNeT++.
    |– ...
|–build

|–data
|–docs
|–include
|–src ← This directory will be referenced in the OMNeT code.
|–tools
|–unitest
|–AUTHORS
|–ChangeLog
|–CMakeLists.txt
|–CMakeSettings.json
|–cmake-variants.yaml
|–CONTRIBUTING.md
|–Jenkinsfile
|–LICENSE
|–NOTICE.md
|–README.md
|–sumo.doxyconf

## 2.2 Installing OMNeT++

Download OMNeT++ 5.5.1 from the site https://omnetpp.org/download/old#omnetpp-551_linux and unzip the package to your dir.

Installing required packages.

```
> sudo apt-get install build-essential clang lld gdb bison flex perl
> sudo apt-get install python3 python3-pip qtbase5-dev qtchooser qt5-qmake
> sudo apt-get install qtbase5-dev-tools libqt5opengl5-dev libxml2-dev zlib1g-dev
> sudo apt-get install doxygen graphviz libwebkit2gtk-4.0-37
> python3 -m pip install --user --upgrade numpy pandas matplotlib scipy
> python3 -m pip install --user --upgrade seaborn posix_ipc
```

And then, start to install OMNeT++.

```
> cd your_omnet_dir
> . setenv
> ./configure
> make && make install
```

Finally, open OMNeT simulator.

```
> omnetpp
```

For more information related to the installation of SUMO and OMNeT, check the site https://sumo.dlr.de/docs/Installing/index.html and https://doc.omnetpp.org/omnetpp/InstallGuide.pdf.

## 2.3 Install INET framework

The network related all protocols, the algorithms and the structures are implemented in the INET framework which developed by OMNeT++ researcher. By default, the INET framework is installed automatically after OMNeT++ installs. However, in the most cases, some scenarios need other version of INET framework. Here, we recommend to install INET framework manually.

Download INET 4.2.2 from the site `https://inet.omnetpp.org/2021-01-11-INET-4.2.2-released.html` and unzip the tar package to *your_inet_dir*.

Next, the detailed installation process to INET framework are presented.

1. Open OMNeT++.

   Click *File* in the menu -> *New* -> *Project....*

   And then select *OMNeT++ Project* from OMNeT++, as shown in Figure 1.

   At last, click *Next* button.

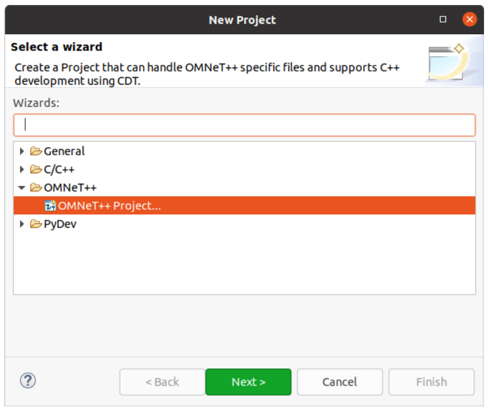
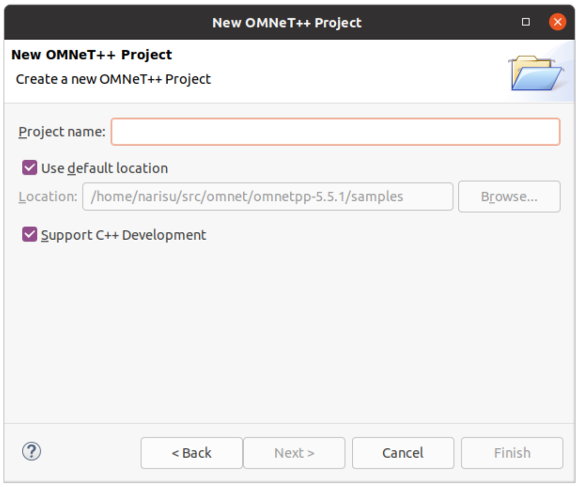


