

Autoware ユーザーズマニュアル

2015/AUG/24

名古屋大学

内容

はじめに	4
概要	4
用語	4
関連文書	5
全体構成	7
構成	7
主な機能	8
環境構築の手順	10
Linux	10
ROS	10
Velodyne ドライバ	11
OpenCV	11
Qt	11
CUDA	12
FlyCapture2	13
Autoware	14
AutowareRider	14
canlib	15
SSH の公開鍵の作成	15
使用手順	16
準備	16
runtime manager の起動	17
ポイントクラウドおよびベクタ地図のロード	17
ドライバのロード	17
位置認識 (NDT)	17
1. Simulation タブの Clock と Sim Time のチェックをつけた上で、Map ボタンを押して ポイントクラウドおよびベクタ地図のロードを行います。Sim Time を有効にしているため この時点では地図は表示されません。(既に Map ボタンを押して起動状態になっている 場合は一旦 Map ボタンを押して終了 -> 再度 Map ボタンを押す、として起動しなおして 下さい)	18
4. Simulation タブのファイル選択ダイアログに実行する rosbag ファイルを指定 し、"Play" ボタンを押します。rosbag のプレイが開始すると地図が表示されます。NDT が実行されはじめるとそれらも表示されます。表示されない場合は rviz の Reset ボタン を押す、または、Displays の一覧にある「Points Map」と「Vector Map」のチェックを 外す -> 再びチェックを入れる、などの操作をして下さい。	18
物体検出	18

経路計画	19
軌跡計画	20
ダイナミックマップ	21
AutowareRider	23
概要	23
起動方法	23
経路データ生成アプリケーションの使用方法	24
ROS PC への経路データ転送手順	25
CAN データ収集アプリケーションの使用方法	25
ROS PC への CAN データ転送手順	26
Launch ファイルの起動方法	26
各機能の説明	27
Runtime Manager	27
概要	27
Quick Start タブ	28
Map タブ	33
Sensing タブ	36
Computing タブ	39
Interface タブ	47
Database タブ	49
Simulation タブ	52
Status タブ	54
Topics タブ	56
ユーザインタフェース	57
概要	57
AutowareRider	58
AutowareRoute	62

はじめに

概要

この文書は、Linux と ROS(Robot OS)をベースとした、自動運転を実現するためのオープンソースのソフトウェアパッケージ「Autoware」のユーザーズマニュアルです。

Autoware と、各種センサ機器もしくはデータを使用して、自動運転もしくはその一部の機能を動作させる手順について記述しています。

用語

- ROS (Robot Operating System)

ロボットソフトウェア開発のためのソフトウェアフレームワーク。ハードウェア抽象化や低レベルデバイス制御、よく使われる機能の実装、プロセス間通信、パッケージ管理などの機能を提供する。

- パッケージ (Package)

ROS を形成するソフトウェアの単位。ノードやライブラリ、環境設定ファイルなどを含む。

- ノード (Node)

単一の機能を提供するプロセス。

- メッセージ (Message)

ノード同士が通信する際のデータ構造。

- トピック (Topic)

メッセージを送受信する先。メッセージの送信を「Publish」、受信を「Subscribe」と呼ぶ。

- OpenCV (Open source Computer Vision library)

コンピュータビジョンを扱うための画像処理ライブラリ。

- Qt

アプリケーション・ユーザ・インタフェースのフレームワーク。

- CUDA (Compute Unified Device Architecture)

NVIDIA 社が提供する、GPU を使った汎用計算プラットフォームとプログラミングモデル。

- FlyCapture SDK

PointGrey 社のカメラを制御するための SDK。

- FOT (Field Operation Test)

実道実験。

- GNSS (Global Navigation Satellite System)
衛星測位システム。
- LIDAR (Light Detection and Ranging または Laser Imaging Detection and Ranging)
レーザー照射を利用して距離などを計測する装置。
- DPM (Deformable Part Model)
物体検出手法。
- KF (Kalman Filter)
過去の観測値をもとに将来の状態を推定する手法。
- NDT (Normal Distributions Transform)
位置推定手法。
- キャリブレーション

カメラに投影された点と 3 次元空間中の位置を合わせるための、カメラのパラメータを求める処理。

- センサ・フュージョン
複数のセンサ情報を組合せて、位置や姿勢をより正確に算出するなど、高度な認識機能を実現する手法。
- TF (TransForm?)
ROS の座標変換ライブラリ?
- オドメトリ(Odometry)
車輪の回転角と回転角速度を積算して位置を推定する手法。
- SLAM(Simultaneous Localization and Mapping)
自己位置推定と環境地図作成を同時に行うこと。
- CAN(Controller Area Network)
自動車等の内部で相互接続された機器間のデータ転送に使用される規格。
- IMU(Inertial Measurement Unit)
慣性計測装置。角速度や加速度を計測する装置。
- DMI(Distance Measuring Instrument)
走行距離計。

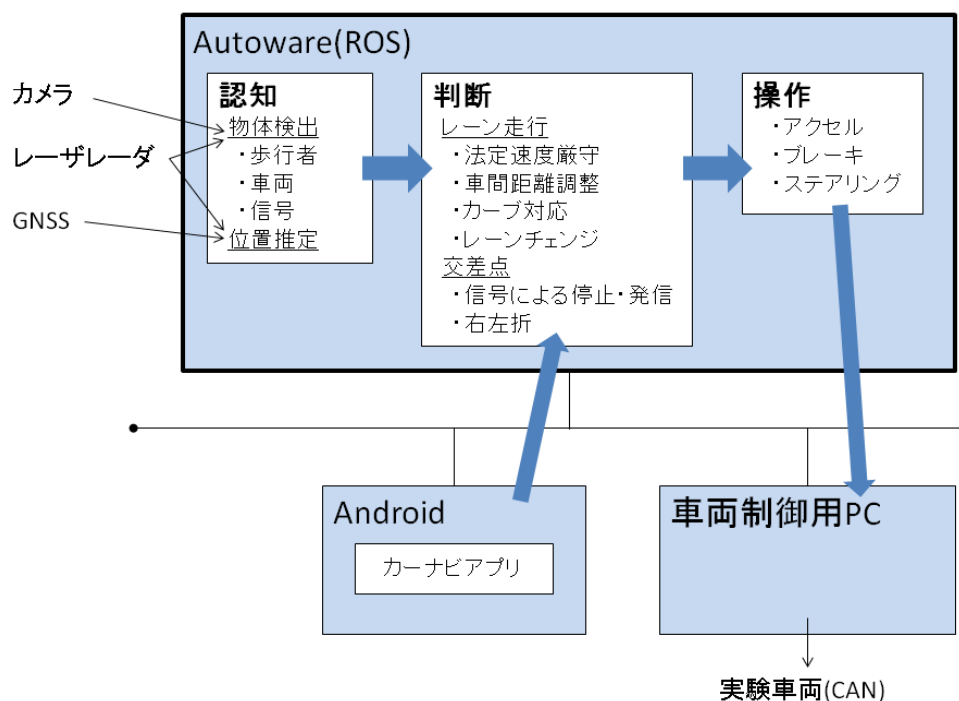
関連文書

- Autoware
<http://www.pdsl.jp/fot/autoware/>
- ROS
<http://www.ros.org/>
- OpenCV
<http://opencv.org/>
<http://opencv.jp/>
- Qt
<http://www.qt.io/>
<http://qt-users.jp/>

- CUDA
http://www.nvidia.com/object/cuda_home_new.html
<http://www.nvidia.co.jp/object/cuda-jp.html>
- FlyCapture SDK
<http://www.ptgrey.com/flycapture-sdk>

全体構成

Autoware は、Linux と ROS をベースとした、自動運転を実現するためのオープンソースのソフトウェアパッケージです。レーザレーダ、カメラ、GNSS などの環境センサを使用して、自車位置や周囲物体を認識しながら、カーナビから与えられたルート上を自立走行することができます。



構成

Autoware による自動運転の機能は、自己位置推定や周囲物体の検出などを行う「認知」、レーンや交差点での走行・停止の「判断」、実際の車両の「操作」、の3つに分けられます。

- `ros/src/computing/perception/`
認知・判断
- `ros/src/computing/planning/`
判断・操作
- `ros/src/data/`
3次元地図などのデータの読み込み(DB、ファイル)
- `ros/src/sensing/`
各種センサドライバ、キャリブレーション、フュージョンなど
- `ros/src/socket/`

スマートフォン用アプリケーションとのインタフェース

- ros/src/util/
Runtime Manager、サンプルデータ、擬似ドライバなど
- ui/tablet/
スマートフォン用アプリケーション
- vehicle/
車両の制御、情報取得など

主な機能

Autoware には以下のような機能があります。

また、これらを実施するためのユーザインタフェース(Runtime Manager)も用意されています。

- 自己位置推定

3次元点群地図と3次元LIDARデータを入力として、NDTアルゴリズムをベースとしたスキャンマッチングを行うことで、自車位置を10cm程度の誤差で推定することができます。

- 3次元地図生成

SLAM技術を用いて、3次元地図をリアルタイムに生成することができます。

生成した3次元地図を、既存の3次元地図に追加することも可能です。この機能により、3次元地図のオンライン更新も実現できます。

3次元地図から地物データを抽出することで、ベクタ形式の3次元地図を生成することもできます。

- 信号機検出

自己位置推定の結果と高精度3次元地図から、信号機の位置を正確に算出し、信号機の3次元位置をセンサフュージョンによってカメラ画像上に射影します。そこから画像処理によって色判別することで、信号機を検出することができます。

- 物体検出

カメラ画像を入力として、DPMアルゴリズムによる画像認識を行うことで、車両や歩行者を検出することができます。

KFを利用してトラッキングを行うことも可能です。トラッキング機能を導入すると、個々の物体を追跡でき、かつ誤認識を削減できます。

また、3次元LIDARデータをフュージョンすることで、検出した物体までの距離も算出できます。

- 経路生成

自動運転の経路は、スマートフォンのカーナビアプリケーション (MapFan を使用した経路データ生成アプリケーション) から入力できます。経路には適切な速度情報も含まれ、その速度を目安に自立走行します。

- 経路追従

生成した経路に 1m 間隔の目印 (way point) を設定し、その目印を追っていくことで経路追従を行います。カーブでは近くの way point、直線では遠くの way point を参照することで、自立走行を安定化しています。

経路から逸脱した場合は、近傍の way point を目指して経路に戻ります。

環境構築の手順

PC に、以下の手順で、Linux、ROS、Autoware などをインストールする手順を示します。

CUDA、FlyCapture SDK および canlib は、必須ではありません。

NVIDIA 社のグラフィックボードに搭載された GPU を使って計算を行う場合は、CUDA が必要です。また、PointGrey 社のカメラを使用する場合は、FlyCapture SDK が必要です。

Linux

現時点で、Autoware が対応している Linux ディストリビューションは以下の通りです。

- Ubuntu 13.04
- Ubuntu 13.10
- Ubuntu 14.04

インストールメディアおよびインストール手順については、以下のサイトを参考にしてください。

- Ubuntu Japanese Team
<https://www.ubuntulinux.jp/>
- Ubuntu
<http://www.ubuntu.com/>

ROS

- Ubuntu14.04 の場合は、下記の手順で ROS および必要なパッケージをインストールします。

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > \
/etc/apt/sources.list.d/ros-latest.list'
$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install ros-indigo-desktop-full ros-indigo-nmea-msgs \
ros-indigo-sound-play
$ sudo apt-get install libnlopt-dev freeglut3-dev qtbase5-dev libqt5opengl5-dev \
libssh2-1-dev
```

- Ubuntu13.10 もしくは 13.04 の場合は、下記の手順で ROS および必要なパッケージをインストールします。

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > \
/etc/apt/sources.list.d/ros-latest.list'
$ sudo apt-get install ros-hydro-desktop-full ros-indigo-nmea-msgs \
```

```
ros-hydro-sound-play
$ sudo apt-get install libnlopt-dev freeglut3-dev libssh2-1-dev
```

- (いずれの場合も) ~/.bashrc などに以下を追加します。
[-f /opt/ros/indigo/setup.bash] && . /opt/ros/indigo/setup.bash

