TOBB University of Economics and Technology

Department of Computer Engineering

BIL496 Senior Design Project Report

Semester	2015-2016 Spring
Group Name	Introbot Team
Student Names and Email Adresses	Alper Taday – st101101020@etu.edu.tr Fırat Top – st101101047@etu.edu.tr Ecem Elvin Çevik – st111101037@etu.edu.tr
Advisor	Assist. Prof. Dr. Esra Kadıoğlu Ürtiş

Table Of Contents

İçindekiler

1. Introduction 4

1.1. Problem Statement 4

1.1.1. Problem Definition 4

1.1.2. Purpose 4

1.1.3. Scope 4

1.2. What is done? 5

1.3. Definitions and Abbreviations 5

1.4. Overview 6

2. Previous Work 6

2.1. Literature Servey 6

2.2. Comparison 8

3. Requirements 8

4. Overall Description 8

4.1. Product Perspective 8

4.2. System Interfaces 8

4.3. Interface Requirements 9

4.3.1. User Interface Requirements 9

4.3.2. Hardware Interfaces Requirements 9

4.3.3. External Interface Requirements 9

4.3.4. Functional Requirements 9

4.4. Non-functional Requirements 9

4.4.1. Performance Requirements 9

4.4.2. Design Constraints 9

4.4.2.1. Software System Attributes 9

5. Design 10

5.1. System Overview 10

5.2. Design Considerations 10

5.2.1. Design Assumptions, Dependencies, Constraints 10

5.2.2. Hardware Constraints 10

5.2.3. Software Constraints 10

5.2.4. Time Constraints **10**
5.2.5. Security Constraints **10**
5.3. Design Goals and Guidelines **11**
5.3.1. Speed Considerations **11**
5.3.2. Reliability **11**
5.3.3. User Interactivity **11**
6. Data Design **11**
6.1. Data Description **11**
7. System Architecture **13**
7.1. Architectural Design **13**
7.2. Description of Modules **13**
7.3. Design Rationale **13**
7.4. User Interface Design **13**
7.4.1. Overview of User Interface **13**
8. Implementation **14**
8.1. Group Members' Work Load **14**
8.2. Configuration Management **15**
8.3. Libraries and Tools **15**
8.3.1. Libraries **15**
8.3.2. Tools **15**
8.4. Project Timeline **15**
8.5. Software Development Process **16**
8.5.1. Process Model **16**
8.5.2. Used Packages and Algorithms **16**
9. User's manual **20**
10. Test Results and Evaluation **22**
11. Conclusion **22**
12. References **23**
13. Lines-of-code Table **24**

1. Introduction

This final report provides a complete description of Introducer Robot System as a final design project of TOBB Economy and Technology University Computer Engineering Department, including all the functions, specifications, test results and manuals. This project is developed by Alper Taday, Fırat Top and Ecem Elvin Çevik.

This document will refer to functionality, that is what the resulting system is supposed to do, external interfaces which interacts the users, performance, attributes, that is if the application is portable, or maintainable, and design constraints imposed on the implementation such as implementation language, input specifications and output expectations. In addition, test results and manuals that user can apply simply are included in this report.

1.1. Problem Statement

The problem statement gives comprehensive description of the problem which is the reason to be developed the Introducer Robot and ease the understanding of the usage and the development of the application.

1.1.1. Problem Definition

Focusing on the real-life problems developed for this purpose, introducer robot will respond to the user's guiding needs.

The aim of the robot acting like an “introducer” student, should get the candidate student from information desk and guide to table from which students would like to get information. The robot will perform this task only in main hall.

1.1.2. Purpose

The purpose of this document is to present a detailed description of the introducer robot. It will explain the aim and features of the system, the interfaces of the system, product function, what the system will do, the constraints under which it must operate and how the system will react to external reviver. This document is planned for both the third parties and the developers of the system and is created by viewing SRS and SDD. This document is the fundamental of all of the system.

1.1.3. Scope

The introducer robot shall perform in same building and same floor. It will be able to plan the route from the entrance to the table of relevant department. It should know their coordinates without the help of users and by the agency of kinect sensor. If someone strayed into the robot, the robot should detect people or some obstacles and the robot goes from around. If people do not located sensor’s visibility, the robot should go on for destination.

More specifically, the robot should require to internet connection. It should know their coordinates from map and user should not interfere the functions of the robot.

1.2. What is done?

In the first place, system's requirement analysis is made based on the problem described in "problem statement" section. Afterwards, the initial timeline is planned and literature survey is started and the similar works are searched deeply in regard to the result of the analysis immediately. After the searches had been done a map of place where robot will perform tasks is made with the help of SLAM algorithm (Simultaneous Localization And Mapping). Then the system which make robot navigate was decided to use in project which is called navigation stack. After, all the factors which effect the timeline is clarified, initial timeline is revised and the implementation is started based on the factors, the timeline and the subjects described in the design report. In accordance with the timeline, implementation had been done within the scope of the project.

1.3. Definitions and Abbreviations

TERM	**DEFINITION**
Turtlebot 2.0	The robot that is used in Project.
Introducer Robot	Turtlebot 2.0
Wireless Network	Any type of computer network that uses wireless data connections for connecting network nodes.
User	Someone who uses the robot from UI.
Kinect Sensor	A line of motion sensing input devices by Microsoft for Xbox 360 and Xbox video game consoles and Windows PCs.
Third Parties	Any person who has interaction with the system who is not a developer.
SLAM	Simultaneous localization and mapping is an algorithm which robot uses while mapping the environment.
AMCL	Adaptive Monte Carlo Localization is an algoritihm which makes robot understand its position on a given map.
Navigation Stack	Mechanism that navigates robot.