**Figure 1:** *Step 1.*

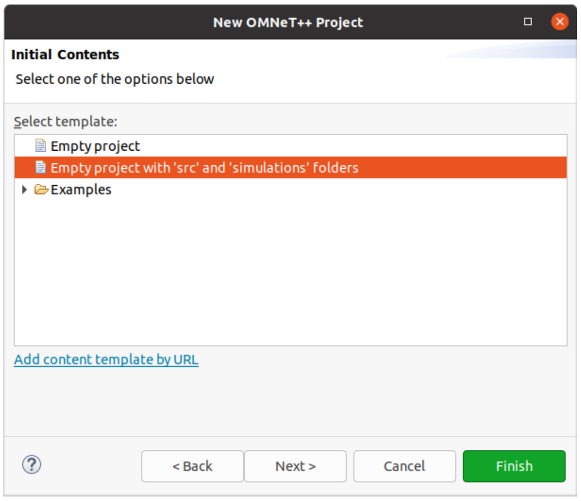
2. Input the project name and select *empty project with 'src' and 'simulation' folders*. And then, click *Finish* button, shown as in Figure 2 and 3.


New OMNeT++ Project

New OMNeT++ Project
Create a new OMNeT++ Project

Project name:
Use default location: ✓
Location: /home/narisu/src/omnet/omnetpp-5.5.1/samples
Support C++ Development: ✓

**Figure 2:** *Step 2.*

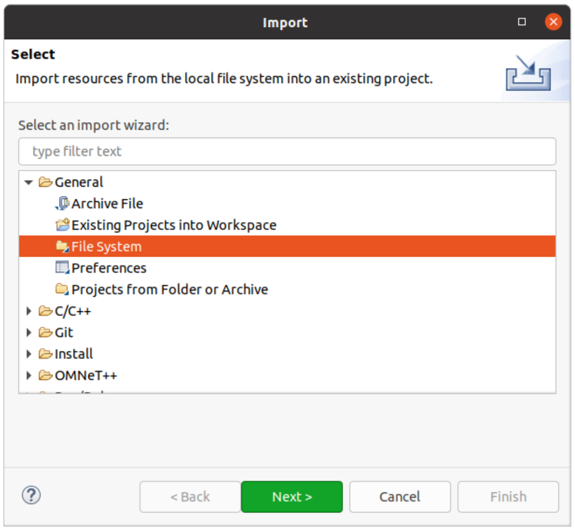

New OMNeT++ Project

Initial Contents
Select one of the options below

Select template:
Empty project
Empty project with 'src' and 'simulations' folders
Examples

**Figure 3:** *Step 3.*

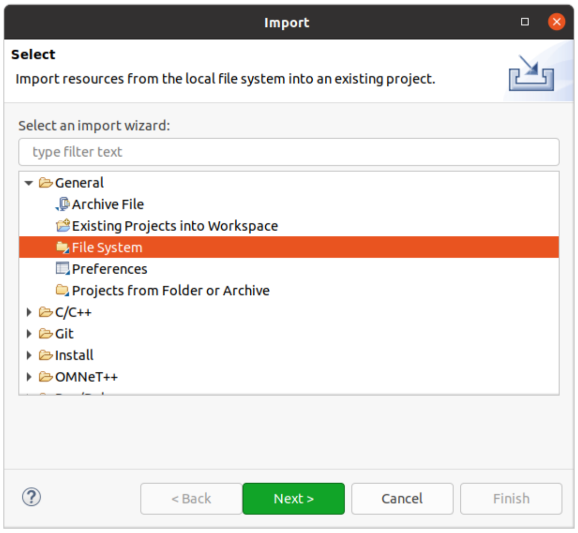
3. Select new project and click *File* in the menu -> *import*, as shown in Figure 4. And then, select *File System* and click *Next* button, as shown in Figure 5.


Import

Select
Import resources from the local file system into an existing project.

