独学ではじめる自動運転入門

in OSC2019北海道

PRESENTED BY RIM友の会
まいむぞう(大路裕介)

大路 裕介

まいむぞう

札幌のフリーランサーで
主にWebシステムやAndroidアプリ
を作っています

Web: http://fromnorth.blogspot.com
Twitter: @maimuzo
facebook: https://www.facebook.com/maimuzo

お品書き

- ROSの世界の復習
- Turtlebot3
- 自律走行に必要な技術

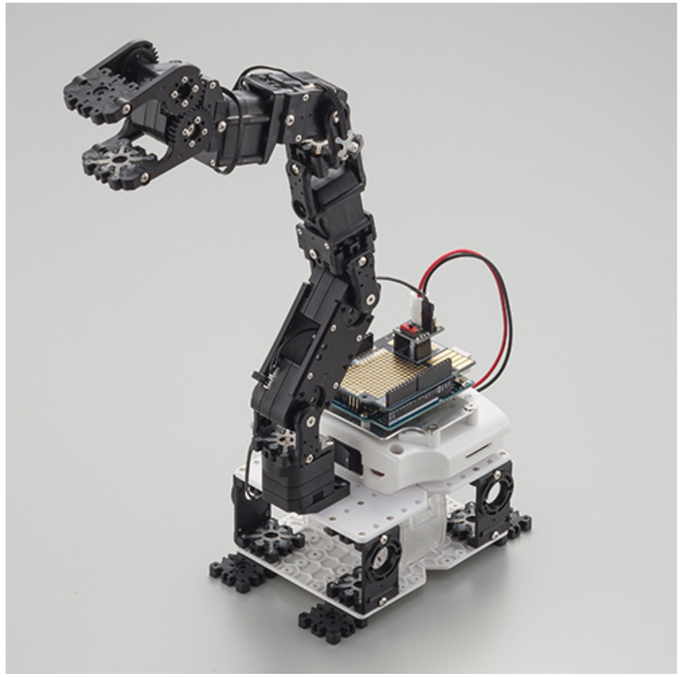
ROSの世界の復習

まず質問

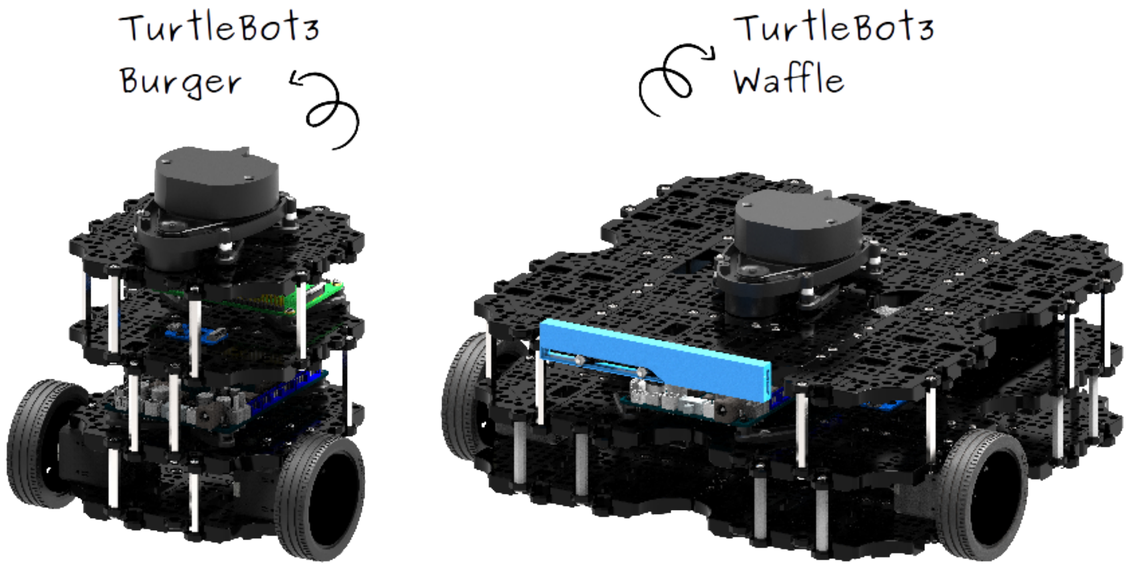
- ROSの世界の復習は、去年のOSC北海道2018での「独学で始めるロボティクス入門」を元にします
- 「独学で始めるロボティクス入門」を聞いてくれた方はどれくらいいますか?

ロボットの構成要素

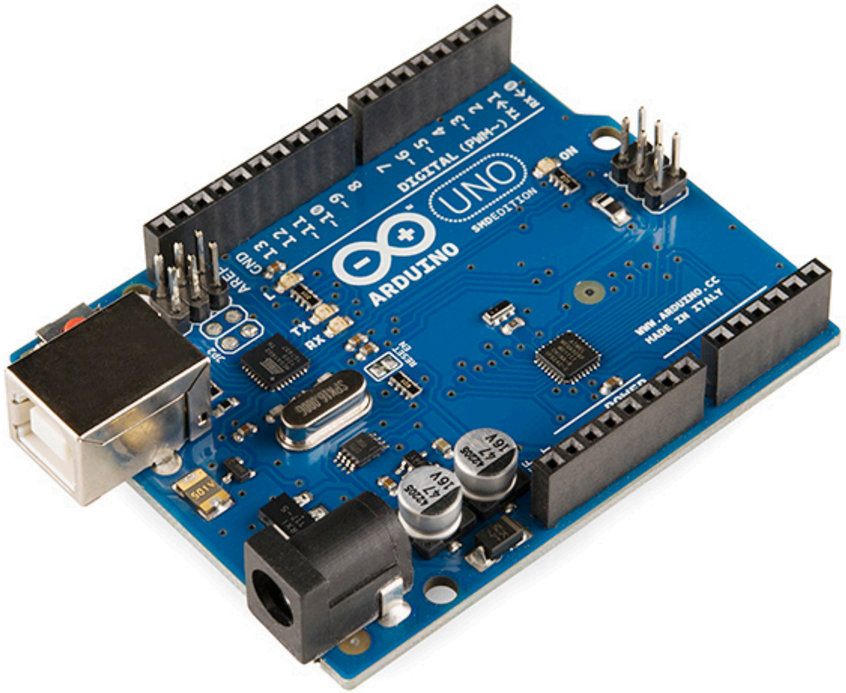
- ロボット本体
- マイコン
- シングルボードコンピューター
- パソコン

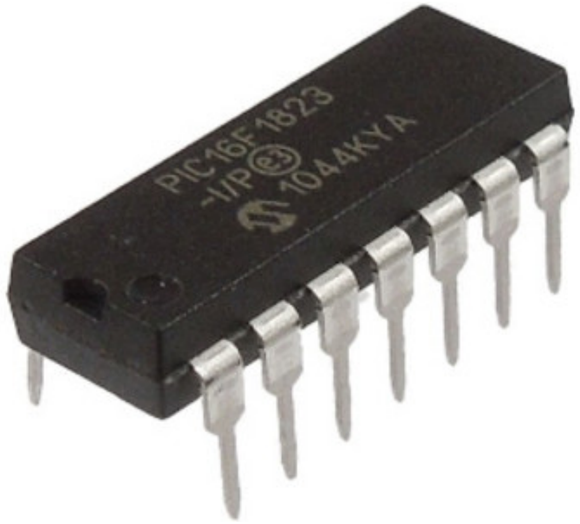


ロボット本体



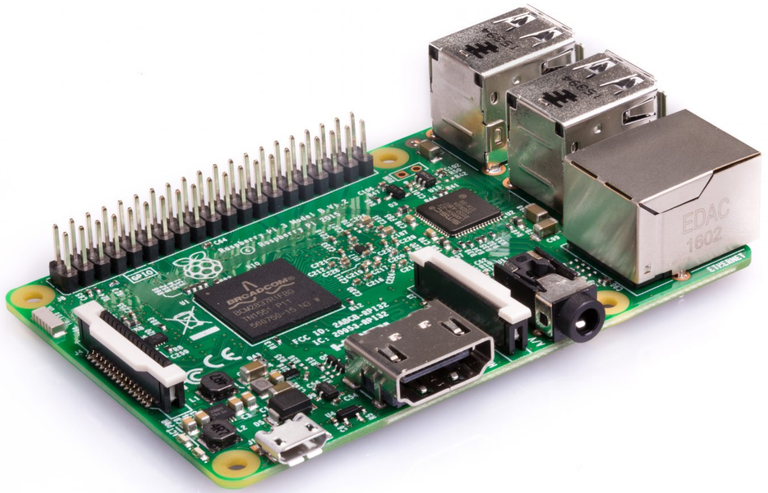
- ロボットのキット、完全自作ロボット、ラジコン改造、ドローンなどなど
- そのロボットの物理的特徴はこれで決まる
- ラジコンとの違いは、センサーがついてること。ロボットには操舵したり前後に動くとき、どれだけ動いたかを測るセンサーが必要です
- ←はマイコンやシングルボード載ってる画像です