Velodyne ドライバ

<https://github.com/ros-drivers/velodyne> からソースコードを入手し、以下の手順でインストールを行います。

```
$ sudo apt-get install libpcap-dev
$ mkdir -p ~/ros_drivers/src
$ cd ~/ros_drivers/src
$ catkin_init_workspace
$ git clone https://github.com/ros-drivers/velodyne.git
$ cd ~/ros_drivers
$ catkin_make
$ source devel/setup.bash
```

OpenCV

OpenCV のサイト(<http://sourceforge.net/projects/opencvlibrary/>)から、バージョン 2.4.8 以降のソースコードを入手し、以下の手順でインストールを行います。

```
$ unzip opencv-2.4.8.zip
$ cd opencv-2.4.8
$ cmake .
$ make
$ sudo make install
```

Qt

1. まず、Qt5 に必要なパッケージを、以下の手順でインストールします。

```
$ sudo apt-get build-dep qt5-default
$ sudo apt-get install build-essential perl python git
$ sudo apt-get install "^libxcb.*" libx11-xcb-dev libglu1-mesa-dev \
libxrender-dev libxi-dev
$ sudo apt-get install flex bison gperf libicu-dev libxslt-dev ruby
$ sudo apt-get install libssl-dev libxcursor-dev libxcomposite-dev libxdamage-dev \
libxrandr-dev libfontconfig1-dev
$ sudo apt-get install libasound2-dev libgstreamer0.10-dev \
libgstreamer-plugins-base0.10-dev
```

- 次に、Qt5 のソースコードを入手して、ビルドおよびインストールを行います。

```
$ git clone git://code.qt.io/qt/qt5.git
$ cd qt5/
$ git checkout v5.2.1
$ perl init-repository --no-webkit
  (webkit は大きいため、--no-webkit を指定しています)
$ ./configure -developer-build -opensource -nomake examples -nomake tests
  (ライセンスを受諾する必要があります)
$ make -j
  (ビルドには数時間かかります)
$ make install
$ sudo cp -r qtbase /usr/local/qtbase5
```

CUDA

<http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-linux/> を参考に、以下の手順でインストールします。

- 環境の確認

```
$ lspci | grep -i nvidia
(NVIDIA のボードの情報が出力されることを確認)

$ uname -m
(x86_64 であることを確認)

$ gcc --version
(インストールされていることを確認)
```

- CUDA のインストール

<http://developer.nvidia.com/cuda-downloads> から CUDA をダウンロード

```
(以下、cuda-repo-ubuntu1404_7.0-28_amd64.deb と想定)
$ sudo dpkg -i cuda-repo-ubuntu1404_7.0-28_amd64.deb
$ sudo apt-get update
$ sudo apt-get install cuda
```

- システムを再起動 (...は不要かもしれませんが)

```
$ lsmod | grep nouveau
(nouveau ドライバがロードされていないことを確認)
```

- 確認

```
$ cat /proc/driver/nvidia/version
```

(カーネルモジュール、gcc のバージョンが表示される)

```
$ cuda-install-samples-7.0.sh ~  
$ cd ~/NVIDIA_CUDA-7.0_Samples/1_Uutilities/deviceQuery/  
$ make  
$ ./deviceQuery
```

5. CUDA を普段から使う場合は、以下の設定を .bashrc などを書く

```
export PATH="/usr/local/cuda:$PATH"  
export LD_LIBRARY_PATH="/usr/local/cuda/lib:$LD_LIBRARY_PATH"
```

FlyCapture2

PointGray 社のカメラを使用する場合は、以下の手順で FlyCapture SDK をインストールします。

2014 年 10 月 28 日に試したときの手順

/radisk2/work/usuda/autoware/doc/MultiCameraEclipse-log-20141028.txt

1. PointGrey 社のサイト (<http://www.ptgrey.com/>) から、FlyCapture SDK をダウンロードします。(ユーザ登録が必要です。)

2. 以下の手順で、事前にパッケージをインストールします。

```
$ sudo apt-get install libglademm-2.4-1c2a libgtkglextmm-x11-1.2-dev libserial-dev
```

3. ダウンロードしたアーカイブを展開します。

```
$ tar xvfz flycapture2-2.6.3.4-amd64-pkg.tgz
```

4. インストーラを起動します。

```
$ cd flycapture2-2.6.3.4-amd64/  
$ sudo sh install_flycapture.sh
```

This is a script to assist with installation of the FlyCapture2 SDK.

Would you like to continue and install all the FlyCapture2 SDK packages?

(y/n)\$ y ← 「y」 と答えます

...

Preparing to unpack updatorgui-2.6.3.4_amd64.deb ...

Unpacking updatorgui (2.6.3.4) ...

updatorgui (2.6.3.4) を設定しています ...

Processing triggers for man-db (2.6.7.1-1ubuntu1) ...

Would you like to add a udev entry to allow access to IEEE-1394 and USB

hardware?

If this is not ran then your cameras may be only accessible by running flycap as sudo.

(y/n)\$ y ← 「y」 と答えます

Autoware

以下の手順で Autoware を入手し、ビルドおよびインストールを行います。

- github から最新を入手する場合

```
$ git clone https://github.com/CPFL/Autoware.git
```

```
$ cd Autoware/ros/src
```

```
$ catkin_init_workspace
```

```
$ cd ../
```

```
$ ./catkin_make_release
```

- アーカイブを使用する場合

```
$ wget http://www.pdsl.jp/app/download/10394444574/Autoware-beta.zip
```

```
$ unzip Autoware-beta.zip
```

```
$ cd Autoware-beta/ros/src
```

```
$ catkin_init_workspace
```

```
$ cd ../
```

```
$ ./catkin_make_release
```

AutowareRider

以下の URL から APK ファイルを入手し、インストールを行います。

- 本体
 - AutowareRider.apk
<https://github.com/CPFL/Autoware/blob/master/ui/tablet/AutowareRider/AutowareRider.apk>
- 経路データ生成アプリケーション
 - AutowareRoute.apk
<https://github.com/CPFL/Autoware/blob/master/ui/tablet/AutowareRoute/AutowareRoute.apk>
- CAN データ収集アプリケーション
 - CanDataSender.apk
<https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CanDataSender/bin/CanDataSender.apk>
 - CanGather.apk
<https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CanGather/apk/CanGather.apk>
 - CarLink_CAN-BT_LS.apk
https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CarLink/apk/CarLink_CAN-BT_LS.apk

- CarLink_CANusbAccessory_LS.apk
https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CarLink/apk/CarLink_CANusbAccessory_LS.apk

CanGather は APK ファイル以外に、設定ファイルを用意する必要があります。

詳細は、以下の URL を参考にしてください。

<https://github.com/CPFL/Autoware/tree/master/vehicle/general/android#cangather-%E3%81%AE%E5%A0%B4%E5%90%88>

canlib

kvaser のサイト(<http://www.kvaser.com/downloads/>) の "Kvaser LINUX Driver and SDK" よりソースコード linuxcan.tar.gz を入手し、以下の手順でインストールを行います。

```
$ tar xzf linuxcan.tar.gz
$ cd linuxcan
$ make
$ sudo make install
```

SSH の公開鍵の作成

pos_db は、SSH を介してデータベースにアクセスします。その際、パスフレーズなしの SSH 鍵を使用します。

そのため、pos_db を使用する場合は、データベースサーバ用の SSH 鍵を以下の手順で作成し、SSH 公開鍵をデータベースサーバに登録する必要があります。

1. SSH 鍵の作成方法
 - 以下のコマンドを実行して鍵を作成します。
 - \$ ssh-keygen -t rsa
 - その際、パスフレーズは空(文字列を入力せずにエンターキーを押す)にして作成してください。
 - DSA を使用する場合は -t dsa と指定してください。
2. SSH 公開鍵をデータベースサーバに登録する
 - 作成した SSH 公開鍵を以下のコマンドでサーバーにコピーします。
 - \$ ssh-copy-id -i ~/.ssh/id_rsa.pub posup@db3.ertl.jp(posup はユーザ名、db3.ertl.jp はデータベースサーバ名)
 - その際にパスワードを聞かれるので適宜入力してください。

使用手順

準備

必要なデータ一式の入ったディレクトリが `~/.autoware/data` にあるという前提で説明します。

サンプルデータは以下のところよりダウンロードして下さい。

デモ用の launch ファイルを生成するスクリプト

http://db3.ertl.jp/autoware/sample_data/my_launch.sh

デモで使うデータ(守山地区の地図・キャリブレーション・経路)

http://db3.ertl.jp/autoware/sample_data/sample_moriyama_data.tar.gz

ROSBAG データ

http://db3.ertl.jp/autoware/sample_data/sample_moriyama_150324.tar.gz

ダウンロードしたデモのデータを `~/.autoware/` 以下に展開します。

```
$ tar xfz sample_moriyama_data.tar.gz -C ~/.autoware/
```

以下のスクリプトを実行して、Qtick Start タブからデモを実行するための launch ファイルを生成します。

```
$ sh my_launch.sh
```

実行すると、以下の launch ファイルが生成されます。

```
my_launch/
  my_map.launch           # 地図のロード
  my_sensing.launch       # ドライバのロード
  my_localization.launch  # 位置認識
  my_detection.launch     # 物体検出
  my_mission_planning.launch # 経路計画
  my_motion_planning.launch # 軌跡計画
```

データを `~/.autoware/data` 以外の場所に配置する場合は、シェルスクリプトを実行する際に、引数にデータを配置したディレクトリの指定を行って下さい。

例)

~/.autoware/data/quick_start/rosbag_sample/ にデータを配置した場合

```
$ sh my_launch.sh ~/autoware/data/quick_start/rosbag_sample/
```

と実行して下さい。

runtime manager の起動

1. ROS PC で Runtime Manager を起動します。
2. runtime manager の右下の "Rviz" ボタンを押して Rviz を起動します。
3. Rviz のファイルメニューから「Open Config」を選択し、
Autoware/ros/src/.config/rviz/default.rviz を「開く」を行います。

ポイントクラウドおよびベクタ地図のロード

1. Quick Start タブの Map のファイル選択ダイアログに準備のところで生成した my_map.launch を指定して "Map" ボタンを押します。
2. 地図のロードが終わったら "Ref" ボタンの右側に 「OK」と表示されます。

ドライバのロード

1. Quick Start タブの Sensing のファイル選択ダイアログに準備のところで生成した my_sensing.launch を指定して "Sensing" ボタンを押します。

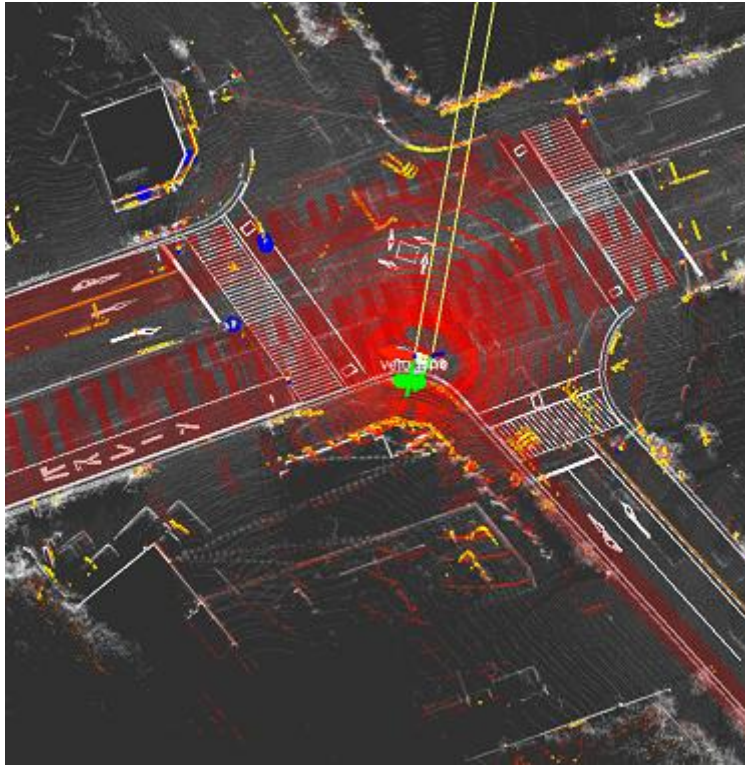
位置認識 (NDT)

roslaunch を使用する場合の手順を説明します

1. Simulation タブの Clock と Sim Time のチェックをつけた上で、Map ボタンを押してポイントクラウドおよびベクタ地図のロードを行います。Sim Time を有効にしているためこの時点では地図は表示されません。(既に Map ボタンを押して起動状態になっている場合は一旦 Map ボタンを押して終了 -> 再度 Map ボタンを押す、として起動しなおして下さい)
2. Computing タブの mdt_matching のリンクをクリックしてダイアログを開き、GNSS にチェックが入っている事を確認した上で OK ボタンを押します。
3. 地図のロードが Quick Start タブの Localization のファイル選択ダイアログに準備のところで生成した my_localization.launch を指定して "Localization" ボタンを押します。
4. Simulation タブのファイル選択ダイアログに実行する rosbag ファイルを指定し、"Play" ボタンを押します。rosbag のプレイが開始すると地図が表示されます。NDT が実行されはじめるとそれらも表示されます。表示されない場合は rviz の Reset ボタンを押す、または、Displays の一覧にある「Points Map」と「Vector Map」のチェックを外す -> 再びチェックを入れる、などの操作をして下さい。
5. NDT の結果が GPS の矢印に追従しない場合は、rviz の上部にある "2D Pose Estimate" をクリックして GPS の矢印の付近にカーソルを合わせてクリックして下さい。

