知能機械設計演習
Practicum in Intelligent Machine Design

MATLAB/SimulinkとROS 2週目
MATLAB/Simulink and ROS 2週目

生命体工学研究科
人間知能システム工学専攻
s-yasukawa@brain.kyutech.ac.jp
安川 真輔
Shinsuke Yasukawa

Outline

13:00-13:30　前回の復習と今回の演習について

13:30-14:10　1班/4班　(7班)

14:10-15:00　2班/5班　(7班)

(間に10分休憩)

15:00-15:40　3班/6班　(7班)

15:40-16:10　Stateflowについて

残りの班

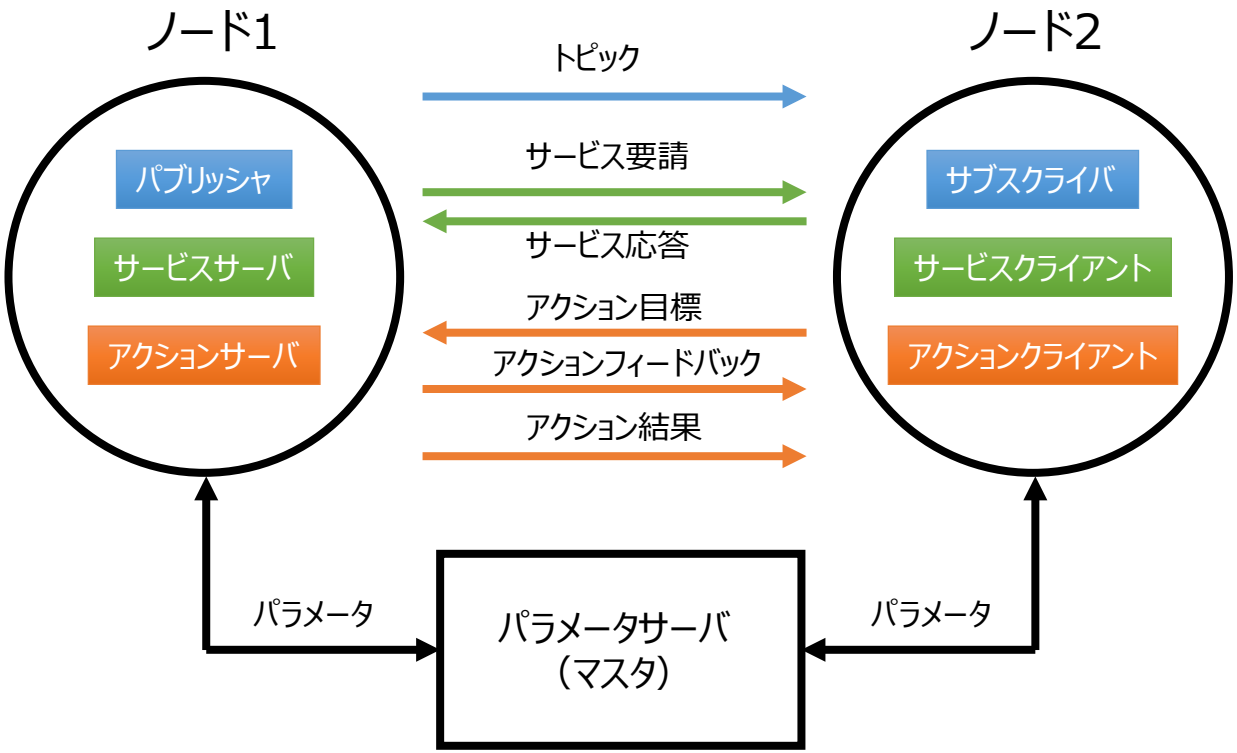
MATLAB/ROS連携
（前回の資料の続き）

1. MatlabからGazebo上の対象の制御
2. 障害物回避及びオブジェクトの追跡と追従

メッセージ通信

A. プログラムの再利用性を高めるため、機能や目的で細分化したノードを用いる。
B. ノード間のメッセージ通信
1. 単方向非同期通信方式のトピック(センサデータの取得など、連続的なデータ通信が必要なとき)
2. 要請と応答から構成される双方向同期通信方式のサービス（ロボットの状態の確認など、要請に対してすぐに応答が必要なとき）
3. 目標、結果、およびフィードバックで構成される双方向非同期通信方式のアクション
（目標への移動を支持されたロボットの現在位置など要請に対する応答に遅延がある場合や、処理中の中間結果が必要なとき）
がある.