1.4. Overview

This document contains a detailed description and usage of Object Tracking System. In the introduction part, it mostly gives a general overview about the project including the definition of a real world problem that project intends to solve. Also in this part, the purpose and the scope of the system and what is done parts are explained. Finally, definitions and abbreviations are described.

Second part of the document is the information about similar projects and how they differs from this project. Moreover, user and literature survey of this part explains the similar projects done before is declared.

Third part of the document is the overall description of the project. This part explains the product perspective of the application the functions included in the application and the constraints, assumptions and dependencies of the desired application. Also, use case diagram are displayed. In additon, specific requirements, data model and description, behavioral model and description are mentioned in this part.

The design of the project are explained in the fourth part of the document. This part includes system overview, design considerations, data design, system architecture, user interface design, libraries and tools.

In the fifth part of this document, implementation of the system is explained detailed.

Next part includes the user's and deploy manual how to user use the system simply.
In the seventh part of the document, test result and evoulation are shown and test results are evaluated.

The final part is conclusion part which gives a brief summary about the whole document.

2. Previous Work

2.1. Literature Servey

- Interactive Map Labeling for Service Robots introduced by Alexander J. B. Trevor, Akansel Cosgun, Jayasree Kumar, Henrik I. Christensen: Maps that include semantic information such as labeled structures, objects, or landmarks can be useful to service robotic systems. [1]

- Bringing Together Human and Robotic Environment Representations introduced by Elin A. Topp, Helge Huettenrauch, Henrik I. Christensen, and Kerstin Severinson Eklundh: Human interaction with a service robot requires a shared representation of the environment for spoken dialogue and task specification where names used for particular locations are depending on personal preferences. [2]

- Topological Modelling for Human Augmented Mapping introduced by Elin A. Topp and Henrik I. Christensen: Service robots designed for domestic settings need to navigate in an environment that they have to share with their users. Thus, they have to be able to report their current state and whereabouts in a way that is comprehensible for the user. [3]

- Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization introduced by Howie Choset, Member, IEEE, and Keiji Nagatani, Member, IEEE: One of the critical components of mapping an unknown environment is the robot's ability to locate itself on a partially explored map. [4]

- Roomba is a series of autonomous robotic vacuum cleaner sold by iRobot. Roomba was introduced in 2002. As of Feb 2014, over 10 million units have been sold worldwide. Roomba features a set of basic sensors that help it perform tasks. For instance, the Roomba is able to change direction on encountering obstacles, detect dirty spots on the floor, and detect steep drops to keep it from falling down stairs. It uses two independently operating wheels that allow 360 degree turns in-place. Additionally, it can adapt to perform other more creative tasks using an embedded computer in conjunction with the Roomba Open Interface. Incomparison to roomba, TurtleBot is also avoid encountering obstacles thanks to it's sensors. [5]

- SaviOne: The first time SaviOne goes to a hotel, it travels around and maps the building. Hotel staff members then need only program in a particular room number and the robot knows how to get there. It uses a variety of sensors to navigate and move around, including depth cameras, sonar and laser rangefinders, and is able communicate with elevators via Wi-Fi. SaviOne is based on Robot Operating System (ROS) technology, which is overseen by the Open Source Robotics Foundation. Our Introducer Robot System also uses ROS (Robotic Operating System) to make hardware abstraction. [6]

- Implementation Of Particle Filtering in a Turtlebot by Frederik Penné: The interest in autonomous mobile robot applications is growing. In order to be autonomous the robots need to have a certain awareness of their environment. Therefore maps are build and sensor data is obtained. One of the problems is to localize a robot with the use of sensor data in a known environment. This can be done with the use of a localization method called particle filter. [7]

- Obstacle Detection and Avoidance Using TurtleBot Platform and XBox Kinect by Sol Boucher: Any robot that is to drive autonomously must be able to detect and avoid obstacles that it might encounter. Traditionally, this problem has been solved using systems of one or more RGB cameras utilizing complicated and computationally-expensive computer vision algorithms, somewhat unreliable ultrasonic distance sensors, or laser-based depth scanners. However, Microsoft's recent release of the XBox Kinect has opened up new areas of research in the areas of computer vision and image understanding, and this same device can be employed for obstacle detection. [8]

2.2. Comparison

Compared to other systems, Introducer Robot has several similiarities. It can be said that Introducer Robot makes map of environment, it takes destination goals and go to that goal without making crashes due to avoiding encountering obstacles thanks to sensors.

3. Requirements

This section provides details about requirements of the application which will be developed based on the problem defined in “problem statement” section. Requirements is for designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators or other external systems.

4. Overall Description

4.1. Product Perspective

This project consists of two parts: computer application and turtlebot. Computer application is a graphical user interface which an user can give commands to the Turtlebot 2.0. The user can select department of university from GUI.

Figure 1

4.2. System Interfaces

The project is modified using ROS programming and Python. ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks,' such as Player, YARP, Orocos, CARMEN, Orca, MOOS and Microsoft Robotics Studio.

The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server. These are explained in greater detail in our Conceptual Overview.

ROS is not a realtime framework, though it is possible to integrate ROS with realtime code. The Willow Garage PR2 robot uses a system called pr2_etherCAT, which transports ROS

messages in and out of a realtime process. ROS also has seamless integration with the Orocos Real-time Toolkit. [9]

4.3. Interface Requirements

4.3.1. User Interface Requirements

- The system shall provide an user interface which will connect user to the robot.
- The terminal shall provide input field which the user will be able to execute program.