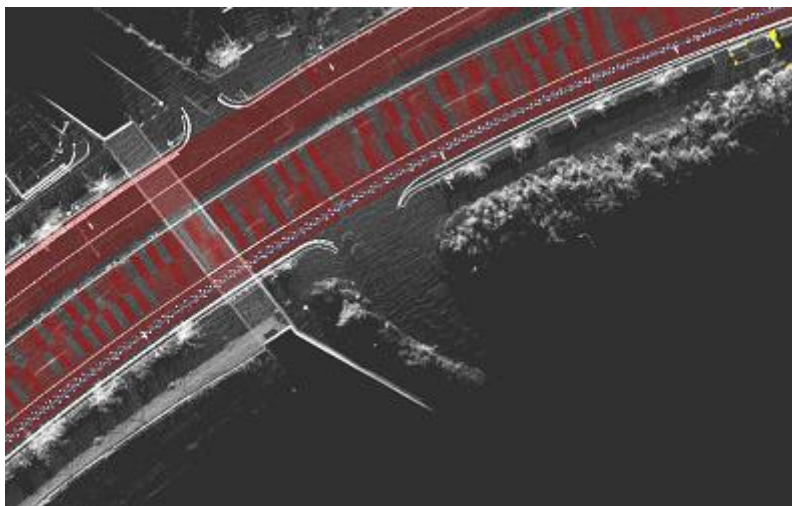
物体検出

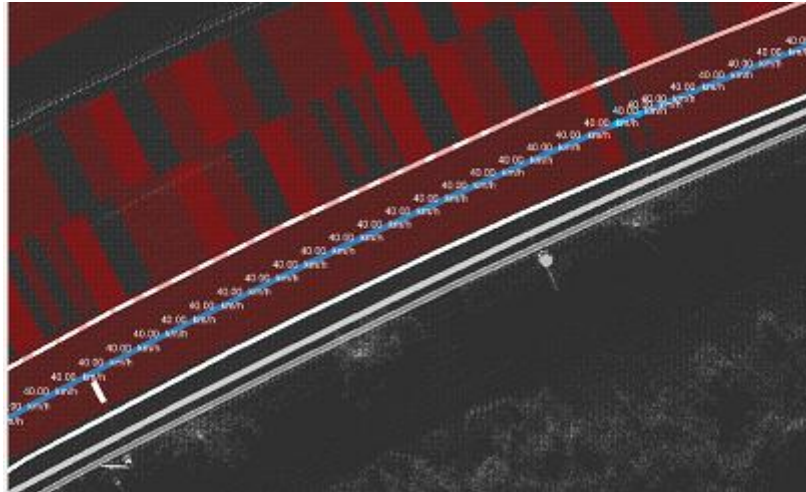
1. Quick Start タブの のファイル選択ダイアログに準備のところで生成した my_detection.launch を指定して "Detection" ボタンを押します
2. NDT 実行中に物体検出を行って成功すると、車両は青い球、歩行者は緑の球で表示されます。



経路計画

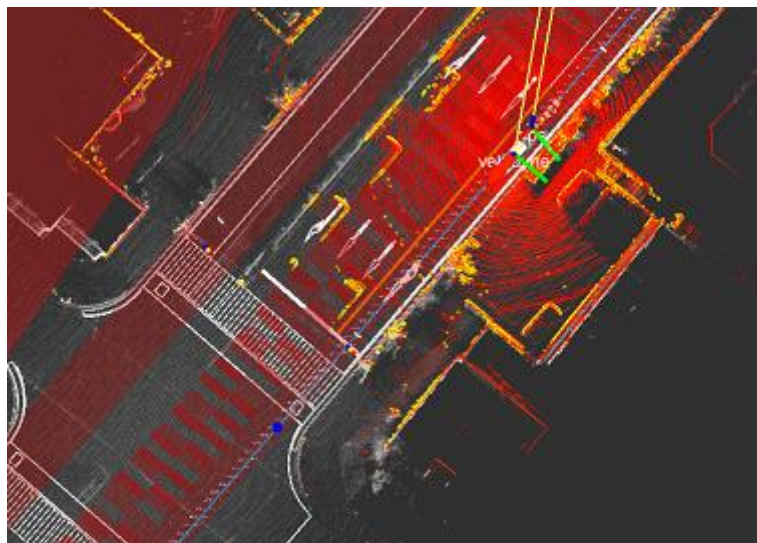
1. Qutick Start タブの Mission Planning のファイル選択ダイアログに準備のところで生成した my_mission_planning.launch を指定して “Mission Planning” ボタンを押します
2. 実行すると、青い線でパスと、速度が表示されます

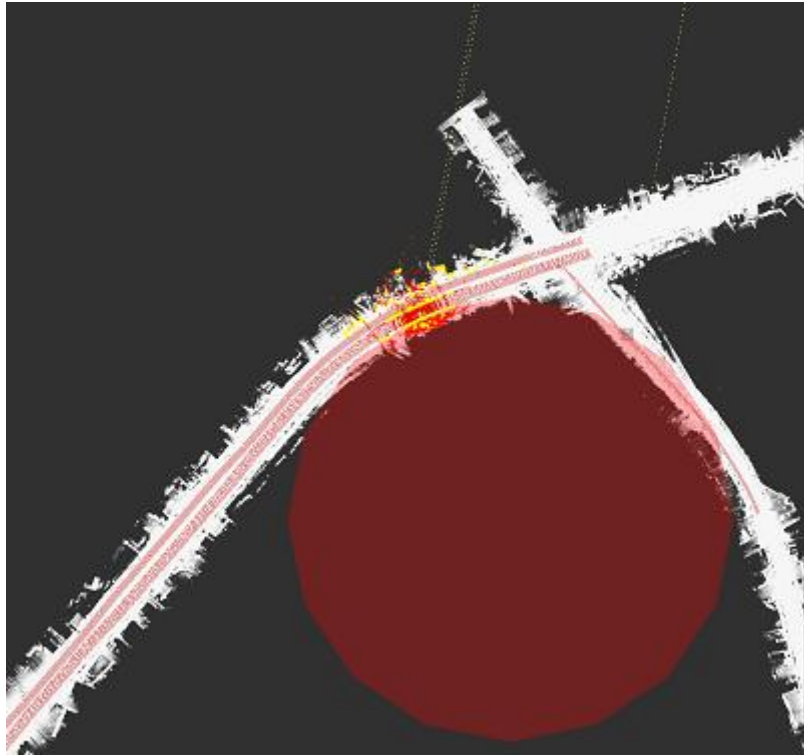




軌跡計画

1. Computing タブの pure pursuit のリンクをクリックしてダイアログを開き、“Waypoint” にチェックが入っている事を確認した上で OK ボタンを押します
2. Qutick Start タブの Mission Planning のファイル選択ダイアログに準備のところで生成した my_motion_planning.launch を指定して “Motion Planning” ボタンを押します
3. 経路計画で設定したパスが表示されているところまで来ると、パス上に青い球、Pure Pursuit により計算される赤い円が表示されます





ダイナミックマップ

自車や他車が認識した車および人の情報を、データベースを使って共有します。

1. 以下のトピックが配信されている状態にします。

すべて必須というわけではなく、以下が配信されていれば、その情報をデータベースに登録します。

ndt_matching (自車, ndt_matching が配信)

obj_car_pose (他車, obj_fusion が配信)

obj_person_pose (人, obj_fusion が配信)

2. (情報を提供する側で) Database タブの pos_uploader のリンクをクリックし、データベースサーバに SSH でアクセスするための情報を入力し、OK ボタンを押します。
(SSH 鍵の生成手順は、環境構築の手順にあります。)

pos_db

User Name: posup ☐ now

DB Server Name db3.ertl.jp DB Port 5678

SSH Public Key /home/posup/.ssh/id_rsa.pub Ref

SSH Private Key /home/posup/.ssh/id_rsa Ref

SSH Port 22 SSH Tunnel Host localhost

OK Cancel

3. (情報を提供する側で) pos_uploader にチェックします(ノードを起動します)。
4. (情報を閲覧する側で) pos_downloader のリンクをクリックし、データベースサーバに SSH でアクセスするための情報を入力し、OK ボタンを押します。
(show my pose にチェックすると、自車の位置を配信します。)

pos_db

User Name: posup ☐ now

DB Server Name db3.ertl.jp DB Port 5678

SSH Public Key /home/posup/.ssh/id_rsa.pub Ref

SSH Private Key /home/posup/.ssh/id_rsa Ref

SSH Port 22 SSH Tunnel Host localhost

OK Cancel

5. (情報を閲覧する側で) pos_downloader にチェックします。
6. (情報を閲覧する側で) rviz で、トピック /mo_marker の Marker を追加すると、認識した自車、他社および人が表示されます。

AutowareRider

概要

AutowareRider は、ROS PC で動作する Autoware をタブレット端末から操作するための、Knight Rider に似た UI を持った、Android アプリケーションです。

AutowareRoute は、MapFan SDK で実装された、経路データ生成のための Android アプリケーションです。

AutowareRider は、以下の機能を提供します。

- AutowareRoute で生成した経路データを ROS PC へ送信
- CAN データ収集アプリケーションを起動
- ボタン操作で ROS PC の Launch ファイルを起動
- ROS PC から受信した CAN データを UI へ反映

ここでは、これらの機能の使用手順を説明します。

起動方法

1. ROS PC で Runtime Manager を起動します。
2. Main タブ[Network Connection] - [Tablet UI]の Active ボタンを押下し、以下を起動します。
 - tablet_receiver
 - tablet_sender
3. Computing タブ[Planning] - [Path]の各アンカーから、以下を設定します。
 - lane_navi
 - vector_map_directory
高精度地図が格納されたディレクトリ
 - lane_rule
 - vector_map_directory
高精度地図が格納されたディレクトリ
 - ruled_waypoint_csv
waypoint が保存されるファイル
 - Velocity
速度（単位: km/h、初期値: 40、範囲: 0~200）

- Difference around Signal
信号の前後で加減速する速度（単位：km/h、初期値：2、範囲：0～20）
 - lane_stop
 - Red Light
赤信号時の速度へ切り替え
 - Green Light
青信号時の速度へ切り替え
- 4. Computing タブ[Planning] - [Path]のチェックボックスを有効にし、以下を起動します。
 - lane_navi
 - lane_rule
 - lane_stop
- 5. Android タブレットのアプリケーション一覧画面から AutowareRider を起動します。
- 6. [右上メニュー]→[設定]から、以下を設定します。
 - ROS PC
 - IP アドレス
ROS PC IPv4 アドレス
 - 命令ポート番号
tablet_receiver ポート番号（初期値：5666）
 - 情報ポート番号
tablet_sender ポート番号（初期値：5777）
- 7. [OK]を押下し、ROS PC へ接続を試みます。
 - このとき設定はファイルに自動的に保存され、次の起動からは保存された設定で接続を試みます。
- 8. 画面中央のバーの色が、明るい赤で表示されている場合は接続に成功しています。
 - バーの色と接続の状態