Select an import wizard:
type filter text

**Figure 4:** *Step 4.*

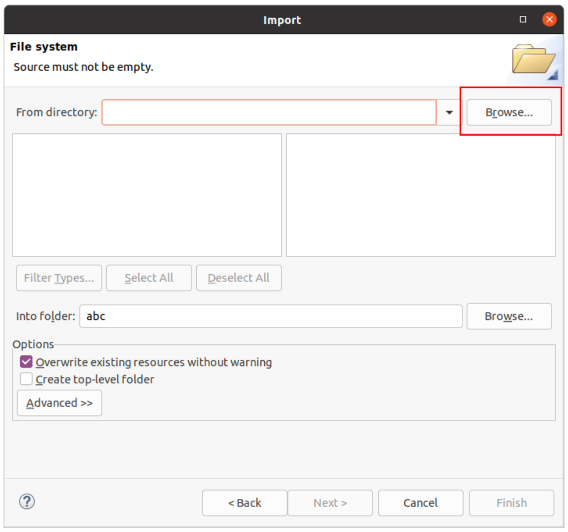

Import

Select
Import resources from the local file system into an existing project.

Select an import wizard:
type filter text

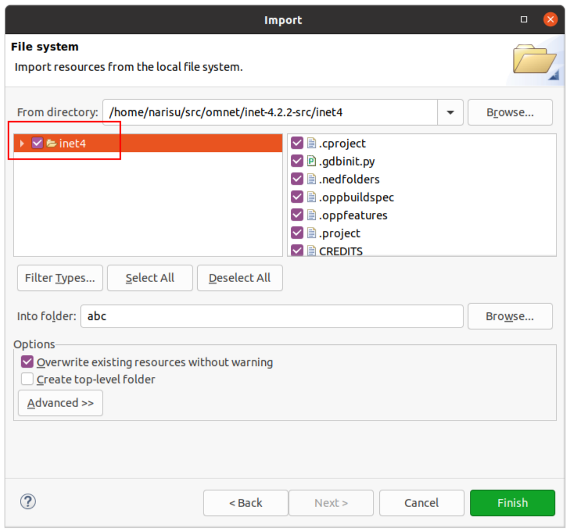
**Figure 5:** *Step 5.*

4. Click *Browse* button and select unzipped INET package from your computer, as shown in Figure 6.



File system
Source must not be empty.
From directory:
Into folder: abc
Options
☑ Overwrite existing resources without warning
☐ Create top-level folder

**Figure 6:** *Step 6.*

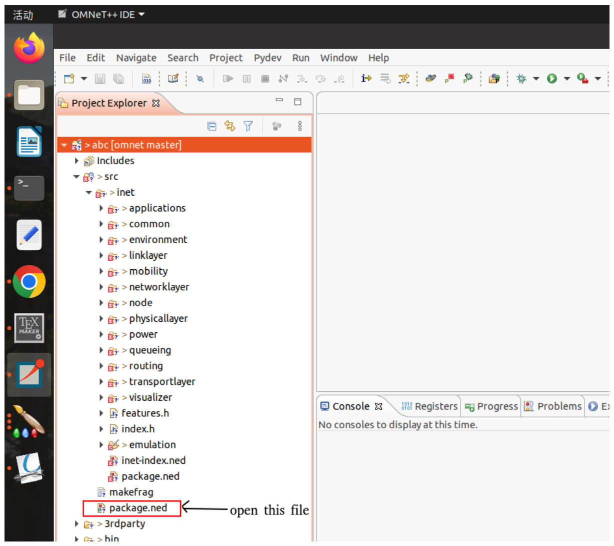


File system
Import resources from the local file system.
From directory: /home/narisu/src/omnet/inet-4.2.2-src/inet4
☑ inet4
☑ .cproject
☑ .gdbinit.py
☑ .nedfolders
☑ .oppbuildspec
☑ .oppfeatures
☑ .project
☑ CREDITS
Into folder: abc
Options
☑ Overwrite existing resources without warning
☐ Create top-level folder

