Autoware ユーザーズマニュアル

2015/AUG/03

名古屋大学

内容

[はじめに 4](#_Toc426387072)

[概要 4](#_Toc426387073)

[用語 4](#_Toc426387074)

[関連文書 6](#_Toc426387075)

[全体構成 7](#_Toc426387076)

[構成 7](#_Toc426387077)

[主な機能 8](#_Toc426387078)

[環境構築の手順 9](#_Toc426387079)

[Linux 9](#_Toc426387080)

[ROS 10](#_Toc426387081)

[OpenCV 10](#_Toc426387082)

[Qt 10](#_Toc426387083)

[CUDA 11](#_Toc426387084)

[FlyCapture2 12](#_Toc426387085)

[Autoware 13](#_Toc426387086)

[AutowareRider 13](#_Toc426387087)

[canlib 14](#_Toc426387088)

[SSHの公開鍵の作成 14](#_Toc426387089)

[使用手順 15](#_Toc426387090)

[センサデータの取得 15](#_Toc426387091)

[自動運転 15](#_Toc426387092)

[AutowareRider 15](#_Toc426387093)

[概要 15](#_Toc426387094)

[起動方法 16](#_Toc426387095)

[経路データ生成アプリケーションの使用方法 17](#_Toc426387096)

[ROS PCへの経路データ転送手順 17](#_Toc426387097)

[CANデータ収集アプリケーションの使用方法 17](#_Toc426387098)

[ROS PCへのCANデータ転送手順 18](#_Toc426387099)

[Launchファイルの起動方法 19](#_Toc426387100)

[各機能の説明 19](#_Toc426387101)

[ROS 19](#_Toc426387102)

[認知 (物体検出, 位置推定) 19](#_Toc426387103)

[判断 (レーン走行, 交差点) 19](#_Toc426387104)

[操作 19](#_Toc426387105)

[データ 19](#_Toc426387106)

[センサ 21](#_Toc426387107)

[非ROSモジュールとの通信 21](#_Toc426387108)

[ユーティリティ 22](#_Toc426387109)

[Runtime Manager 22](#_Toc426387110)

[概要 22](#_Toc426387111)

[Quick Startタブ 23](#_Toc426387112)

[Mapタブ 27](#_Toc426387113)

[Sensingタブ 29](#_Toc426387114)

[Computingタブ 33](#_Toc426387115)

[Interfaceタブ 38](#_Toc426387116)

[Databaseタブ 40](#_Toc426387117)

[Simulationタブ 43](#_Toc426387118)

[Statusタブ 45](#_Toc426387119)

[ユーザインタフェース 46](#_Toc426387120)

[概要 46](#_Toc426387121)

[AutowareRider 46](#_Toc426387122)

[AutowareRoute 50](#_Toc426387123)

[車の制御 52](#_Toc426387124)

[一般 53](#_Toc426387125)

[ZMP 53](#_Toc426387126)