バーの色	接続の状態
暗い赤	ROS PC 未接続
明るい赤	ROS PC 接続
明るい青	自動運転（mode_info: 1）
明るい黄	異常発生（error_info: 1）

経路データ生成アプリケーションの使用方法

1. AutowareRider の NAVI ボタンを押下し、経路検索を起動します。

2. 地図を長押しして、以下を順番に実行します。
 - 出発地に設定
 - 目的地に設定
 - ルート探索実行
3. ルート探索の実行後に経路検索を終了することで、ROS PC へ経路データが転送されます。
 - このとき経路データはファイルに自動的に保存され、次回からはルート探索を省略して経路データを転送できます。
4. 転送後は、再び AutowareRider へ画面が戻ります。

ROS PC への経路データ転送手順

上記の経路データ生成アプリケーションの使用方法 手順 3. を参照してください。

CAN データ収集アプリケーションの使用方法

1. AutowareRider の[右上メニュー]→[設定]から、以下を設定します。

これらの設定は AutowareRider から起動された、CanDataSender が使用します。

 - データ収集
 - テーブル名
データ転送先 テーブル名
 - SSH
 - ホスト名
SSH 接続先 ホスト名
 - ポート番号
SSH 接続先 ポート番号 (初期値: 22)
 - ユーザ名
SSH でログインするユーザ名
 - パスワード
SSH でログインするパスワード
 - ポートフォワーディング
 - ローカルポート番号
ローカルマシンの転送元ポート番号 (初期値: 5558)

- リモートホスト名
リモートマシン ホスト名 (初期値: 127.0.0.1)
- リモートポート番号
リモートマシンの転送先ポート番号 (初期値: 5555)

2. [OK]を押下することで、設定がファイルに保存されます。
 - ただし、SSH のパスワードはファイルに保存しません。AutowareRider を起動している間だけ、メモリにのみ保持しています。
3. [右上メニュー]→[データ収集]から、以下のいずれかを起動します。
 - CanGather
 - CarLink (Bluetooth)
 - CarLink (USB)
4. アプリケーション起動後の使用方法は、それぞれを単独で起動した場合と同様です。
 - 詳細は、以下の URL を参考にしてください。
<https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/README.md>

ROS PC への CAN データ転送手順

上記の CAN データ収集アプリケーションの使用方法 手順 4. を参照してください。

Launch ファイルの起動方法

1. AutowareRider の S1 ボタン、S2 ボタンは、それぞれが以下の Launch ファイルに対応しています。
 - check.launch
 - set.launch
2. ボタンを押下することで、ROS PC で Launch ファイルが起動します。
 - ボタンと Launch ファイルの状態

ボタン	Launch ファイルの状態
押下 (文字色: 黒)	起動 (<code>{ndt, lf}_stat: false</code>)
押下 (文字色: 赤)	起動 (<code>{ndt, lf}_stat: true</code>)

各機能の説明

Runtime Manager

```
# ros/src/util/packages/runtime_manager
```

概要

Runtime Manager は runtime_manager パッケージに含まれる Python スクリプト (scripts/runtime_manager_dialog.py) を rosrun コマンドで起動し使用する。

```
$ rosrun runtime_manager runtime_manager_dialog.py
```

Runtime Manager を起動すると、画面にダイアログが表示される。

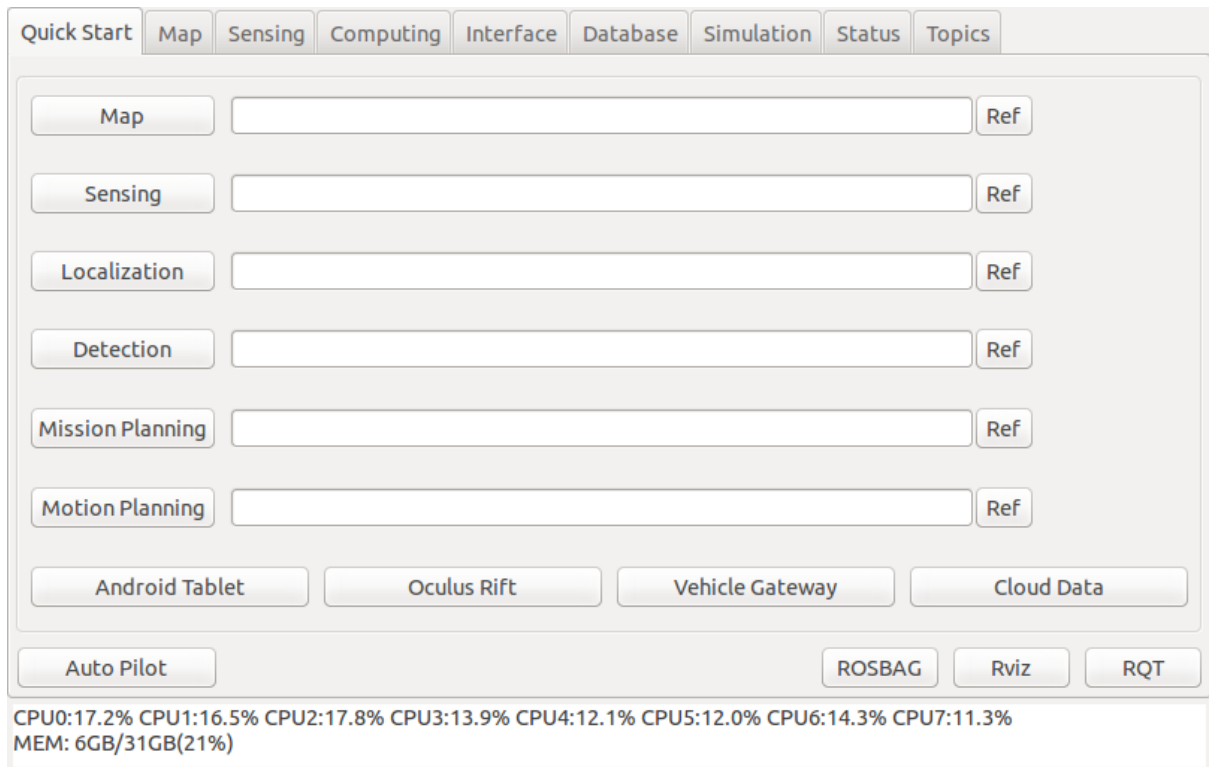
Runtime Manager のダイアログ操作により、

Autoware で使用する各種 ROS ノードの起動・終了処理や、
起動した各種 ROS ノードへのパラメータ用のトピックの発行処理などを行なう事ができる。

Runtime Manager のダイアログの画面は、複数のタブ画面で構成される。

各種 ROS ノードを起動・終了するためのボタン類は、
ノードの機能により、各タブ画面に分類・配置されている。

各タブ画面の表示は、画面上部のタブにより切替える。



Runtime Manager 起動画面

Quick Start タブ

Map トグルボタン

Map テキストボックスで指定されている.launch スクリプトを起動・終了する。

Map テキストボックス

Map トグルボタンから起動・終了させる.launch スクリプトのパスを指定する。
(フルパスで指定する)

Map Ref ボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Map テキストボックスに設定される。

Sensing トグルボタン

Sensing テキストボックスで指定されている.launch スクリプトを起動・終了する。

Sensing テキストボックス

Sensing トグルボタンから起動・終了させる.launch スクリプトのパスを指定する。
(フルパスで指定する)

Sensing Ref ボタン

ファイル選択ダイアログが表示される。
選択したファイルは、Sensing テキストボックスに設定される。

Localization トグルボタン

Localization テキストボックスで指定されている.launch スクリプトを起動・終了する。

Localization テキストボックス

Localization トグルボタンから起動・終了させる.launch スクリプトのパスを指定する。
(フルパスで指定する)

Localization Ref ボタン

ファイル選択ダイアログが表示される。
選択したファイルは、Localization テキストボックスに設定される。

Detection トグルボタン

Detection テキストボックスで指定されている.launch スクリプトを起動・終了する。

Detection テキストボックス

Detection トグルボタンから起動・終了させる.launch スクリプトのパスを指定する。
(フルパスで指定する)

Detection Ref ボタン

ファイル選択ダイアログが表示される。
選択したファイルは、Detection テキストボックスに設定される。

Mission Planning トグルボタン

Mission Planning テキストボックスで指定されている.launch スクリプトを起動・終了する。

Mission Planning テキストボックス

Mission Planning トグルボタンから起動・終了させる.launch スクリプトのパスを指定する。
(フルパスで指定する)

Mission Planning Ref ボタン

ファイル選択ダイアログが表示される。
選択したファイルは、Mission Planning テキストボックスに設定される。

Motion Planning トグルボタン

Motion Planning テキストボックスで指定されている.launch スクリプトを起動・終了する。

Motion Planning テキストボックス

Motion Planning トグルボタンから起動・終了させる.launch スクリプトのパスを指定する。
(フルパスで指定する)

Motion Planning Ref ボタン

ファイル選択ダイアログが表示される。
選択したファイルは、Motion Planning テキストボックスに設定される。

Android Tablet トグルボタン

runtime_manager/tablet_socket.launch スクリプトを
起動・終了する。

Oculus Rift トグルボタン

〈未実装〉

Vehicle Gatewat トグルボタン

runtime_manager/vehicle_socket.launch スクリプトを
起動・終了する。

Cloud Data トグルボタン

obj_db/obj_downloader ノードを起動・終了する。

Auto Pilot トグルボタン

ボタンの状態に応じた mode_cmd トピックを発行する。

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

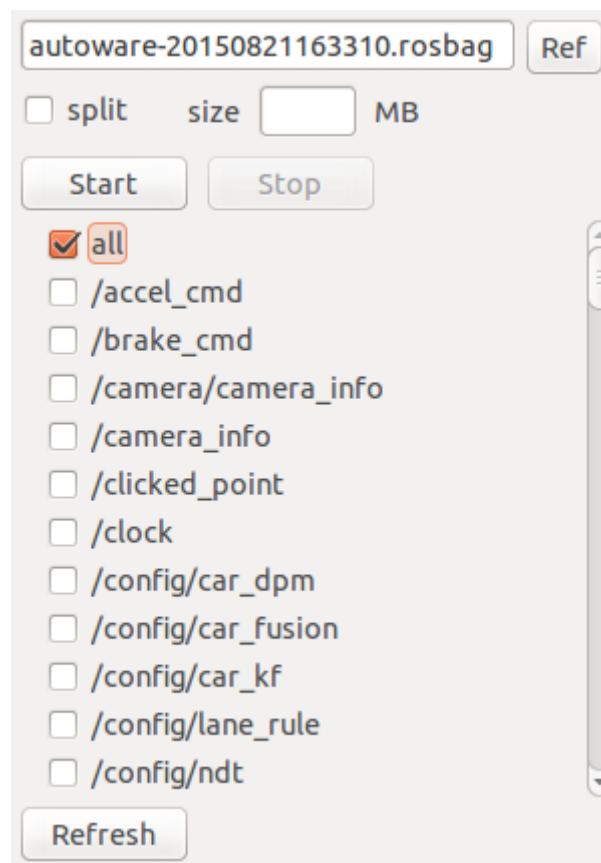
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

ROSBAG Record ダイアログ



ROSBAG Record ダイアログ

上部テキストボックス

rosvag record コマンドを実行する際の、bag ファイルを指定する。
(フルパスで指定する)

Ref ボタン

保存ファイル指定ダイアログが表示される。
指定したファイルは、上部テキストボックスに設定される。

Start ボタン

上部テキストボックスに設定された bag ファイルを指定して、
rosvag record コマンドを起動する。

Stop ボタン

起動している rosvag record コマンドを終了する。

All チェックボックス

チェックボックスが ON の場合、rosbag record コマンドを起動する際に、
-a オプションが指定される。

その他チェックボックス群

rosbag record コマンドを起動する際に、
チェックボックスが ON のトピックを指定する。
(ただし、All チェックボックスが OFF の場合のみ有効)

Refresh ボタン

rostopic list コマンドを実行し、現在有効なトピックを調べ、
その他のチェックボックス群を更新する。

最下行の情報表示

CPU(コア)の負荷状況とメモリ使用量を表示する。

Map タブ

The screenshot shows the 'Map' tab of the ROS Bag Recorder interface. At the top, there is a row of tabs: 'Quick Start', 'Map' (selected), 'Sensing', 'Computing', 'Interface', 'Database', 'Simulation', 'Status', and 'Topics'. Below the tabs, there are three main sections for map visualization:

- Point Cloud:** Includes a 'Point Cloud' button, a text input field, and a 'Ref' button.
- Area List:** Includes an 'Auto Update' checkbox, a '1x1' dropdown menu, an 'Area List' dropdown menu (currently set to 'None'), and a 'Ref' button.
- Vector Map:** Includes a 'Vector Map' button, a text input field, and a 'Ref' button.
- TF:** Includes a 'TF' button, a text input field, and a 'Ref' button.