メッセージ通信（トピック、サービス、アクション、パラメータ）



コンピュテーショングラフレベル（通信・演算に関するレベル）

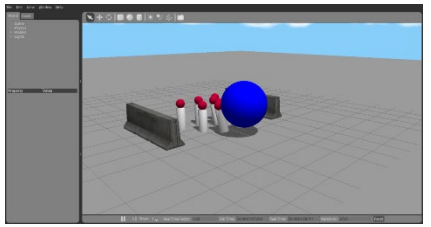
1. MatlabからGazebo上の対象の制御

1. Gazebo からのモデルおよびシミュレーションの特性の読み取り

https://jp.mathworks.com/help/robotics/examples/read-model-and-simulation-properties-in-gazebo.html

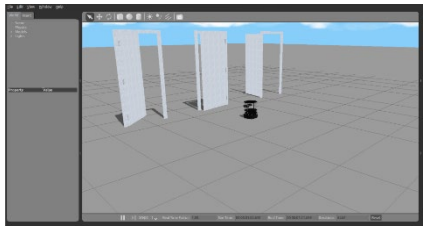
2. Gazebo でのオブジェクトの追加、作成、および削除

https://jp.mathworks.com/help/robotics/examples/add-build-and-remove-objects-in-gazebo.html



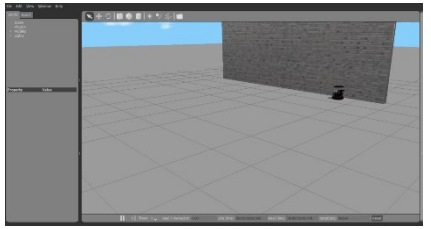
3. Gazebo での力とトルクの適用

https://jp.mathworks.com/help/robotics/examples/apply-forces-and-torques-in-gazebo.html



4. シミュレーションでのロボットの自律性のテスト

https://jp.mathworks.com/help/robotics/examples/test-autonomy-in-simulation.html



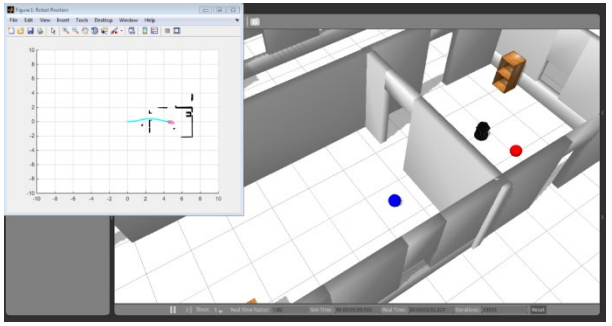
(その他)

rosbag ログファイルの操作

https://jp.mathworks.com/help/robotics/examples/work-with-rosbag-logfiles.html

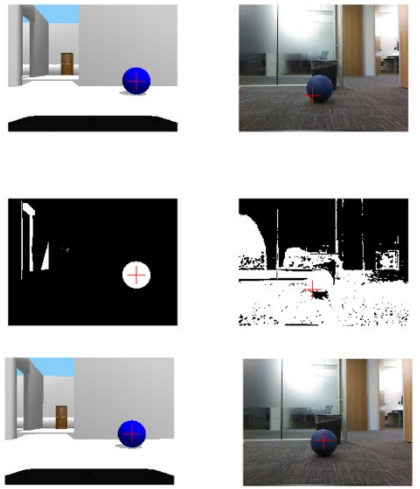
2. 障害物回避及びオブジェクトの追跡と追従

TurtleBot による障害物回避



https://jp.mathworks.com/help/robotics/examples/obstacle-avoidance-using-turtlebot-robot.html

