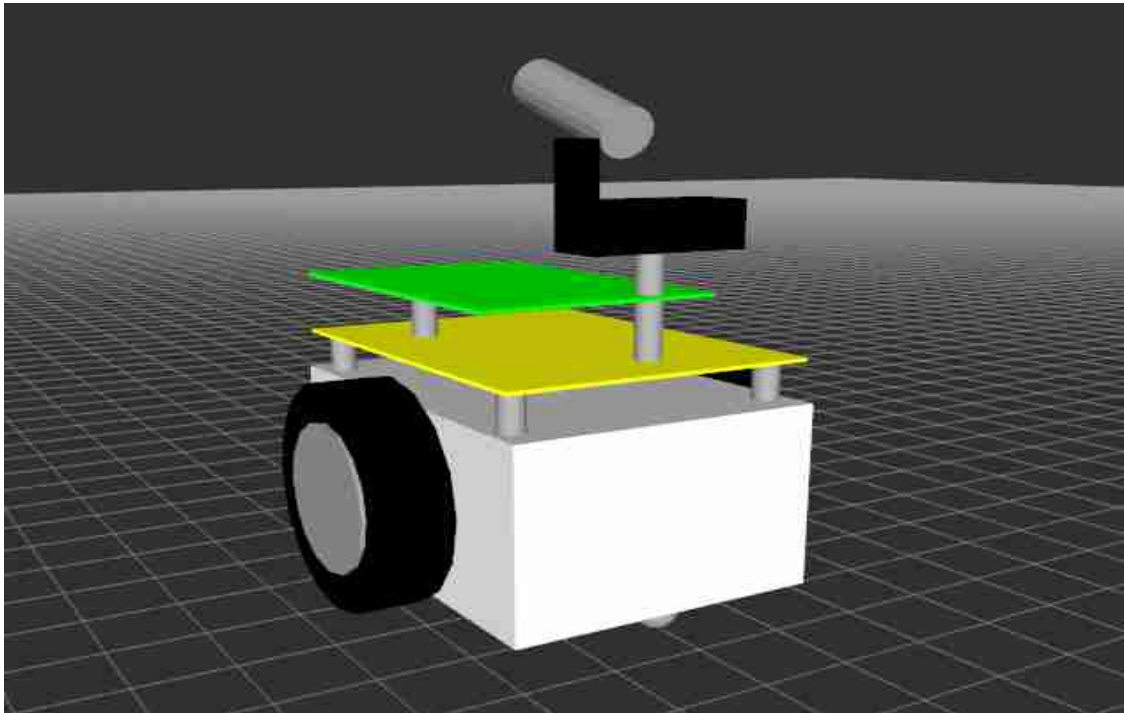
Turtlereal2でNavigation

ー GazeboとNavigationを動かすのに苦労した話 ー



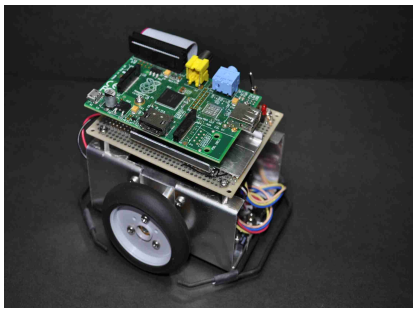
第八回ROS勉強会 2016/6/26

目次

1. TurtleReal2とは

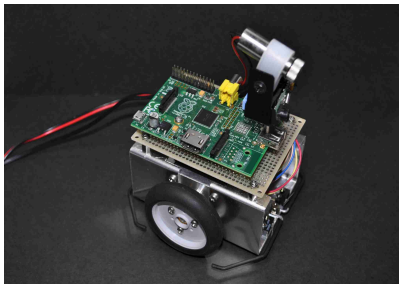
1万円台でNavigationが動くRaspberry Pi搭載のロボット

