The ROS Benders

Project Report for BLG456E - Robotics Course

17 January 2017

This project report is prepared

by

Can Yılmaz Altıniğne - 150130132
Ece Naz Sefercioğlu - 150130140
Gamze Akyol - 150140142
Oğuzhan Can - 150160710

- Table of Contents -

1. Abstract

In this document, all the implementation progress of the term project for Robotics course is explained in terms of the main problem that our robot handles, design of this problem, technical details of the solution for this problem, analysis of our original proposal and the final results after the implementation phase by the members of The ROS Benders Team.

2. The Problem & Motivation

There are a lot of newcomers to the faculty every year and our faculty Electrical and Electronics Faculty (EEF) sometimes become a little complex for foreigners. We are focusing on people who do not know places of the rooms in EEF. Since only few foreigners are brave enough to ask their questions right away, lots of them get lost in EEF. When somebody asks “Do you know the room 1302 ?”, there are really few people who can answer. To overcome this problem and some moral issues, we decided to make a robot that show rooms in EEF and it starts working with a small “Hello!” message from user.

3. Problem Background & Review

The problem consists of place finding issues. People who are new to the Electrical and Electronics Faculty building, suffer while searching for the places they want to go. There is a possibility that they can not find the place because of their inability to find the place, they may not find anyone to guide them to their target place or they may be shy or abstentious about asking guidance from strangers. In the light of the results of a study, Horsch carried on on a group of students, she states that even there is an offer for help shy people feels reluctant to accept it. The reason is stated as they want to keep human interaction minimum [1]. Therefore, this kind of inability to find target places can be observed in museums, exhibition halls, libraries etc.

Moreover, problems can arise for people guiding as well. As it is stated by Montemerlo et al., nurses may endure difficulties while guiding elderlies to their target places to console with their doctors, consume regular meals, and many other destinations. These difficulties may include, keeping up with the slow pace of elderlies, loss of time while assisting simple actions for elderlies and not being able to spend this time slot on more constructive assignments [2]. All in all, human to human guidance could arise problems for both sides of the interaction.

[1] http://journals.sagepub.com/doi/pdf/10.2466/pr0.98.1.199-204

[2] http://robots.stanford.edu/papers/Pineau02f.pdf

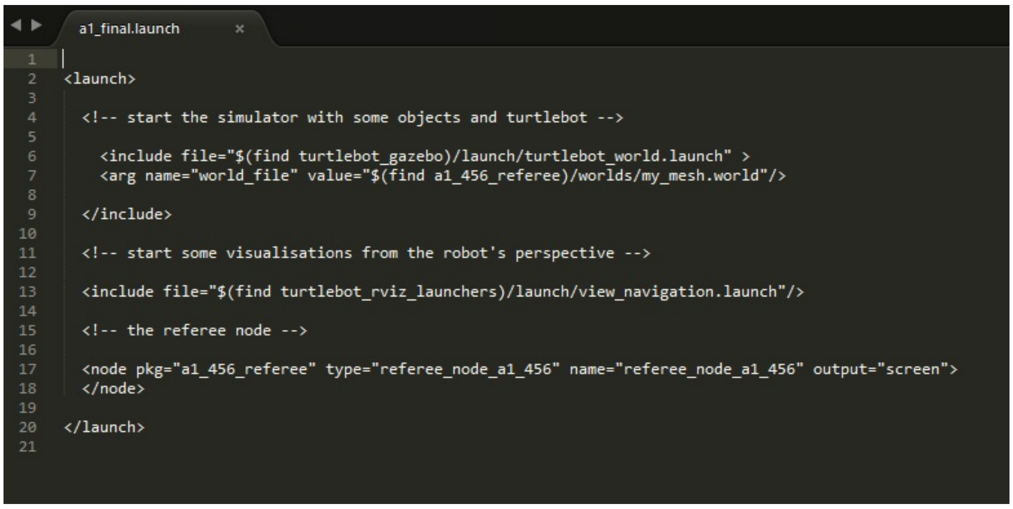
4. The Design of the Solution

There will be a robot that will wait for a 'Hello!' voice input from the user to start running the guidance program. After the robot starts running, it will ask for the digits of the room number where the user wants to go. After taking four digits, it will require an approval from the user whether the room number taken as an input is correct. If the inputs were wrong, it will ask for them again. Otherwise, it will start to drive the target classroom after telling user, ' ***I am going to an adventure Hoooooooray!*** '. When the target classroom is reached, the robot will stop near to the door of the classroom and its task is finished. In conclusion, user will be guided to his/her target classroom with the aid of a robot.