オブジェクトの追跡と追従



https://jp.mathworks.com/help/robotics/examples/track-and-follow-an-object.html

*シミュレーション側のみ

3. Simulink coderによるROSノードの生成

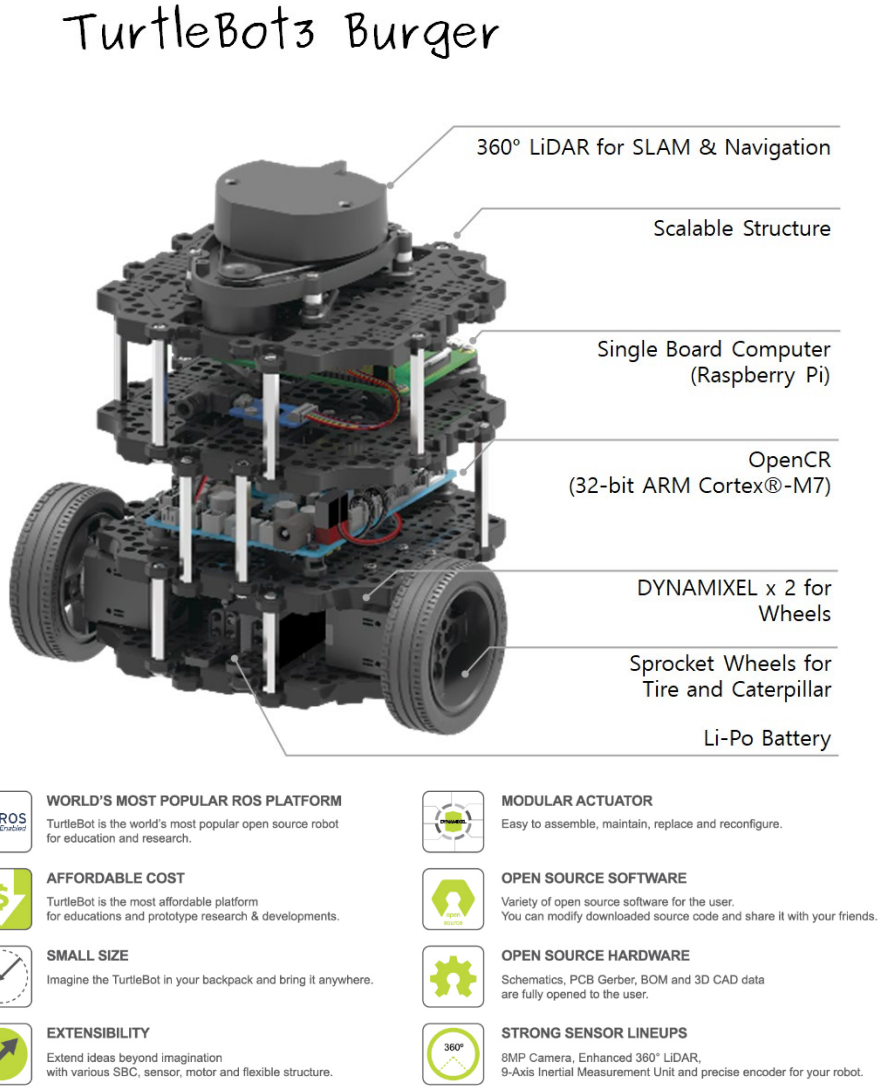
Simulink® からのスタンドアロン ROS ノードの生成

https://jp.mathworks.com/help/robotics/examples/generate-a-standalone-ros-node-in-simulink.html

Turtlebot3

移動ロボット turtlebot3 -1/2-

Items	Burger
Maximum translational velocity	0.22 m/s
Maximum rotational velocity	2.84 rad/s (162.72 deg/s)
Maximum payload	15kg
Size (L x W x H)	138mm x 178mm x 192mm
Weight (+ SBC + Battery + Sensors)	1kg
Threshold of climbing	10 mm or lower
Expected operating time	2h 30m
Expected charging time	2h 30m
SBC (Single Board Computers)	Raspberry Pi 3 Model B and B+
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Remote Controller	-
Actuator	Dynamixel XL430-W250
LDS(Laser Distance Sensor)	360 Laser Distance Sensor LDS-01
Camera	-
IMU	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis
Power connectors	3.3V / 800mA 5V / 4A 12V / 1A
Expansion pins	GPIO 18 pins Arduino 32 pin
Peripheral	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
Dynamixel ports	RS485 x 3, TTL x 3
Audio	Several programmable beep sequences
Programmable LEDs	User LED x 4
Status LEDs	Board status LED x 1 Arduino LED x 1 Power LED x 1
Buttons and Switches	Push buttons x 2, Reset button x 1, Dip switch x 2
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
PC connection	USB
Firmware upgrade	via USB / via JTAG
Power adapter (SMPS)	Input : 100-240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A



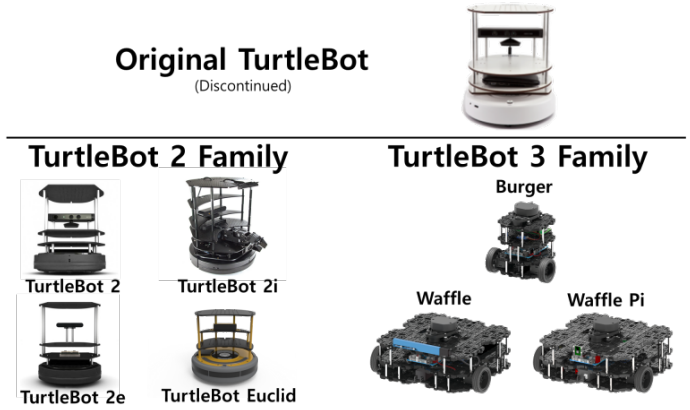
http://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#specifications

移動ロボット turtlebot3 -2/2-

TurtleBotシリーズ

・ROSの標準ロボットプラットフォーム

・教育用コンピュータプログラミング言語logoの使用例としてのカメ型ロボットに大きく影響を受けている



TurtleBot3

オープンソースハードウェア

・主な構成要素

アクチュエータDINAMIXEL*2:駆動用
センサLDS：SLAMとナビゲーション用
SBC(Single Board Computer)：ROS動作可能
リチウムポリマーバッテリ:11v