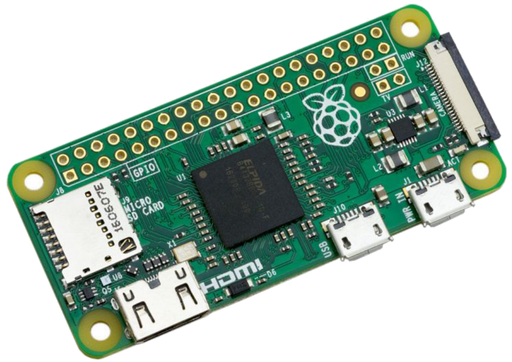


マイコン

- PICやArduinoなど
- プログラミング可能なもの
- センサーからの情報取得や、モーターなどのコントロールが得意。リアルタイム性高い
- 1つのプログラムが無限ループで常に動いている(OS無しも多い)
- 基本的にスタンドアローン
- 電源断に強い(壊れない)



シングルボードコンピュータ



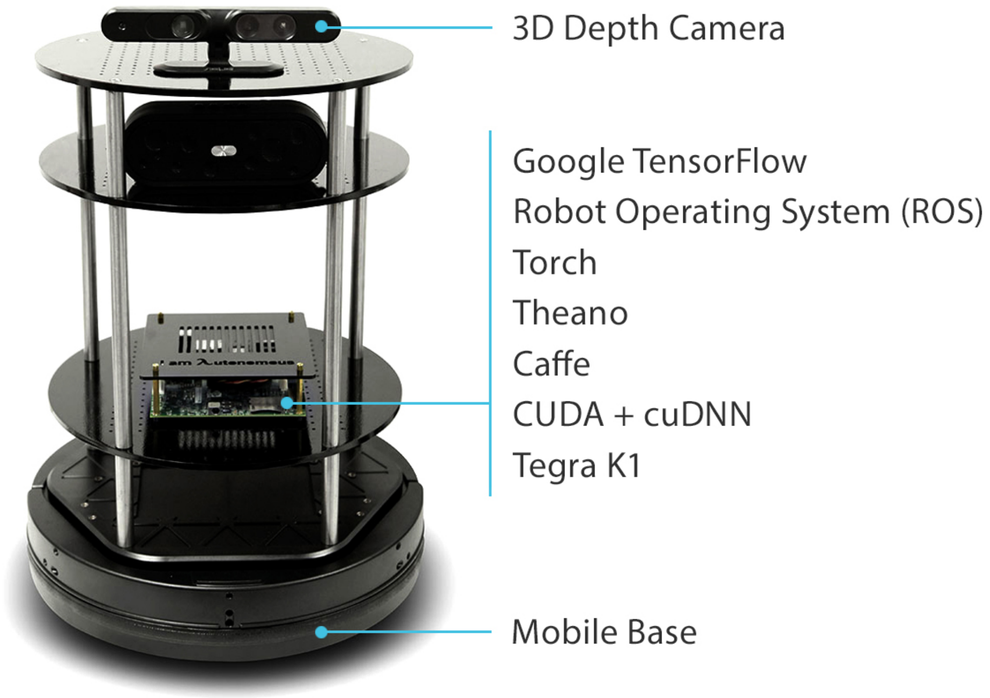
- Raspberry piやNVIDIA Jetsonなど
- Linuxベースが多い
- ソフトがたくさん
- ネットワークまわり得意
- ちっちゃいけどパソコンみたいなもの
- 小さくてバッテリー駆動できるのでロボットに搭載される
- マイコンとUSB接続されることが多い

パソコン

- みなさんが使っているもの
- プログラムを書いたり、ロボットに指示を出したり、モニタリングしたり、シュミレーションしたりできる
- 処理能力が高く、コンセントから電力を無限に使えるので、大規模演算が必要なものはパソコンで(機械学習など)
- シングルボードコンピューターとはWiFi接続

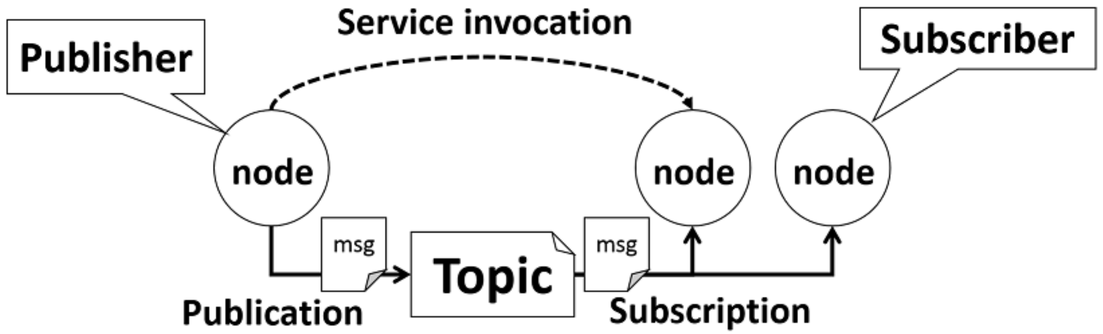
ROS

ROS



- ROS: Robot Operating System
- ラジコン以上のことをしたいなら必要
- ロボットのための基盤技術を提供
  - 各構成要素間の通信方法
  - センサーデータなどのロギング
  - シュミレーションなど
- 現状ではLinuxなどのOS上のミドルウェアとして実装
- 最先端の技術がapt installの感覚でインストール可能

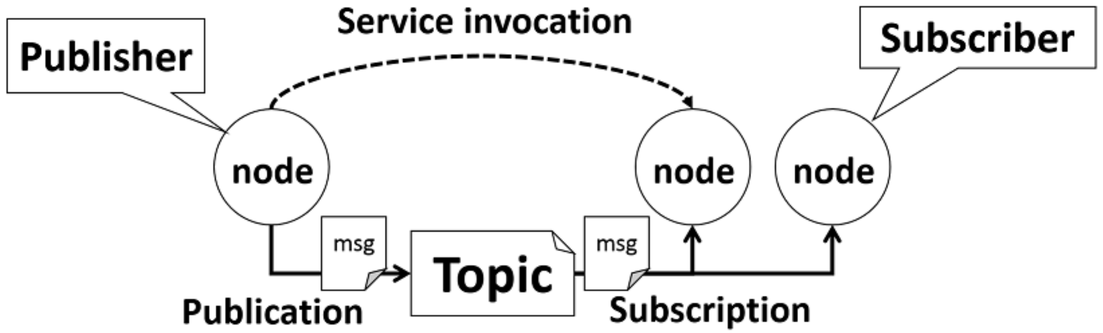
ROSの基本コンセプト



- ROSにはいろいろな機能がありますが、一番の基本になるのがTopicを使ったメッセージング機能です。
- node: 処理単位の小さなプログラムです。例えばセンサーを読み続けるnodeや、モーターを回転させるnodeがあります
- nodeはC/C++や、Pythonを使って書くことができ、nodeごとに1プロセスとして動きます

