サンプルアプリケーション各機能の内部仕様の概要

地図の作成

地図を作るプロセスは実際のところ、gmapping というオープンソースのパッケージを用いて行なっており、これをROBOTIS Japan が公開している、サンプルプログラムを通して呼び出しています。 ROBOTIS Japan 公開のサンプルプログラムは robot_ws/src/robotis3 配下に保存されており、これは 環境構築時に https://github.com/ROBOTIS-JAPAN-GIT/turtlebot3.git からコピーしてきたものです。 コマンド roslaunch で呼び出した slam.launch は ソースコード robot_ws/src/delivery_challenge_sample/launch/slam.launch にあります。このファイルは中で turtlebot3_slam パッケージの turtlebot3_slam.launch を呼び出すよう記述されており、 turtlebot3_slam.launch は今回利用しているロボットで gmapping を動作させるのに最適なパラメータを与えて gmapping を起動します。 gmapping は SLAM という手法を使って地図の作成を行います。 SLAMおよびROBOTIS のサンプルアプリケーションについてより詳しくは次が参考になります。
http://emanual.robotis.com/docs/en/platform/turtlebot3/slam/#slam

（*メモ*：このページは AWS RoboMaker を使わない通常の ROS 開発環境での説明になっています。ここで [TurtleBot] となっている箇所は RoboMaker のシミュレーションと解釈してください。[TurtleBot] の部分はシミュレーションが起動した時点で動作しています。[Remote PC] の部分が皆さんのアプリケーションを動作させる部分で、 RoboMaker 開発環境で robot_ws 配下に展開されている箇所に相当します）

ブラウザーインターフェースで [Save Map] ボタンをクリックするとファイルは一度 /tmp 配下に保存され、次に Amazon S3 に保存されます。 Amazon S3 での保存先は robot_ws/src/delivery_robot_sample/settings/settings.yaml ファイルに記録されています。 保存されるファイルは map.pgm と map.yaml ファイルの２つです。 map.pgm ファイルはグレースケールの画像ファイルで、作成された地図データになります。これは画像ファイルなので、画像編集ソフトで編集することも可能です。画像データの中で、黒い箇所が障害物として認識されます。例えば地図の中でロボットを走行させたくない領域がある場合、その周辺を黒く塗りつぶすことで、そこに障害物があると解釈させることができます。あまり大きく編集すると、ロボットのセンサーが捉えた情報と地図が一致しなくなり、ロボットが自身の位置を判断することができなくなるので、行きすぎた編集には注意が必要です。

SLAMの起動までの流れは次の通りです
RoboMaker シミュレーションの起動 (roboMakerSettings.json に定義されている）
↓
robot_ws/src/delivery_robot_sample/launch/controller.launch
・リモートコントローラーの起動

ターミナルから roslaunch delivery_robot_sample slam.launch を実行
↓
robot_ws/src/delivery_challenge_sample/launch/slam.launch
↓
turtlebot3_slam.launch
↓
turtlebot3_gmapping.launch
↓・gmapping パラメータの設定
gmapping 開始

ナビゲーション

ナビゲーションは、地図情報を読み取り、地図の中での自身の場所を特定し、目的地が指定された時、その目的地に移動する経路を計算し、ロボットを移動させます。途中地図にない障害物などが検出された場合でも、その障害物にぶつからないよう経路を修正しながら目的地を目指します。一連の機能は ROS のナビゲーションスタック により実現しています。AWS RoboMaker から navigation アプリケーションを実行すると、まず Amazon S3 から地図情報をダウンロード、これを /tmp ディレクトリ配下に保存、このファイルを読み込ませてナビゲーションスタップを実行しています。地図は robot_ws/src/delivery_robot_sample/settings/settings.yaml に指定されている Amazon S3 のバケットからダウンロードしてきます。この地図のダウンロードとナビゲーションスタックの起動は robot_ws/src/delivery_robot_sample/scripts の中の launch_navigation_stack.sh で行われています。このファイルはROBOTIS Japanのサンプルアプリケーションの turtlebot3_navigation.launch を呼び出しており、turtlebot3_navigation.launch はナビゲーションスタックを起動するための各種パラメータを設定し、実際にナビゲーションスタックを起動しています。 ROBOTIS Japan 公開のサンプルプログラムは robot_ws/src/robotis3 配下に保存されており、これは 環境構築時に https://github.com/ROBOTIS-JAPAN-GIT/turtlebot3.git からコピーしてきたものです。

ナビゲーションに関してより詳しくはこちらが参考になります。 （メモ：このページは AWS RoboMaker を使わない通常の ROS 開発環境での説明になっています。ここで [TurtleBot] となっている箇所は RoboMaker のシミュレーションと解釈してください。[TurtleBot] の部分はシミュレーションが起動した時点で動作しています。[Remote PC] の部分が皆さんのアプリケーションを動作させる部分で、 RoboMaker 開発環境で robot_ws 配下に展開されている箇所に相当します）