オープンソースソフトウェア

・OpenCRのファームウェア(turtlebot_core)

・4つのROSパッケージ

turtlebot3
 turtlebot3のロボットモデルに関するファイル
turtlebot3_msg
 メッセージファイルの集合
turtlebot3_simulations
 シミュレーションに関する機能
turtlebot3_applications
 アプリケーション例

ROBOTIS TurtleBot3 Burger

https://cad.onshape.com/documents/2586c4659ef3e7078e91168b/w/14abf4cb615429a14a2732cc/e/9ae9841864e78c02c4966c5e

セットアップ

TurtleBot

ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
ROS_HOSTNAME = IP_OF_TURTLEBOT

・ロボットの動作
・センサデータの取得

Remote PC

ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
ROS_HOSTNAME = IP_OF_REMOTE_PC

・マスタの役割
・遠隔操作, SLAM, ナビゲーションなどの
上位レベルのコントローラ

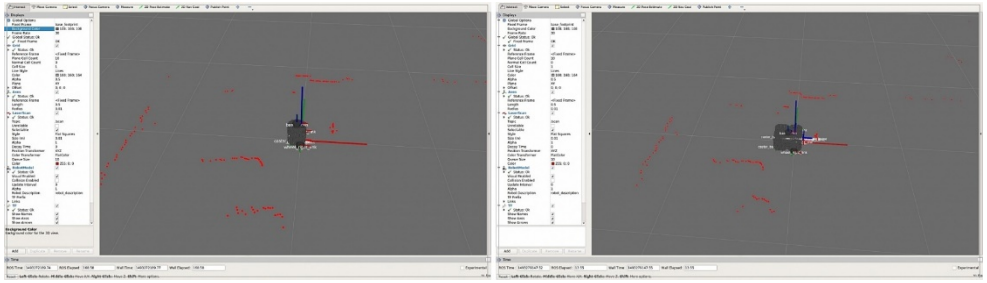
SSHを用いてRemotePCからTurtleBotSBCに接続して作業する.
SSH pi@

立ち上げ方法とその確認

[Remote PC]
（手順①）
$ roscore

[TurtleBot SBC(Single Board Computer)]
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch (手順②)

[Remote PC] TurtleBot3の表示
(手順③ Load a TurtleBot3 on Rviz)
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_bringup turtlebot3_model.launch



[Remote PC] TurtleBot3のキーボードによる操作
(手順④)
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

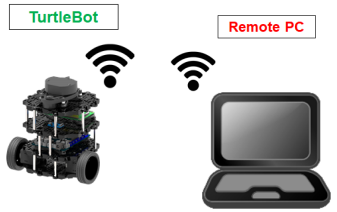
[Remote PC] トピック通信の可視化
(手順⑤)
一旦turtlebot3_teleopを終了
$ rqt_graph
$ rostopic list
もう一度turtlebot3_teleopを立ち上げ
$ rqt_graph

＊端末を実行するためのショートカットキー
Ctrl- Alt- T

Launch
1. turtlebot3_core：OpenCRと通信する
2. hls_lfcd_lds_driveノード：LDSの駆動

・ロボットの各関節におけるrfが
RGBの座標軸と共に表示

・ロボット搭載のLDSにより取得された障害物
の距離データも表示



サブスクライブトピックとパブリッシュトピックの確認 -1/2-

○サブスクライブトピックによるロボット制御

[Remote PC]
停止
$ rostopic pub /motor_power std_msg/Bool “data: 0”
TurtleBot3を動かすための速度のパブリッシュ
$ rostopic pub /cmd_vel geometry_msgs/Twist “linear:
x : 0.02
y : 0.0
z : 0.0
angular:
x : 0.0
y : 0.0
z : 0.0”

$ rostopic pub /cmd_vel geometry_msgs/Twist “linear:
x : 0.0
y : 0.0
z : 0.0
angular:
x : 0.0
y : 0.0
z : 1.0”

○パブリッシュトピックを用いたロボットの状態の確認

[Remote PC]
sensor_state：OpenCRに接続されたアナログセンサのデータ
$ rostopic echo /sensor_state
stamp: ...

odem：ジャイロやエンコーダから出力したオドメトリ情報
$ rostopic echo /odom
header: ...

tf：XY平面上のロボットの位置(base_footprint), オドメトリ情報(odom),
相対座標変換で記述されるロボットの各関節のpose(位置と姿勢)
$ rostopic echo /tf
transforms: ...