At the bottom right, there are three buttons: 'ROSBAG', 'Rviz', and 'RQT'. At the very bottom, a status bar displays system information: 'CPU0:17.2% CPU1:16.5% CPU2:17.8% CPU3:13.9% CPU4:12.0% CPU5:12.0% CPU6:14.3% CPU7:11.3% MEM: 6GB/31GB(21%)'.

Map タブ

Point Cloud トグルボタン

map_file/points_map_loader ノードを起動・終了する。

Point Cloud テキストボックス

Point Cloud トグルボタンで map_file/points_map_loader を起動する際に引数で渡す、pcd ファイル群のパスを指定する。

(フルパスを','で区切り指定する)

Point Cloud Ref ボタン

ファイル選択ダイアログが表示される。

複数のファイルが選択可能。(ただし同一ディレクトリに限る)

選択したファイル群は、Point Cloud テキストボックスに設定される。

Auto Update チェックボックス

Point Cloud トグルボタンで map_file/points_map_loader を起動する際の、自動アップデートの有無を指定する

Auto Update メニュー

Point Cloud トグルボタンで map_file/points_map_loader を起動する際の、自動アップデート有効時の、シーン数を指定する。

(Auto Update チェックボックスで ON が指定された場合のみ有効)

Area List テキストボックス

Point Cloud トグルボタンで map_file/points_map_loader を起動する際に引数で渡す、area list ファイルのパスを指定する。

(フルパスで指定する)

Area List Ref ボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Area List テキストボックスに設定される。

Vector Map トグルボタン

map_file/vector_map_loader ノードを起動・終了する。

Vector Map テキストボックス

Vector Map トグルボタンで map_file/vector_map_loader を起動する際に引数で渡す、
csv ファイル群のパスを指定する。

(フルパスを','で区切り指定する)

Vector Map Ref ボタン

ファイル選択ダイアログが表示される。

複数のファイルが選択可能。(ただし同一ディレクトリに限る)

選択したファイル群は、Vector Map テキストボックスに設定される。

TF トグルボタン

TF テキストボックスに設定されている launch ファイルを起動・終了する。

TF テキストボックスに launch ファイルが設定されていない場合は、

次のパスの launch ファイルを起動・終了する。

~/.autoware/data/tf/tf.launch

TF テキストボックス

TF トグルボタンにより起動・終了させる launch ファイルのパスを指定する。

(フルパスで指定する)

TF Ref ボタン

ファイル選択ダイアログが表示される。

選択したファイルは、TF テキストボックスに設定される。

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

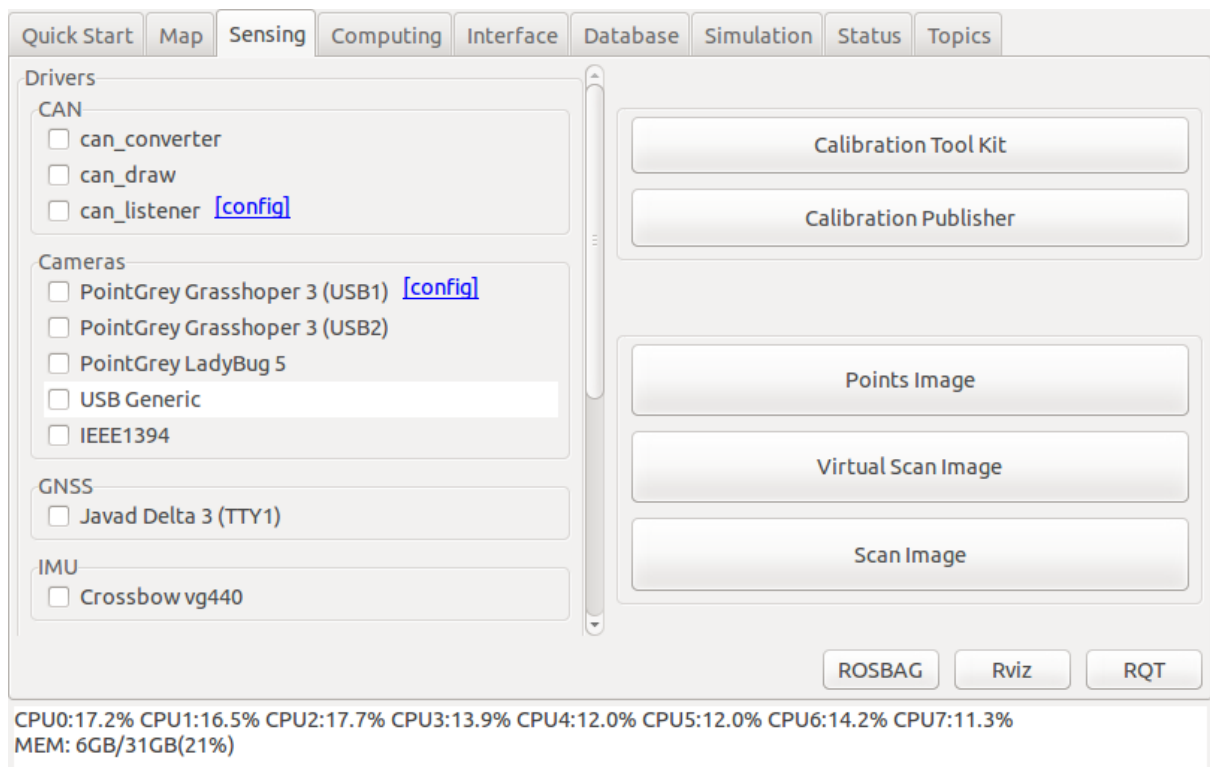
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

Sensing タブ



Sensing タブ

Drivers/CAN 欄

can_converter 項目

kvaser/can_converter ノードを起動・終了する。

can_draw 項目

kvaser/can_draw ノードを起動・終了する。

can_listener 項目

kvaser/can_listener ノードを起動・終了する。

config リンク

can_listener ダイアログを表示する。

ノード起動時に指定するチャンネルを設定する。

Drivers/Cameras 欄

PointGrey Grasshopper 3 (USB1)項目

pointgrey/grasshopper3.launch スクリプトを起動・終了する。

config リンク

calibration_path_grasshopper3 ダイアログを表示する。

スクリプト起動時に指定する CalibrationFile の path を設定する。

PointGrey Grasshopper 3 (USB2)項目

〈未実装〉

PointGray LadyBug 5 項目

〈未実装〉

USB Generic 項目

uvc_camera/uvc_camera_node ノードを起動・終了する。

IEEE1394 項目

〈未実装〉

Drivers/GNSS 欄

Javad Delta 3 (TTY1)項目

javad/gnss.sh スクリプトを起動・終了する。

Drivers/IMU 欄

Crossbow vg440 項目

〈未実装〉

Drivers/LIDARs 欄

Velodyne HDL-64e 項目

velodyne/velodyne_hdl64e.launch スクリプトを起動・終了する。

config リンク

calibration_path ダイアログを表示する。

スクリプト起動時に指定する calibration の path を設定する。

Velodyne HDL-32e 項目

velodyne/velodyne_hdl32e.launch スクリプトを起動・終了する。

config リンク

calibration_path ダイアログを表示する。

スクリプト起動時に指定する calibration_path の値を設定する。

Hokuyo TOP-URG 項目

hokuyo/top_urg スクリプト

Hokuyo 3D-URG 項目

hokuyo/hokuyo_3d ノードを起動・終了する。

SICK LMS511 項目

〈未実装〉

IBEO 8L Single 項目

〈未実装〉

Drivers/OtherSensors 欄

〈項目なし〉

Calibration Tool Kti トグルボタン

camera_lidar3d/camera_lidar3d_offline_calib ノードを起動・終了する。

Calibration Publisher トグルボタン

calibration_camera_lidar/calibrtion_publisher ノードを起動・終了する。

起動時に、calibration_publiher ダイアログを表示するので、
ノード起動時に指定する YAML ファイルのパスを設定する。
(フルパスで指定する)

Points Image トグルボタン

points2image/points2image ノードを起動・終了する。

Scan Image トグルボタン

scan2image/scan2image ノードを起動・終了する。

Virtual Scan Image トグルボタン

runtime_manager/vscan.launch スクリプトを起動・終了する。

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

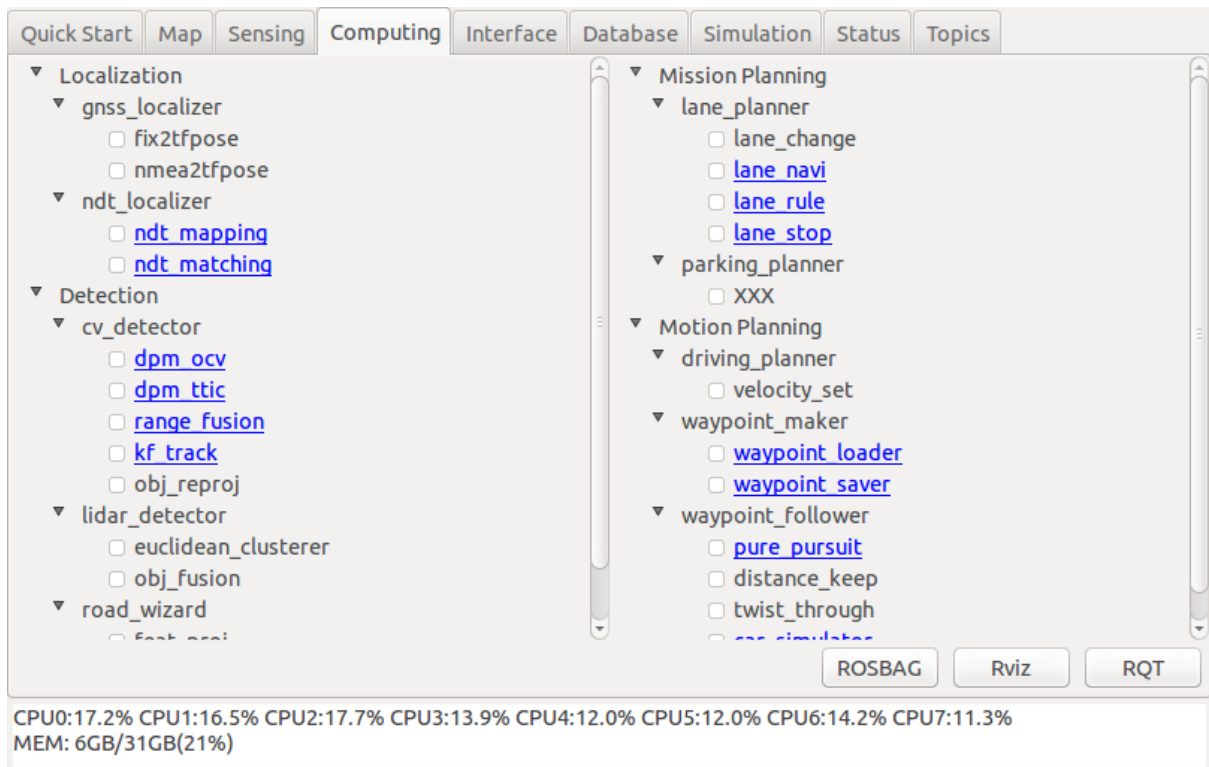
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