# 先頭が「#」で始まる行は、コメントです。

# 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|デベロッパーズ}マニュアルです。

(ユーザーズマニュアルでは、) 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。

(デベロッパーズマニュアルでは、) Autowareに独自の機能を追加するために必要な開発手順、その助けとなる情報について記述しています。

## 用語

# 自動運転に関する用語も、統一したいので追加する。

* 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

# 文書ではなくURLになっていますが…

* 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>

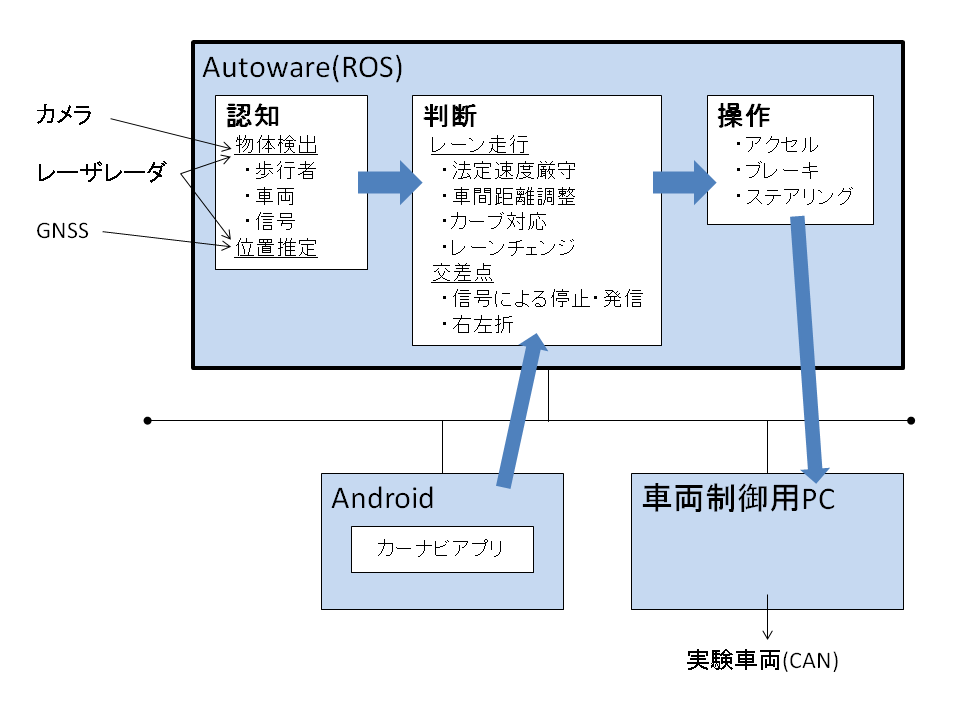
# 全体構成

# AutowareのPC + 各種センサ機器 の図と説明を書く。

# Autowareの中は、加藤先生の仕様書などを参考に。

# http://www.pdsl.jp/fot/autoware/

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

* ros/src/computing/planning/

判断

* ros/src/computing/control/

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とFlyCapture SDK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

1. 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-velodyne-pointcloud \

ros-indigo-nmea-msgs

$ sudo apt-get install libnlopt-dev freeglut3-dev qtbase5-dev libqt5opengl5-dev

1. 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。

# sources.listの設定など必要

$ sudo apt-get install ros-hydro-desktop-full ros-hydro-velodyne-pointcloud \

ros-indigo-nmea-msgs

$ sudo apt-get install libnlopt-dev freeglut3-dev

1. ~/.bashrcなどに以下を追加します。

[ -f /opt/ros/indigo/setup.bash ] && . /opt/ros/indigo/setup.bash

## OpenCV

OpenCVのサイト(<http://sourceforge.net/projects/opencvlibrary/>)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。

# 現在、2.4.8が入手不可だが、他のバージョンでもOKか?

$ 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

1. 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 https://git.gitorious.org/qt/qt5.git qt5

$ 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

1. 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)

1. 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

1. システムを再起動 (…は不要かもしれません)

$ lsmod | grep nouveau

(nouveauドライバがロードされていないことを確認)

1. 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\_Utilities/deviceQuery/

$ make

$ ./deviceQuery

1. CUDAを普段から使う場合は、以下の設定を .bashrc などに書く

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

1. 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

1. 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。

$ git clone https://github.com/CPFL/Autoware.git  
 $ cd Autoware/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://](http://sourceforge.net/projects/opencvlibrary/)[www.kvaser.com/downloads/](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 はデータベースにアクセスする際に SSH の公開鍵認証を使用します。

そのため、pos\_db ノードを動かすユーザであらかじめ作成して公開鍵の情報を サーバに反映しておく必要があります。

1. 鍵の作成方法
   * 以下のコマンドを実行して鍵を作成する
     + $ 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. 公開鍵をサーバーにコピーする
   * 作成した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
   * 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

# デモなどで使われているものをいくつか説明

# https://github.com/CPFL/Autoware/wiki/5.-Moriyama-FOT-(ja)

# 基本は Runtime Manager から制御

# センサ, AutowareRider, AutowareTouch も使う例を入れる

## センサデータの取得

## 自動運転

## 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。
   * ui\_receiver
   * ui\_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  
       ui\_receiver ポート番号 (初期値: 5666)
     + 情報ポート番号  
       ui\_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、次回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 ({ndt, lf}\_stat: false) |
| 押下 (文字色: 赤) | 起動 ({ndt, lf}\_stat: true) |