ナビゲーションスタックの起動までの流れは次の通りです

RoboMaker シミュレーションの起動 (roboMakerSettings.json に定義されている）
↓
robot_ws/src/delivery_robot_sample/launch/navigation_robot.launch
・リモートコントローラーを起動
・navigation.launch を読み込み
↓
robot_ws/src/delivery_robot_sample/launch/navigation.launch
・Amazon S3 からのファイルのダウンロード
・turtlebot3_navigation.launch の読み込み
↓
robot_ws/src/turtlebot3/turtlebot3_navigation/launch/turtlebot3_navigation.launch
↓・ナビゲーションスタックパラメータの設定
ナビゲーションスタック開始

ブラウザーインターフェース

サンプルアプリケーションではブラウザーからロボットアプリケーションにアクセスできるインターフェースを提供しています。ブラウザーインターフェースは次を提供しています。

- ロボットの移動方向をコントロール
- ロボットの位置の表示
- 地図の保存のリクエスト、これは地図の作成アプリケーションを実行している間だけ機能します。
- 指定された場所へのナビゲーションによる移動、これはナビゲーションを実行している間だけ機能します。

ブラウザーでは JavaScript が動いており、JavaScript から送られたメッセージは /awsiot_to_ros という ROS トピックとしてROSアプリケーションに届けらえれます。 ROS アプリケーションで /ros_to_awsiot という ROSトピックにメッセージを送ると ROS アプリケーションから JavaScipt にメッセージを送ることができます。

/ros_to_awsiot、 /awsiot_to_ros それぞれを処理して移動方向のコントロールなどを行なっています。これらの処理は robot_ws/src/delivery_robot_sample/nodes の中にあります。

ブラウザーと ROS アプリケーションとの通信は AWS IoT を利用して行なっており、その通信の処理は robot_ws/src/deps/aws_game_manager の中で実装されています。
robot_ws/src/deps/aws_game_manager/certs の中には AWS IoT で通信をするためのキーファイルが含まれています。このフォルダーの中のファイルは外部に共有しないようにしてください。また robot_ws/src/deps/ 配下にはシミュレーション予選用機能が実装されているので、robot_ws/src/deps/ 配下はコンテスト参加者は変更禁止です。

環境構築時の自動処理 ws_setup.sh について

環境構築時に実行する ws_setup.sh はシステムを最新にして、システムの情報を確認、アプリケーションを動かすために必要な AWS のリソースを作成、そしてアプリケーション内の設定ファイルを更新しています。 作成された AWS リソースの情報は ws_resources.yaml ファイルに記録されます。 途中でエラーが出た時は、問題を修正して、再度 ws_setup.sh を実行してください。 ws_setup.sh 実行の際、ws_resources.yaml に記録されているリソースについては、再度生成することなく、ws_resources.yaml　の内容を引き継ぎます。 ws_setup.sh は ws_resources.yaml　に加えアプリケーションの設定ファイルの更新状況を記録した ws_settings.yaml とアプリケーション実行先の環境情報を記録した ws_info.yaml というファイルも書き出します。途中でエラーが出たことが理由で 再度 ws_setup.sh を実行する場合は ws_settings.yaml と ws_info.yaml については削除してから実行を行なった方がより確実な再実行が行われます。 （これら３つのファイルは ws_setup.sh の中で行われている一つ一つの処理を記録しているもので、記録が残っている作業については再実行の際はスキップするよう作用します)

ビルド、バンドルを行う

プログラムコードを変更した場合、これをアプリケーションに反映させる必要があります。 今回提供のアプリケーションは大きく次の２つのパートからなっています。

simulation_ws 配下のファイル・・シミュレーションアプリケーション。これはシミュレーション環境を立ち上げるためのアプリケーションです。一度構築が完了すると、基本的に変更、再構築は必要ありません。このフォルダー配下は本戦では不要です。

robot_ws 配下のファイル・・ロボットアプリケーション。このフォルダーの中、src フォルダーの中にロボットを操作するアプリケーションコードが入ります。この中のファイルを変更した場合、変更をアプリケーションに反映させるためにビルドとバンドルという操作を行う必要があります。

ビルド、バンドルの方法： 開発環境、メニューより Run -> Workflow -> Delivery Challenge Robot app build -> bundle を選択します。

> メモ：バンドルとは？ バンドルは、ROS のアプリケーションを一つのアーカイブファイルにパッケージするプロセスで、RoboMaker では、アプリケーションを環境間で移動させるために、このバンドルというアーカイブファイルを交換する方法を採用しています。 参考：
> https://aws.amazon.com/jp/blogs/news/building-bundling-ros-app-aws-robomaker/