4.3.2. Hardware Interfaces Requirements

- The system shall comprise a Kinect-360 sensor.
- The system shall comprise a Kobuki Base.
- The system shall comprise bumper sensors.
- The system shall comprise a speaker.
- The system shall comprise a telephone for robot head and image. [10]

4.3.3. External Interface Requirements

- The system shall communicate to the robot with the help of wireless connections.

4.3.4. Functional Requirements

- The robot shall perform tasks between desks which are given as inputs from GUI.

4.4. Non-functional Requirements

4.4.1. Performance Requirements

- The robot's Kinect sensor senses at minimum 2.3 ft. distance.
- The robot's Kinect sensor senses at maximum 19.7 ft. distance.

4.4.2. Design Constraints

4.4.2.1. Software System Attributes

- **Usability:** The system will be able to used easily by users, with help of GUI.
- **Reliability:** There are no explicitly stated reliability requirements.
- **Availability:** The user will be able to use the robot whenever the user desires.
- **Security:** There are no explicitly stated reliability requirements.
- **Maintainability:** The introducer robot must be clearly separate from the UI to allow different user interfaces to be developed in the future.

5. Design

5.1. System Overview

The Introducer Robot performs between desks which are given as inputs from GUI by user in same area. While the robot does task, it requires to wireless network.

The robot can do task of guide easily. It knows their coordinates without the help of users and by the agency of kinect sensor, it abstains from obstacles and it can move without crashing. If obstacle does not located sensor's visibility, the robot detects obstacles through the agency of crashing sensor and the robot changes the direction.

This project consist of two parts: computer application and turtlebot. Computer application is a graphical user interface. The user will be able to navigate the robot to any requested rooms which are given as inputs by using ubuntu terminal.

5.2. Design Considerations

5.2.1. Design Assumptions, Dependencies, Constraints

- The tasks must be in the same location.
- Wireless connection must be stable.

5.2.2. Hardware Constraints

- The robot senses obstacles by using kinect sensor but if obstacle does not located sensor's visibility, the robot detects obstacles through the agency of crashing sensor and the robot changes the direction.
- The robot perceives the object by using laser scan. Extreme sun light affects negatively the laser scan. Hence, the robot cannot senses the object.
- Voice of turtlebot out from the speaker. We link the speaker between the turtlebot with aux cable. The speaker's charge can withstand approximately 5 hours. The speaker can be charged from computer's USB port.

5.2.3. Software Constraints

- We are restricted to using ROS.
- We are restricted to using Python for writing and editing ROS packages.

5.2.4. Time Constraints

- The robot will wait approximately just a five second after going to relevant desk.

5.2.5. Security Constraints

- There are not security constraints.

5.3. Design Goals and Guidelines

5.3.1. Speed Considerations

- The robot's maximum speed is 65 cm/s.
- The robot's maximum rotational speed is 180 deg/s.

5.3.2. Reliability

- There are no explicitly stated reliability requirements.

5.3.3. User Interactivity

The user is able to click relevant department button from GUI, then the robot goes towards destination point and returns to the starting point.

6. Data Design

6.1. Data Description

The robot makes mapping. The desks' coordinates are marked on the map which is created by the robot using SLAM gmapping algorithm. The system has data model which includes department desks' coordinates. We get the desks' coordinate from the rviz. Rviz is a simulation program.

