

NEDO 講座

SEED-R7: noid/mover

実機操作紹介

02/10/2023

Release: 0.0.1

目次

1. SEED-R7 noid/mover の特徴と起動手順	4
1.1. SEED-noid	4
1.2. SEED-lifter.....	4
1.3. SEED-mover	5
1.4. 起動手順.....	5
2. SEED-noid-mover との通信方法	7
2.1. マスターノードのセッティング	7
2.2. マスターノードの接続確認.....	9
3. ロボット実践操作.....	10
3.1. SEED-mover 単純移動.....	10
3.2. SEED-noid ハンド操作	10

1. SEED-R7 noid/mover の特徴と起動手順

SEED-R7 シリーズの中で今回扱っていくロボットタイプは [typeG] で3つのユニットとロボットハンド TRX が取り付けられたものです。扱っていくロボットタイプでは、USB ケーブルを PC に接続するだけで、モーションプレイ・遠隔操作・自律動作ができます。

typeG ロボット図



本講座で扱う簡易的な情報を以降で記述していますが、詳細な情報に関しましては、下記のリンクから確認することができます。

https://www.seed-solutions.net/sites/default/files/documents/OperationManual_SEED-R7_No.805.pdf

1.1. SEED-noid



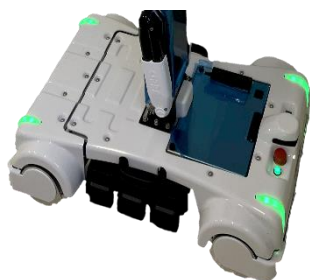
SEED-noid は上半身型ヒューマノイドでロボットハンド TRX を除いた重さを片腕のみで 1.5kg の物体を取り扱うことができます。スレンダーボディーで人の形にこだわった設計となっており、また低出力モーターで構築されている為、本質的な安全設計となっています。

1.2. SEED-lifter



SEED-lifter は上下の昇降と前後の移動が可能でリンク機構とスプリングのアシストのよって、10~20kg の積載ができます。また、スプリングを変更することで可搬質量の調整が可能となっています。