ROSの基本コンセプト

※いわゆるPubSubモデルです

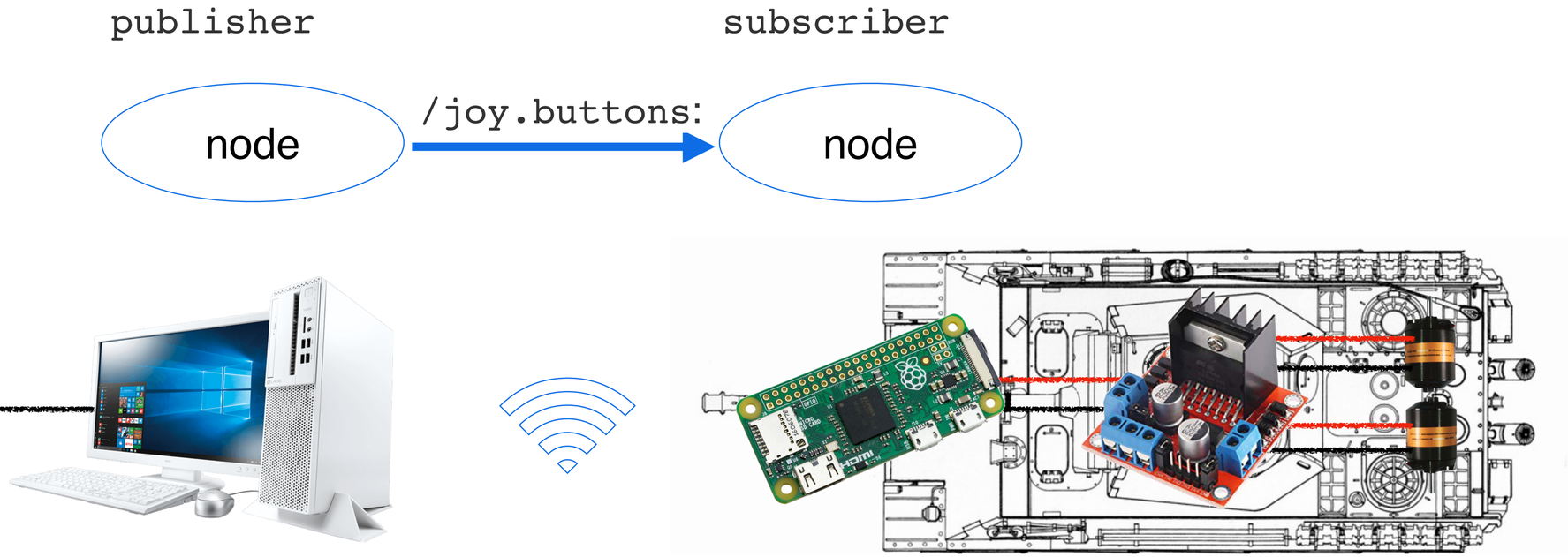


/my_data
  sensor_id(int): 1
  time(double): 12345
  angle(float): 60.5

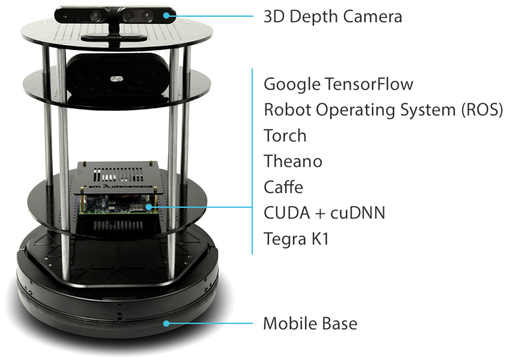
- 例えば、センサーの値を読むnodeはTopicのPublisherとなり、読んだデータをブロードキャストのように送信することができます
- 図のmsgはメッセージであり、C言語の構造体のように、名前と型と値のセットを一塊として送信できます(←参照)
- このメッセージを使いたいnodeは、Subscriberとなることで、メッセージを受信できます