サブスクライブトピックとパブリッシュトピックの確認 -2/2-

表 10.1　TurtleBot3 のサブスクライブトピック

トピック名	メッセージ型	機　能
motor_power	std_msgs/Bool	Dynamixel モータを On/Off
reset	std_msgs/Empty	オドメトリ（odometry）と IMU をリセット
sound	turtlebot3_msgs/Sound	ブザー音の出力
cmd_vel	geometry_msgs/Twist	移動ロボットの並進、回転速度を制御 単位はそれぞれ m/s、rad/s

表 10.2　TurtleBot3 のパブリッシュトピック

トピック名	メッセージ型	機　能
sensor_state	turtlebot3_msgs/SensorState	TurtleBot3 のセンサデータを格納
battery_state	sensor_msgs/BatteryState	バッテリの電圧などの状態を格納
scan	sensor_msgs/LaserScan	TurtleBot3 の LDS のスキャンデータを格納
imu	sensor_msgs/Imu	加速度／ジャイロセンサのデータに基づいて得られたロボットの姿勢データを格納
odom	nav_msgs/Odometry	エンコーダと IMU データに基づいて得られた TurtleBot3 のオドメトリを格納
tf	tf2_msgs/TFMessage	TurtleBot3 の base_footprint、odom などの座標変換を格納
joint_states	sensor_msgs/JointState	左右の両輪を関節として見たときの位置、速度、力（それぞれの単位は m、m/s、N・m）
diagnostics	diagnostic_msgs/DiagnosticArray	自己診断から得られた情報を格納
version_info	turtlebot3_msgs/VersionInfo	TurtleBot3 のハードウェア、ファームウェア、ソフトウェアなどに対する情報を格納
cmd_vel_rc100	geometry_msgs/Twist	Bluetooth コントローラ RC-100 を用いるときに使用するトピック。移動ロボットの速度制御に使用される。単位は m/s、rad/s

表 允哲, 倉爪亮, 鄭 黎蝘, “ROSロボットプログラミングバイブル”, 2018.

Rvizを用いたTurtleBot3のシミュレーション

○仮想ロボットの実行

[Remote PC]