TurtleReal2

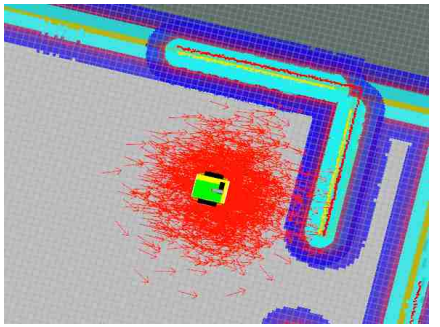


レーザレンジファインダ

TurtleReal2
+
レーザレンジファインダ

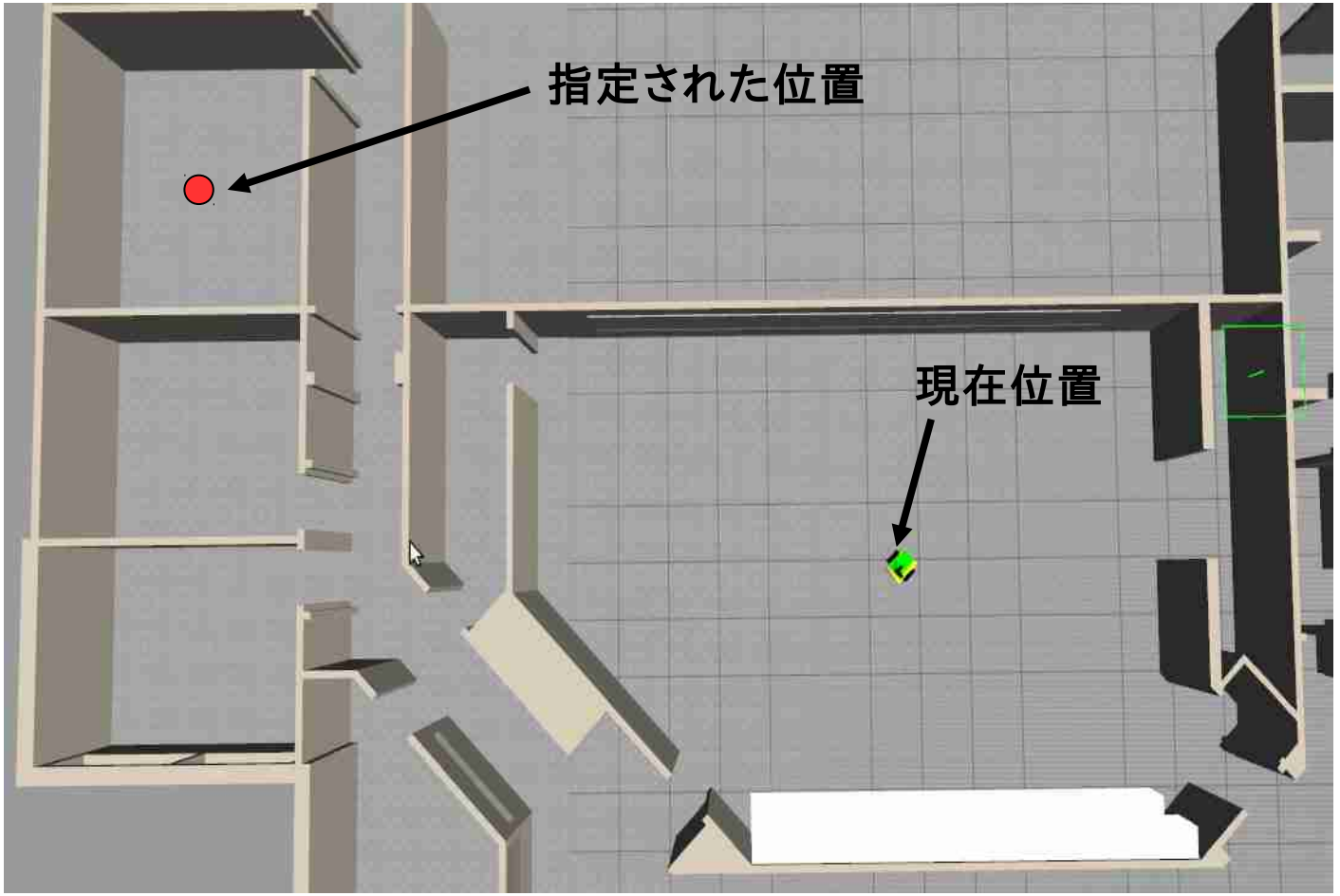


Navigationなど
ソフトウェアをメインと
した次のステップへ



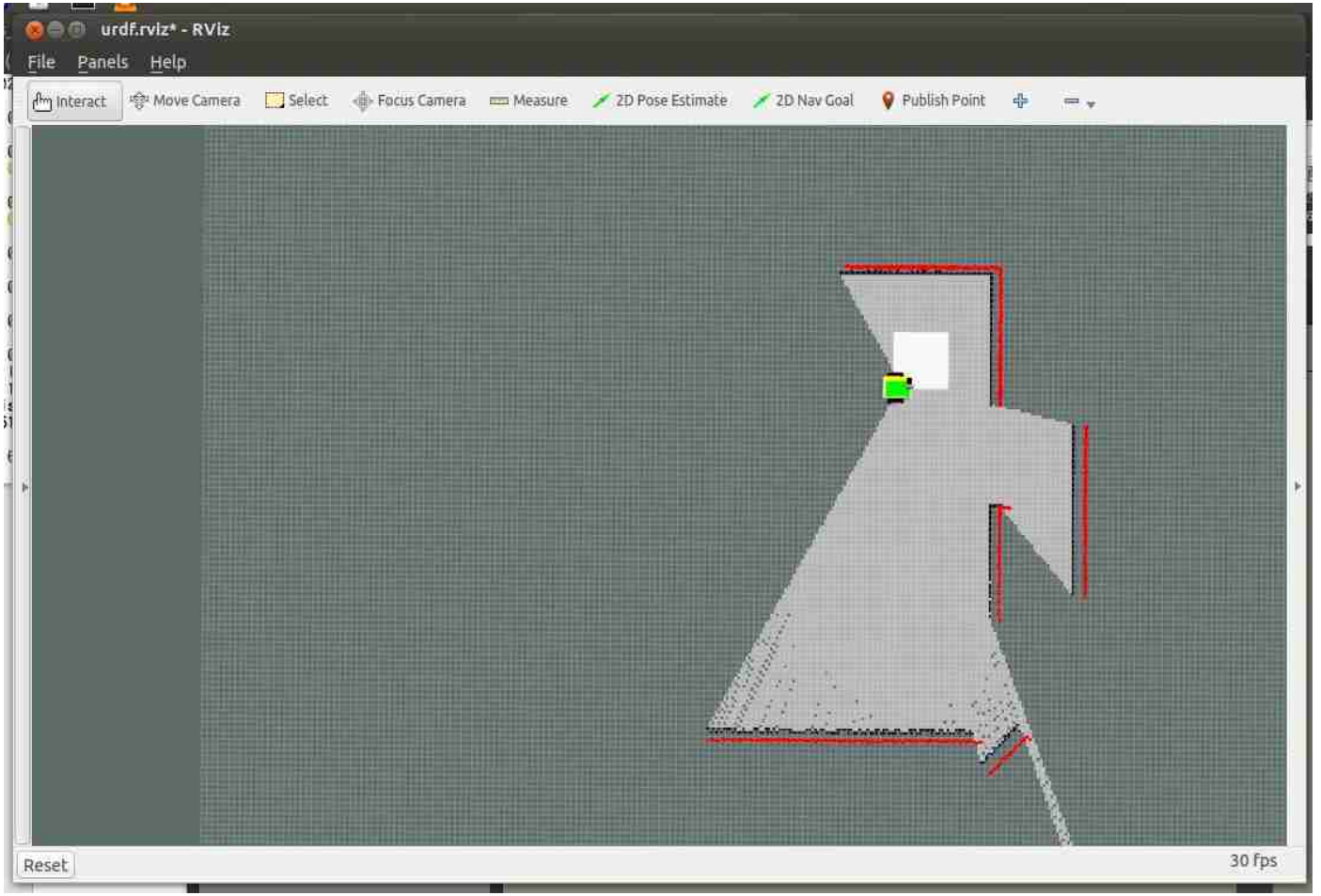
2. Navigation (1/4)