**Figure 7:** *Step 7.*

5. Click the check box of inet4 in the left bar, and then clikc ***Finish*** button, as shown in Figure 7.

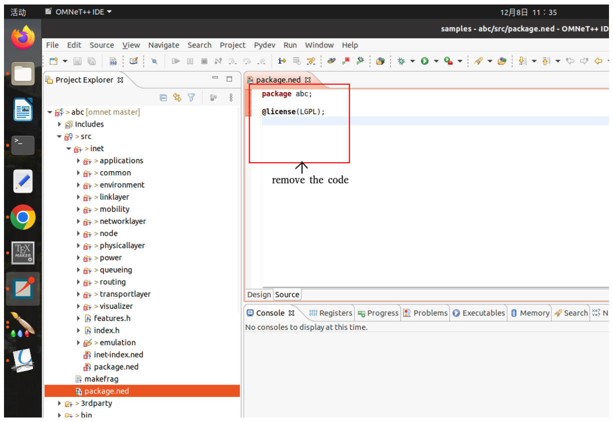
6. Imported INET framework seems to be errors, as shown in Figure 8. In this time, open the file *package.ned* from src/inet folder and remove the code, as shown in Figure 9.



```
> abc [omnet master]
  Includes
  > src
    > inet
      > applications
      > common
      > environment
      > linklayer
      > mobility
      > networklayer
      > node
      > physicallayer
      > power
      > queueing
      > routing
      > transportlayer
      > visualizer
      features.h
      index.h
      > emulation
      inet-index.ned
      package.ned
    makefrag
    package.ned
  > 3rdparty
  > bin
```

No consoles to display at this time.

**Figure 8:** *Step 8.*



```
package abc;

@license(LGPL);
```

**Figure 9:** *Step 9.*

7. Select this project and built the project. By now, INET framework has been imported into OMNeT++.

For the convenient, other unrelated projects are closed.

8. The following is the process for linking *.so and including libsumo library. To compile and link libsumo library and OMNeT, some option of make files needs to be modified. The file *Makefile.inc* locates in the OMNeT root directory, as shown in Figure 10. And open *Makefile.inc*.

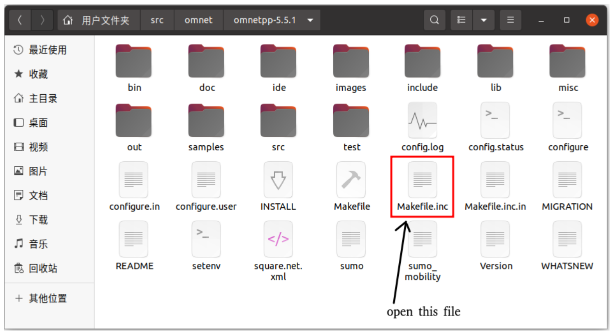
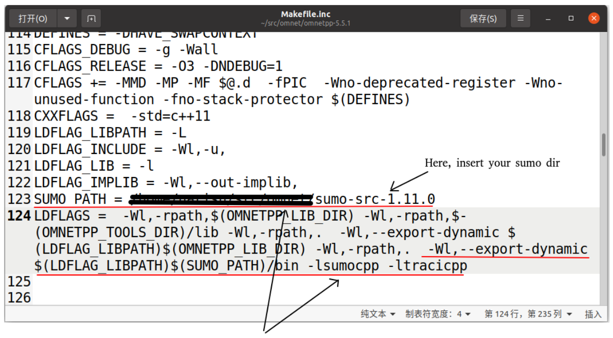


**Figure 10:** *Step 10.*

9. *Makefile.inc* needs to be edited, as shown in Figure 11. *Makefile.inc* is also uploaded to Github. The purpose of the modification is that two files libsumocpp.so and libtracicpp.so are linked to the omnet project when the OMNeT project compiles.



```
DEFINES = -DHAVE_SWAPCONTEXT
CFLAGS_DEBUG = -g -Wall
CFLAGS_RELEASE = -O3 -DNDEBUG=1
CFLAGS += -MMD -MP -MF $@.d  -fPIC  -Wno-deprecated-register -Wno-
unused-function -fno-stack-protector $(DEFINES)
CXXFLAGS =  -std=c++11
LDFLAG_LIBPATH = -L
LDFLAG_INCLUDE = -Wl,-u,
LDFLAG_LIB = -l
LDFLAG_IMPLIB = -Wl,--out-implib,
SUMO_PATH = [redacted]/sumo-src-1.11.0
LDFLAGS =  -Wl,-rpath,$(OMNETPP_LIB_DIR) -Wl,-rpath,$-
(OMNETPP_TOOLS_DIR)/lib -Wl,-rpath,.  -Wl,--export-dynamic $
(LDFLAG_LIBPATH)$(OMNETPP_LIB_DIR) -Wl,-rpath,.  -Wl,--export-dynamic
$(LDFLAG_LIBPATH)$(SUMO_PATH)/bin -lsumocpp -ltracicpp
```

Here, insert your sumo dir
The red line part is the content that needs to be inserted.