There are also possible matters that should be taken into consideration while the implementation of the solution. In technical aspects, environment noise may make it harder for robot to understand the user and cause time loss. The obstacles may fully close the path making robot unable to guide to the target place. Lastly, turtlebot may be perceived short for human height and can not be spotted for asking help. Socially concerning, ill thought people may think of stealing the robot or harm it for no reason. To sum up, the hardship this solution should handle arises from environmental and humane problems.

5. Main Technical Details of the Implementation

We used Assignment 1's launch file as base and kept working with it. We made sure that it is in the shape we want. We deleted some parameters and of course our dear polaris cars. After setting certain parameters, paths and nodes, we were done with the launch file.

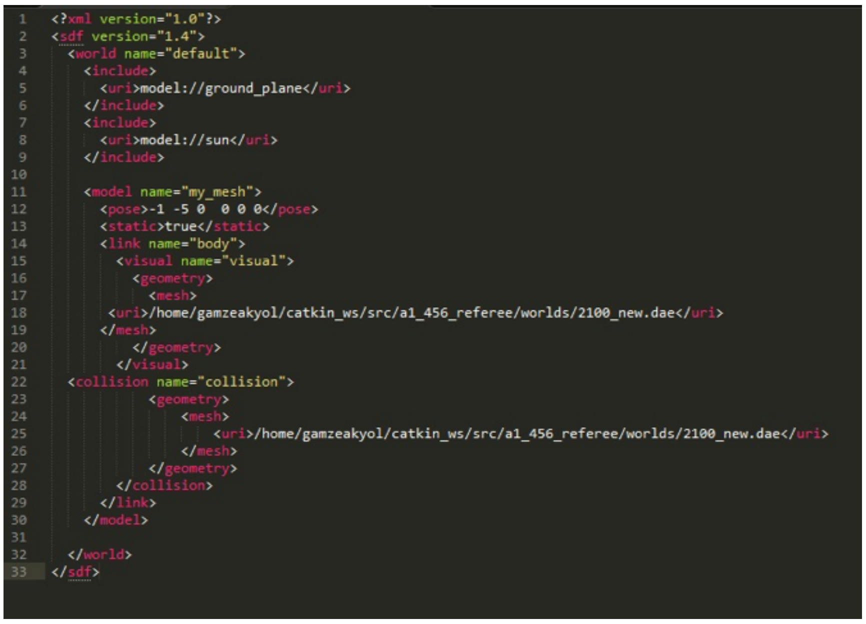


a1_final.launch

```

<launch>

  <!-- start the simulator with some objects and turtlebot -->

    <include file="$(find turtlebot_gazebo)/launch/turtlebot_world.launch" >
    <arg name="world_file" value="$(find a1_456_referee)/worlds/my_mesh.world"/>

  </include>

  <!-- start some visualisations from the robot's perspective -->

  <include file="$(find turtlebot_rviz_launchers)/launch/view_navigation.launch"/>

  <!-- the referee node -->

  <node pkg="a1_456_referee" type="referee_node_a1_456" name="referee_node_a1_456" output="screen">
  </node>

</launch>

```

Figure 1: Launch file

While creating world file main problem was opening the custom model in launch file. We solved the problem by giving the full path to the world file.



```
<?xml version="1.0"?>
<sdf version="1.4">
  <world name="default">
    <include>
      <uri>model://ground_plane</uri>
    </include>
    <include>
      <uri>model://sun</uri>
    </include>

    <model name="my_mesh">
      <pose>-1 -5 0  0 0 0</pose>
      <static>true</static>
      <link name="body">
        <visual name="visual">
          <geometry>
            <mesh>
     <uri>/home/gamzeakyol/catkin_ws/src/a1_456_referee/worlds/2100_new.dae</uri>
      </mesh>
          </geometry>
        </visual>
  <collision name="collision">
            <geometry>
                <mesh>
                    <uri>/home/gamzeakyol/catkin_ws/src/a1_456_referee/worlds/2100_new.dae</uri>
                </mesh>
            </geometry>
        </collision>
      </link>
    </model>

  </world>
</sdf>
```