機能：現在位置から指定された位置への移動



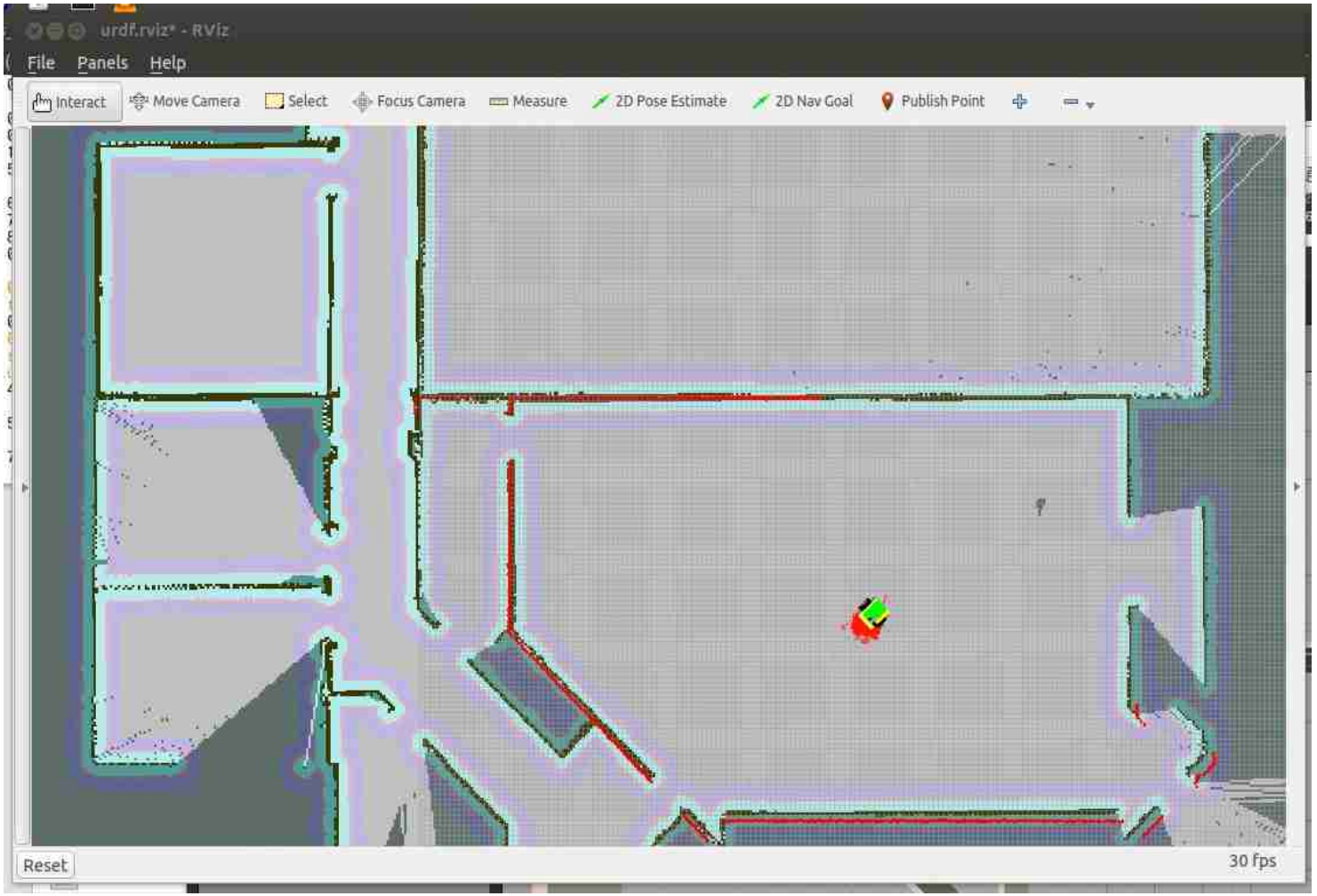
2. Navigation (2/4)

要素1：地図作成 (slam_gmapping)