Computing タブ



Computing タブ

Localization/gnss_localiser 欄

fix2tfpose 項目

gnss_localizer/fix2tfpose ノードを起動・終了する。

nmea2tfpose 項目

gnss_localizer/nmea2tfpose.launch スクリプトを起動・終了する。

Localization/ndt_localiser 欄

ndt_mapping 項目

ndt_localizer/ndt_mapping.launch スクリプトを起動・終了する。

リンク

ndt_mapping ダイアログを表示する。

パラメータ変更後

/config/ndt_mapping トピックを発行する。

ndt_matching 項目

ndt_localizer/ndt_matching.launch スクリプトを起動・終了する。

リンク

ndt ダイアログを表示する。

パラメータ変更後

/config/ndt トピックを発行する。

Detection/cv_detector 欄

dpm_ocv 項目

runtime_manager/dpm_ocv.launch スクリプトを起動・終了する。

起動時に dpm_ocv ダイアログが表示される。

パラメータを設定後、Detection Start ボタンでスクリプトが起動する。

リンク

チューニングするパラメータの種別(Car 又は Pedestrian)を選択すると、

car_dpm ダイアログ又は pedestrian_dpm ダイアログを表示する。

パラメータ変更後

/config/car_dpm 又は /config/pedestrian_dpm トピックを発行する。

dpm_ttlic 項目

cv_tracker/dpm_ttlic.launch スクリプトを起動・終了する。

起動時に dpm_ttlic ダイアログが表示される。

パラメータを設定後、Detection Start ボタンでスクリプトが起動する。

リンク

チューニングするパラメータの種別(Car 又は Pedestrian)を選択すると、

car_dpm ダイアログ又は pedestrian_dpm ダイアログを表示する。

パラメータ変更後

/config/car_dpm 又は /config/pedestrian_dpm トピックを発行する。

fusion_ranging 項目

cv_tracker/ranging.launch スクリプトを起動・終了する。

起動時に car_fusion ダイアログが表示される。

パラメータを設定後、Start ボタンでスクリプトが起動する。

リンク

チューニングするパラメータの種別(Car 又は Pedestrian)を選択すると、

car_fusion ダイアログ又は pedestrian_fusion ダイアログを表示する。

パラメータ変更後

/config/car_fusion 又は /config/pedestrian_fusion トピックを発行する。

kf_tracking 項目

cv_tracker/kf_tracking.launch スクリプトを起動・終了する。

起動時に car_kf ダイアログが表示される。

パラメータを設定後、Start ボタンでスクリプトが起動する。

リンク

チューニングするパラメータの種別(Car 又は Pedestrian)を選択すると、

car_kf ダイアログ又は pedestrian_kf ダイアログを表示する。

パラメータ変更後

/config/car_kf 又は /config/pedestrian_kf トピックを発行する。

obj_reproj 項目

cv_tracker/reprojection.launch スクリプトを起動・終了する。

起動時に obj_reproj ダイアログが表示される。

パラメータを設定後、Start ボタンでスクリプトが起動する。

Detection/lidar_detector 欄

euclidean_clustering 項目

lidar_tracker/euclidean_clustering.launch スクリプトを起動・終了する。

obj_fusion 項目

lidar_tracker/obj_fusion.launch スクリプトを起動・終了する。

起動時に obj_fusion ダイアログが表示される。

パラメータを設定後、Start ボタンでスクリプトが起動する。

Detection/road_wizard 欄

feat_proj 項目

road_wizard/feat_proj ノードを起動・終了する。

region_tlr 項目

road_wizard/traffic_light_recognition.launch スクリプトを起動・終了する。

Detection/viewers 欄

image_viewer 項目

viewers/image_viewer ノードを起動・終了する。

image_d_viewer 項目

viewers/image_d_viewer ノードを起動・終了する。

points_image_viewer 項目

viewers/points_image_viewer ノードを起動・終了する。

points_image_d_viewer 項目

viewers/points_image_d_viewer ノードを起動・終了する。

vscan_image_viewer 項目

viewers/vscan_image_viewer ノードを起動・終了する。

vscan_image_d_viewer 項目

viewers/vscan_image_d_viewer ノードを起動・終了する。

traffic_light_viewer 項目

viewers/traffic_light_viewer ノードを起動・終了する。

Mission Planning/lane_planner 欄

lane_change 項目

lane_planner/lane_chabge ノードを起動・終了する。

lane_navi 項目

lane_planner/lane_navi ノードを起動・終了する。

リンク

lane_navi ダイアログを表示する。

パラメータ変更後

rosparam /lane_navi/velocity,
rosparam /lane_navi/output_file を設定する。

lane_rule 項目

lane_planner/lane_rule ノードを起動・終了する。

リンク

lane_rule ダイアログを表示する。

パラメータ変更後

rosparam /lane_rule/vector_map_directory,
rosparam /lane_rule/ruled_waypoint_csv を設定し、
/config/lane_rule トピックを発行する。

lane_stop 項目

lane_planner/lane_stop ノードを起動・終了する。

リンク

lane_stop ダイアログを表示する。

パラメータ変更後

/traffic_light トピックを発行する。

Mission Planning/parking_planner 欄

XXX 項目

<未実装>

Motion Planning/driving_planner 欄

velocity_set 項目

driving_planner/velocity_set ノードを起動・終了する。

Motion Planning/waypoint_maker 欄

waypoint_loader 項目

waypoint_maker/waypoint_loader.launch スクリプトを起動・終了する。

[リンク](#)

waypoint_loader ダイアログを表示する。

パラメータ変更後

/waypoint_loader/vector_map_directory トピックを発行し、
rosparam /waypoint_loader/ruled_waypoint_csv を設定し、
/config/waypoint_loader トピックを発行する。

waypoint_saver 項目

waypoint_maker/waypoint_saver.launch スクリプトを起動・終了する。

[リンク](#)

waypoint_saver ダイアログを表示する。

スクリプト起動時に指定する save_filename と Interval の値を設定する。

Motion Planning/waypoint_follower 欄

pure_pursuit 項目

waypoint_follower/pure_pursuit_sim.launch スクリプトを起動・終了する。

[リンク](#)

waypoint_follower ダイアログを表示する。

パラメータ変更後

/config/waypoint_follower トピックを発行する。

distance_keep 項目

waypoint_follower/distance_keep.launch スクリプトを起動・終了する。

twist_through 項目

waypoint_follower/twist_through ノードを起動・終了する。

car_simulator 項目

lane_follower/car_simulator.launch スクリプトを起動・終了する。

リンク

car_simulator ダイアログを表示する。

パラメータ変更後

```
rosparam /odom_gen/use_pose  
rosparam /odom_gen/initial_pos_x  
rosparam /odom_gen/initial_pos_y  
rosparam /odom_gen/initial_pos_z  
rosparam /odom_gen/initial_pos_roll  
rosparam /odom_gen/initial_pos_pitch  
rosparam /odom_gen/initial_pos_yaw
```

を設定する。

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

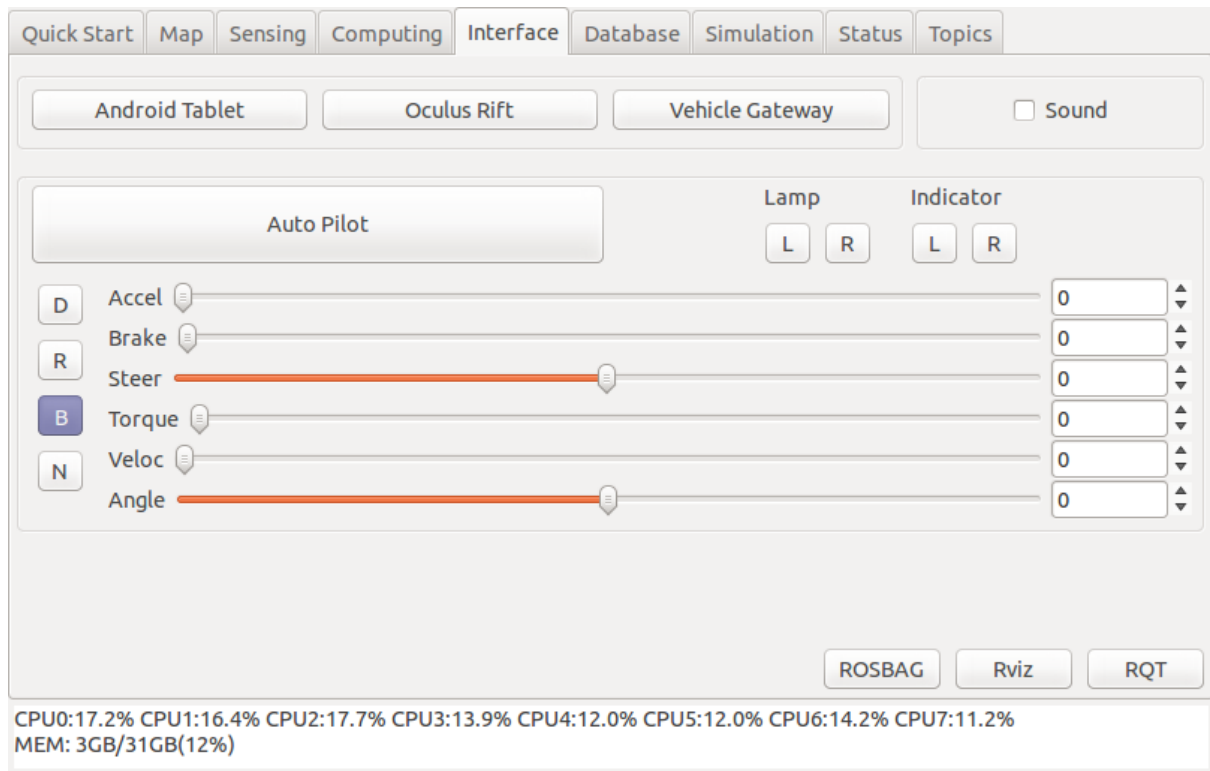
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

Interface タブ



Interface タブ

Android Tablet トグルボタン

runtime_manager/tablet_socket.launch スクリプトを
起動・終了する。

Oculus Rift トグルボタン

〈未実装〉

Vehicle Gatewat トグルボタン

runtime_manager/vehicle_socket.launch スクリプトを
起動・終了する。

Sound チェックボックス

sound_player/sound_player.py スクリプトを起動・終了する。

Auto Pilot トグルボタン

ボタンの状態に応じた mode_cmd トピックを発行する。

Lamp L, R トグルボタン

ボタンの状態に応じた lamp_cmd トピックを発行する。

Indicator L, R トグルボタン

ボタンの状態に応じた indicator_cmd トピックを発行する。

D,R,B,N ボタン

ON 操作したボタンに応じた gear_cmd トピックを発行する。

Accel スライダー

accel_cmd トピックを発行する。

Brake スライダー

brake_cmd トピックを発行する。

Steer スライダー

steer_cmd トピックを発行する。

Torque スライダー

〈未実装〉

Veloc スライダー

twist_cmd トピックを発行する。

(スライダーの値はメッセージの twist.linear.x フィールドに反映)

Angle スライダー

twist_cmd トピックを発行する。

(スライダーの値はメッセージの twist.angular.z フィールドに反映)

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

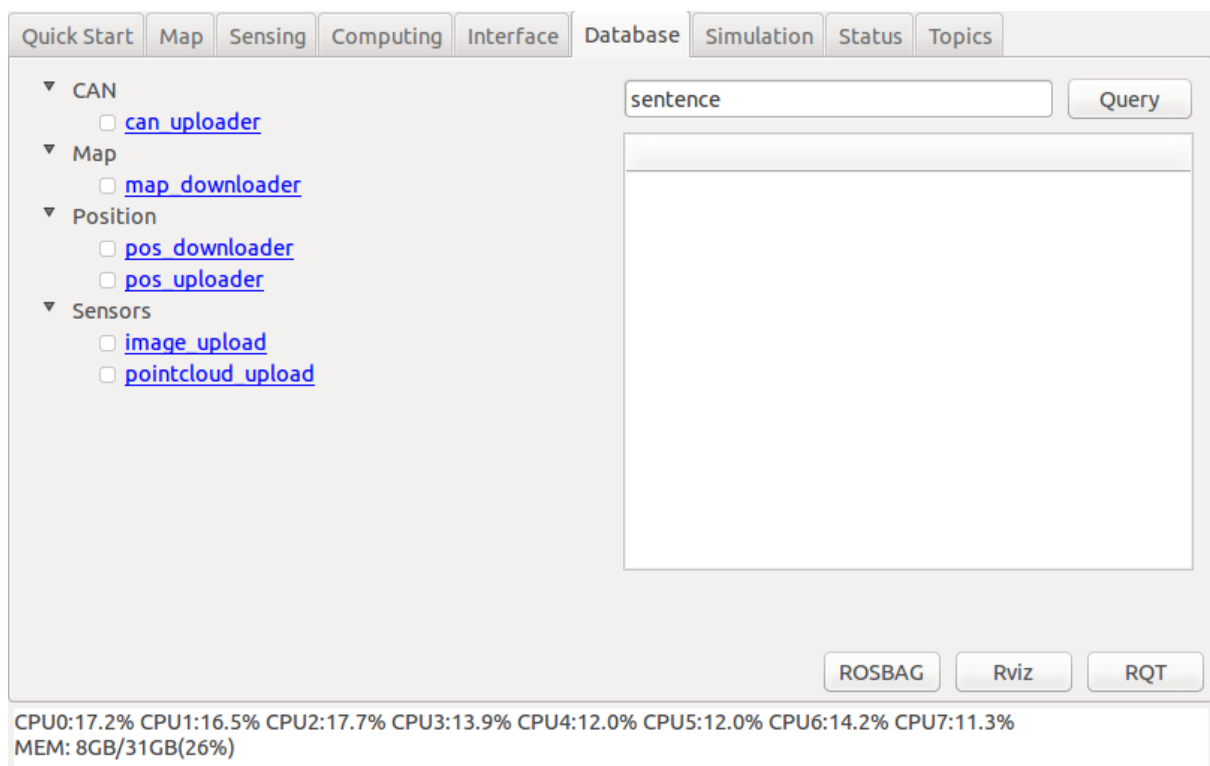
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