Figure 2: Social Facilities 4th hall

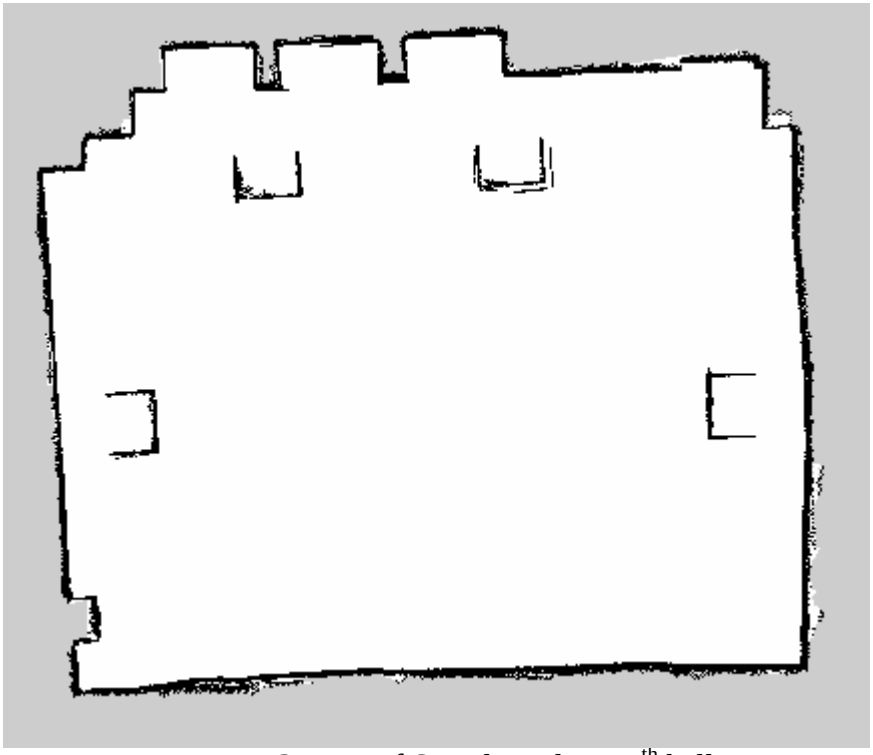

Figure 3: Map of Social Facilities 4th hall

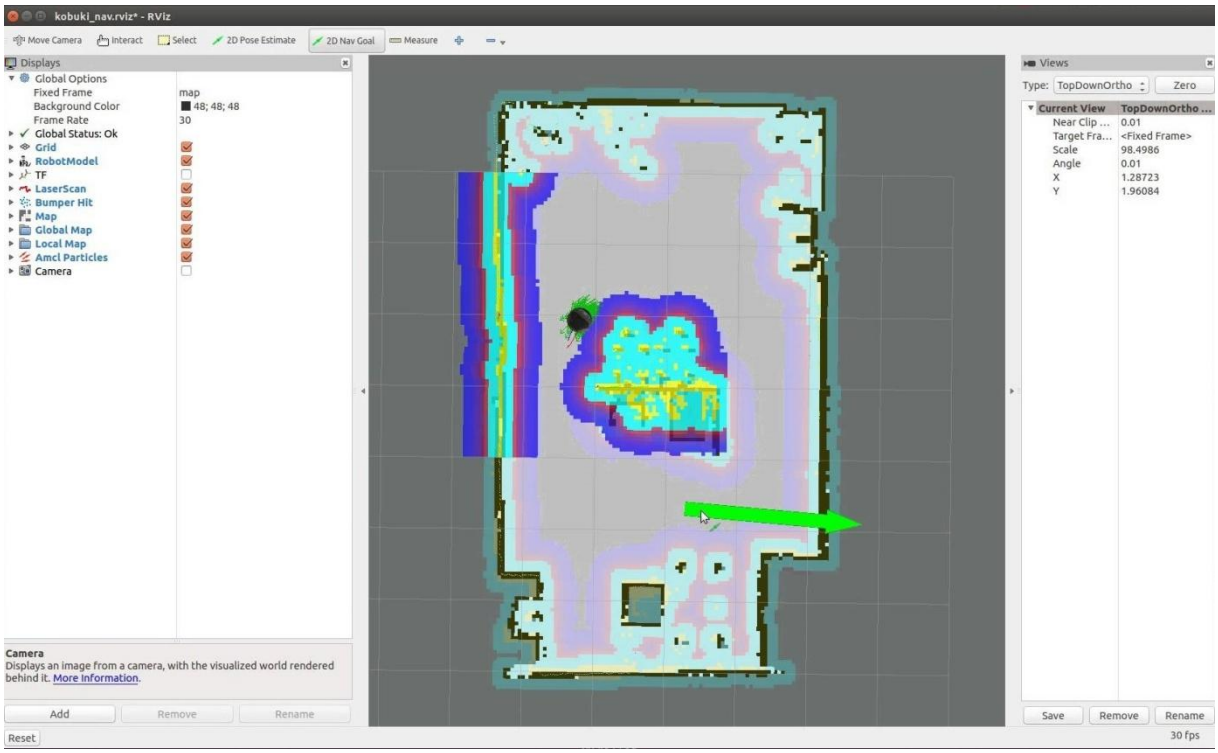

Figure 4: Rviz [11]

There is another data model for previously created ROS packages objects. The data model has direct impact on how the content exist in the system.

7. System Architecture

Sensor – Actuator – Controller Architecture is used in this system.

7.1. Architectural Design

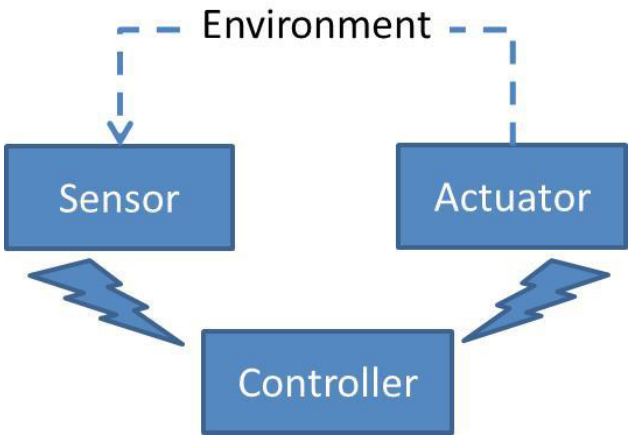

Figure 5 [12]

7.2. Description of Modules

- **Sensors:** This module is about turtlebot's sensors; bumper sensor and Kinect 360 sensor and provides information about environment.
- **Controller:** Graphical user interface is a controller.
- **Actuator:** The robot's wheels are prompted by DC Motors

7.3. Design Rationale

Generally, in control systems Sensor – Actuator – Controller Architecture is used. Because of this reason, we prefer this design architecture.

7.4. User Interface Design

7.4.1. Overview of User Interface

The user selects relevant desk of department button from GUI, and the robot goes towards destination point and returns to the starting point. There are 4 buttons on the GUI. These are department's buttons. When the user clicks any button, turtlebot goes to relevant department and say "i have brought you to the desk of … engineering." for candidate student and returns to the starting point. If turtlebot stacks or aborts in any way, an error message appears for attendant student and robot says "an error occured". For this situation there is a 'recover' button on the menubar. When the user clicks this button, turtlebot returns to the starting point and is recovered system of the turtlebot. Also there is a 'help' button for the user. This button opens the 'ReadMe' document for user.