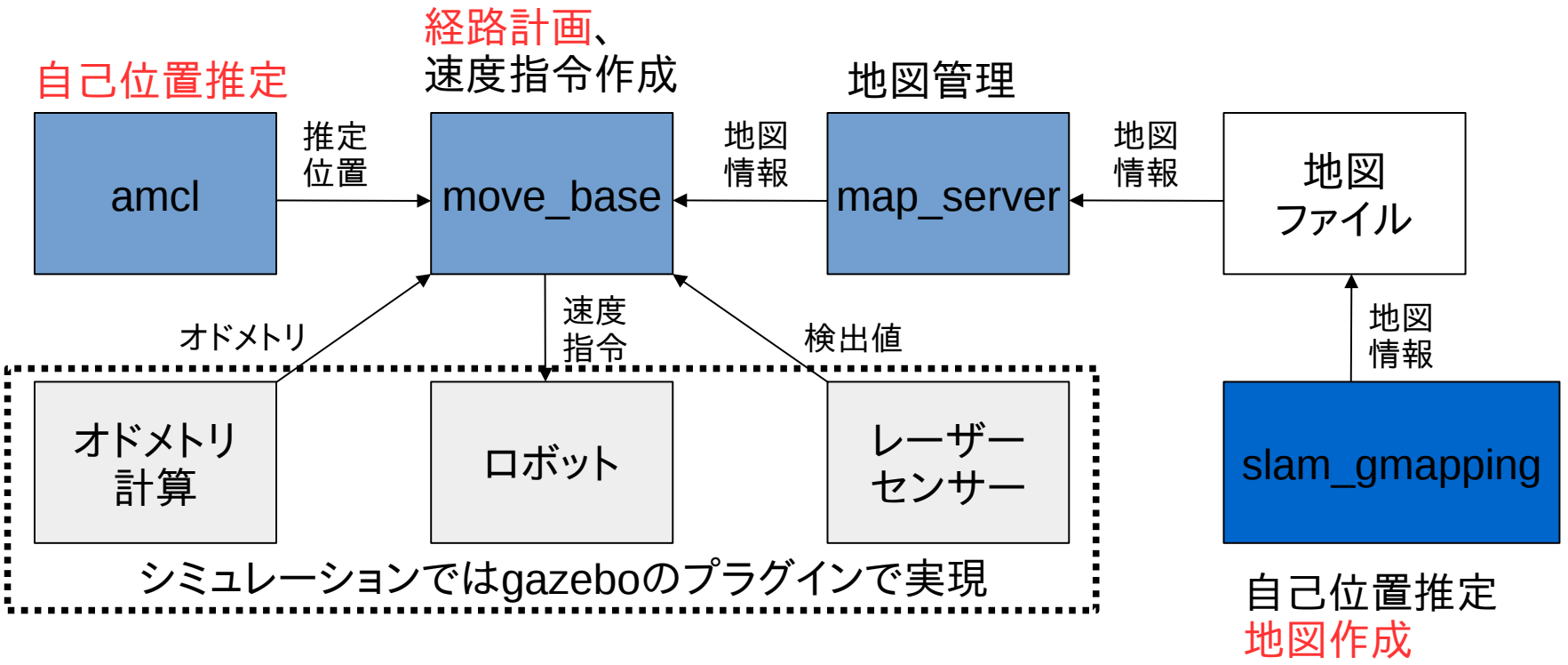
2. Navigation　(3/4)

要素2：自己位置推定と経路計画 (amcl, move_base)



2. Navigation　(4/4)

Navigationを実現するためのプログラム構成



amcl : Adaptive Monte Carlo Localization
slam : Simultaneous Localization And Mapping

3. そこに壁が…

Raspberry PiでNavigationパッケージのビルドに5昼夜！
Navigationのパラメータ設定も面倒らしい。

↓

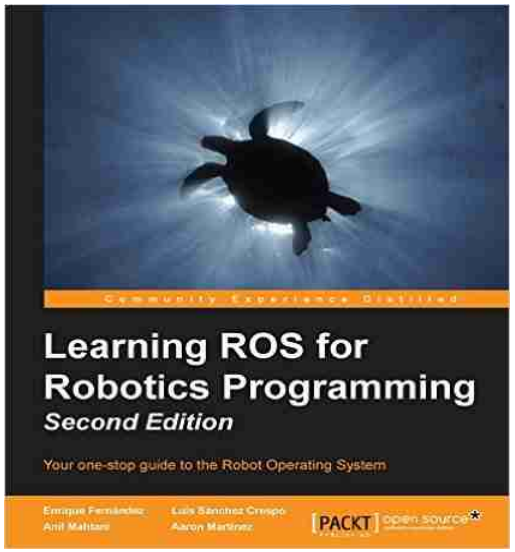
先にシミュレーションで試してみよう。

↓

動かない！！！

4. ネタ元

Navigation組み込みに関する日本語の情報が少ない。
この本とROS Wikiがネタ元



サンプルコードをダウンロードしても動かない!!!

しょうがないので最初からステップ・バイ・ステップで実施

5. Navigationを動かすステップ

(1) 必要なパッケージのインストール
(2) ジョイスティックでcmd_vel発行
(3) URDFロボットモデルをrvizに表示
(4) ロボットモデルをgazeboに表示
(5) gazeboにwillow garageを表示
(6) gazeboにレーザーセンサープラグインを組み込み
(7) gazeboにロボットのプラグインを組み込み動かす
(8) slam_gmappingを使ってマップを作成
(9) move_baseとamclを使ってロボットを動かす

ステップ(7)、(9)が大変!!!

6. 必要なパッケージのインストール

以下コマンドでパッケージをインストール

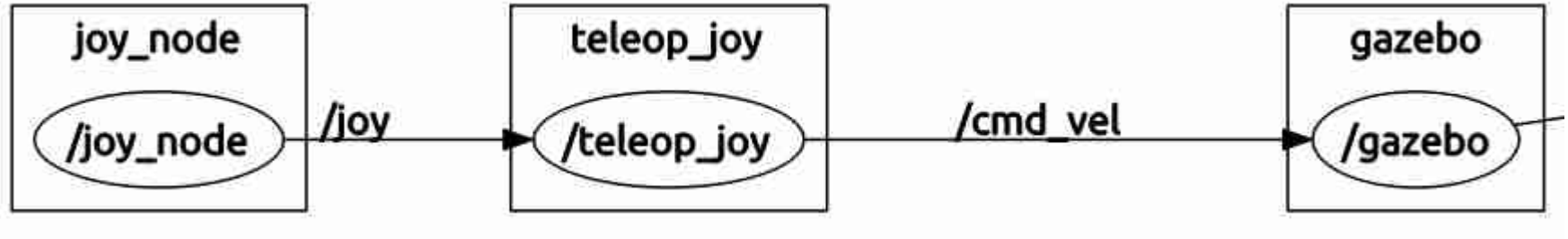
■ジョイスティック用ドライバ
$ sudo apt-get install ros-indigo-joystick-drivers