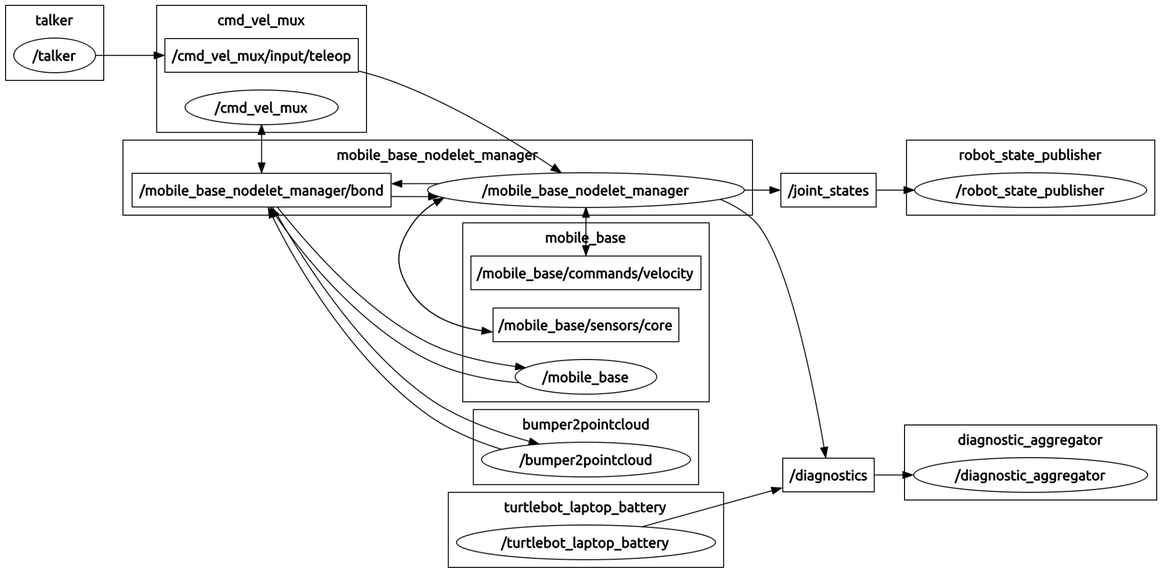
簡単な構成例

/joy.buttons:
  A: 0
  B: 1
  X: 0
  Y: 0
  LB: 0.899
  RB: 0.0
  START: 0
  …



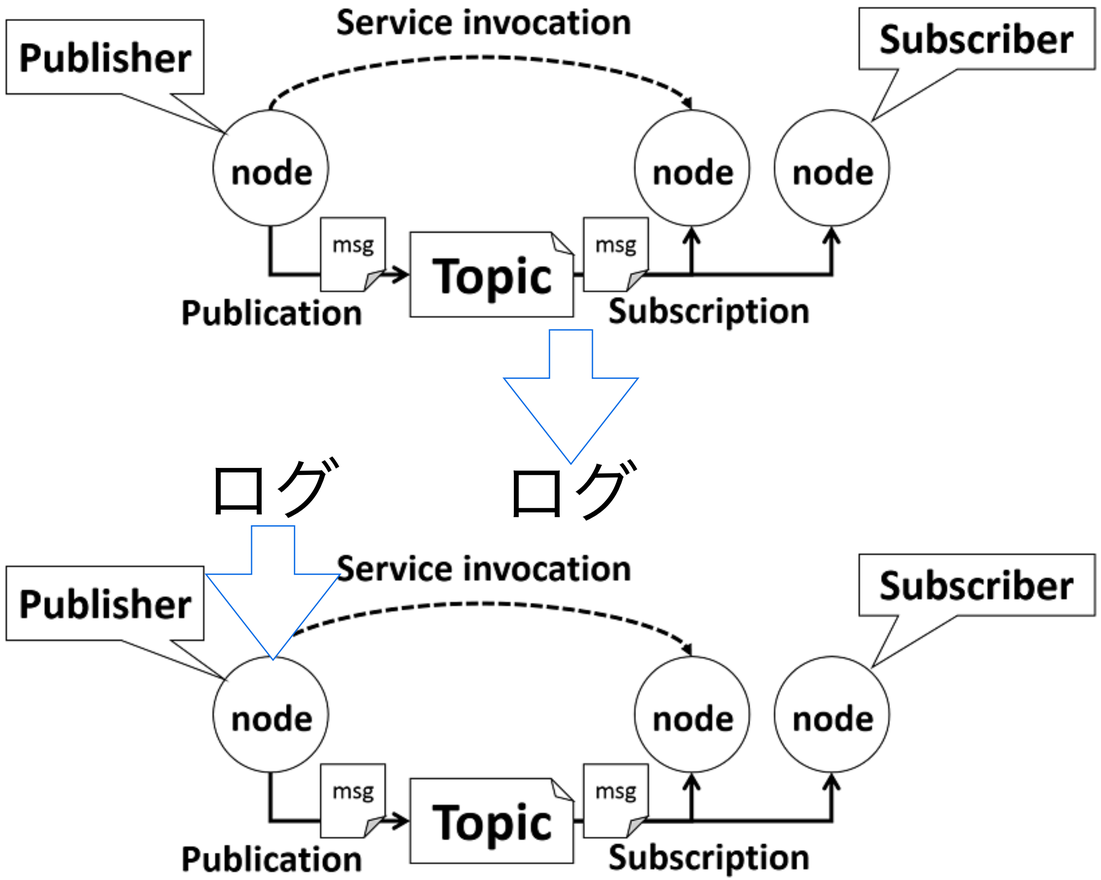
Topicネットワーク





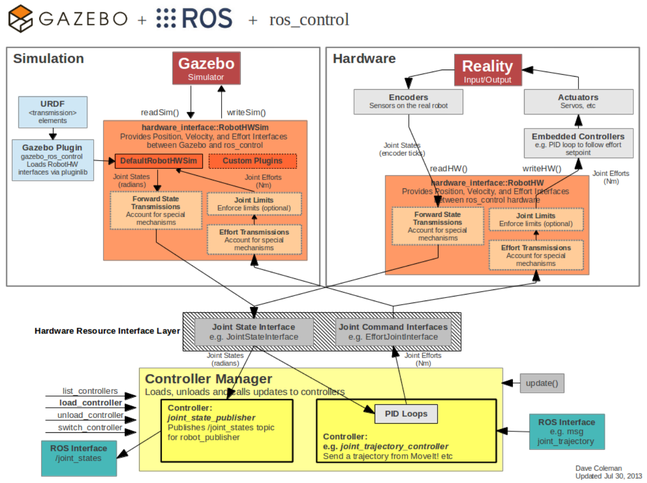
- nodeは、何個でもpublishできますし、何個でもsubscribeできます
- たとえば、センサーの値をsubscribeして姿勢を計算し、その結果をpublishすることもできます
- ある程度の規模になると、センサー数百個、モーターやアクチュエーター何十個、姿勢制御などの計算ネットワークが多段・並列で何個も、という規模になりますが、基本的にはPubSub型で実現できます

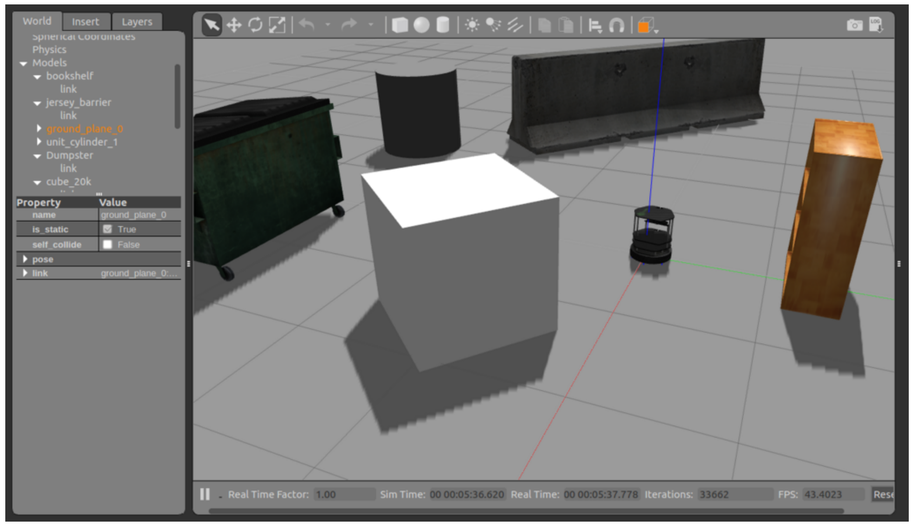
ロギングとシュミレーション



- スマホのアプリやWebシステムと異なり、ロボットは物理的なセンサー値やモーターなどを扱うので、テストやデバッグが大変です
- 例えば、Topicのデータをログとして出力しておくことで、生のデータを見ながらプログラムを改良できます
- 外に出なくても、センサーの値を読むnodeの代わりに、ログのデータを読むnodeを起動しておくと、シュミレーションできます

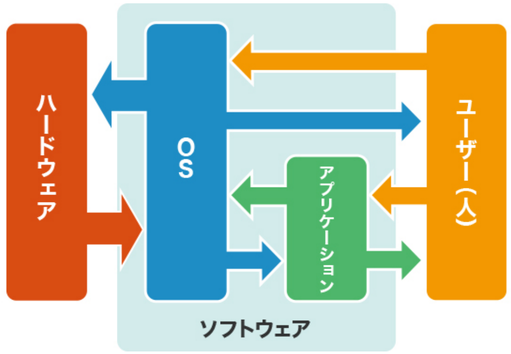
ROS × Gazebo





- こういったシュミレーション環境を応用したものにGazeboがあります
- ロボットの3Dモデル(形状など)と、地形データの3Dモデル(障害物など)を与えることで、実際の動作ネットワークをシュミレーションできます。
  - 例:入力と出力部分をシュミレーションすることで、自律走行部分を改良するなど

ROS=ロボット用OS



- ロボットの作り方の標準的な仕組みを提供
  - pubsub(入出力の方法)とメッセージ(データの型)で機能を表現できるので、標準的な作り方が統一できる
  - 標準的な作り方に則った、そのデータの可視化機能や、シュミレーション機能を提供する
- 作り方が統一できるので、よく使う機能がプラグイン的に導入できる
  - aptで導入できたり、gitからソースを取ってきてコンパイルするだけで使える
  - 最先端の機能が試せる(学習・プロトタイプ・比較)