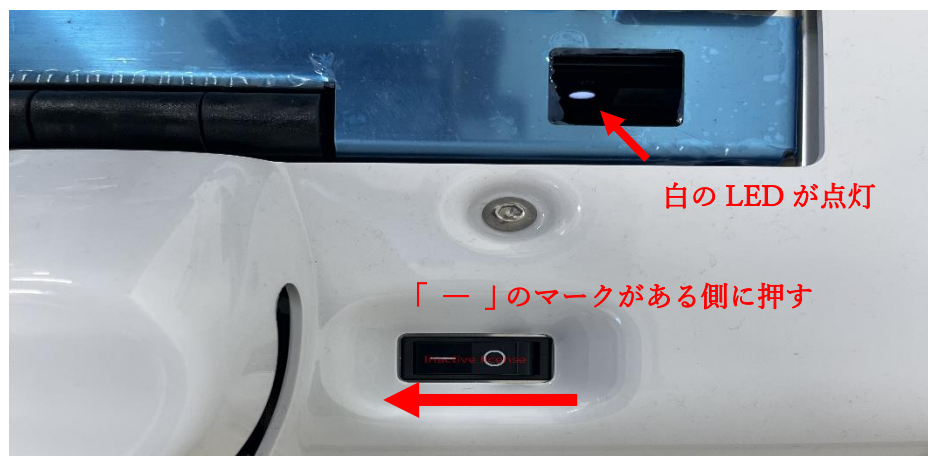
1.3. SEED-mover



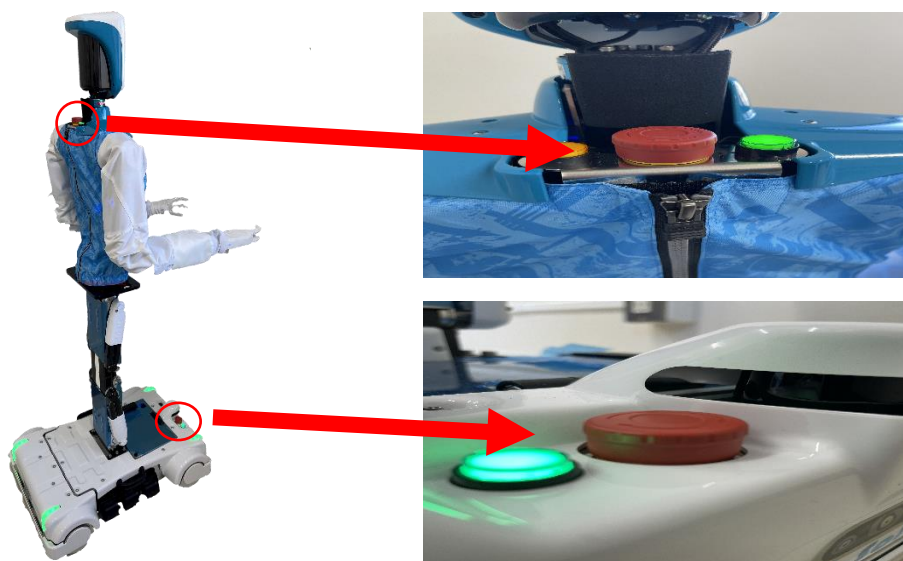
SEED-mover はメカナムホイールにより全方位置移動台車で最大 35kg までの運搬が可能です。自己位置推定が可能な SLAM 式で PC とレーザーセンサーが搭載されています。

1.4. 起動手順

SEED-Noid-mover の起動手順として、SEED-mover にある下記のスイッチを左に押します。そうすると白の LED が点灯し、他 4 か所の角の LED が点灯し点滅します。



続いて 2 か所にある非常停止ボタンの解除を行います。解除方法として、右にねじるか、もしくは上に引っ張るかです。（上に引っ張る方法を推薦します。）



非常停止ボタンの解除を行ったら、点滅している起動ボタン(緑のボタン)を押します。はじめに SEED-mover のある起動ボタンから押していきます。



起動ボタンを押すと点滅から
通常点灯に切り替わる



※注意 SEED-mover にある起動スイッチを押す際
上半身のアームを確認しながら押します

原点復帰が行われる為、突然アームが動き出します。

以上で起動の手順となります。

2. SEED-noid-mover との通信方法

SEED-mover には PC とアクセスポイントとなる WIFI ルーターが搭載されており、搭載されている PC には ROS 環境がセットされています。他の PC で SEED-mover にある PC に接続するには下記のアプリケーション



Remmina



Ultra VNC

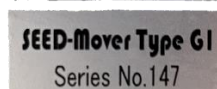
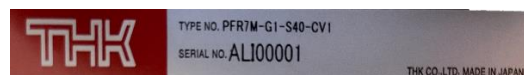


VNC Viewer

を使用するか、または、SSH を使用した通信方法と ROS のマスターノードを使用した方法の 3 つの接続タイプがあります。本講座ではマスターノード使用した通信を行って行きます。

2.1. マスターノードのセッティング

マスターノードのセッティングには、SEED-mover に搭載されている WIFI ルーターに接続する必要があります。また、SEED-mover 側の PC と 接続する側の PC にも ROS 環境を構築する必要があります。あらかじめ、SEED-mover の左側面に記載されている WIFI ルーターの SSID とパスワードをメモし、操作する PC 側で WIFI のセッティングを行います。



PASSWORD

SSID

WIFI ルーターに接続しましたら、下記のように IP の固定化を行います。

The screenshot shows the 'Wired' network configuration window. The 'IPv4 Method' is set to 'Manual'. Under 'Addresses', the first entry is 192.168.0.150 with netmask 255.255.255.0 and gateway 192.168.13.1. The 'DNS' and 'Routes' sections are also visible, both with 'Automatic' toggled 'ON'.

IP の固定化に続いて、操作側の PC の環境変数に ROS_IP と ROS_MASTER_URI の記述を行います。新規ターミナルから gedit で ~/.bashrc ファイルを開き、一番下に下記の記述を行い保存します。