■地図作成
$ sudo apt-get install ros-indigo-slam_gmapping

■ナビゲーション
$ sudo apt-get install ros-indigo-navigation

7. ジョイスティックでcmd_vel発行

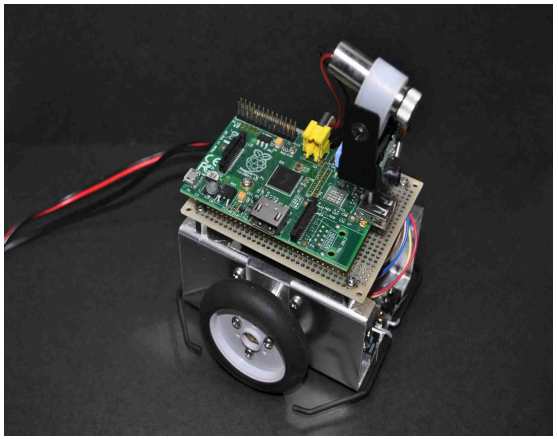
ロボットをジョイスティックで操作するため、以下のような環境をつくる。



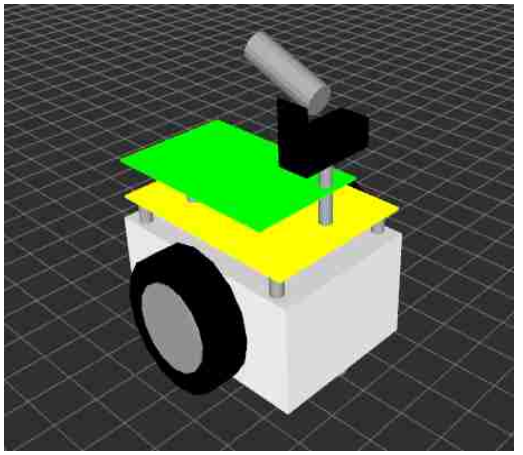
同じメーカーのジョイスティックでも吐き出すデータとボタンの割付が違ので注意！

8. URDFロボットモデルをrvizに表示

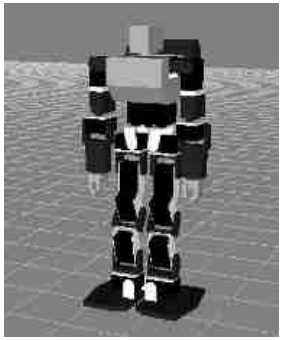
URDFでロボットモデルを書いてrvizに表示
以下のモデルで400行くらい、簡略化して100行～200行



モデル化



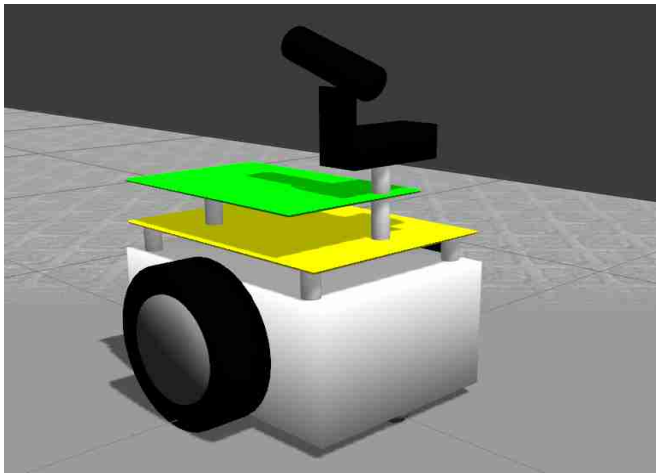
ちなみに…以下のようなロボットもURDFだけでモデル化可能



9. ロボットモデルをgazeboに表示

慣性行列、衝突検出、摩擦係数、gazebo用表面色設定をURDFファイルに追加

```
<gazebo reference="wheel_1_link">
  <mu1 value="10.0"/>
  <mu2 value="1.0"/>
  <kp value="10000000.0"/>
  <kd value="1.0"/>
  <fdir1 value="1 0 0"/>
  <material>Gazebo/Black</material>
  <turnGravityOff>false</turnGravityOff>
</gazebo>
```

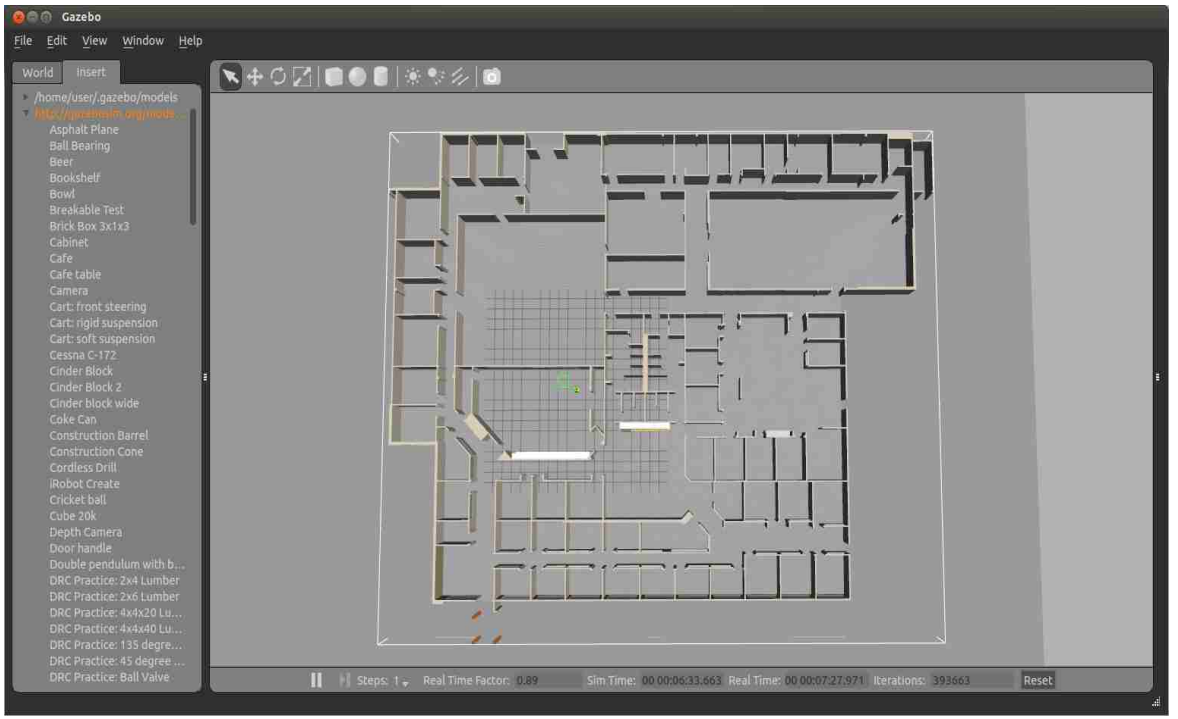


```
<collision>
  <geometry>
    <box size="${base_size_x} ${base_size_y} ${base_size_z}"/>
  </geometry>
</collision>
<inertial>
  <mass value="${base_mass}"/>
  <inertia ixx="${1.0 / 12.0 * base_mass * (base_size_y * base_size_y + base_size_z * base_size_z)}"
ixy="0.0" ixz="0.0" iyy="${1.0 / 12.0 * base_mass * (base_size_x * base_size_x + base_size_z *
base_size_z)}" iyz="0.0" izz="${1.0 / 12.0 * base_mass * (base_size_x * base_size_x + base_size_y *
base_size_y)}"/>
</inertial>
```