**Figure 11:** *Step 11.*

! *your_sumo_dir* needs to be replaced with the address in your computer.

10. All dirs and files in the folder *your_sumo_dir/src* are copied into *omnet_dir/include*. Figure 12 is before the copying, and Figure 13 is after the copying.

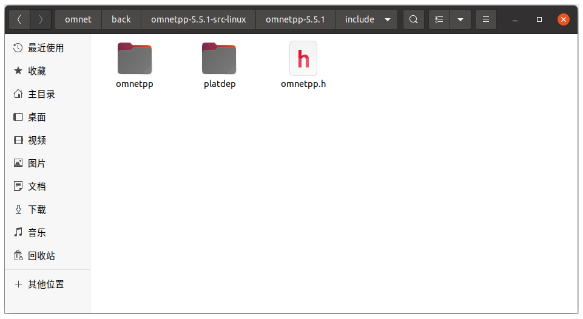


**Figure 12:** *Step 12.*

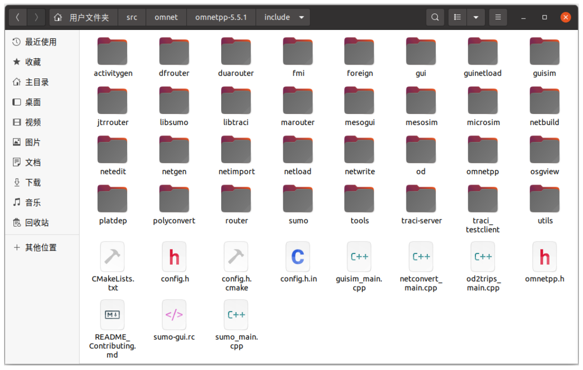


**Figure 13:** *Step 13.*

By now, the configuration for SUMO has finished. The following is the installing example.

11. The source code can be downloaded from the Github site `https://github.com/nrs018/omnet_libsumo` and unzip the *.tar package. Containing files are shown in Figure 14.

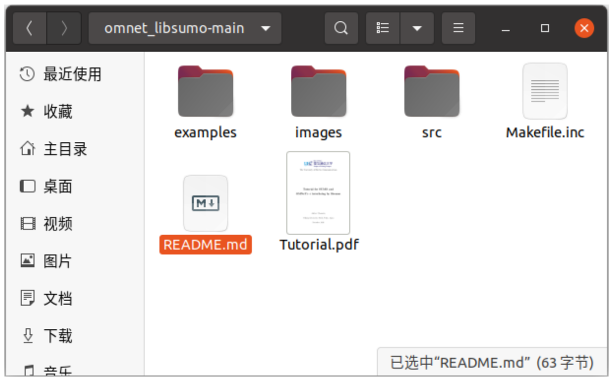


**Figure 14:** *Step 14.*

12. The direcories *src* and *examples* are copied into new created project, as shown in Figure 15.

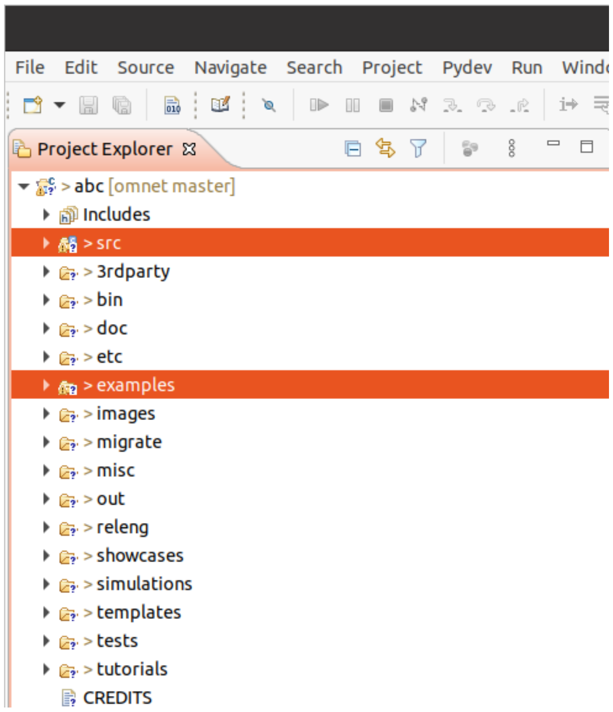


**Figure 15:** *Step 15.*

13. *Makefile.inc* and *images* dictectory are copied into *omnet_dir*, as shown in Figure 16.