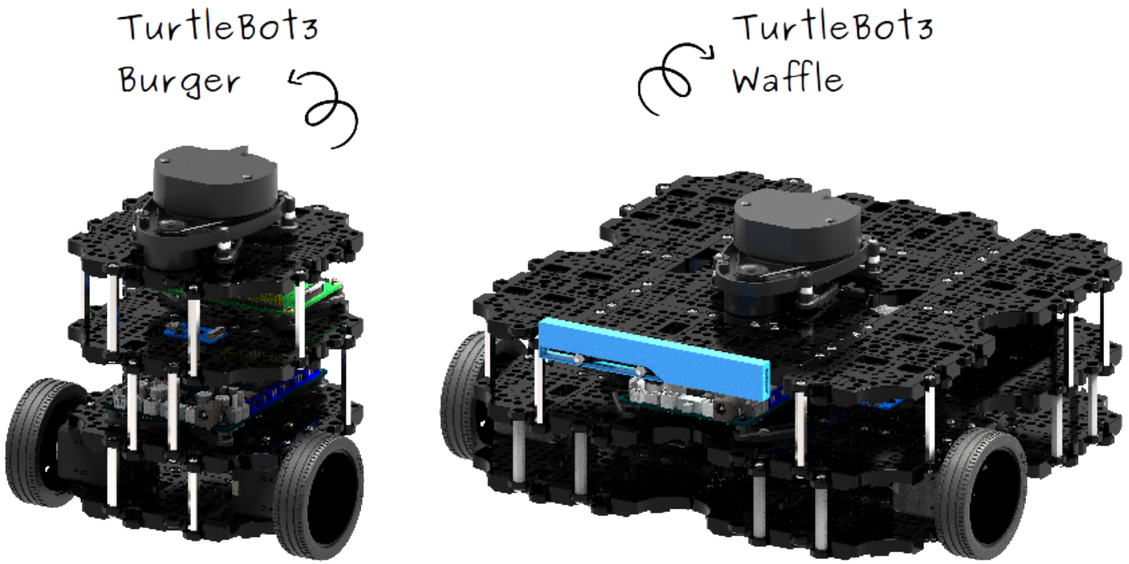
個人開発者にとっての至福

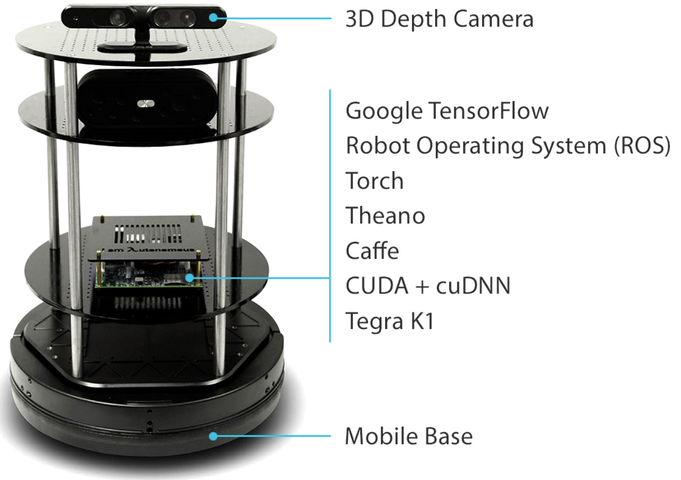
- リファレンス機(burgerなど)で作った機能が、自作機で使える
  - 設置場所や計算能力によって、複数の計算機を組み合わせて使える
  - ロボットに乗って制御するラズパイ、ラズパイからのデータを使って制御するPCなど
- これが無料です

Turtlebot3

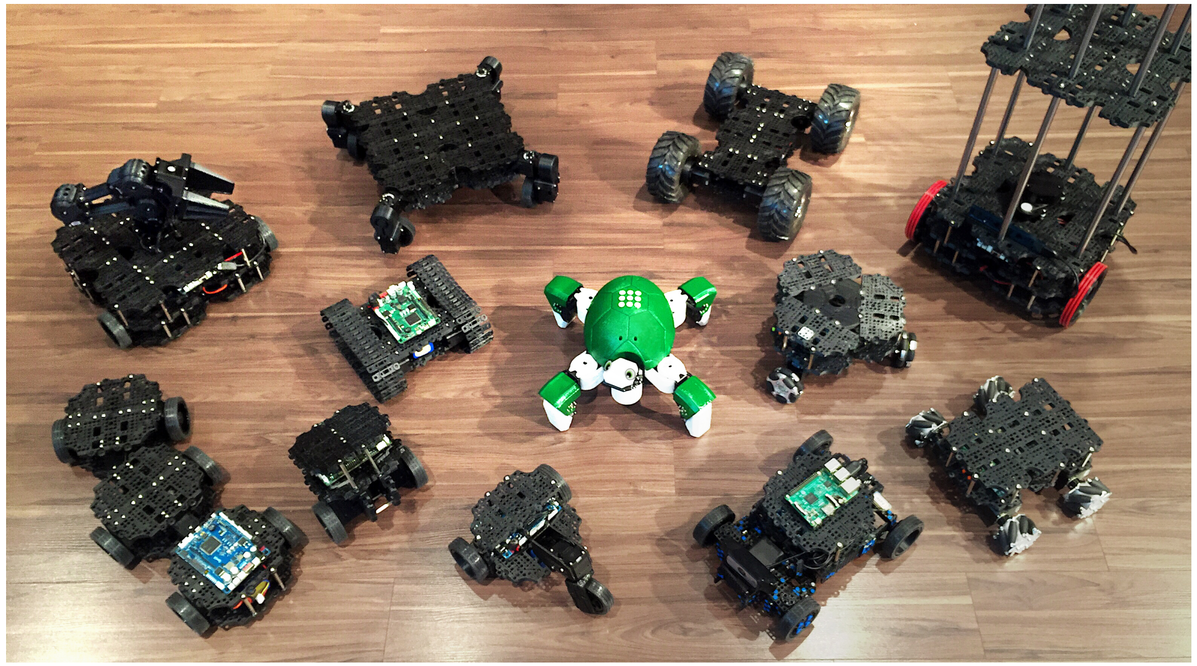
Turtlebotとは

- Turtlebotは研究・学習目的のとっかかりとしてのリファレンス機
- 自立走行するためのソフトウェアスタックや、シュミレーション環境が用意されている
- Turtlebot1や2もある



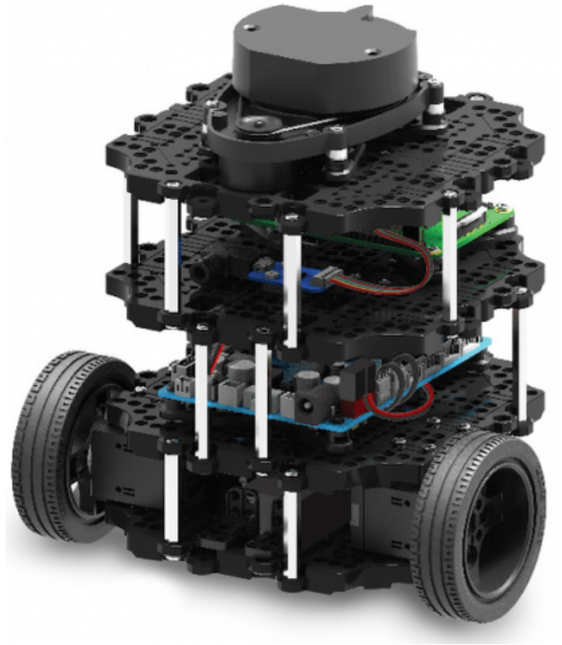


Turtlebotとは



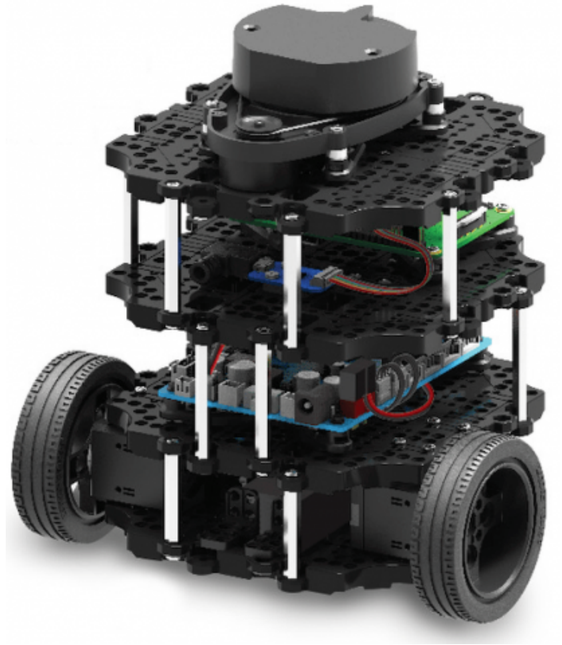
- Turtlebot3をベースにした親戚
  - 4輪カー
  - 3輪カー
  - 4足歩行
  - オムニホイール(全方向移動可能)
  - キャタピラー
  - セグウェイ
- これらはオープンソフトウェア、かつオープンハードウェアです