Figure 2: World file

We used SketchUp to implement a .dae file to our simulation. Only 2100 corridor is used for testing purposes. After creating the world file, hard part has come - embedding the .dae file to our world and launch file. After embedding the dae file by using simple "roslaunch" command and giving package and launch files name, our world was ready to create the map. To create the map we used the instructions listed in this link. (http://learn.turtlebot.com/2015/02/03/8/) We created our map using gmapping.

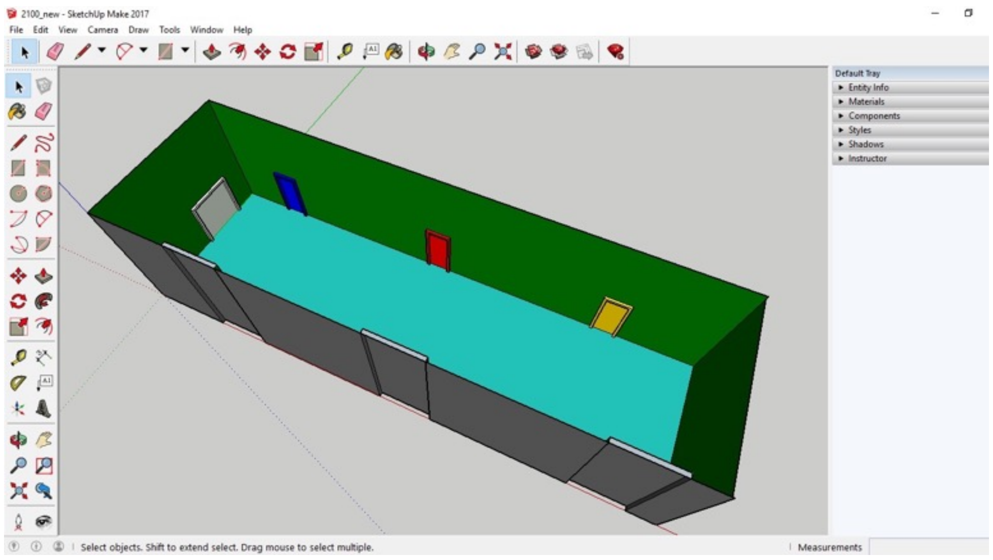

2100_new - SketchUp Make 2017
File Edit View Camera Draw Tools Window Help
Default Tray
Entity Info
Materials
Components
Styles
Shadows
Instructor
Select objects. Shift to extend select. Drag mouse to select multiple.
Measurements

Figure 3: SketchUp for implementation

The first pgm (tutorial.pgm) file that we created, turned out to messy and blurry. So we decided to use a manually created map file (2100.pgm). In the very

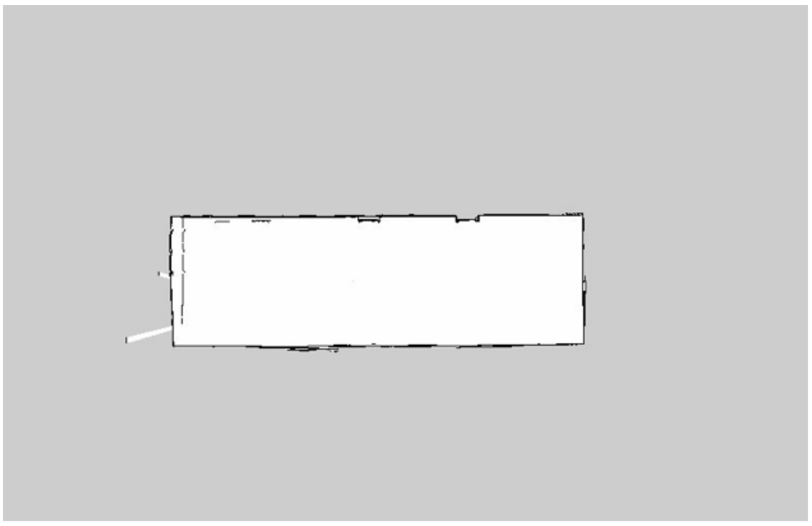


Figure 4: 2100.pgm file

end the map file we created manually did not work as we wanted; so, we returned to the first pgm file (tutorial). The issue we encountered here was creating the map. Since it is a long corridor and turtlebot's sensors have a limited range, robot kept losing itself in the map. After a few tries we managed to create a somewhat successful map.