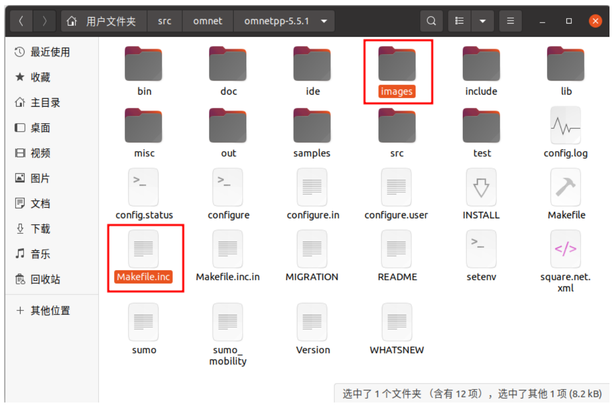


**Figure 16:** *Step 16.*

14. Finally, build the project. By now, the whole installation is finished.

## 3 Running example

This ssection presents an example of SUMO coupling with OMNeT++ interfacing by libsumo library.

We upload the core code of the example to Github and the source code is downloaded from the site https://github.com/nrs018/omnet_libsumo.

The source code dowloaded from the Github can not directly run in your environment. Because Makefile.inc file needs to be modified.

Before starting the example, the settings accordding to Section 2 must be configured. And then, unzip the package downloaded from Github.

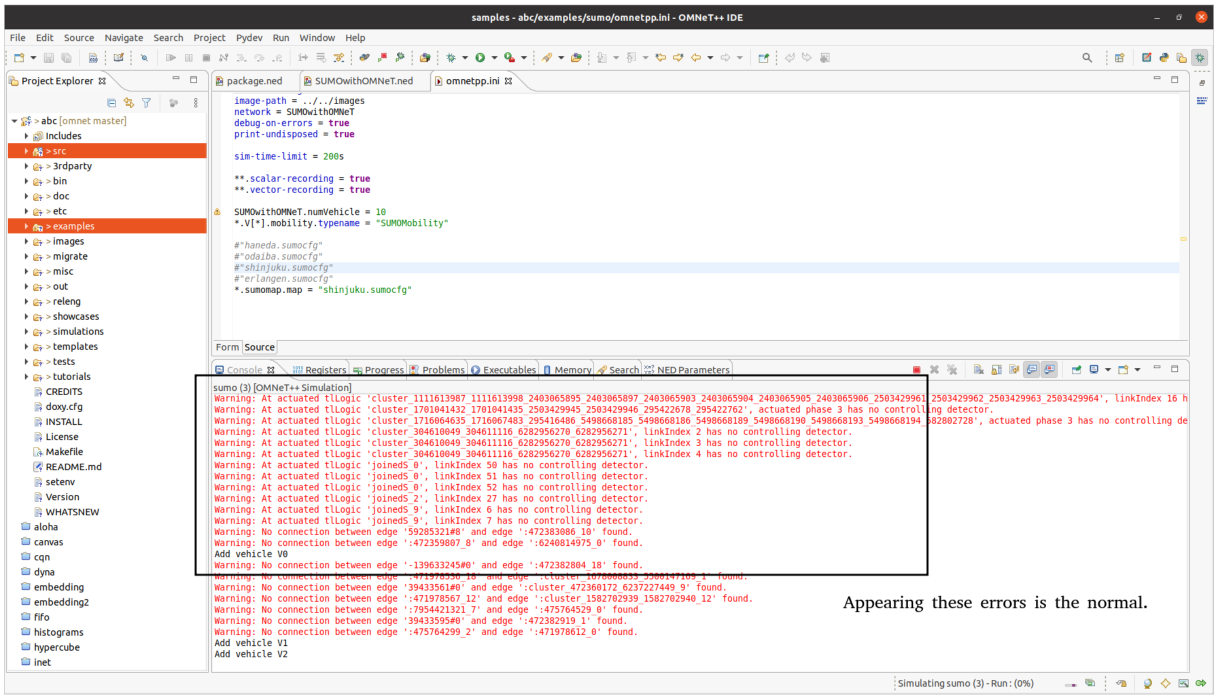
The directory includes the following.

```
|
|–src
    |–mobility
        |–single
            |–SUMOMap.cc
            |–SUMOMap.h
            |–SUMOMap.ned
            |–SUMOMobility.cc
            |–SUMOMobility.h
            |–SUMOMobility.ned
    |–visualizer
        |–roads
            |–SUMORoadsCanvasVisualizer.cc
```

```
                    |–SUMORoadsCanvasVisualizer.h
                    |–SUMORoadsCanvasVisualizer.ned
    |–examples
        |–SUMO
            |–erlangen.net.xml
            |–erlangen.sumocfg
            |–shinjuku.net.xml
            |–shinjuku.sumocfg
            |–odaiba.net.xml
            |–odaiba.sumocfg
            |–handeda.net.xml
            |–handeda.sumocfg
            |–omnetpp.ini
            |–SUMOwithOMNeT.ned
    |–Tutorial.pdf
```