Database タブ



Database タブ

CAN 欄

can_uploader 項目

obj_db/can_uploader ノードを起動・終了する。

[リンク](#)

other ダイアログを表示する。

Map 欄

map_downloader 項目

＜未実装＞

[リンク](#)

map_file ダイアログを表示する。

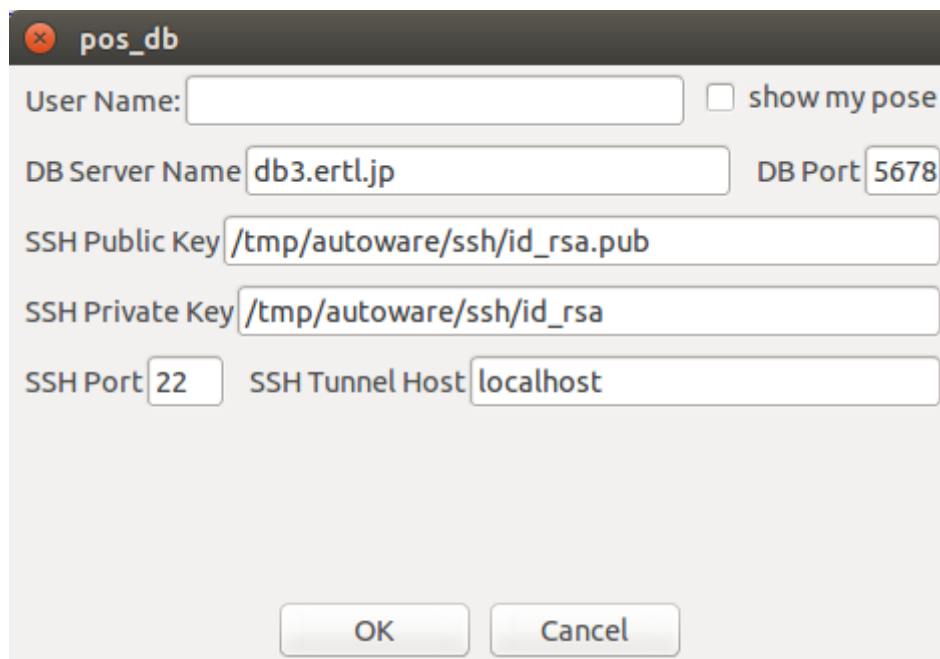
Position 欄

pos_downloader 項目

pos_db/pos_downloader ノードを起動・終了する。

[リンク](#)

pos_db ダイアログを表示する。



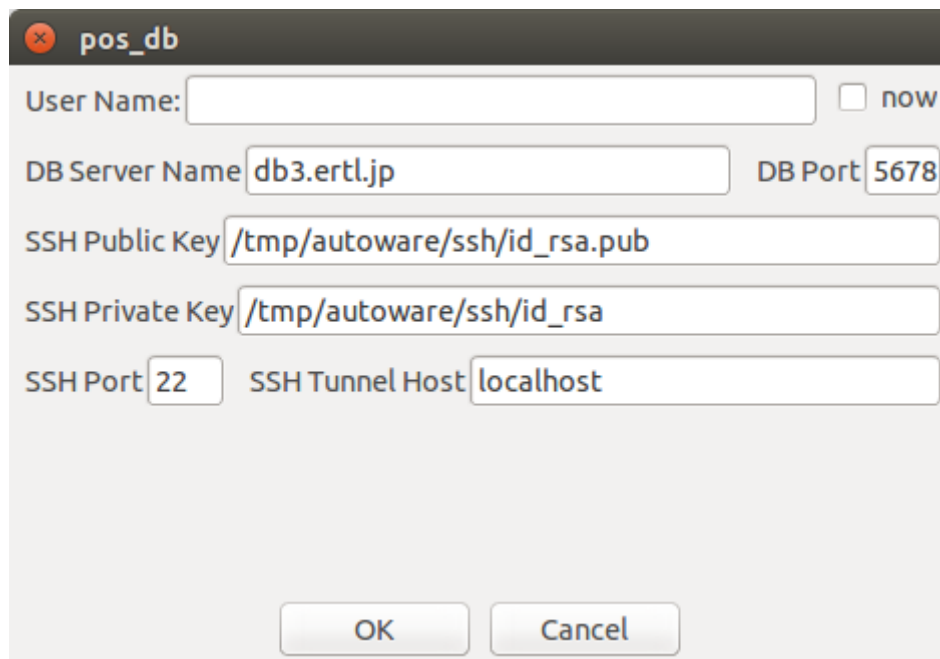
pos_db ダイアログ(pos_downloader)

pos_uploader 項目

pos_db/pos_uploader ノードを起動・終了する。

[リンク](#)

pos_db ダイアログを表示する。



pos_db

User Name: ☐ now

DB Server Name DB Port

SSH Public Key

SSH Private Key

SSH Port SSH Tunnel Host

OK Cancel

pos_db ダイアログ(pos_uploader)

Sensors 欄

image_upload 項目

〈未実装〉

リンク

other ダイアログを表示する。

pointcloud_upload 項目

〈未実装〉

リンク

other ダイアログを表示する。

Query テキストボックス

〈未実装〉

Query ボタン

〈未実装〉

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

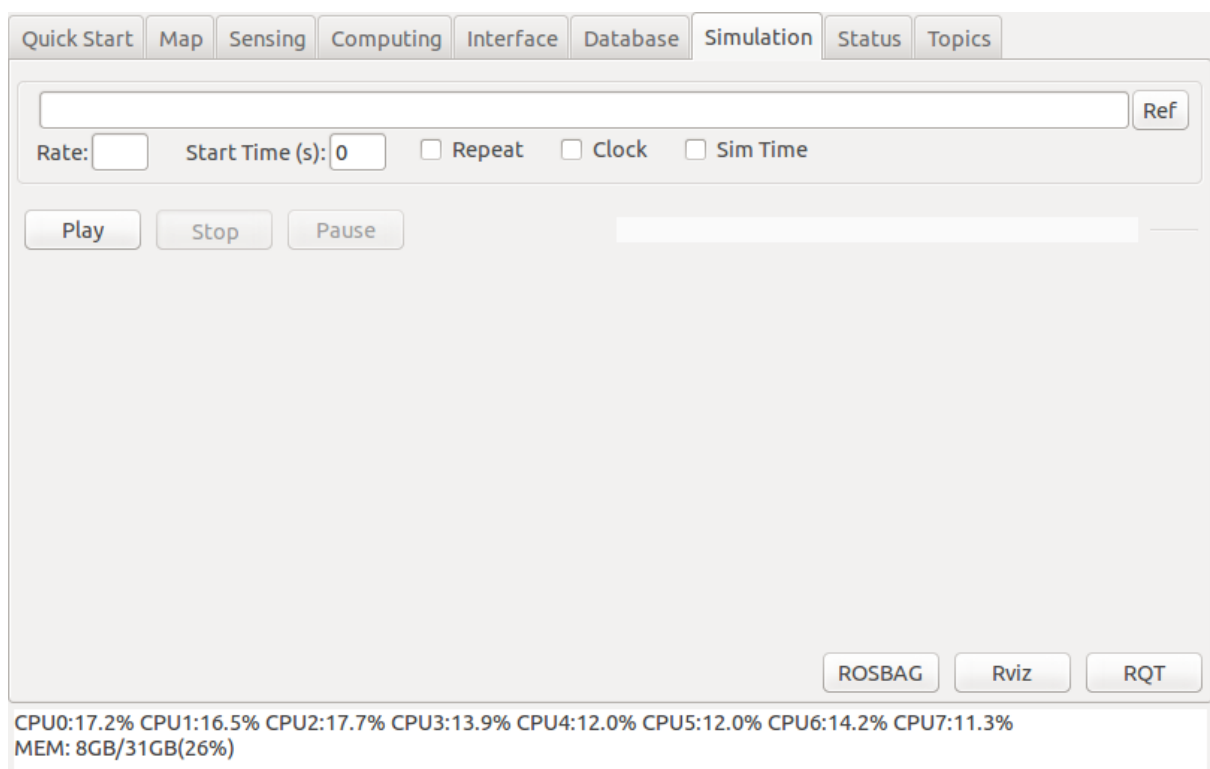
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

Simulation タブ



Simulation タブ

ROSBAG テキストボックス

Play ボタンで rosbag play コマンドを実行する際の、bag ファイルを指定する。
(フルパスで指定する)

ROSBAG Ref ボタン

ファイル選択ダイアログが表示される。

選択したファイルは、ROSBAG テキストボックスに設定される。

Rate テキストボックス

rosbag play コマンドを起動する際の `-r` オプションで指定する数値を指定する。

未設定の場合は `-r` オプションを指定しない。

Start Time(s) テキストボックス

rosbag play コマンドを起動する際の `--start` オプションで指定する開始位置の秒数を指定する。

未設定の場合は `--start` オプションを指定しない。

Repeat チェックボックス

チェックボックスが ON の場合、rosbag play コマンドを起動する際に、
`--loop` オプションが指定される。

Clock チェックボックス

チェックボックスが ON の場合、rosbag play コマンドを起動する際に、
`--clock` オプションが指定される。

(この設定は終了時に保存されない)

Sim Time チェックボックス

rosparam `/use_sim_time` の設定値 (true,false) を表示する。

チェックボックスを操作すると、値を rosparam `/usr_sim_time` に設定する。

(この設定は終了時に保存されない)