10. gazeboにwillow garageを表示

走行環境としてgazeboへwillow garageオフィスのモデルをロードするようにlaunchファイルに追加

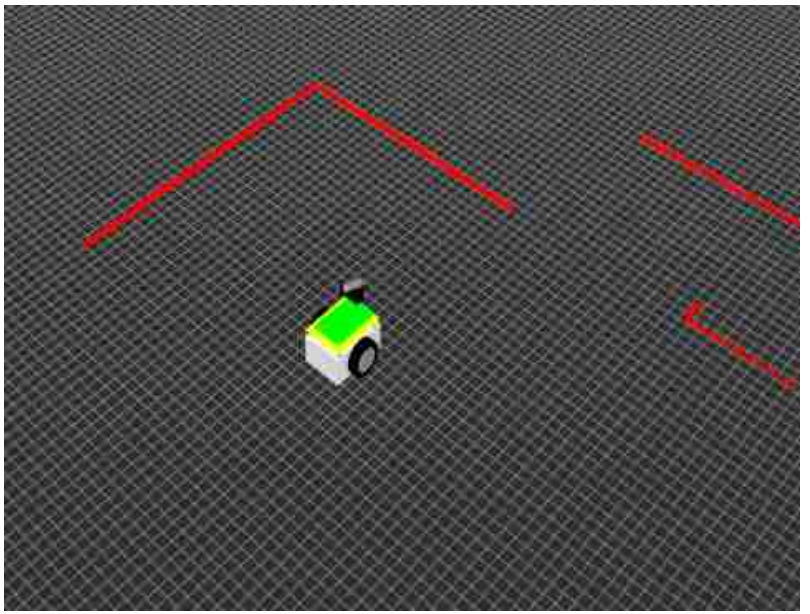
```
<include file="$(find gazebo_ros)/launch/willowgarage_world.launch"/>
```



11. gazeboにレーザーセンサープラグインを組み込み

レーザーセンサプラグインをURDFファイルに追加

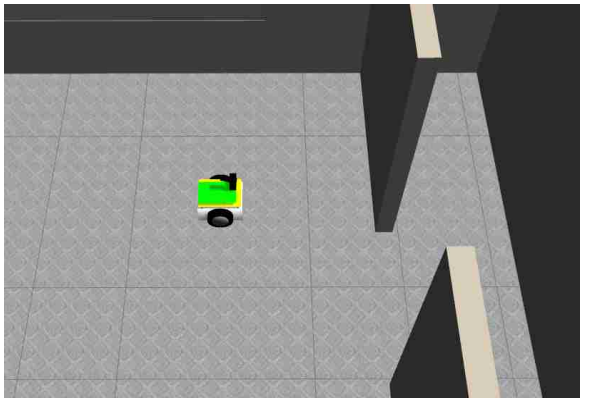
```
<gazebo reference="laser">
  <sensor type="ray" name="head_hokuyo_sensor">
    <pose>0 0 0 0 0 0</pose>
    <visualize>false</visualize>
    <update_rate>10</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>667</samples>
          <resolution>1</resolution>
          <min_angle>-2.094395</min_angle>
          <max_angle>2.094395</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>${0.05 * model_scale}</min>
        <max>${4.5 * model_scale}</max>
        <resolution>0.01</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>
    </ray>
    <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
      <topicName>/scan</topicName>
      <frameName>laser</frameName>
    </plugin>
  </sensor>
</gazebo>
```