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_fake turtlebot3_fake.launch
```

「Displays」->Global Options->fixed frame
/odom
「Add」: RobotModelを追加

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

w, x, a, d, s, cキー

Turtlebot3_fake_node:
Turtlebot3_descriptionパッケージにある
Turtlebot3の3次元モデルの読み込み
実際のロボットと同様にトピックをパブリッシュ

robot_state_publisher:
ロボットの車輪回転データから得られる両車輪
及び各関節の姿勢を、tf形式でパブリッシュ

カメラ・距離センサはGazebo上で扱う

○可視化

「Add」: By Topic -> Odometry
「Displays」-> Odometry Convarianのチェックを外す
「Shape」Shaft Length Shape/Head Lengthの値を適切に

GazeboをTurtleBot3のシミュレーション

[Remote PC]

$ gazebo

〇仮想TurtleBot3の起動（カメラを使いたいのでwaffleで）

$ export TURTLEBOT3_MODEL=waffle
$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
or
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

〇例．仮想環境上での仮想TurtleBot3のランダム移動及び障害物回避

$ export TURTLEBOT3_MODEL=waffle
$ roslaunch turtlebot3_gazebo turtlebot3_simulation.launch

Rvizでの表示

$ export TURTLEBOT3_MODEL=waffle
$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch

例．仮想環境上でのSLAM(自己位置と地図の同時推定)

$ export TURTLEBOT3_MODEL=waffle
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