# 各機能の説明

## ROS

### 認知 (物体検出, 位置推定)

# ros/src/computing/perception/{detection,localization,...}

### 判断 (レーン走行, 交差点)

# ros/src/computing/planning/{mission, motion, path,...}

### 操作

# ros/src/computing/control

### データ

# ros/src/data (ファイルやDBからデータを取得)

data のノード

1. points\_map\_loader

path: ros/src/data/packages/map\_file/nodes/points\_map\_loader/

publish\_msg: points\_map, pmap\_stat

subscribe\_msg: -

parameter: -

description: ポイントクラウド地図のファイルを読み込んで rviz に表示します

1. vector\_map\_loader

path: ros/src/data/packages/map\_file/nodes/vector\_map\_loader/

publish\_msg: vector\_map, vmap\_stat, vector\_map\_info

subscribe\_msg: -

parameter: -

argument: 高精度地図のCSVファイル

description: 高精度地図のCSVファイルを読み込んで rviz に表示します

1. pos\_uploader

path: ros/src/data/packages/pos\_db/nodes/pos\_uploader/

publish\_msg: -

subscribe\_msg: current\_pose, car\_pose, pedestrian\_pose

parameter: pos\_db/db\_host\_name

pos\_db/db\_port

pos\_db/sshpubkey

pos\_db/sshprivatekey

pos\_db/ssh\_port

pos\_db/sshtunnelhost

argument: データベースサーバのユーザ名,

now(省略するとsimulation timeを使用)

description: current\_pose, car\_pose, pedestrian\_pose のいずれかが publish された際に、それらの位置情報をデータベースサーバに登録を行う

1. pos\_downloader

path: ros/src/data/packages/pos\_db/nodes/pos\_downloader/

publish\_msg: mo\_marker

subscribe\_msg: -

parameter: pos\_downloader/time

pos\_downloader/delay

pos\_downloader/life\_time

pos\_downloader/posup\_dz

pos\_downloader/pedestrian\_dz

pos\_db/db\_host\_name

pos\_db/db\_port

pos\_db/sshpubkey

pos\_db/sshprivatekey

pos\_db/ssh\_port

pos\_db/sshtunnelhost

argument: データベースサーバのユーザ名,

show\_my\_pose(省略すると自己車両を表示しない)

description: データベースサーバから取得した位置情報を rviz に表示します

### センサ

# ros/src/sensing/{calibration, drivers, fusion, sync, ...}

### 非ROSモジュールとの通信

# ros/src/socket

ui\_socketのノード

1. ui\_receiver

path: ros/src/socket/packages/ui\_socket/nodes/ui\_receiver/

publish\_msg: gear\_cmd mode\_cmd route\_cmd

subscribe\_msg: -

parameter: ui\_receiver/port (default: 5666)

description: ROS非対応のAndroidアプリケーションなどからのデータをROSのメッセージに変換してpublishするノードです。5666/TCP(パラメータで変更可能)で待ち受けます。

1. ui\_sender

path: ros/src/socket/packages/ui\_socket/nodes/ui\_sender/

publish\_msg: -

subscribe\_msg: error\_info can\_info mode\_info ndt\_stat lf\_stat

parameter: ui\_sender/port (default: 5777)

description: ROS非対応のAndroidアプリケーションなどへ、ROSのメッセージの情報を送信するノードです。5777/TCP(パラメータで変更可能)で待ち受けます。

