自律運転トレーラー模型
取扱い説明書

saoto-tech

[履歴]
Rev. 1.0：2018/7/10　初版（土屋）
Rev. 2.0：2020/03/23　更新（土屋）

目次

1. はじめに

トレーラーラジコン模型ベース自律運転試験車両の取扱説明。

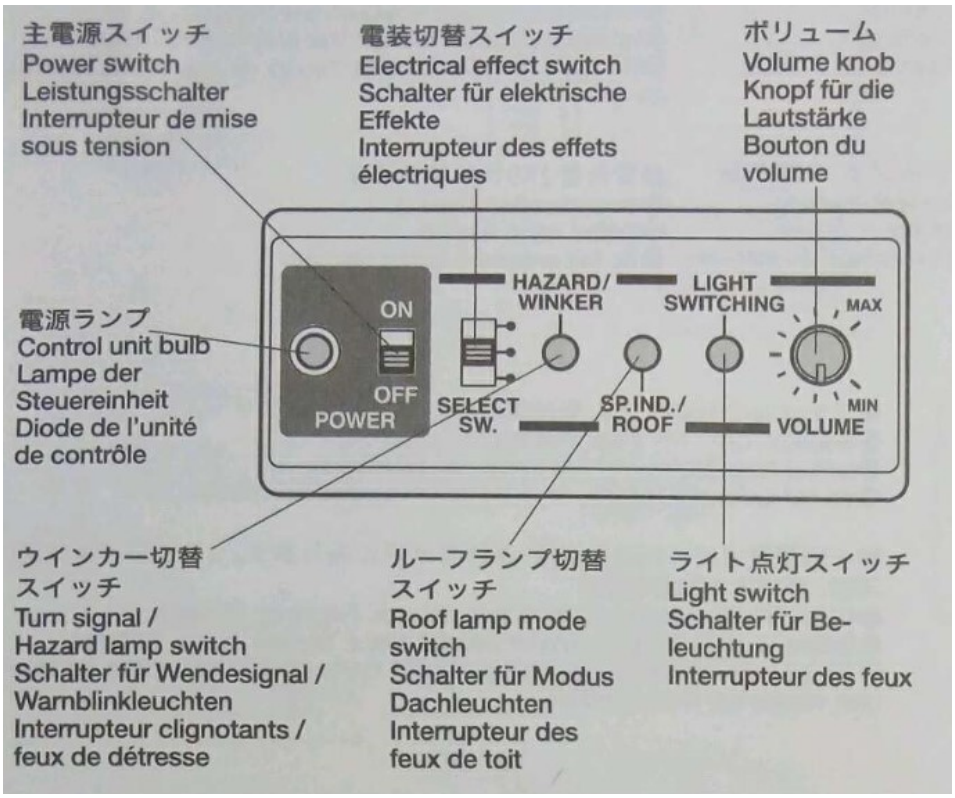
ソフトウェアの操作は PC 上で行いますが、PC の OS は Windows ではなく、Linux 系の Ubuntu です。Ubuntu の使い方は、参考書籍「ROS ロボット入門」の p19 あたりと第 C 章が参考になります。または、下記サイトなど。
https://www.sejuku.net/blog/84286

2. ハードウェア

キャビンをかぶせると、LIDAR の視野が狭くなるので、初めはかぶせないで評価したほうがや効率的です。

ラジコン

送信機・受信機共に電源を入れる。
車両の「電装切替スイッチ」は中央へ（上だとライト類だけ、下だと走行制御だけになる）。



車両(受信機)スイッチ BOX

制御モード

デフォルトはラジコン状態（受信機の信号がモータコントローラへ入る）。
ライト点灯モードをフォグランプ点灯にすると Photo Relay が切り替わり自動制御状態（Arduino の信号がモータコントローラへ入る）になる。

フォグランプ点灯にするには、２つの方法がある。
（１）車両スイッチ BOX の「ライト点灯スイッチ」により
（２）送信機の 3ch（右レバーの上下）のトリムをいっぱいまで下げた状態で、レバーを下へ操作
上記（１）または（２）の操作により、ライト点灯モードは次のように切り替わる。
「消灯」→「ルーフランプ点灯」→「ヘッドライト点灯」→「**フォグランプ点灯**」→消灯へ