SLAMの実行

$ export TURTLEBOT3_MODEL=waffle
$ roslaunch turtlebot3_slam turtlebot3_slam.launch

RVizの実行

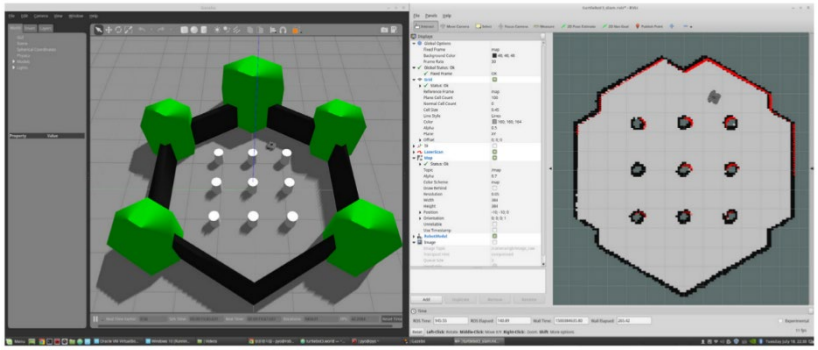
$ export TURTLEBOT3_MODEL=waffle
$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_sim.rviz

***Gazebo**

物理エンジンを搭載し、接触や力のバランスなどの
動力学的な変化を伴う実世界を再現し、そこにロボット、
センサなどのモデルを置くことで3次元シミュレーションを行うことができる

Gazeboの特徴

1. 動力学シミュレーション
2. 3次元グラフィックス
3. センサノイズ
4. プラグイン追加機能
5. ロボットモデル
6. TCP/IPデータ転送
7. クラウドシミュレーション
8. コマンドラインツール



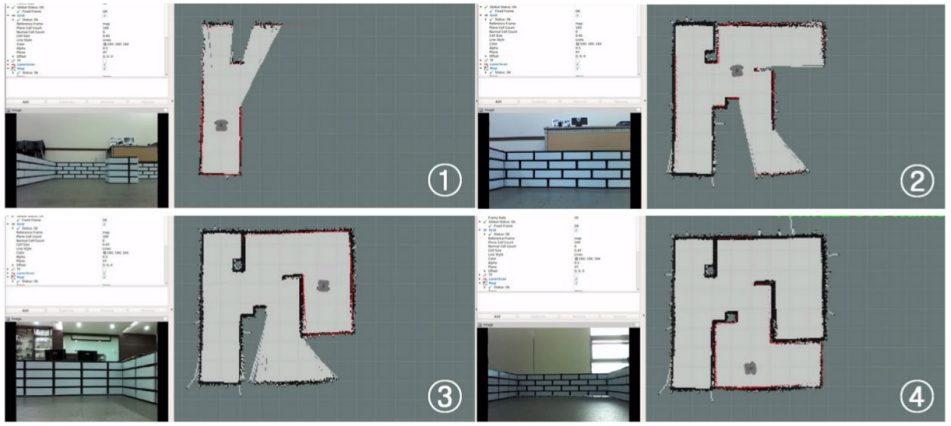
RVizの実行

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

地図の保存

$ rosrun map_server map_saver –f ~map

SLAM(自己位置と地図の同時推定)



[リモートPC]
$ roscore

[TurtleBot]
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

[リモートPC]

$export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping

Slam_methods & Tuning

gmapping/cartographer/hector/karto/frontier_exploration

http://emanual.robotis.com/docs/en/platform/turtlebot3/slam/#slam

[リモートPC] Run Teleoperation Node

$ export TURTLEBOT3_MODEL=%{TB3_MODEL}
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