ui\_socketのメッセージ

1. gear\_cmd

Header header

int32 gear

1. mode\_cmd

Header header

int32 mode

1. route\_cmd

Header header

Waypoint[] point

1. Waypoint

float64 lat

float64 lon

1. error\_info

Header header

int32 error

1. mode\_info

Header header

int32 mode

### ユーティリティ

# ros/src/util

## 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

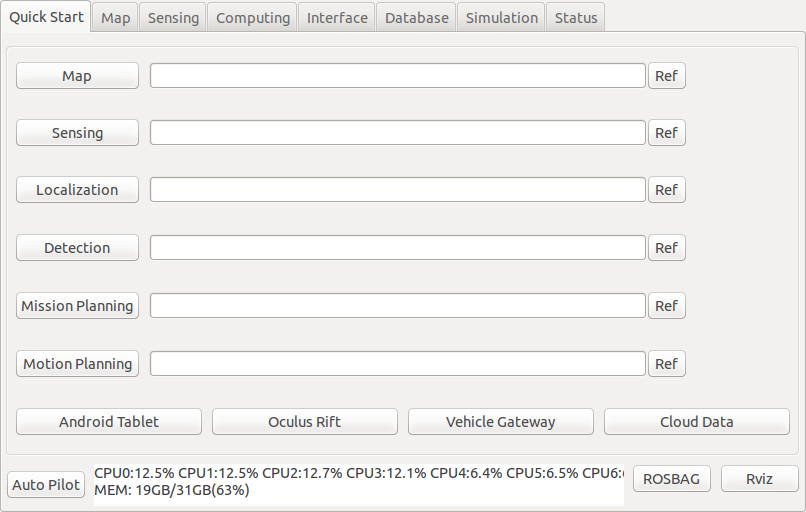
行なう事ができる。

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

ui\_socket/ui\_receiver, ui\_socket/ui\_sender ノードを

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

vehicle\_socket/vehicle\_receiver, vehicle\_socket/vehicle\_sender

ノードを起動・終了する。

#### Clound 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/rivzノードを起動・終了する。

#### ROSBAG Recordダイアログ

##### 上部テキストボックス

rosbag 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、

rosbag recordコマンドを起動する。

##### Stopボタン

起動しているrosbag 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

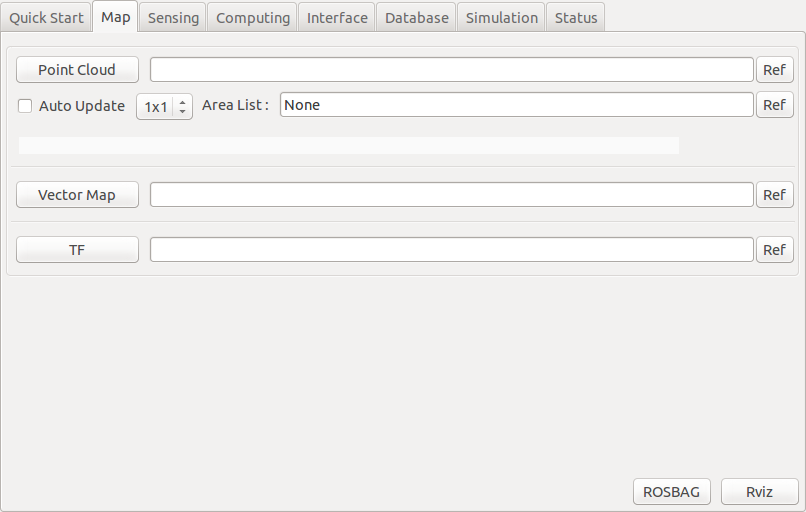
rostopic listコマンドを実行し、現在有効なトピックを調べ、

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



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lauch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

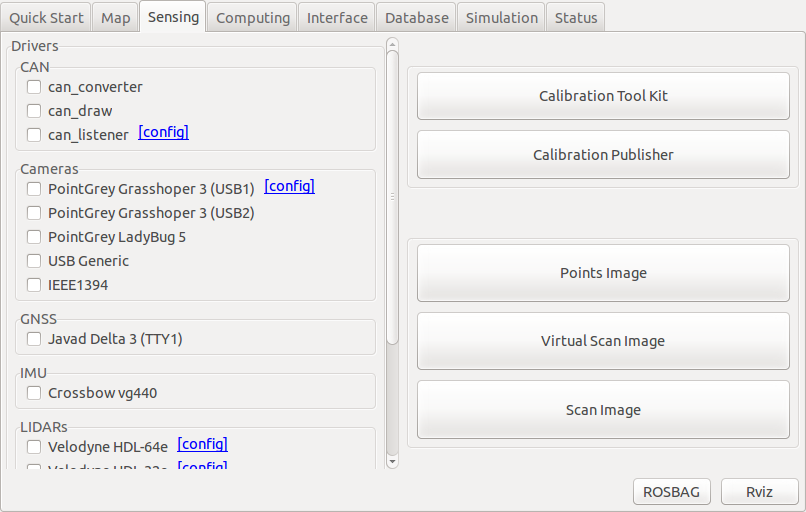
ROSBAG Recordダイアログを表示する。

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/rivzノードを起動・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 Grasshoper 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 Grasshoper 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