緊急停止

走行中はライト点灯切替ができない。従って、自動運転中に車両を止めたい場合は、送信機の電源を切る。（送信機からの信号がないと走行できない仕様になっている）

3. PC 準備

Raspberry Pi より先に PC を起動する。

① 電源オン

ubuntu 起動

ログイン ID：test、パスワード：test で設定してあります。（自動ログインの設定がされていれば、入力の必要はありませんが。）

② Virtual Wifi Router　起動

ターミナルを開く。ターミナルを開くには右クリックメニュー（タッチパッドでは 2 本指クリック）の一番下の「端末を開く」、または Ctrl+Alt+T。

ターミナルで、virtual_router シェルスクリプトを実行

```
test@note:~$ ./virtual_router
```

これで、PC が router として機能する。

4. Raspberry Pi 準備

以下、RasPi と省略。

① 電源オン

横の microUSB 端子から 5V を供給すると起動する。

RasPi が virtual_wifi の SSID：note に接続されるように設定されています。

② PC から RasPi へ接続

PC で新しいターミナルを開き、RasPi が起動する（数十秒）待ってから、下記コマンドで ssh 接続

```
test@note:~$ ssh ubuntu@192.168.x.x        # アドレスチェックで得た IP アドレス
```

現在の設定では：192.168.12.109

（ssh_raspi というシェルスクリプトが作ってあるので、$ ./ssh_raspi　でも接続できます。）

初期パスワード：ubuntu

成功すると、コマンドプロンプトが下記に変わる。

```
ubuntu@ubuntu:~$
```

なお、RasPi の電源を切る際は、以下のコマンドで終了させるのが安全。

```
ubuntu@ubuntu:~$ sudo poweroff
```

ssh 接続を抜けるには、Cntl+d

5. 地図作成

車両をラジコンで手動操作し、環境マップを作成する方法。

① PC　roscore　起動

roslaunch で自動に roscore も起動するが、PC 側を MASTER に設定しているので、PC で先に roscore を立ち上げておいた方が確実。新たなターミナルにて、

```
test@note:~$ roscore
```

② RasPi　rplidar 起動

```
ubuntu@ubuntu:~$ roslaunch rplidar_ros rplidar.launch
```

③ PC　マッピング用 launch ファイルの起動

マッピング用 launch ファイルの起動。rviz も立ち上がる。

```
test@note:~$ roslaunch auto_drive_model lidar_mapping.launch
```

あとは、ラジコンで車両を動かして、マップを作成する。マップがうまくできれば、次の④マップ保存へ。

地図がうまく作成できない場合は、hector_mapping の設定を調整するのも良し。

また、測定時にトピックデータ（/scan　と/tf）を rosbag で記録すれば、計測した後で何度でもパラメータを変更して mapping を試すことができる。

データの記録方法

```
test@note:~$ rosbag record /scan /tf
```

データの再生方法

lidar を作動させる代わりに下記コマンドでトピックを再生し、マッピングを試せる。

```
test@note:~$ rosparam set use_sim_time true
test@note:~$ rosbag play –clock data_file_name.bag
```

（1 行目で、記録された時間を使用するように設定）

④ PC　マップ保存

rviz に表示されているマップを保存。あらたなターミナルにて、

```
test@note:~$ rosrun map_server map_saver
```

map.pgm と map.yaml の 2 つのファイルが作られる。ファイル名を変更する際は、yaml ファイルの 1 行目も変更すること。

6. 自律走行

mapping から継続する場合、ROS プログクラムは全部終了させ、roscore も再起動する。

roscore は実行ごとに再起動したほうが無難である。

① PC　roscore　起動

roslaunch で自動に roscore も起動するが、PC 側を MASTER に設定しているので、PC で先に roscore を立ち上げておいた方が確実。新たなターミナルにて、

```
test@note:~$ roscore
```

② RasPi 自律走行用 launch ファイル起動

```
ubuntu@ubuntu:~$ roslaunch auto_drive_model raspi.launch
```

以下の起動をしている。

- rplidar_node の起動
- rossrial の起動
- laser_frame の transform
- (odometry 計算は、laser_scan_matcher にさせている)
- laser_scan_matcher の起動
- amcl の起動

③ PC 自律走行用 launch ファイル起動

```
test@note:~$ roslaunch auto_drive_model auto_drive.launch
```

以下を起動している。

- map_server の起動（launch ファイルを編集して、作成した地図を map_file にする）
- move_base の起動
- robot_description の設定
- joint_state_pubrisher と robot_state_publisher の起動
- rviz の起動