12. gazeboにロボットのプラグインを組み込み動かす

gazebo内でロボットの動きをシミュレーションするプラグインをURDFファイルに追加

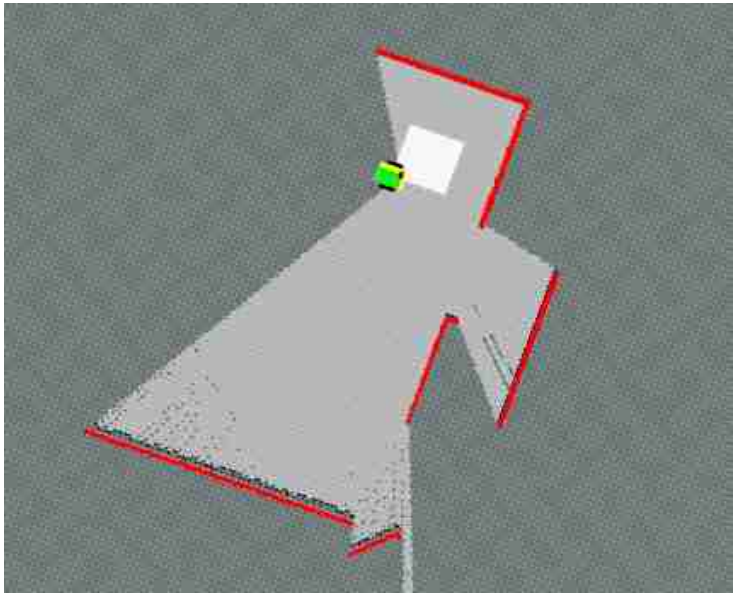
```
<gazebo>
  <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
    <publishWheelTF>false</publishWheelTF>
    <robotNamespace>/</robotNamespace>
    <publishTf>1</publishTf>
    <publishWheelJointState>false</publishWheelJointState>
    <alwaysOn>true</alwaysOn>
    <updateRate>100.0</updateRate>
    <leftJoint>wheel_2_joint</leftJoint>
    <rightJoint>wheel_1_joint</rightJoint>
    <wheelSeparation>${base_size_y + wheel_thickness + 2 * wheel_gap}</wheelSeparation>
    <wheelDiameter>${wheel_radius * 2.0}</wheelDiameter>
    <broadcastTF>1</broadcastTF>
    <wheelTorque>60</wheelTorque>
    <wheelAcceleration>1.8</wheelAcceleration>
    <commandTopic>cmd_vel</commandTopic>
    <odometryFrame>odom</odometryFrame>
    <odometryTopic>odom</odometryTopic>
    <robotBaseFrame>base_footprint</robotBaseFrame>
  </plugin>
</gazebo>
```



13. slam_gmappingを使ってマップを作成

slam_gmappingノードの起動をlaunchファイルに追加

```
<node name="slam_gmapping" pkg="gmapping" type="slam_gmapping">
  <param name="base_link" value="base_footprint"/>
</node>
```



14. move_baseとamclを使ってロボットを動かす (1/3)

slam_gmappingをlaunchファイルから外し、ロボットの軌道制御を行うmove_baseとamcl、地図管理のmap_severの起動を追加

```
<node name="map_server" pkg="map_server" type="map_server" args="$(find
turtlereal)/maps/test2.yaml"/>

<include file="$(find amcl)/examples/amcl_diff.launch" >
</include>

<node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
  <param name="controller_frequency" value="10.0"/>
  <param name="controller_patiente" value="15.0"/>
  <rosparam file="$(find turtlereal)/launch/costmap_common_params.yaml" command="load"
ns="global_costmap" />
  <rosparam file="$(find turtlereal)/launch/costmap_common_params.yaml" command="load"
ns="local_costmap" />
  <rosparam file="$(find turtlereal)/launch/local_costmap_params.yaml" command="load" />
  <rosparam file="$(find turtlereal)/launch/global_costmap_params.yaml" command="load" />
  <rosparam file="$(find turtlereal)/launch/base_local_planner_params.yaml" command="load" />
</node>
```

14. move_baseとamclを使ってロボットを動かす (2/3)

amclの起動と設定 (amcl_diff.launch)

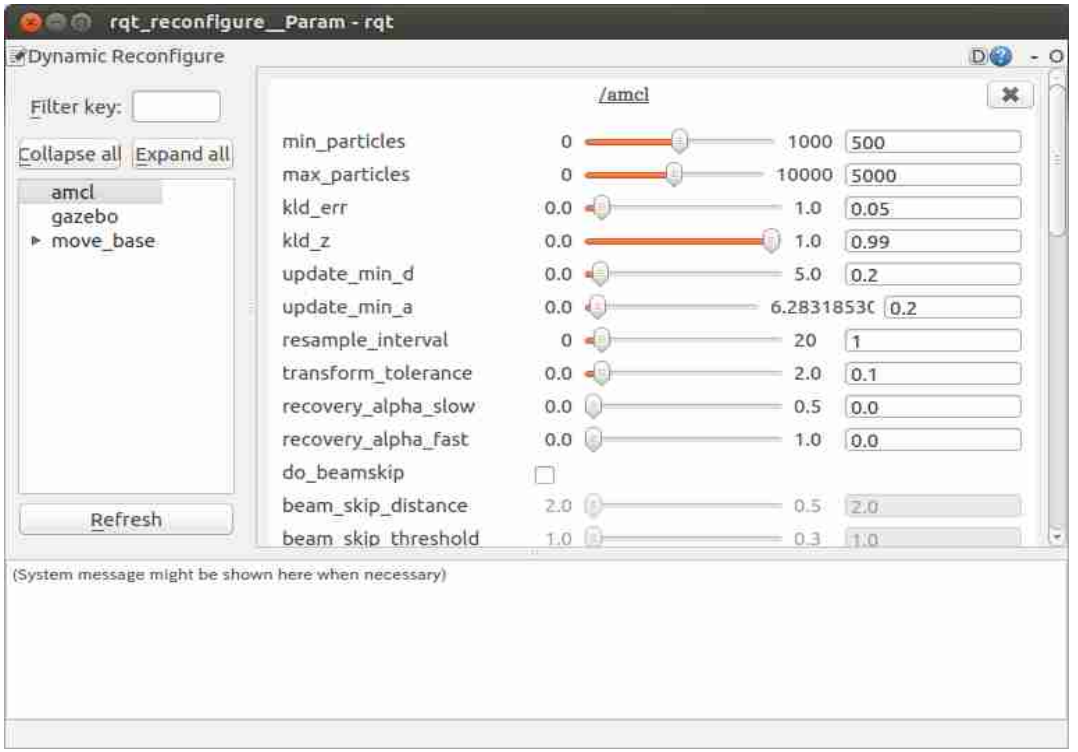
```
<launch>
<node pkg="amcl" type="amcl" name="amcl" output="screen">
  <!-- Publish scans from best pose at a max of 10 Hz -->
  <param name="odom_model_type" value="diff"/>
  <param name="odom_alpha5" value="0.1"/>
  <param name="transform_tolerance" value="0.2" />
  <param name="gui_publish_rate" value="10.0"/>
  <param name="laser_max_beams" value="30"/>
  <param name="min_particles" value="500"/>
  <param name="max_particles" value="5000"/>
  <param name="kld_err" value="0.05"/>
  <param name="kld_z" value="0.99"/>
  <param name="odom_alpha1" value="0.8"/>
  <param name="odom_alpha2" value="0.8"/>
  <!-- translation std dev, m -->
  <param name="odom_alpha3" value="0.8"/>
  <param name="odom_alpha4" value="0.2"/>
  <param name="laser_z_hit" value="0.5"/>
  <param name="laser_z_short" value="0.05"/>
  <param name="laser_z_max" value="0.05"/>
  <param name="laser_z_rand" value="0.5"/>
  <param name="laser_sigma_hit" value="0.2"/>
  <param name="laser_lambda_short" value="0.1"/>
  <param name="laser_lambda_short" value="0.1"/>
  <param name="laser_model_type" value="likelihood_field"/>
  <!-- <param name="laser_model_type" value="beam"/> -->
  <param name="laser_likelihood_max_dist" value="2.0"/>
  <param name="update_min_d" value="0.2"/>
  <param name="update_min_a" value="0.2"/>
  <param name="odom_frame_id" value="odom"/>
  <param name="resample_interval" value="1"/>
  <param name="transform_tolerance" value="0.1"/>
  <param name="recovery_alpha_slow" value="0.0"/>
  <param name="recovery_alpha_fast" value="0.0"/>
</node>
</launch>
```