A map for the first floor's second corridor is implemented on SketchUp 3D. The map generated by using SLAM map building technique by following the instructions guided in this link, using RViz. The main map consists of three doors with numbers, 2102,2104 and 2106 respectively and a symbolic door represents where the second corridor intersects with the fourth, fifth and third corridors.

For robot to achieve target place while avoiding obstacles, *move_base* package is used. Move_base package consists of *Adaptive Monte Carlo Localization (AMCL)* algorithm for localization and *A* algorithm* for path planning.

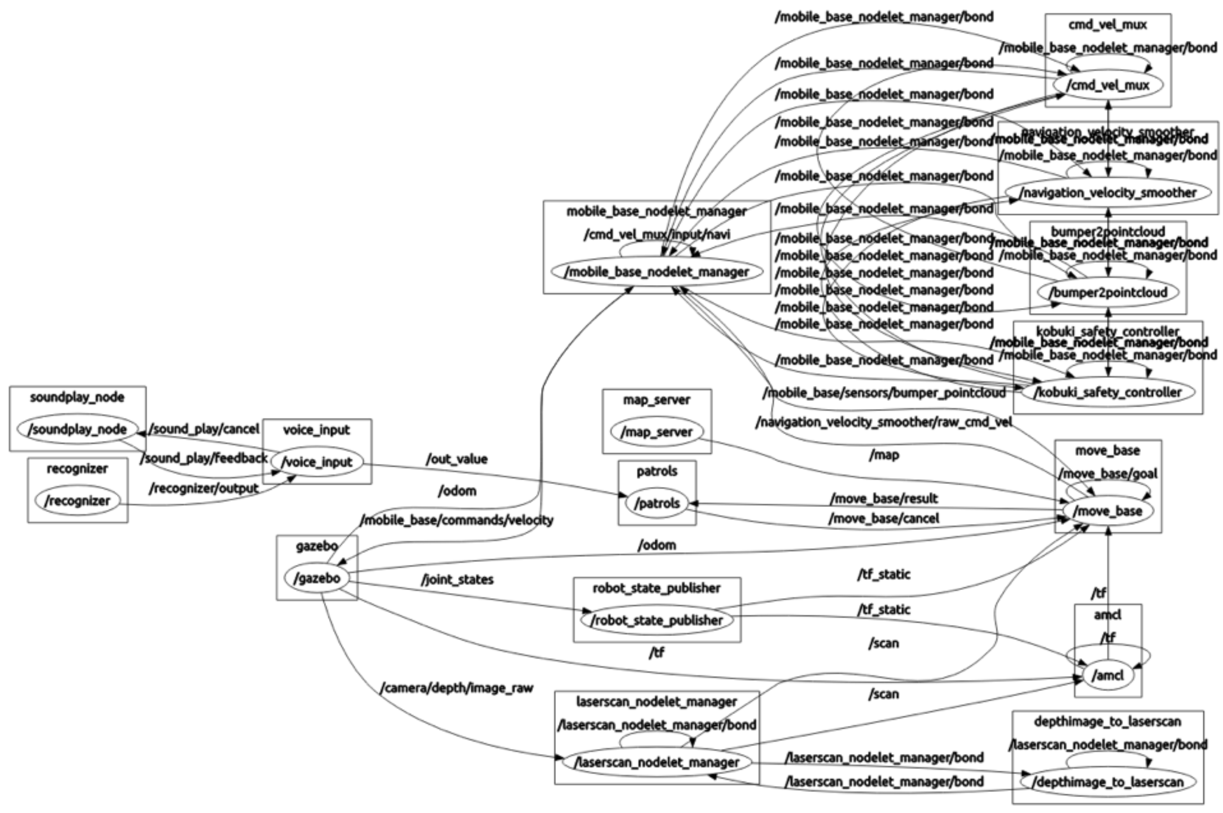


Figure 5 : RQT Graph for the project

AMCL calculates possible places that can be occupied by the robot. Then, by continuous calculations, it will calculate a most probable place for the robot and the robot can know where it is. As it is stated in 2015 by Quigley, Gerkey and Smart, in Rviz, green arrows near the robot show that estimation about where the robot might be, after AMCL works [3].