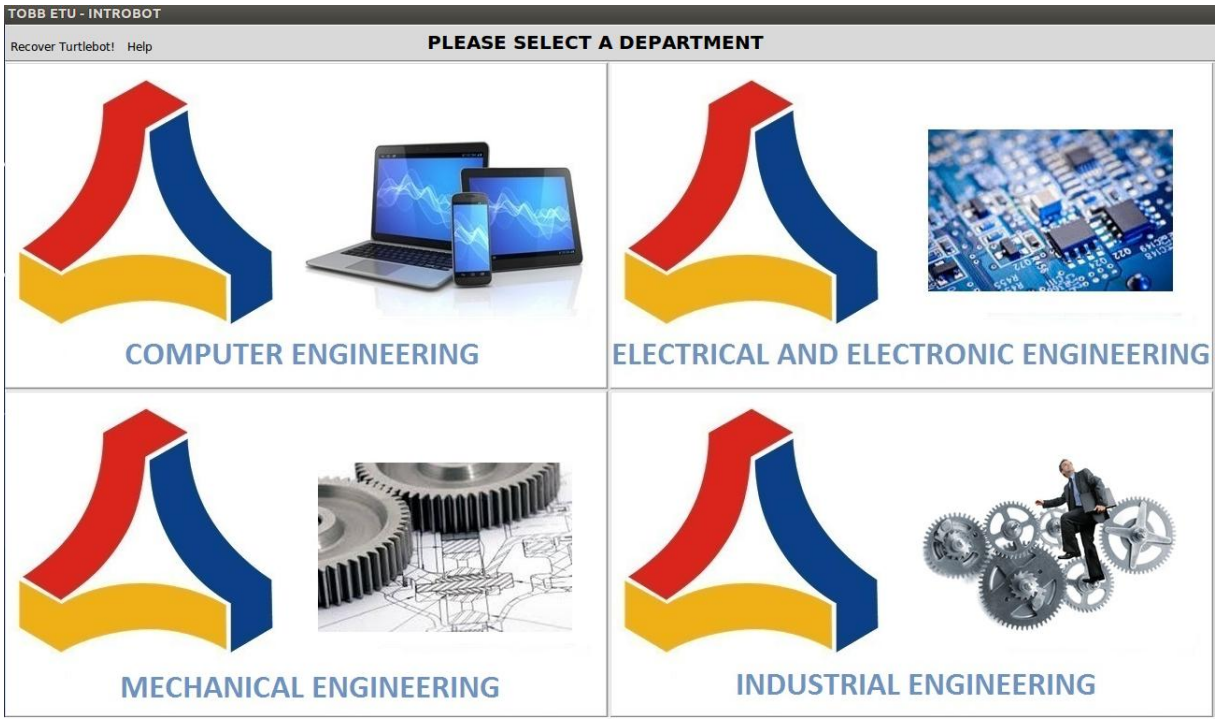

Figure 6: Graphical User Interface of Project

8. Implementation

8.1. Group Members' Work Load

Alper Taday

- Creating a ROS package and launch file
- Integrate ROS commands with GUI
- Writing linux script for executable program
- Killing all process for exitting of system
- Preparing SRS, SDD and final report

Fırat Top

- Creating a map and getting coordinates
- Design of GUI
- Embed sounds libraries to the program
- Preparing read me file and final report

Ecem Elvin Çevik

- Design turtlebot
- Hardware setup

8.2. Configuration Management

System versions and parts are kept in backups. So, version recycling and maintaining can be done simply.

8.3. Libraries and Tools

8.3.1. Libraries

- ROS Packages
- Python
- Tkinter Libraries
- Script Language
- Several Linux Commands

8.3.2. Tools

- Guake Terminal: Guake is a drop-down terminal for GNOME Desktop Environment. Like similar terminals, it is invoked with a single key, and hidden by pressing the same key again. [13]

- Open SSH Server: OpenSSH is a set of computer programs that provides encrypted communication sessions over a computer network using the SSH protocol. It was created as an open source alternative to the proprietary Secure Shell software suite offered by SSH Communications Security. [14]

- Sublime Text: It is a sophisticated text editor for code, markup and prose. [15]

8.4. Project Timeline

3 - 4 Weeks:
- ROS libraries setup
- Preparation SRS - SDD report
- Mapping

5 - 6 Weeks:
- Specification of department desk on the map

7 - 11 Weeks:
- Navigation process
- First demo
- First presentation

12 - 14 Weeks:
- Test
- Poster
- Final demo
- Final presentation

- Final report

8.5. Software Development Process

8.5.1. Process Model

In this project, waterfall process model is used.

Waterfall Process: The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases, typically:

- Requirements analysis resulting in a software requirements specification
- Software design
- Implementation
- Testing
- Integration, if there are multiple subsystems
- Deployment
- Maintenance