Turtlebot3 burgerの紹介



- Turtlebot3のうち、小型・安価な方がburger
  - burgerは大体6万円ぐらい。(waffleは22万円ぐらい)
- 小さくても、自律走行が可能な構成になっている
- システム全体の構成要素
  - OpenCR Arduino互換のマイコン+電源+センサーやモーター制御回路
  - Raspberry Pi 3B+ ネットワークとROSノードとして
  - パソコン Ubuntu 16.04ベース 送られてきたセンサーの値を使って計算する部分

Turtlebot3 burgerの紹介



- burgerの構成とセンサー・モーター類
  - ラズパイ3とOpenCR
  - 12Vバッテリー
  - ステッピングモーター ×2
  - Lidar ×1
  - IMU(6自由度+地磁気) ×1
- 詳しいスペック表
  - http://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#specifications

Turtlebot3 burger動作デモ

- 動作デモとして以下を実行してみます
  - 起動とキーボード操作とrviz
  - rqtとメッセージ
  - シュミレーション
- 準備
  - Turblebot3の電源ONしてMacからラズパイにSSH接続
  - Mac上のVMWareでUbuntu16.04を起動

Turtlebot3 burger動作デモ

- 起動とキーボード操作
- パソコン(VM)側
  - roscore
    - TCP/IPネットワーク上のどこかに、roscore(ROS管理プログラム)を起動しておく必要がある。すべてのNodeはこのroscoreに接続する。
- Turtlebot側
  - roslaunch turtlebot3_bringup turtlebot3_robot.launch
    - Turtlebot側(ラズパイ)では turtlebot3_robot.launchを起動しておくだけで良い
    - センサー値の送信と、移動指令に対してモーターを回転するだけ。どこに移動するかなど、複雑なことはすべてパソコン側で行っている

Turtlebot3 burger動作デモ

- 起動とキーボード操作
- パソコン(VM)側
  - roslaunch turtlebot3_bringup turtlebot3_remote.launch
    - パソコン(VM)側ではturtlebot3_remote.launchが基本的な機能を提供している
  - roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
    - turtlebot3_teleop_key.launchを起動すると、キーボードで操作できる
- roslaunchを使うと、複数のNodeやパラメタを一気に起動できる
  - 普通は機能ごとに.launchファイルを作っておく

Turtlebot3 burger動作デモ

- rvizとrqtとメッセージ
- パソコン(VM)側
  - rosrun rviz rviz -d `rospack find turtlebot3_description`/rviz/model.rviz
    - 起動オプションにより、見せ方が異なる
    - 現在はburger中心に、Lidarが検出した点を表示している
  - rqt
    - GUIが開くのでplugin → Topics → Topic Monitor
    - 見たいTopicにチェックを付けると、その時の値が表示される

Turtlebot3 burger動作デモ

- シュミレーションその1(Gazeboなし)
- 前提条件
- Turtlebot側
  - turtlebot3_robot.launch を止める
- パソコン(VM)側
  - turtlebot3_remote.launchを止める
  - roslaunch turtlebot3_fake turtlebot3_fake.launch
    - つまりturtlebot3_robot.launchと turtlebot3_remote.launch の代わり
  - roslaunch turtlebot3_gazebo turtlebot3_world.launch
    - 仮想ワールド起動

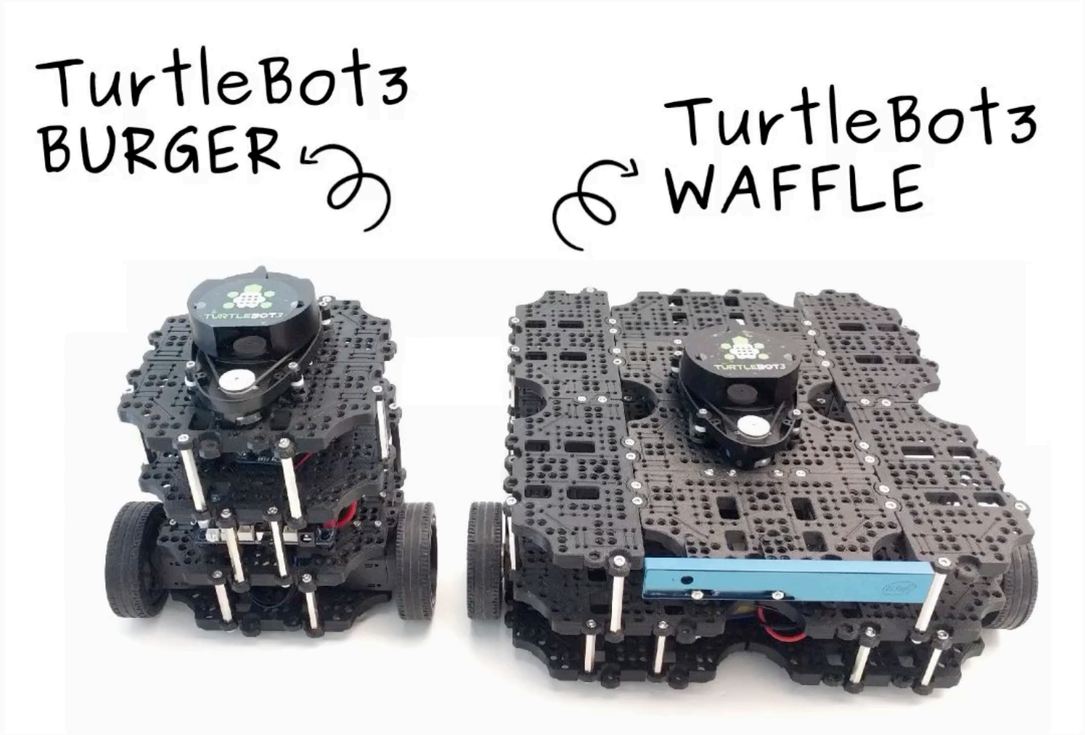
自律走行に必要な技術

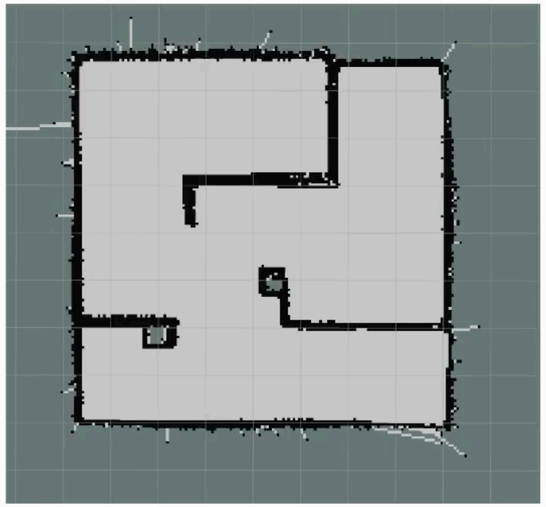
SLAM

一言で言うと

地図を作り、その地図の中のどこに
自分がいるかを推定する技術

SLAM





SLAM Demo

https://youtu.be/7mEKrT_cKWI

SLAM

- Simultaneous Localization And Mapping
  - 高度に数学的・統計的で、様々な種類があり、活発に研究されている
  - ここで作った地図を使って自律走行の経路を決定します
- SLAMの種類
  - マップの次元の種類(2次元、3次元)
  - センサーの種類 (LIDAR、単眼RGBカメラ、複眼RGBカメラ、単眼RGBDカメラ)
  - アルゴリズムの種類(Gmapping、Hector、Karto、Cartographer、Ethzasl icp mapping、TUM-RGB-D、LSD-SLAM、ORB-SLAM、PTAM)
  - マップの見た目(人間にわかりやすい、人間には分かりづらい)

SLAM

- Gmappingのデモ
  - burgerに搭載されている2次元Liderを使って、2次元の地図を作るGmappingを使います
- パソコン(VM)側
  - roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
    - SLAM開始
  - rosrun map_server map_saver -f ~/map
    - 地図を保存