14. move_baseとamclを使ってロボットを動かす (3/3)

move_base設定(base_local_planner_params.yaml)

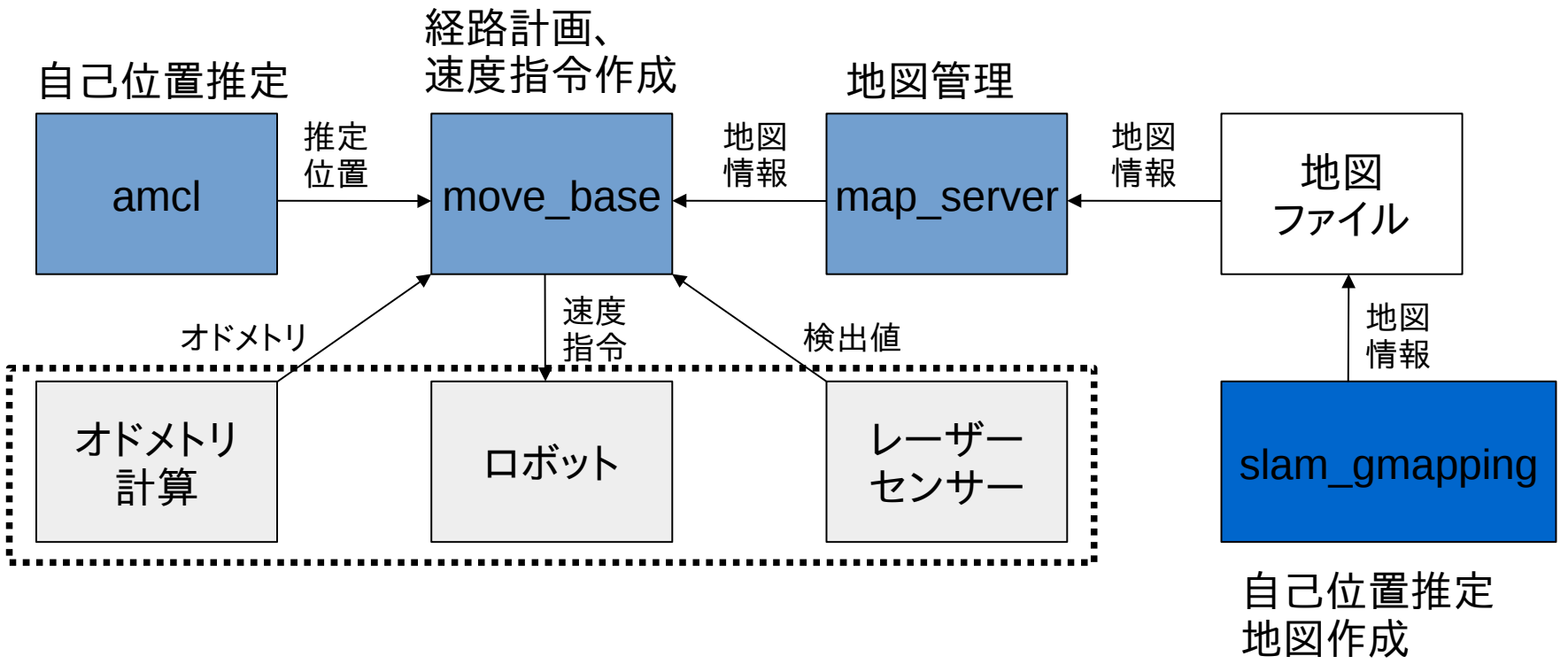
```
TrajectoryPlannerROS:
  max_vel_x: 0.4
  min_vel_x: 0.01
  max_rotational_vel: 0.4
  min_in_place_rotational_vel: 0.4
  min_in_place_vel_theta: 0.01
  max_vel_theta: 20.0
  min_vel_theta: -20.0
  acc_lim_th: 0.5
  acc_lim_x: 0.5
  acc_lim_y: 0.0

  holonomic_robot: false

  path_distance_bias: 3.0
  goal_distance_bias: 0.5
```

パラメータはrqt_reconfigureを使って動的に調整可能



15. 次のステップ

Turtlereal2で動かすために点線部分のプログラムを作成



ご静聴ありがとうございました。

詳しくは「安曇野 ROS」で検索！