In the example, four maps are provided to test, including shinjuku (Japan), odaiba (Japan), haneda (Japan) and erlangen (Germany).

Run omnetpp.ini in the *examples/sumo*, as shown in Figure 17.



```
image-path = ../../images
network = SUMOwithOMNeT
debug-on-errors = true
print-undisposed = true

sim-time-limit = 200s

**.scalar-recording = true
**.vector-recording = true

SUMOwithOMNeT.numVehicle = 10
*.V[*].mobility.typename = "SUMOMobility"

#"haneda.sumocfg"
#"odaiba.sumocfg"
#"shinjuku.sumocfg"
#"erlangen.sumocfg"
*.sumomap.map = "shinjuku.sumocfg"
```

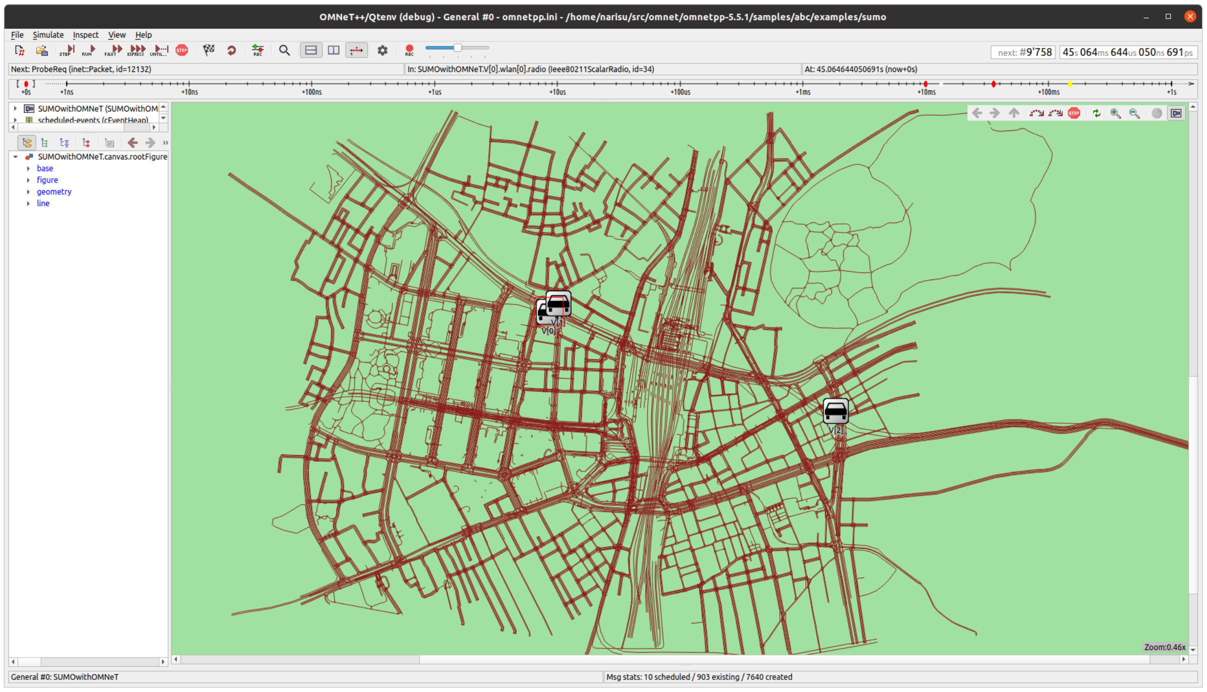
```
sumo (3) [OMNeT++ Simulation]
Warning: At actuated tlLogic 'cluster_1111613987_1111613998_2403065895_2403065897_2403065903_2403065904_2403065905_2403065906_2503429961_2503429962_2503429963_2503429964', linkIndex 16 h
Warning: At actuated tlLogic 'cluster_1701041432_1701041435_2503429945_2503429946_295422678_295422762', actuated phase 3 has no controlling detector.
Warning: At actuated tlLogic 'cluster_1716064635_1716067483_295416486_5498668185_5498668186_5498668189_5498668190_5498668193_5498668194_582802728', actuated phase 3 has no controlling de
Warning: At actuated tlLogic 'cluster_304610049_304611116_6282956270_6282956271', linkIndex 2 has no controlling detector.
Warning: At actuated tlLogic 'cluster_304610049_304611116_6282956270_6282956271', linkIndex 3 has no controlling detector.
Warning: At actuated tlLogic 'cluster_304610049_304611116_6282956270_6282956271', linkIndex 4 has no controlling detector.
Warning: At actuated tlLogic 'joinedS_0', linkIndex 50 has no controlling detector.
Warning: At actuated tlLogic 'joinedS_0', linkIndex 51 has no controlling detector.
Warning: At actuated tlLogic 'joinedS_0', linkIndex 52 has no controlling detector.
Warning: At actuated tlLogic 'joinedS_2', linkIndex 27 has no controlling detector.