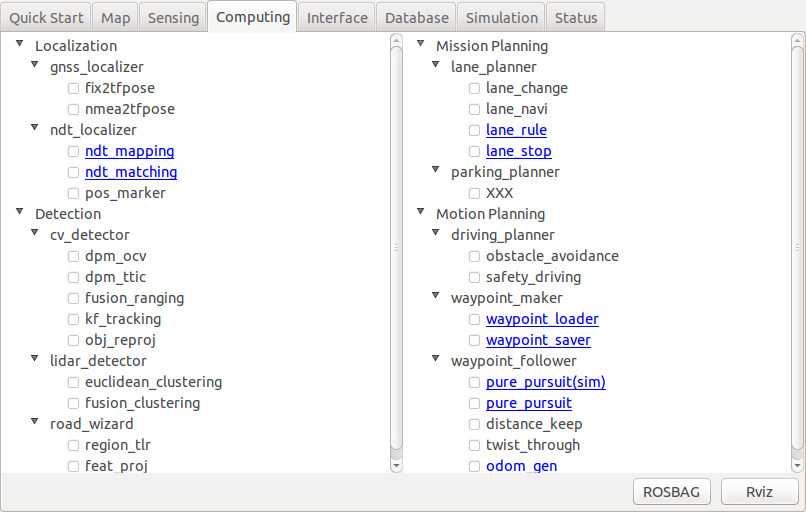
ROSBAG Recordダイアログを表示する。

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/rivzノードを起動・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

<未実装>

##### dpm\_ttic項目

<未実装>

##### fusion\_ranging項目

<未実装>

##### kf\_tracking項目

<未実装>

##### obj\_reproj項目

<未実装>

#### Detection/lidar\_detector欄

##### euclidean\_clustering項目

<未実装>

##### fusion\_clustering項目

<未実装>

#### Detection/road\_wizard欄

##### region\_tlr項目

<未実装>

##### feat\_proj項目

<未実装>

#### 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。

##### 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/obstacle\_avoidance欄

##### obstacle\_avoidance項目

driving\_planner/obstacle\_avoidanceノードを起動・終了する。

##### safety\_driving項目

driving\_planner/safety\_driving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(sim)項目