A* algorithm is a complete algorithm that can be used in path planning problem comfortably. This algorithm calculates the cost of the path that it has gone and also calculates the cost of the path to goal by computing Euclidean distance. By this approach, this algorithm always computes the path that has the least cost. In addition, if there is a trouble on the path, this algorithm can also compute the new path which has the minimum cost.

In obstacle avoidance and path planning problems, we used move_base because of its useful tools described above.

Human - Robot Interaction is achieved by letting user speak to robot for the initialization phase. For voice recognition, '*Pocketsphinx*' package for ROS is used. This package can recognize voices through microphone in conformity with the created dictionary. Firstly, exact words that the user wants package to recognize are added to a text file. Then this text file is converted into dictionary and pronunciation files from 'http://www.speech.cs.cmu.edu/tools/lmtool-new.html' . '*Pocketsphinx*' uses these files to choose what words to recognize. Robot's vocabulary is limited to recognize the digits, yes & no and the hello word. '*Pocketsphinx*' publishes the words that have been recognized through '*/recognizer/output*' topic. We subscribe this topic to take the words in the module and the module performs operations to transform the user's word into the room number which is used by patrol code.

To make robot speak, '*sound_play*' package for ROS is used. This package can make text to speech conversion by benefiting library functions. The module is written for ensuring the connection between recognizer code and the patrol code. This module determines the room number that user said and publishes this number to 'out_value' topic. The patrol code subscribes this topic, gets the value and starts moving.

[3] Programming Robots with ROS: A Practical Introduction to the Robot Operating System, p.173

6. Evaluation of Our Solution

The performance was unexpectedly well; since, the robot went to the all desired rooms without any problem. The robot can avoid obstacles (if it meets any) and detect some words (digits, yes, no, hello).

However, there are of course some issues. The voice package is not perfect and it needs a quiet environment to work, any noise other than the user interrupts process. The generated map is not excellent; because, there is an imaginary wall at the left of the map (pgm) file. Since it does not affect the performance of the robot, we did not fix it. Obstacle avoidance package also needs a little time to recognize the obstruction. But, this time interval is so small that it only becomes an issue if the obstacle appears in an instance at the very front of the robot. Robot also moves at a small speed; its speed can be increased.

7. Analysis of the Original Proposal

In the original proposal, there was also a work package including color recognition. But, a team member of us have received a VF and we had to remove this package. Also, we expected the robot to return to its starting place, however we did not manage to do it, since we ran out of time. Other parts concerning the team members had done according to proposal.

The robot went to the desired room even if it encountered some obstacles throughout the path. The AMCL based map and patrol (robot driving one) packages worked interestingly well together. The main issue we encountered is integrating the voice recognition package to the system, make it publish some data and making patrol package subscribing to this published data. In the end all went according to plan and system was completed as it was described.

Each member learned a lot of things such as the working system of ROS, creating some packages etc. However we wanted to show the most important things we learned;

- Can – "Never underestimate the difficulty of project integration."
- Ece – "Do not stop trying, as it will end in success"
- Gamze – "A person's motivation impacts all the team."
- Oğuzhan – "Always, but always; check the launch file!"

8. Detailed Instruction for the Software

There are package that are needed to be installed by using the bash commands below.

- sudo apt-get install gstreamer0.10-pocketsphinx
- sudo apt-get install ros-indigo-pocketsphinx
- sudo apt-get install ros-indigo-audio-common
- sudo apt-get install libasound2

Since the robot is initialized as it hears the word 'Hello', you definitely need a microphone to make the project work. Also a silent environment would be a great choice.

