EN613 Final Project Template (v2)

Overview

In this zip file you will find all the files you need to get started on the final project. This template includes the following two folders

1. robot_spawner_pkg

This package includes an example world file, model files, and ros2 launch files for launching gazebo and spawning a basic robot. The current robot model uses a differential drive plugin for steering and publishes laser scan messages from a 2D laser range finder.

2. robot_control_pkg

This package includes a placeholder controller node and placeholder estimator node. The placeholder controller node subscribes to the estimated state and publishes a desired velocity message, currently all zeros. The placeholder estimator node accepts the laser scan message and odometry message and publishes a current estimated state message, currently all zeros. These nodes

How to Run

1. Create a workspace

The first step is to download the final_project_template.zip file and create a ROS2 workspace. Use the following commands as originally described in Assignment 1.

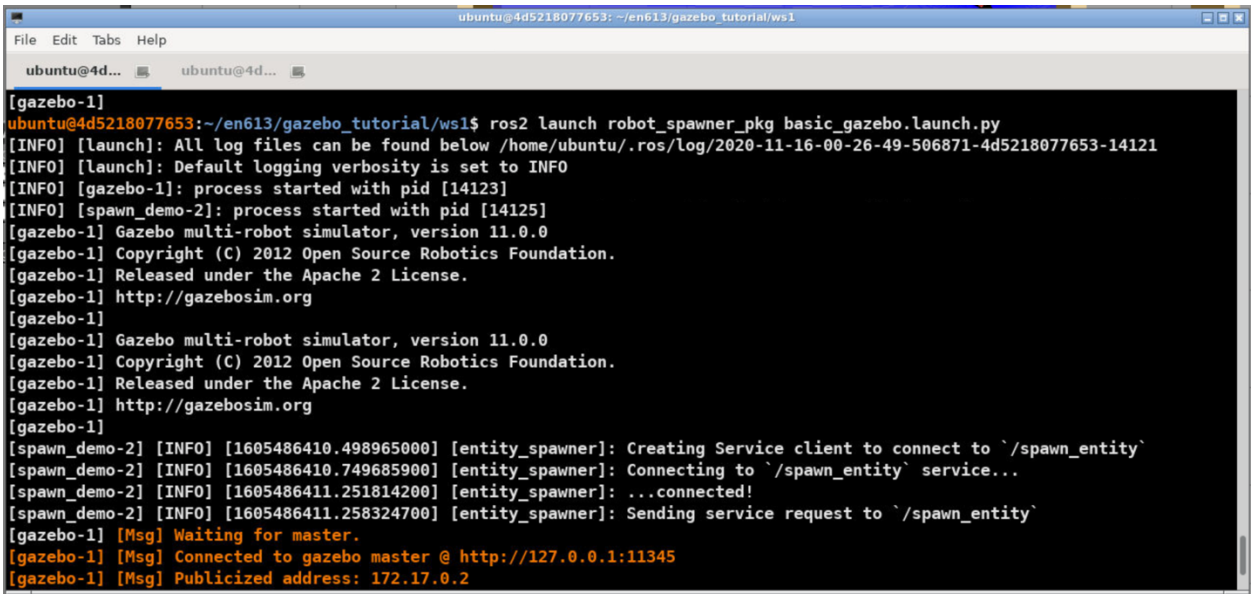
```
mkdir -p ~/final_ws/src
cp ~/Downloads/final_project_template.zip ~/final_ws/src/
cd ~/final_ws/src
unzip final_project_template.zip
cd ..
colcon build –symlink-install
```