lane\_follower/pure\_pursuit\_sim.launchスクリプトを起動・終了する。

###### リンク

lane\_fllowerダイアログを表示する。

パラメータ変更後

/config/lane\_followerトピックを発行する。

##### pure pursuit項目

lane\_follower/pure\_pursuit.launchスクリプトを起動・終了する。

###### リンク

lane\_fllowerダイアログを表示する。

パラメータ変更後

/config/lane\_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。

##### odom gen項目

lane\_follower/odom\_gen.launchスクリプトを起動・終了する。

###### リンク

odom\_genダイアログを表示する。

パラメータ変更後

/odom\_gen/use\_pose

/odom\_gen/initial\_pos\_x

/odom\_gen/initial\_pos\_y

/odom\_gen/initial\_pos\_z

/odom\_gen/initial\_pos\_roll

/odom\_gen/initial\_pos\_pitch

/odom\_gen/initial\_pos\_yaw

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。

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/rivzノードを起動・終了する。

### Interfaceタブ

#### interface.png

Interfaceタブ

#### Android Tabletトグルボタン

ui\_socket/ui\_receiver, ui\_socket/ui\_sender ノードを

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

vehicle\_socket/vehicle\_receiver, vehicle\_socket/vehicle\_sender

ノードを起動・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

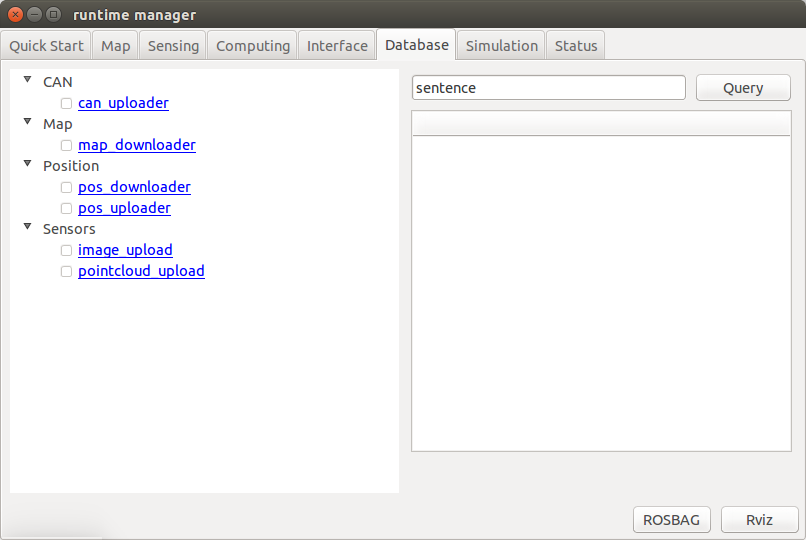
ROSBAG Recordダイアログを表示する。

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/rivzノードを起動・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。

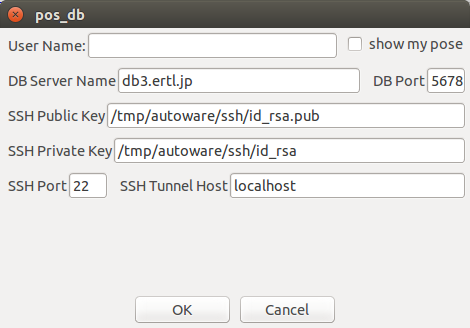
#### Position欄

##### 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。



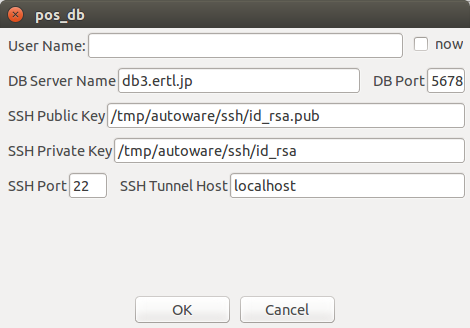
pos\_dbダイアログ(pos\_downloader)

##### 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

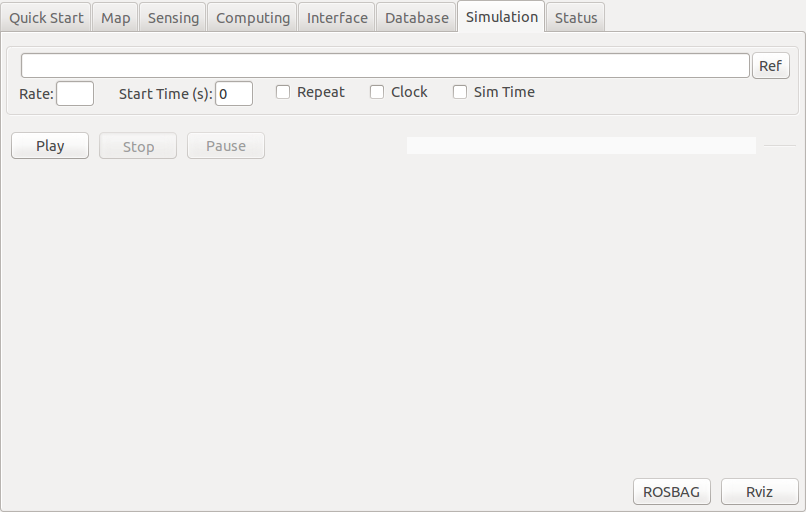
ROSBAG Recordダイアログを表示する。

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/rivzノードを起動・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。

#### 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。

#### 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。

#### ROSBAGボタン

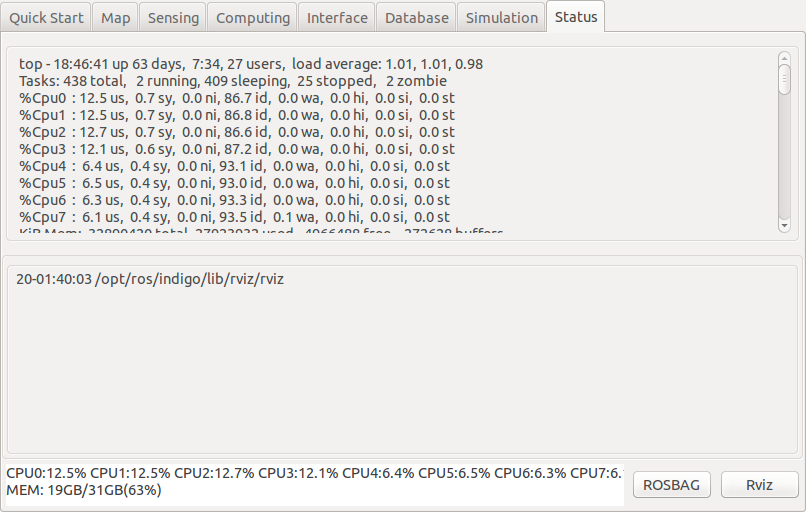
ROSBAG Recordダイアログを表示する。

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/rivzノードを起動・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。