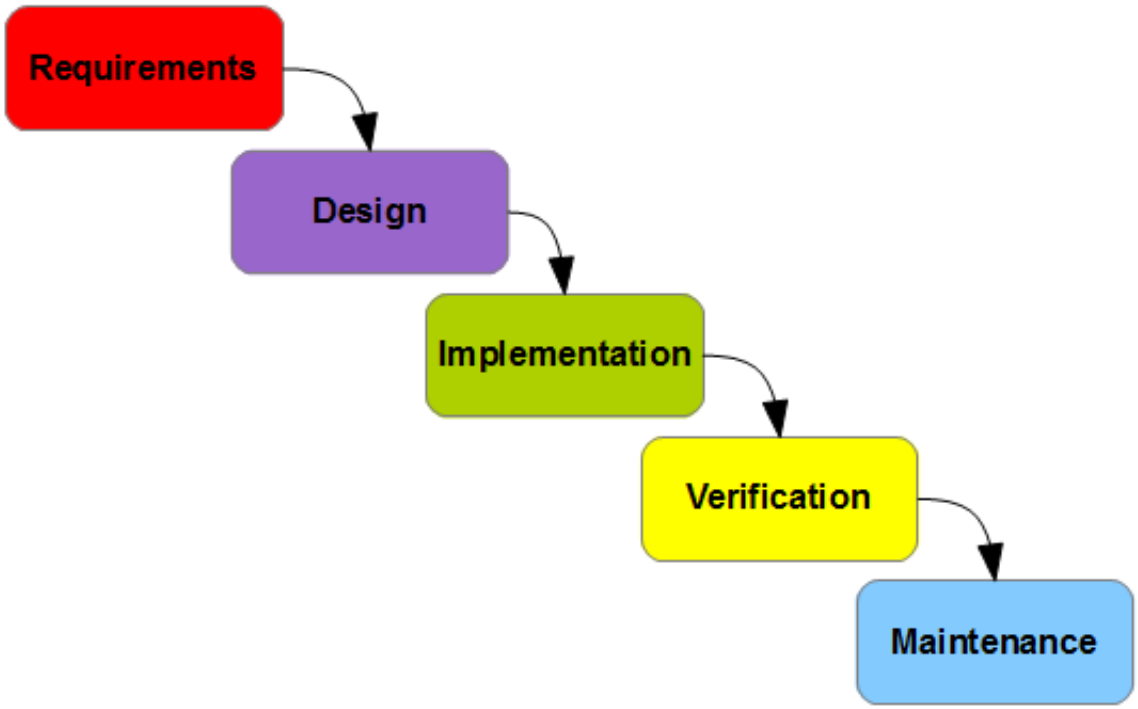

Figure 7: The activities of the software development process represented in the waterfall mode. [16]

8.5.2. Used Packages and Algorithms

Slam algorithm and amcl packages are used.

SLAM (Simultaneous localization and mapping):
In robotics, Simultaneous localization and mapping(SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping

track of an agent's location within it. While this initially appears to be a chicken-and-egg problem there are several algorithms known for solving it, at least approximately, in tractable time for certain environments.

SLAM algorithms are tailored to the available resources, hence not aimed at perfection, but at operational compliance. Published approaches are employed in self-driving cars, unmanned aerial vehicles, autonomous underwater vehicles, planetary rovers, newly emerging domestic robots and even inside the human body. Slam algorithm is used mapping. [17]

AMCL(Adaptive Monte Carlo Localization):
AMCL is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map. [18]

AMCL algorithn ise used for localization of robot in a given map which is made by using SLAM algorithm. With this approach robot's location and movements in map are determined.

i. move_base_msgs Package:

This package contains the messages used to communicate with the move_base node. These messages are auto-generated from the MoveBase.action action specification. [19]

ii. MoveBase.action:

Action Definition

- geometry_msgs/PoseStamped target_pose
- geometry_msgs/PoseStamped base_position

The target_pose is the goal that the navigation stack attempts to achieve. The base_position given as feedback is the current position of the base in the world as reported by tf. For the move_base node, the target_pose is projected into the XY plane with the Z axis pointing up when attempting to achieve a goal. [19]

iii. TF:

Tf is a package that lets the user keep track of multiple coordinate frames over time. Tf maintains the relationship between coordinate frames in a tree structure buffered in time and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time. [20]

Actionlib:
The actionlib stack provides a standardized interface for interfacing with preemptable tasks. Examples of this include moving the base to a target location, performing a laser scan and returning the resulting point cloud, detecting the handle of a door, etc. [21]

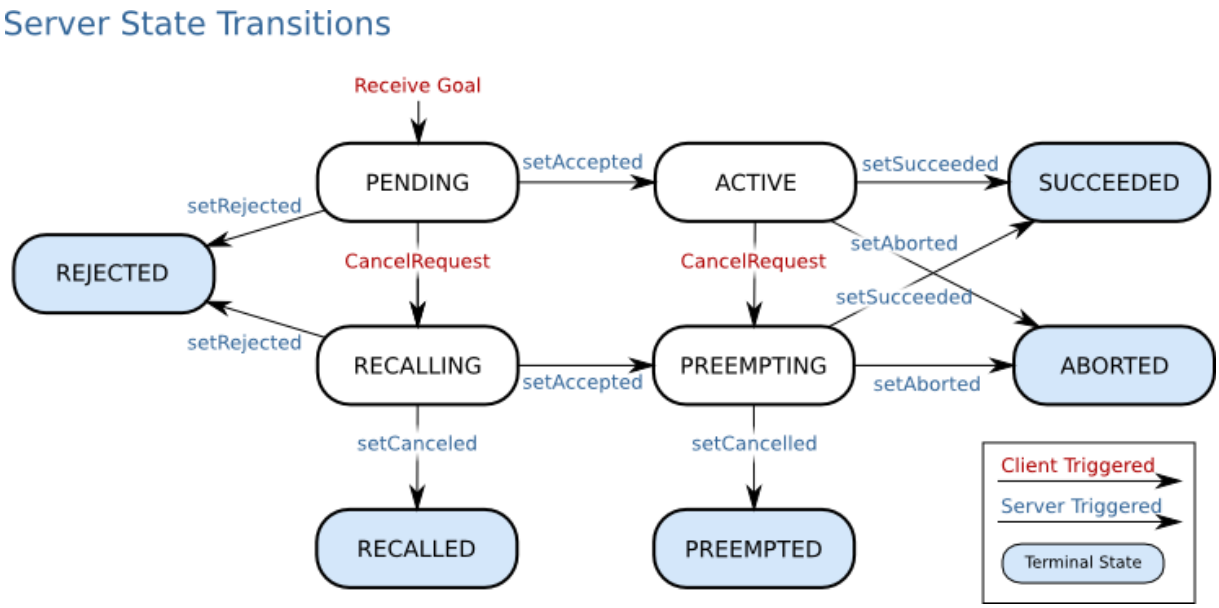

Figure 8: Server State Transitions [22]