2. Launch Gazebo using the robot_spawner_pkg

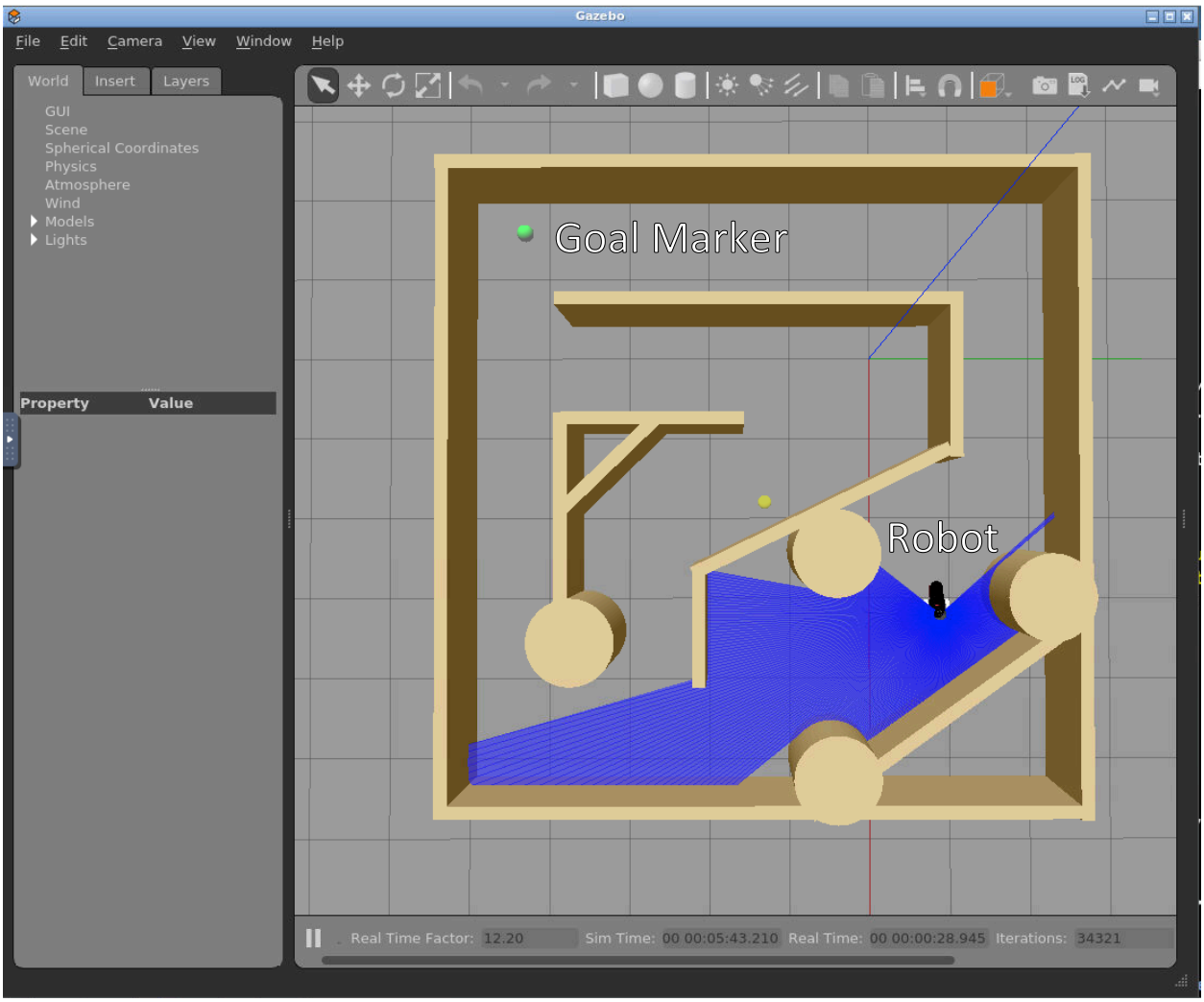
The robot_spawner_pkg includes a launch file for starting gazebo with the ROS2 plugins already enabled and then spawning the BasicBot inside the gazebo simulation. To test this run the following command.

```
cd ~/final_ws/
source install/setup.bash
Ros2 launch robot_spawner_pkg final_gazebo_spawn.launch.py
```

The following output should appear in your terminal.



Then you should see the following window pop up.



Finally this file will launch a node publishing the goal position under the name '/en613/goal' with the type geometry_msgs/Pose.