Play ボタン

ROSBAG テキストボックスに設定された bag ファイルを指定して、
rosbag play コマンドを起動する。

Stop ボタン

起動している rosbag play コマンドを終了する。

Pause ボタン

起動している rosbag play コマンドを一時停止する。

中央部

ROSBAG テキストボックスに設定している .bag ファイルについて、
rosbag info コマンドの実行結果を表示する。

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

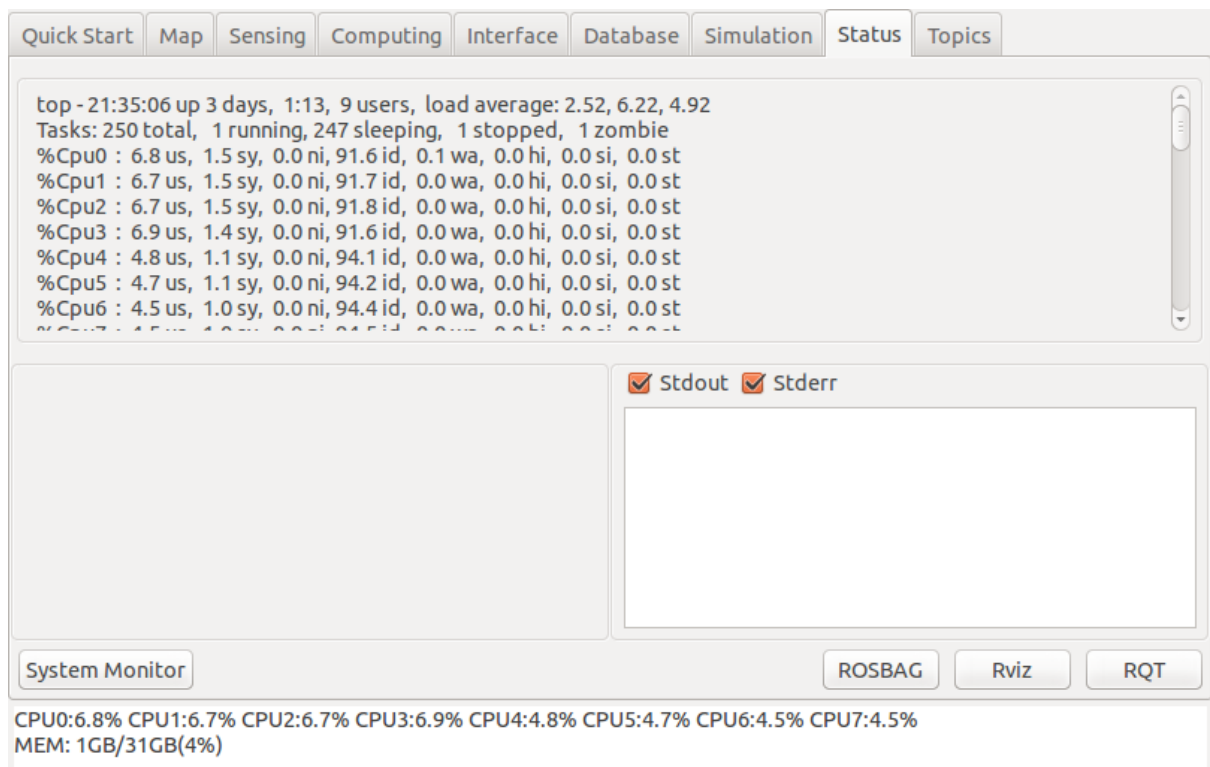
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

Status タブ



Status タブ

上段の表示

内部で実行している top コマンドの実行結果を表示する。

下段左側の表示

関連のノードが発行する周期実行時間を表示する。

下段右側の表示

起動したノード、スクリプトの標準出力、標準エラー出力の内容を表示する。
(ただし、プログレスバー表示を行う一部のノードでは表示されない)

最下行の情報表示

CPU(コア)の負荷状況とメモリ使用量を表示する。

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

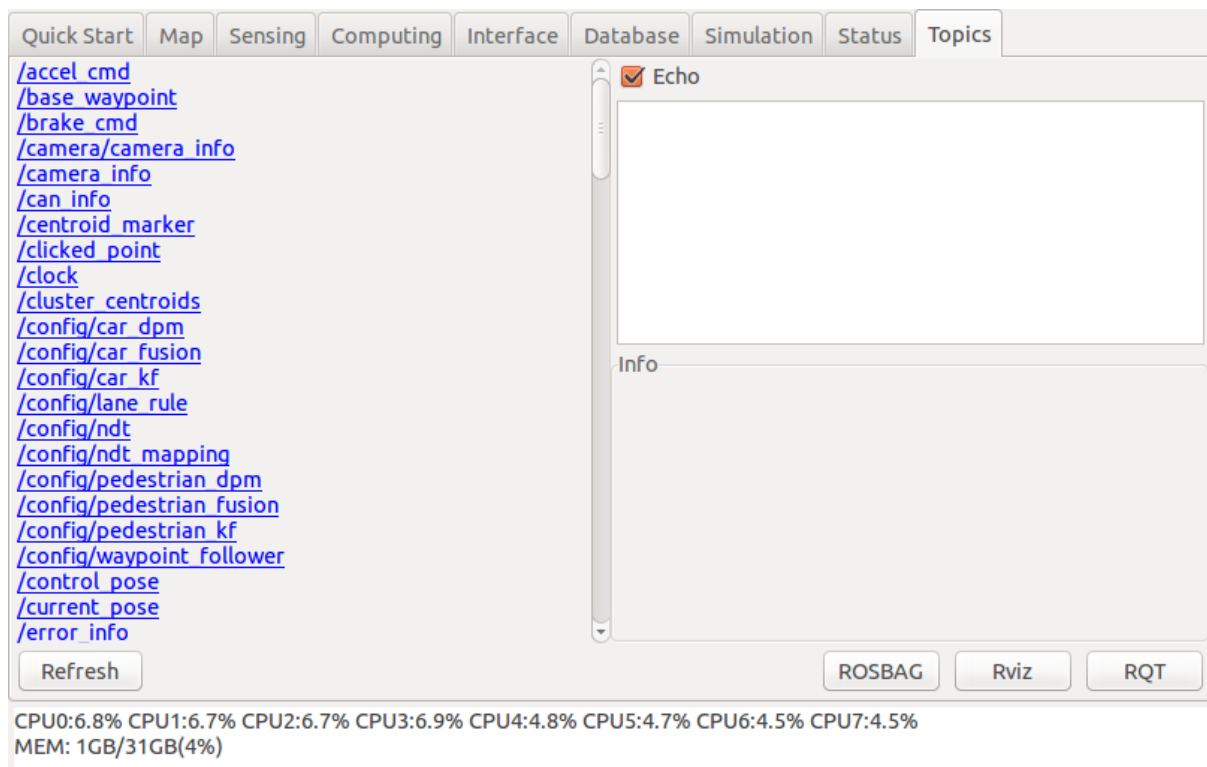
Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

Topics タブ



Topics タブ

左側の表示

トピック名の一覧を表示する。

リンクをクリックすると、

rostopic echo <対象トピック> コマンドを実行し、右側上段に結果を表示し、
rostopic info <対象トピック> コマンドを実行し、右側下段に結果を表示する。

右側上段の表示

rostopic echo コマンドの実行結果を表示する。

右側下段の表示

rostopic info コマンドの実行結果を表示する。

Refresh ボタン

トピック名の一覧を取得しなおして更新する。

また、リンククリックにより rostopic echo コマンド実行中ならば停止させる。

最下行の情報表示

CPU(コア)の負荷状況とメモリ使用量を表示する。

ROSBAG ボタン

ROSBAG Record ダイアログを表示する。

ROSBAG Record ダイアログの詳細は Quick Start タブを参照。

Rviz トグルボタン

rviz/rviz ノードを起動・終了する。

RQT ボタン

rqt を起動・終了する。

ユーザインタフェース

概要

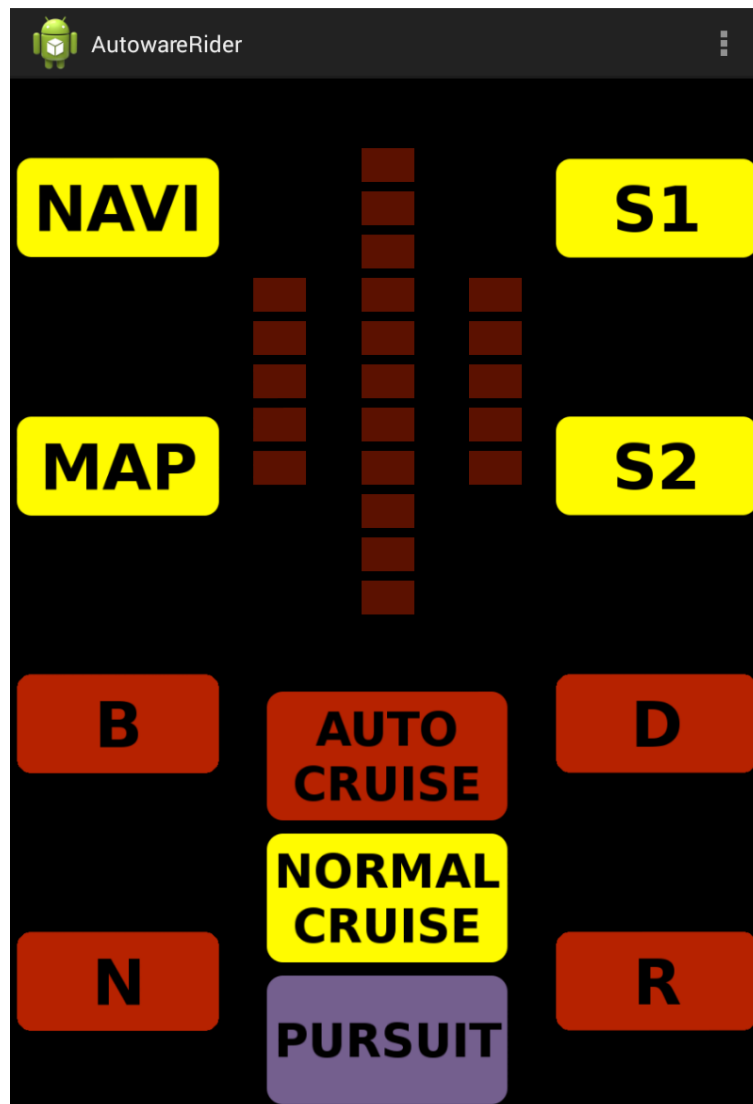
AutowareRider は、ROS PC で動作する Autoware をタブレット端末から操作するための、Knight Rider に似た UI を持った、Android アプリケーションです。

AutowareRoute は、MapFan SDK で実装された、経路データ生成のための Android アプリケーションです。

ここでは、これらの UI の機能を説明します。

AutowareRider

以下が起動時の画面です。



図の各ボタンの機能は以下です。

- NAVI
 - AutowareRoute.apk の起動
- MAP
 - 未実装
- S1
 - check.launch を ROS PC で起動
- S2
 - set.launch を ROS PC で起動
- B
 - ギア情報 B を ROS PC へ送信

- N
 - ギア情報 N を ROS PC へ送信
- D
 - ギア情報 D を ROS PC へ送信
- R
 - ギア情報 R を ROS PC へ送信
- AUTO CRUISE
 - 未実装
- NORMAL CRUISE
 - 未実装
- PURSUIT
 - 未実装（現状はアプリケーションの終了）

[右上メニュー]から以下が選択できます。

- [設定]
- [データ収集]

以下が[設定]の画面です。

設定

ROS PC

IPアドレス: 192.168.0.10

命令受信ポート番号: 5666

情報送信ポート番号: 5777

データ収集

テーブル名: candata

SSH

ホスト名: candb.jp

ポート番号: 22

ユーザ名: autoware

パスワード:

ポートフォワーディング

ローカルポート番号: 5558

リモートホスト名: 127.0.0.1

リモートポート番号: 5555

キャンセル OK

図の各項目の説明は以下です。

- ROS PC
 - IP アドレス
ROS PC IPv4 アドレス
 - 命令ポート番号
tablet_receiver ポート番号（初期値：5666）
 - 情報ポート番号
tablet_sender ポート番号（初期値：5777）
- データ収集
 - テーブル名
データ転送先 テーブル名
- SSH
 - ホスト名
SSH 接続先 ホスト名

- ポート番号
SSH 接続先 ポート番号 (初期値: 22)
- ユーザ名
SSH でログインするユーザ名
- パスワード
SSH でログインするパスワード
- ポートフォワーディング
 - ローカルポート番号
ローカルマシンの転送元ポート番号 (初期値: 5558)
 - リモートホスト名
リモートマシン ホスト名 (初期値: 127.0.0.1)
 - リモートポート番号
リモートマシンの転送先ポート番号 (初期値: 5555)

以下が[データ収集]の画面です。



図の各ボタンの機能は以下です。

- CanGather
 - CanGather.apk の起動
- CarLink (Bluetooth)
 - CarLink_CAN-BT_LS.apk の起動
- CarLink (USB)
 - CarLink_CANusbAccessory_LS.apk の起動

AutowareRoute

以下が起動時の画面です。



地図を長押しすることで、以下のダイアログが表示されます。



図の各ボタンの機能は以下です。

- 出発地に設定
 - 長押しした地点を経路データの出発地として設定
- 立寄地に設定
 - 長押しした地点を経路データの立寄地として設定
- 目的地に設定
 - 長押しした地点を経路データの目的地として設定
- ルート消去
 - ルート探索実行によって生成された経路データの消去
- ルート探索実行
 - 出発地、立寄地、目的地に応じた経路データの生成