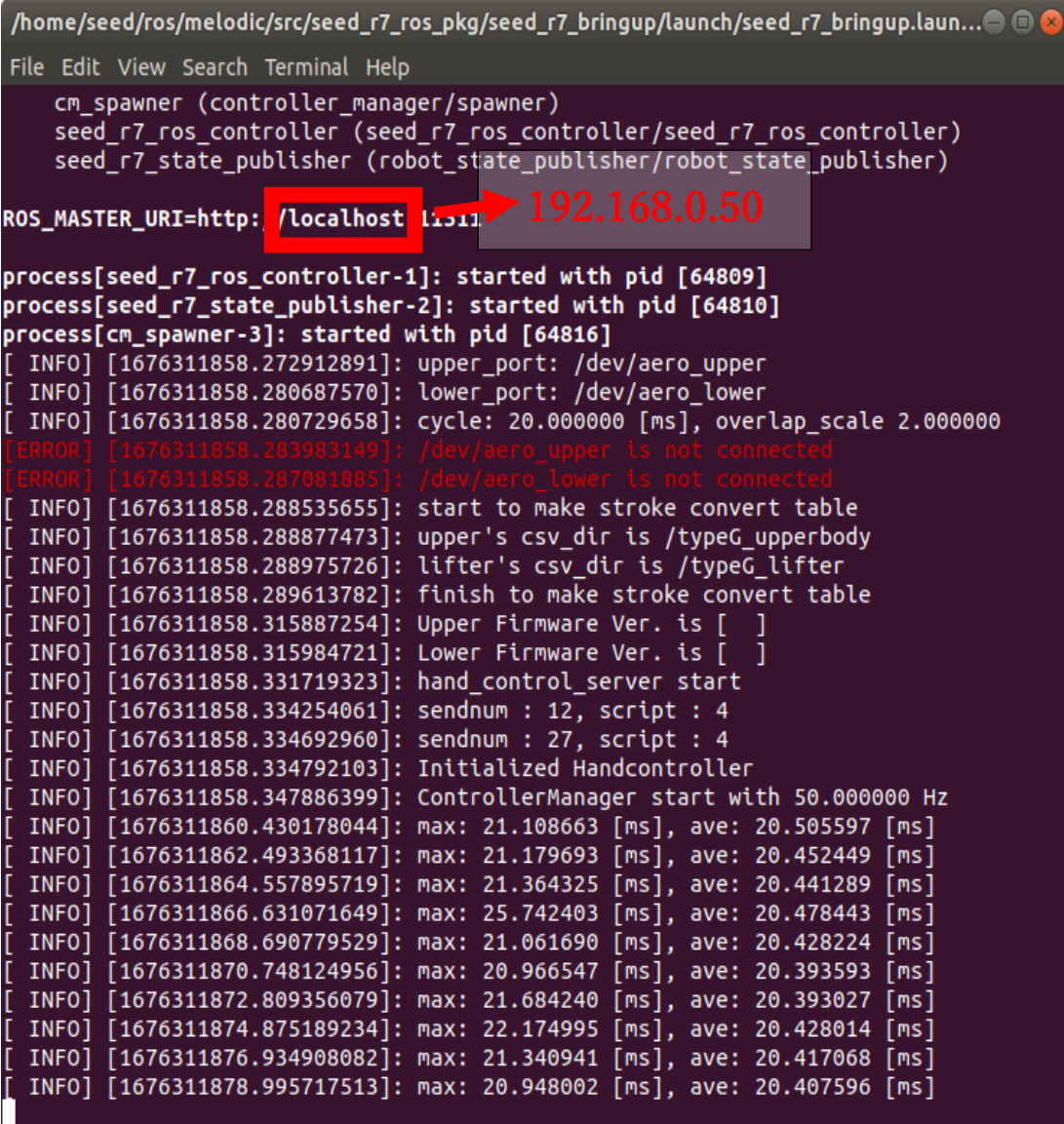
```
$ gedit ~/.bashrc
# Local IP Setting
export ROS_IP=192.168.0.150
# Ros Master IP Setting
export ROS_MASTER_URI=http://192.168.0.50:11311
```


2.2. マスターノードの接続確認

マスターノードの IP の固定化が完了しましたら、新規ターミナルから、

```
$ roslaunch seed_r7_bringup seed_r7_bringup.launch robot_model:=typeg
```