If you run the "ros2 topic echo /en613/goal" command you will see the following terminal output.

```
ubuntu@4d5218077653:~/en613/final_project$ ros2 topic echo /en613/goal
position:
  x: -1.5
  y: -4.25
  z: 0.1
orientation:
  x: 0.0
  y: 0.0
  z: 0.0
  w: 1.0
```

3. Start the placeholder controller and estimator

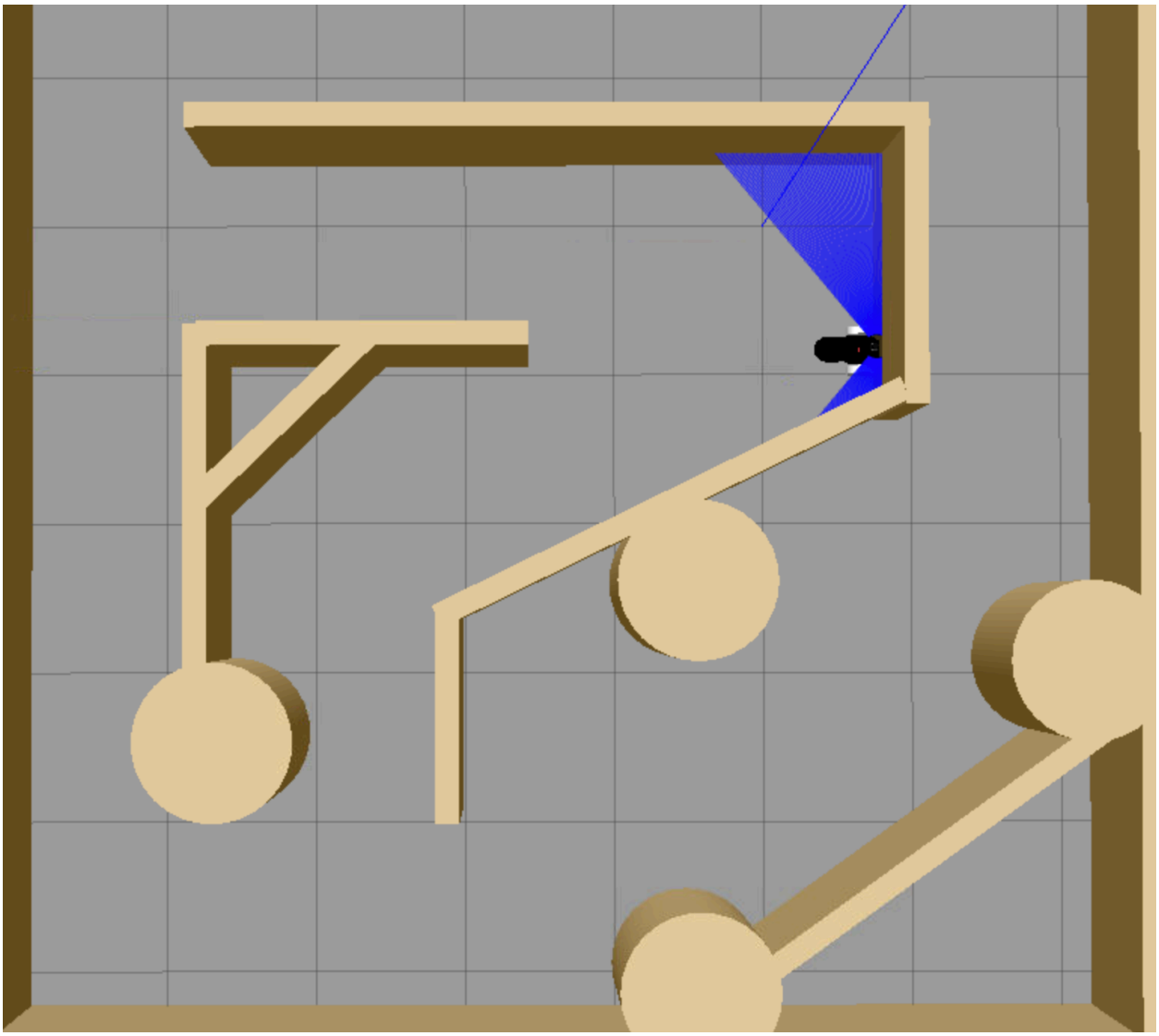
Now start the placeholder controller and estimator nodes. These nodes do not provide any particular functionality but provide examples of how to subscribe to the output of the gazebo simulation and publish commands to control the robot. Use the following command to launch these placeholder nodes.

```
ros2 launch robot_control_pkg placeholder.launch.py
```

You should see the following output from the terminal.

```
ubuntu@4d5218077653:~/en613/gazebo_tutorial/ws1$ ros2 launch robot_control_pkg placeholder.launch.py
[INFO] [launch]: All log files can be found below /home/ubuntu/.ros/log/2020-11-16-00-37-48-343198-4d5218077653-14807
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [placeholder_control-1]: process started with pid [14809]
[INFO] [placeholder_estimator-2]: process started with pid [14811]
[placeholder_estimator-2] [INFO] [1605487076.093187100] [PlaceholderEstimator]: Transform isn't available, waiting...
[placeholder_estimator-2] [INFO] [1605487082.107755800] [PlaceholderEstimator]: Transform isn't available, waiting...
›
```

Then after several seconds the robot should begin moving slowly forward until it collides with a wall. As shown below.