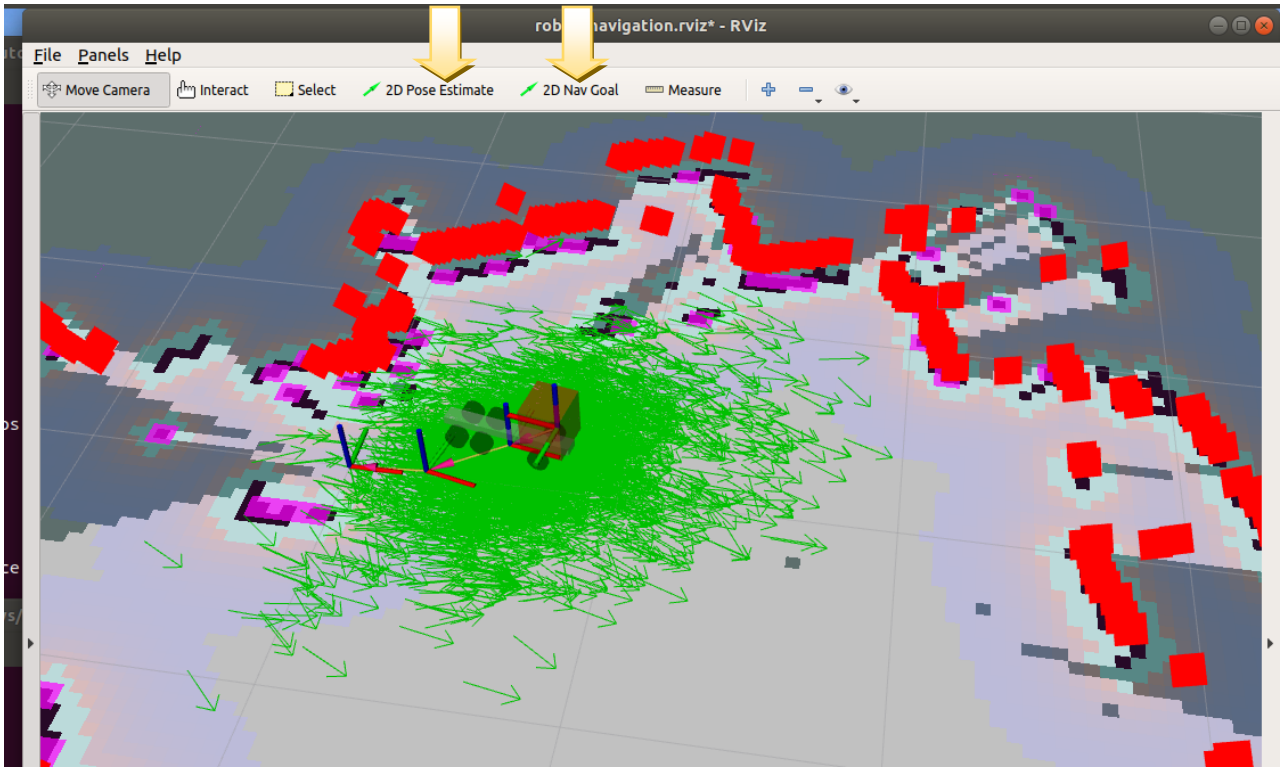
④ PC rviz で操作

A) 初期位置設定

初期位置がずれているようであれば、rviz メニューの「2D Pose Estimate」で合わせる。

B) ゴール地点設定

rviz メニューの「2D Nav Goal」でゴールを設定すれば、経路探索をして自律走行する。
（フォグランプ点灯状態にして、自動制御モードにしておく）



7. 追記

Error が発生した際

ROS を使っていると、いろいろとうまくいかない時があると思います。そんな時は WEB で検索すれば、往々にして世界中の誰かが解決策を提示していたりします。英語のサイトも恐れずに。そして、ゆくゆくはあなたも他のユーザーのための発信をしてください。

スキャンマッチングエラー：

スキャンマッチングや自位置推定がエラーになる要因として、下記があげられる。

① 炎天下等で Lidar がスキャン不能になる
② 路面の傾斜で、壁が見えなかったり、路面を見てしまったりする
③ 対象物が少なくて、マッチングできない（広い空間や後方が見えないと陥りやすい）
④ PC の処理が追い付いていない
⑤ 車両の移動・旋回が早すぎる（スキャン頻度 10Hz なので、限界はあまり高くない）

処理待ち：

PC や Raspberry Pi の処理能力を超える処理をさせようとすると、処理待ちが発生し制御が正常に働かなくなる。

top コマンドや uptime コマンドなどで、システム稼働状況を確認するべき