のノードを起動し、下記の赤い枠が設定したマスターノードの IP となっていれば接続可能です。



```
/home/seed/ros/melodic/src/seed_r7_ros_pkg/seed_r7_bringup/launch/seed_r7_bringup.laun...
File Edit View Search Terminal Help

cm_spawner (controller_manager/spawner)
seed_r7_ros_controller (seed_r7_ros_controller/seed_r7_ros_controller)
seed_r7_state_publisher (robot_state_publisher/robot_state_publisher)

ROS_MASTER_URI=http://localhost:11311 192.168.0.50

process[seed_r7_ros_controller-1]: started with pid [64809]
process[seed_r7_state_publisher-2]: started with pid [64810]
process[cm_spawner-3]: started with pid [64816]
[ INFO] [1676311858.272912891]: upper_port: /dev/aero_upper
[ INFO] [1676311858.280687570]: lower_port: /dev/aero_lower
[ INFO] [1676311858.280729658]: cycle: 20.000000 [ms], overlap_scale 2.000000
[ERROR] [1676311858.283983149]: /dev/aero_upper is not connected
[ERROR] [1676311858.287081885]: /dev/aero_lower is not connected
[ INFO] [1676311858.288535655]: start to make stroke convert table
[ INFO] [1676311858.288877473]: upper's csv_dir is /typeG_upperbody
[ INFO] [1676311858.288975726]: lifter's csv_dir is /typeG_lifter
[ INFO] [1676311858.289613782]: finish to make stroke convert table
[ INFO] [1676311858.315887254]: Upper Firmware Ver. is [ ]
[ INFO] [1676311858.315984721]: Lower Firmware Ver. is [ ]
[ INFO] [1676311858.331719323]: hand_control_server start
[ INFO] [1676311858.334254061]: sendnum : 12, script : 4
[ INFO] [1676311858.334692960]: sendnum : 27, script : 4
[ INFO] [1676311858.334792103]: Initialized Handcontroller
[ INFO] [1676311858.347886399]: ControllerManager start with 50.000000 Hz
[ INFO] [1676311860.430178044]: max: 21.108663 [ms], ave: 20.505597 [ms]
[ INFO] [1676311862.493368117]: max: 21.179693 [ms], ave: 20.452449 [ms]
[ INFO] [1676311864.557895719]: max: 21.364325 [ms], ave: 20.441289 [ms]
[ INFO] [1676311866.631071649]: max: 25.742403 [ms], ave: 20.478443 [ms]
[ INFO] [1676311868.690779529]: max: 21.061690 [ms], ave: 20.428224 [ms]
[ INFO] [1676311870.748124956]: max: 20.966547 [ms], ave: 20.393593 [ms]
[ INFO] [1676311872.809356079]: max: 21.684240 [ms], ave: 20.393027 [ms]
[ INFO] [1676311874.875189234]: max: 22.174995 [ms], ave: 20.428014 [ms]
[ INFO] [1676311876.934908082]: max: 21.340941 [ms], ave: 20.417068 [ms]
[ INFO] [1676311878.995717513]: max: 20.948002 [ms], ave: 20.407596 [ms]
```

3. ロボット実践操作

ロボット扱って SEED-mover の XY 方向の単純移動や向きの変更, 他にマッピング済みの状態での移動, ハンド周りの角度や上半身の操作を解説していきます. 事前に必要なパッケージをお使いの PC に取り入れます. 新規ターミナルを開き下記のコマンドでディレクトリに移動します

```
$ cd ~/ros/melodic/src
```

ディレクトリに移動したら, Github から

```
$ git clone https://github.com/skrjtech/seedNoidPkgs.git
```

のパッケージをクローンします. ビルドを行うパッケージに移動

```
$ cd seedNoidPkgs/package/simple_pkg
```

移動後にビルドを行う.

```
$ catkin build -this
```

以上で必要なパッケージの構築が完了します.

3.1. SEED-mover 単純移動

マッピング済みの環境で SEED-mover の操作を行います. マッピングされた状態で, X 方向に 1 メートルといった SEED-mover の移動操作を行います. プログラミング済みの python スクリプトを新規ターミナルで記述します.

```
$ rosrun simple_pkg one_meters_move.py
```

コマンドの実行により 10 秒かけて 1 メートル移動します. 向きや方向の変更は simple_pkg 内で用意されているスクリプトを起動することで向きが変えられます. 左方向の場合では,

```
$ rosrun simple_pkg angle_left_rotation.py
```

を実行することで左に回転します. また右の場合では,

```
$ rosrun simple_pkg angle_right_rotation.py
```

を実行することで右に回転します.

3.2. SEED-noid ハンド操作

上半身のハンド操作で, task_programmer のパッケージにいくつかの動きをとるプログラムが用意されています. その中で, あいさつするモーションのプログラムを起動していきます. 実機ロボットが起動状態であれば下記のコマンドを実行します.

```
$ rosrun task_programmer motion.py
```

もし実機ロボットが起動状態でなければ,

```
$ roslaunch seed_r7_bringup moveit.launch robot_model:=typeg
```

を実行し, その後, motion.py を実行します.

コマンドの起動後, 下記のイメージのようにロボットが動作します.