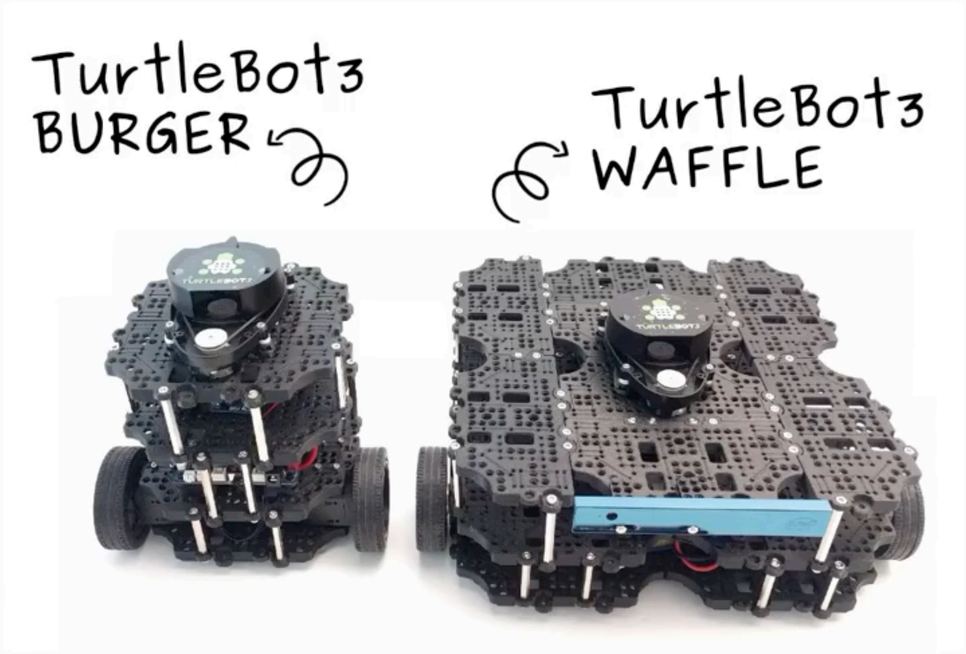
Navigation

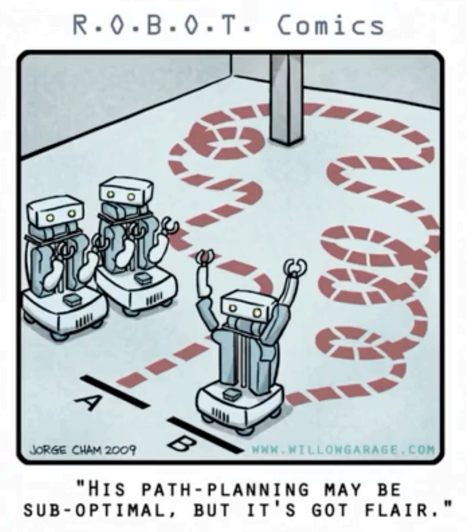
一言で言うと

地図がある状態で、自分の位置と
目的地の位置を指定することで、
その移動経路を決める技術

※Navigation自体は、特定の技術を指す固有名詞ではありません

Navigation



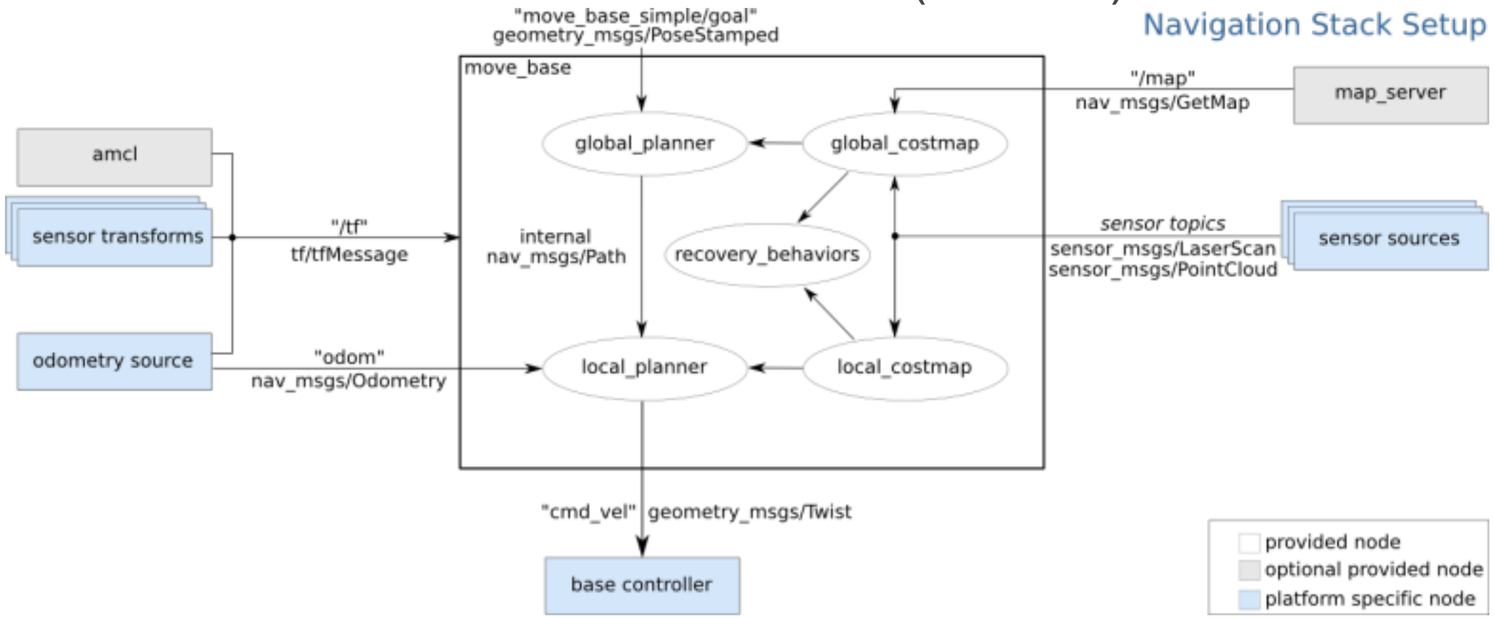


Navigation Demo

https://youtu.be/VYlMywwYALU

Navigation

- Turtlebot3はGmapping(2次元マップ)を前提としているので、Turtlebot3用の経路計画パッケージであるNavigationも2次元を対象としています
  - Global planner ～ 車でのナビ相当。どこへ、どういう道順で行くのか
  - Local planner ～ 運転席から見える障害物(人や車)をどうやって迂回するのか



Navigation

- Navigationのデモ
  - Gmappingで作った地図をベースに、rvizで指定した場所まで移動します
- パソコン(VM)側
  - roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
    - ナビゲーション開始
  - rviz -d `rospack find turtlebot3_navigation`/rviz/turtlebot3_navigation.rviz
    - ゴール地点を指定するためにrviz起動