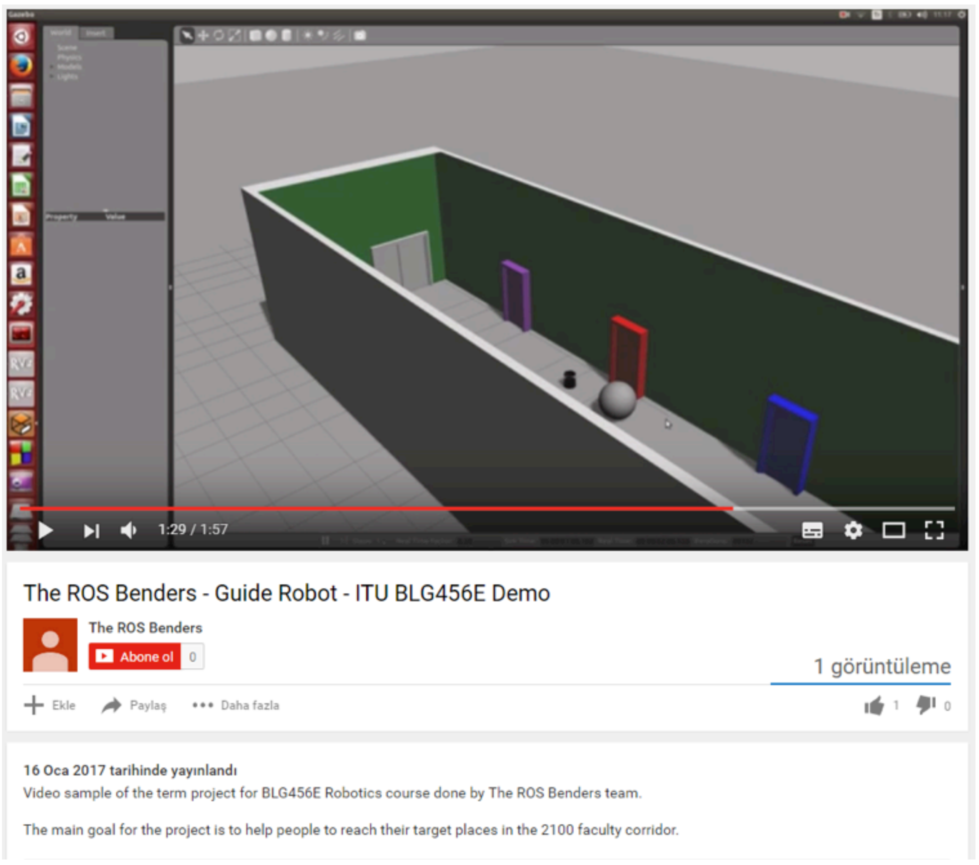
How to run the program through consoles:

- The user should use 5 terminals, so that s/he can run whole program.
- Each console's destination should be set to catkin workspace, and have their source command given as well.
- For each console, run: source devel/setup.bash

After source command, run below commands respectively:

- roslaunch a1_456_referee a1.launch
- roslaunch turtlebot_gazebo amcl_demo.launch map_file:=<file path to map's yaml file>
- roslaunch voice_recognition recognizer.launch
- rosrun voice_recognition voice_input.py
- rosrun patrol patrols.py

9. Video Demo



The ROS Benders - Guide Robot - ITU BLG456E Demo

The ROS Benders

16 Oca 2017 tarihinde yayınlandı

Video sample of the term project for BLG456E Robotics course done by The ROS Benders team.

The main goal for the project is to help people to reach their target places in the 2100 faculty corridor.

Figure 6 : Video Demo of the Project

We have added a video to demonstrate our project. Video demo can be streamed on Youtube through this link ‘https://www.youtube.com/watch?v=lojQ6psmjqw'.

GitHub repository for the project can be found in this address ‘https://github.com/The-ROS-Benders/ITU_BLG456E_ROBOT-ASSIGNMENT'.

10. References

- http://journals.sagepub.com/doi/pdf/10.2466/pr0.98.1.199-204

- http://robots.stanford.edu/papers/Pineau02f.pdf

- Programming Robots with ROS: A Practical Introduction to the Robot Operating System, p.173