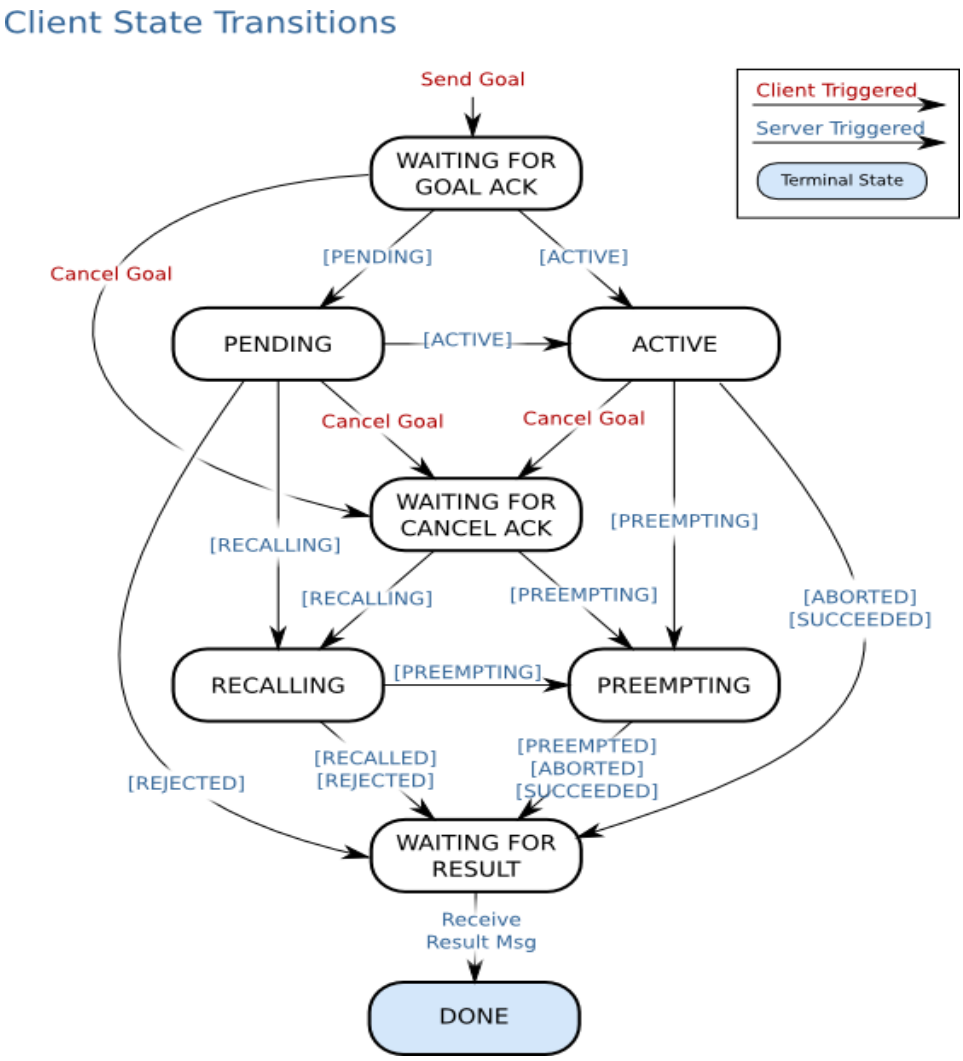

Figure 9: Client State Transitions [22]

Figure 8 and Figure 9 shows us what happens when a goal is sent by client to the server in state diagrams.

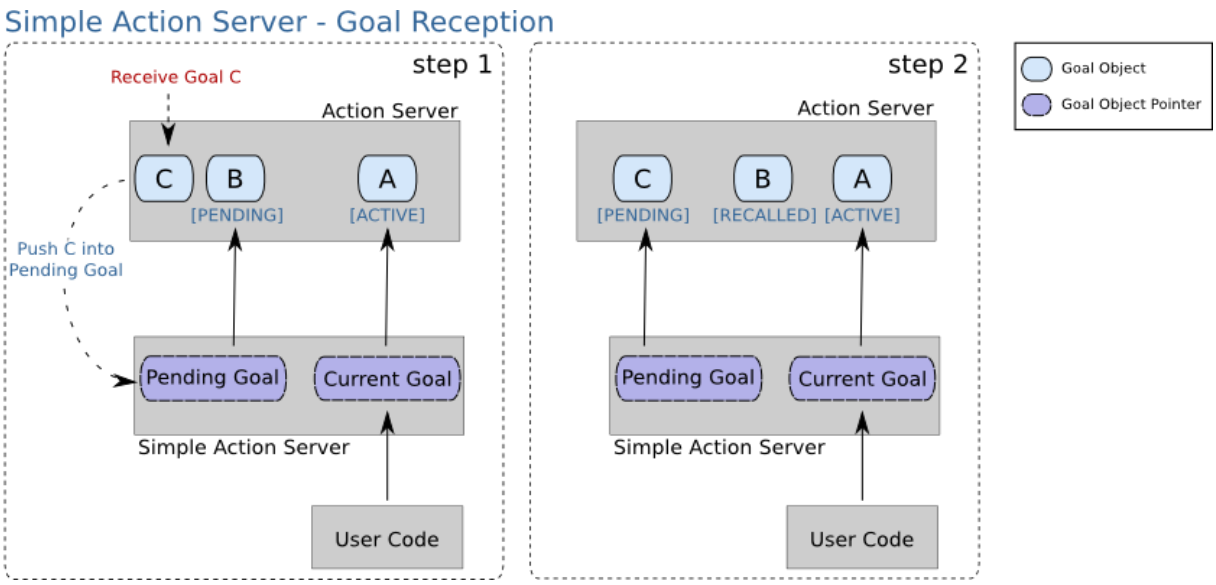

Figure 10: Goal Reception of Action Server [22]