Robot_spawner_pkg

This package includes all of the model files and launch files necessary to start gazebo and spawn the basic robot model. This section will briefly address the important content of this package.

Launch

This directory contains the **basic_gazebo.launch.py** file which is used to start gazebo and execute the **spawn_demo.py** file. You can change location that the robot is spawned at by editing the arguments that are passed to the **spawn_entity** object at line 49.

```
spawn_entity = Node(package='robot_spawner_pkg', executable='spawn_demo',
                    arguments=['BasicBot', 'en613', '0.0', '0.0', '0.1'],
                    output='screen')
```

The first two arguments passed to the nodes are the name of the robot and the namespace which will be prepended to all published ros topics that are associated with the robot. The last three numbers are the x,y, and z positions of the robot's spawning location.

Models

This directory includes the models for both the robot [basic_robot] and the maze object [Maze_ql_1]. In these directory you can find a **model.config** file that provides a high level description of what the folder contains and a **model.sdf** that actually defines the model itself. It is recommended that you experiment with the basic_robot/model.sdf file to make it more efficient and adjust it to your tastes.

Robot_spawn_pkg

This folder contains the spawn_demo.py file which is the executable for actually spawning a robot in gazebo. This file is hardcoded to spawn the model in models/basic_robot/ but accepts the following command line arguments.

1. Robot name
2. Robot namespace
3. X position
4. Y position
5. Z position

Worlds

This folder contains the **maze.world** file. Another type of SDF file that is read by gazebo at launch and used to set all the environmental parameters and spawn entities.

Robot_control_pkg

The robot_control_pkg contains example files for creating a controller node and estimator node. These files do not do anything currently except drive the robot very slowly forward. It is up to you to populate this package with functional components that will allow it to drive through a maze.

Launch

This folder contains the placeholder.launch.py file that will launch both the placeholder controller and placeholder estimator nodes.

Robot_control_pkg

This file contains source code for the placeholder controller and estimator.

placeholder_estimator

> Currently all this node does is subscribe to the odom and scan messages published by gazebo. Then publishes the transform between odom and the robot's chassis frame as an estimated state. It publishes this state every time it receives an odom message.

placeholder_controller.

> Currently all this node does is subscribe to the estimated state message and publishes a twist message to the cmd_vel topic. To make this function properly it will also need to either subscribe to a goal pose message or receive a goal location as a command line argument.

Initial Steps

The files in this package should provide you with a good start towards building your own robotic control system. Here are a few things you can do to start working with these packages.

Fix the sensor on the basic robot

Currently the laser sensor on the robot is partially obscured by the robot body. Causing some of the laser scans to always return 0.0. Try to stop this from occurring. Either by editing the sensor to either reduce the field of vision or altering its position on the robot.

Update the controller to accept a goal

Currently the controller doesn't have any knowledge of where to go. Try writing a new node that publishes the goal position and add it to the launch file. Then update the controller code to subscribe to this topic. Finally see if you can cause the vehicle to drive in a straight line towards this goal.

Instructions

Generate two videos of different scenarios

The spawn_scenario node supports 4 prepackaged scenarios which spawn the robot and goal position in different positions. For this project you will need to record two videos of the robot completing two of the available scenarios. To do this you will need to edit the launch file. Before installation this is located in "**robot_spawner_pkg/launch**" and after you build the package using "**colcon build**" it will be located in "**install/robot_spawner_pkg/share/robot_spawner_pkg/launch**".

To change the scenario being launched locate the following entry in the final_gazebo_spawn.launch.py.

```
spawn_entity = Node(package='robot_spawner_pkg', executable='spawn_scenario',
                    arguments=['BasicBot', 'en613', '3'],
                    output='screen')
```

The scenario number is the last item in the arguments list, passed in after 'BasicBot' and 'en613'. It has a value between [0,3] and in this example is set to 3.