[リモートPC] Save Map

$ rosrun map_server map_saver -f ~/map

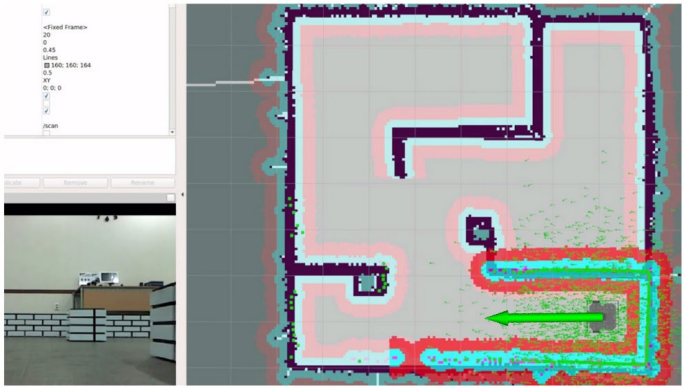
ナビゲーション

[リモートPC]
$ roscore

[TurtleBot]
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

[リモートPC] Launch the navigation file.

$ export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml



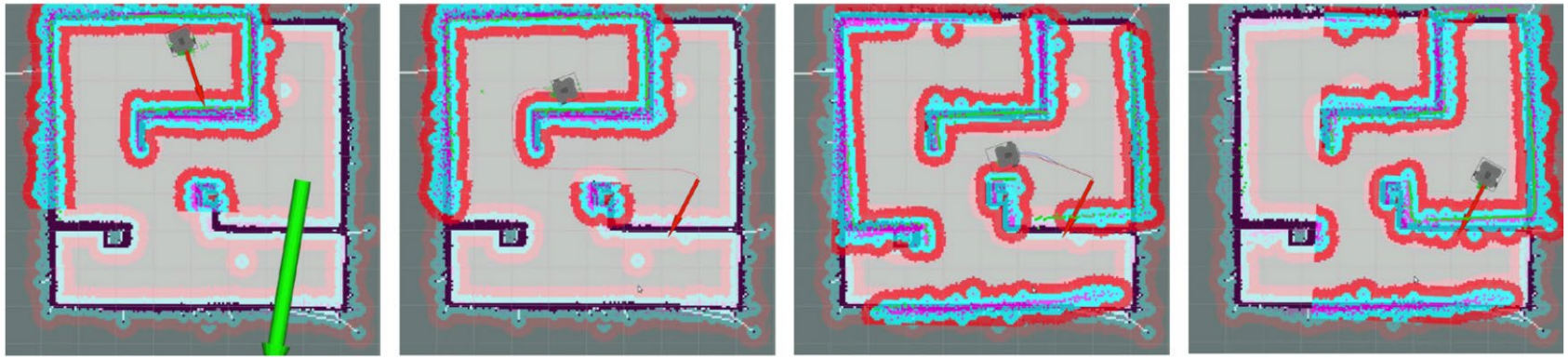
[リモートPC] Estimate Initial Pose

位置の確認　　2D Pose Estimateボタンをクリック

$ export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

[リモートPC] Send Navigation Goal

目標地点の設定 2D Nav Goalボタンをクリック



http://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/#navigation

Stateflowの基礎

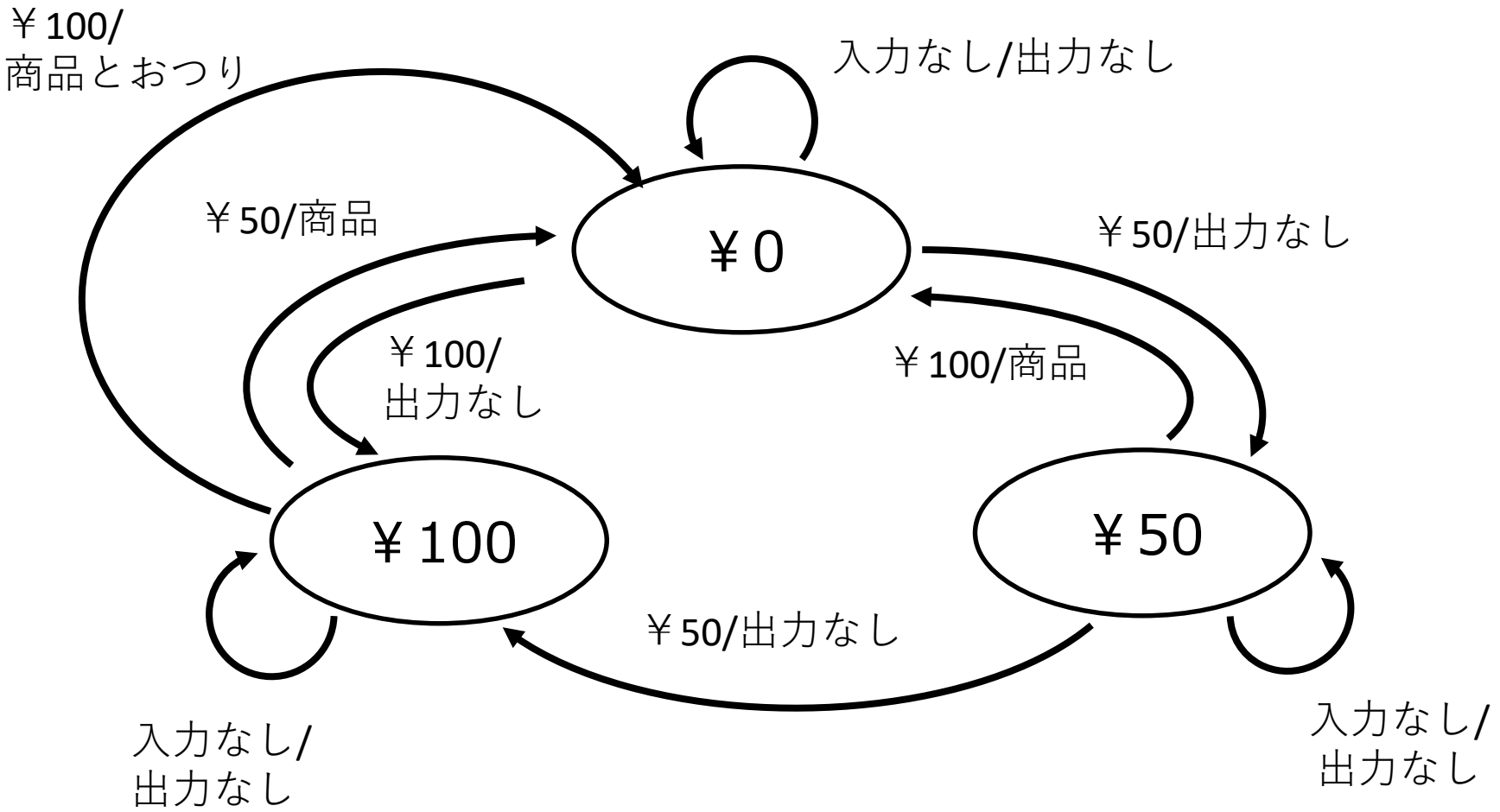
Stateflowの基礎(1/4)

例. Stateflowによる自動販売機(vending machine)の論理設計

50円，100円の2種類に硬貨を受け付け，
150円の商品1種類を販売する自動販売機の設計

・input： {なし，50円投入，100円投入}
・output： {なし，商品排出，おつり50円排出，
商品とおつり50円排出}
・state： {累積金額0円，累積金額50円，
累積金額100円}

Stateflowの基礎(2/4)

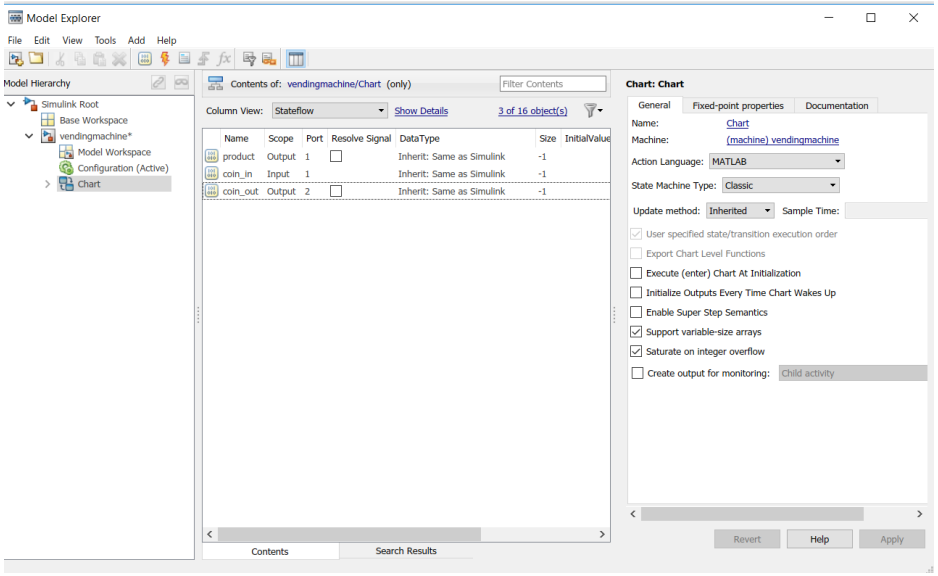


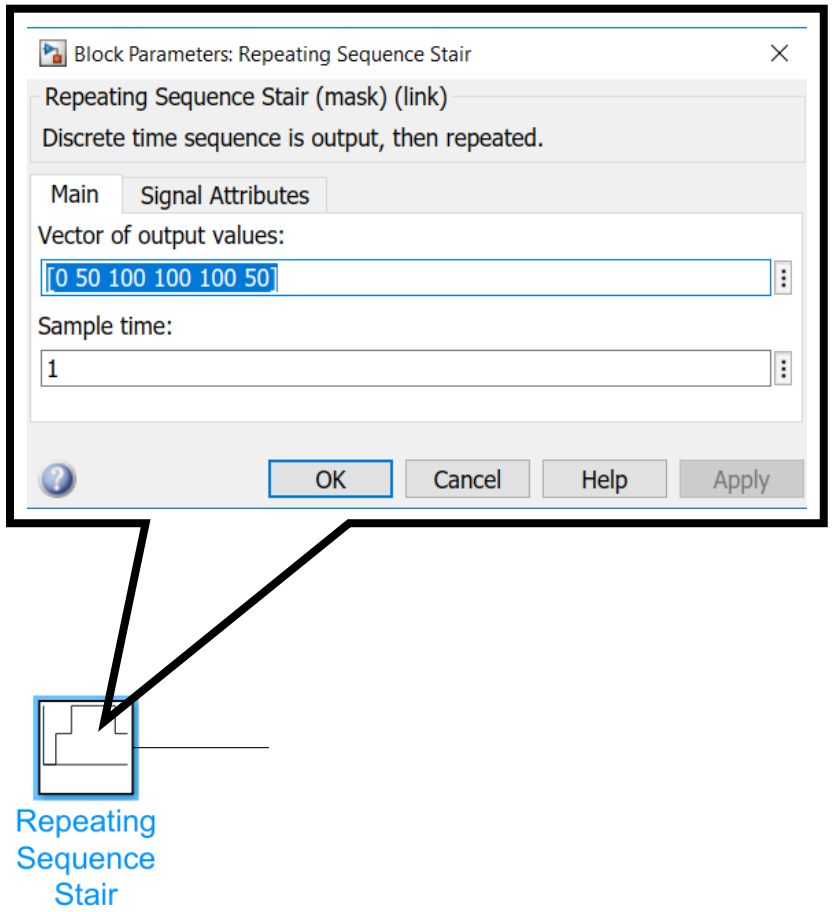
Stateflowの基礎(3/4)

状態遷移図

Stateflow: input port/output port

Model Explorer





Simulink: Sample rate

Stateflowの基礎(4/4)