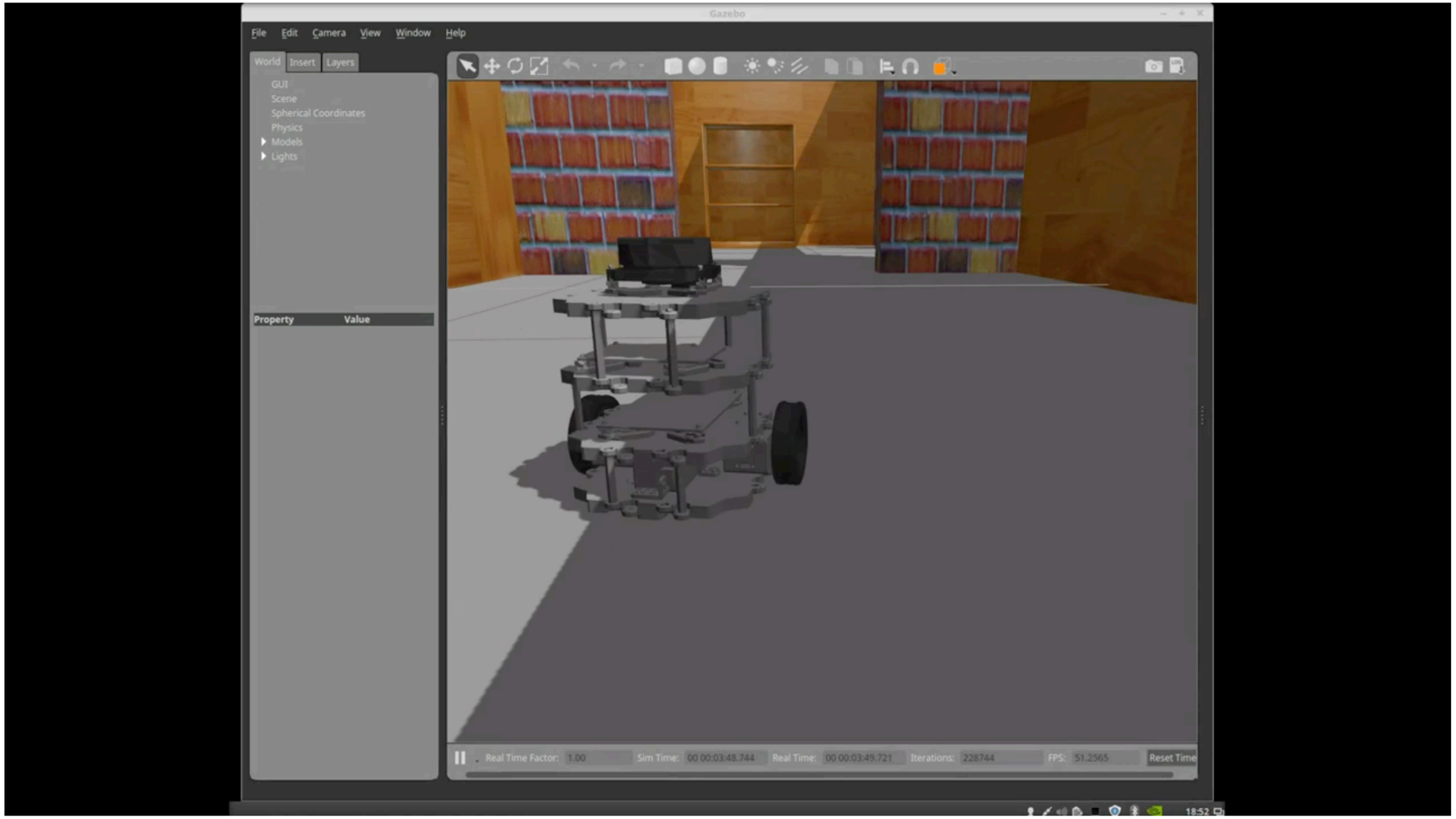
Simulation

一言で言うと

何らかのシステムの挙動を、
それとほぼ同じ法則に支配される
他のシステムやコンピュータ
などによって模擬する技術

※Simulation自体も、特定の技術を指す固有名詞ではありません

Simulation



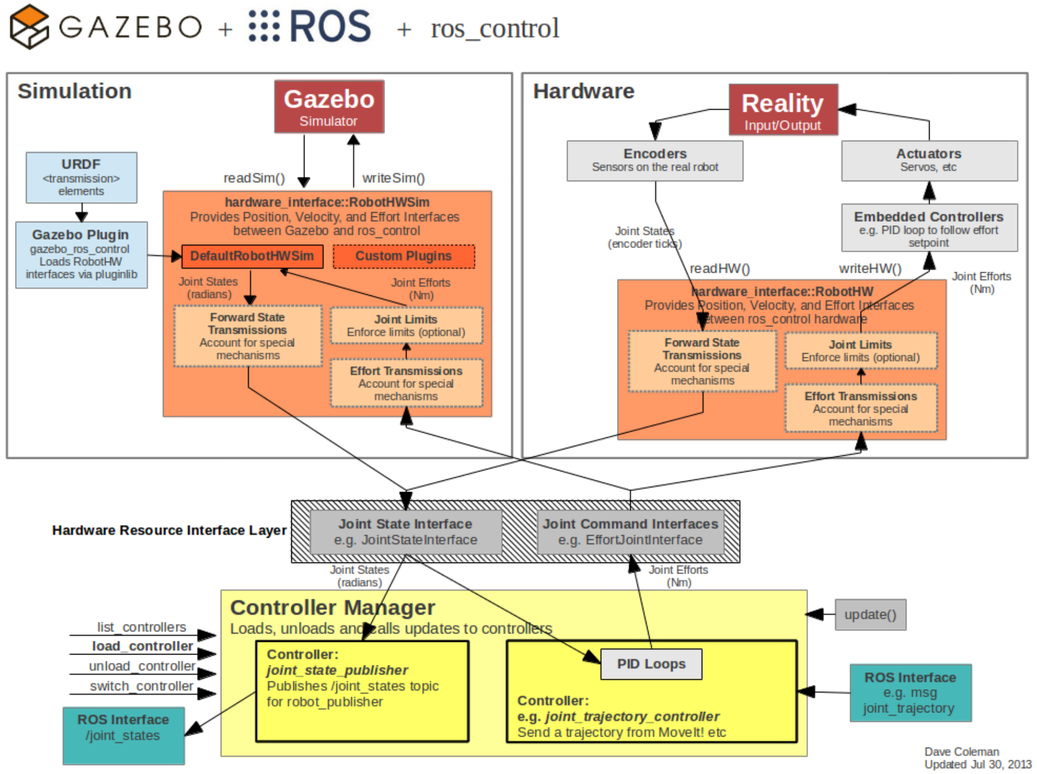
https://youtu.be/UzOoJ6a_mOg

Simulation

- なにが嬉しいのか
  - 実際のロボットを作る前に、仮想空間上でうまく動くかお試しできる
    - 事前に当たりをつけれる
  - 腕と足の相対位置関係などの物理的なパラメタを定義することで、実際の制御プログラムで使うことができる(実装の一部とすることができる)
  - 実際にロボットを動かさなくても、仮想空間上で動かすことができる
    - 制御部分をお手軽に作り込みできる
    - ちょっとした修正で現場に行かずに済む
    - 特定の状況を再現できる・再現しやすい

↑これかなり重要です

Simulation



- Turtlebot3の場合、2種類の方法があります
  - 単純に、移動指示通りに動いたことにする

    fake_node
  - GAZEBOを使った物理シュミレーション
- GAZEBOなら
  - 精密な3Dオブジェクト空間上に、自分で定義したロボットを動かせる
  - 挙動は物理シュミレーションされる
  - カメラ画像取得やLiderによる距離計測もできる

Simulation

- Simulationのデモ
  - GAZEBO上でSLAM(Gmapping)します
- パソコン(VM)側
  - roslaunch turtlebot3_gazebo turtlebot3_world.launch
    - 仮想ワールド起動
  - roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
    - SLAM実行
  - roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
    - キーボードで操作

まとめ

まとめ

- ROSは、ロボットとしての標準的なプログラム構造を定め、部品としての再利用性を高めることで、ロボット用OSとして成功しました
- SLAMで地図を作り、その地図を元にNavigationで目的地まで自律走行できました。
- これらは、非常に高度な技術の上に成り立ちますが、Turtlebot3 burgerを使えば、比較的簡単に実際に試してみることができました。
- また、Turtlebot3などを持っていなくても、Simulationを使うことで仮想空間で試すことができました。

フォローアップ

- ROS公式
  - http://wiki.ros.org/ja
- ROS2 (ただし現在移行期につき不安定)
  - https://index.ros.org/doc/ros2/
- 本当の車の自動運転 AutoWare
  - https://www.slideshare.net/takuyaazumi3/autoware-ros
- 書籍
  - SLAM入門: ロボットの自己位置推定と地図構築の技術
  - 実用ロボット開発のためのROSプログラミング

Thanks!

Any questions?

You can find me at: @maimuzo on twitter or maimuzo@gmail.com