Warning: At actuated tlLogic 'joinedS_9', linkIndex 6 has no controlling detector.
Warning: At actuated tlLogic 'joinedS_9', linkIndex 7 has no controlling detector.
Warning: No connection between edge '59285321#8' and edge ':472383086_10' found.
Warning: No connection between edge ':472359807_8' and edge ':6240814975_0' found.
Add vehicle V0
Warning: No connection between edge '-139633245#0' and edge ':472382804_18' found.
Warning: No connection between edge ':471978536_18' and edge ':cluster_1678008833_5500147169_1' found.
Warning: No connection between edge '39433561#0' and edge ':cluster_472360172_6237227449_9' found.
Warning: No connection between edge ':471978567_12' and edge ':cluster_1582702939_1582702940_12' found.
Warning: No connection between edge ':7954421321_7' and edge ':475764529_0' found.
Warning: No connection between edge '39433595#0' and edge ':472382919_1' found.
Warning: No connection between edge ':475764299_2' and edge ':471978612_0' found.
Add vehicle V1
Add vehicle V2
```

Appearing these errors is the normal.

**Figure 17:** *Step 17.*

The errors/warnings occurred in the IDE console is normal. Because the departure and destination is generated randomly when adding a vehicle. So, the route validation is necessary. The errors/warning is appared when a route is not reachable from the departure to the destination. At this time, the code reroutes for the vehicle until the route is reachable.

Qtenv is showed in Figure 18 and using map is shinjuku in Japan.


OMNeT++/Qtenv (debug) - General #0 - omnetpp.ini - /home/narisu/src/omnet/omnetpp-5.5.1/samples/abc/examples/sumo

Next: ProbeReq (inet::Packet, id=12132)
In: SUMOwithOMNeT.V[0].wlan[0].radio (Ieee80211ScalarRadio, id=34)
At: 45.064644050691s (now+0s)

next: #9'758
45s 064ms 644us 050ns 691ps

General #0: SUMOwithOMNeT
Msg stats: 10 scheduled / 903 existing / 7640 created

**Figure 18:** *Step 18.*