#### 下段の表示

登録ノードの起動してからの経過時間を表示する。

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。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。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。 (仕様の変更予定有り)

#### 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/rivzノードを起動・終了する。

## ユーザインタフェース

### 概要

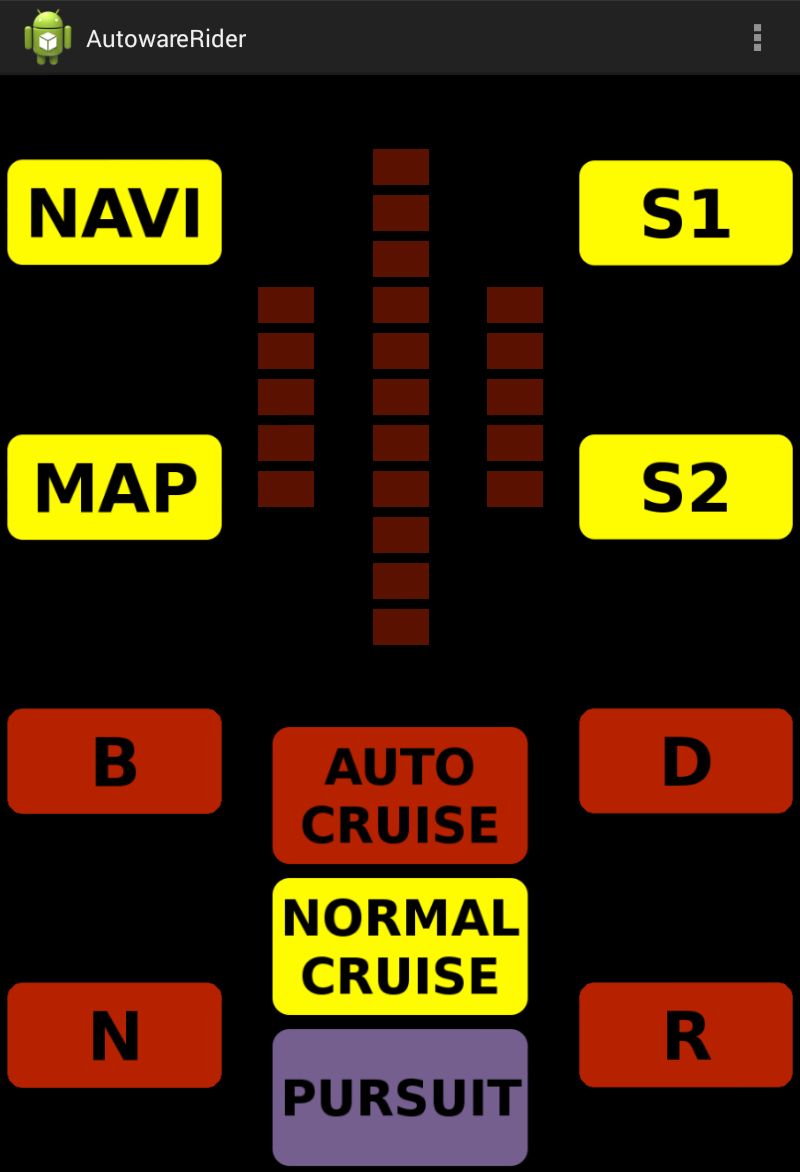
AutowareRiderは、ROS PCで動作するAutowareをタブレット端末から操作するための、Knight Riderに似たUIを持った、Androidアプリケーションです。

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。



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  
    ui\_receiver ポート番号 (初期値: 5666)
  + 情報ポート番号  
    ui\_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

## 車の制御

### 一般

# vehicle/general

### ZMP

# vehicle/zmp