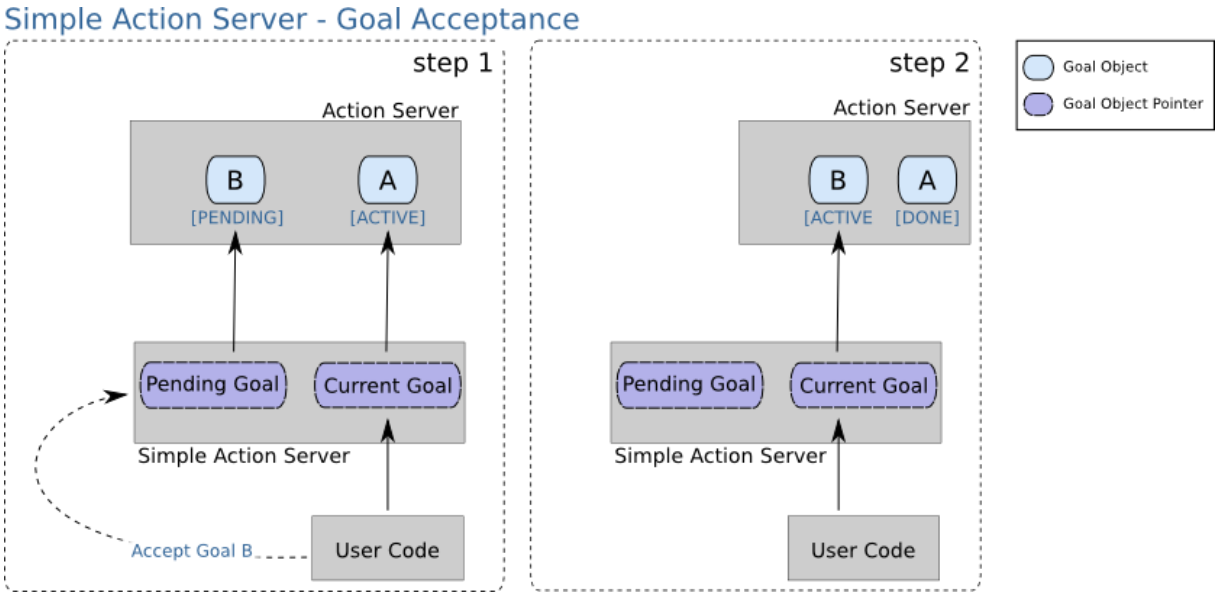

Figure 11: Goal Acceptance of Action Server [22]

iv. geometry_msgs:

Geometry_msgs provides messages for common geometric primitives such as points, vectors and poses. These primitives are designed to provide a common data type and facilitate interoperability throughout the system. [23]

9. User's manual

Problems that user may encounter:

1. If there is no network connection robot can't perform intended job, because it is controlled via another device which uses internet connection.

2. If floor that robot moves is not appropriate, there can be problems in rotational movements because of excessive friction.

3. If there are many obstacles around robot which cause robot to not move, sending goal mission aborts due to the high amount risk of crashing. But fort his case the system has recovery mechanism.

4. Robot should not be lifted while it is moving or performing a job, if it is lifted kidnapped robot problem appears and robot can not localize itself within the map.

   In robotics, the kidnapped robot problem commonly refers to a situation where an autonomous robot in operation is carried to an arbitrary location. The kidnapped robot problem creates significant issues with the robot's localization system, and only a subset of localization algorithms can successfully deal with the uncertainty created; it is commonly used to test a robot's ability to recover from catastrophic localization failures. [24]

5. Robot's Kinect sensor and cliff sensor may be affected by dense sunlight, because of this there can be problems about navigation of robot.

Deployement's manual:

1. In order to make work with robot, it is needed to make connection to the robot. This is done with creating ssh connection to the device's ip that is connected to the robot.

2. After connection is established turtlebot's all necessary packages and nodes must be run in order to make robot fully functional. In other words TurtleBot Software must be bringed up.

3. It is enough to use gmapping node of ROS to make mapping of environment. Mapping can be done manual or automated. Network connection is necessary while mapping because all operations about robot is implemented with this connection. Furthermore robot should not be lifted while mapping due to the kidnapped robot problem which is explained in user's manual. If robot is lifted, it is needed to start gmapping again.

4. Another point that is to be considered while mapping is odometry. Odometry is the use of data from motion sensors to estimate change in position over time. Odometry is used by some robots, whether they be legged or wheeled, to estimate (not determine) their position relative to a starting location. This method is sensitive to errors due to the integration of velocity measurements over time to give position estimates. Rapid and accurate data collection, equipment calibration, and processing are required in most cases for odometry to be used effectively. It is necessary not to navigate robot forward and then backward or vice versa, because it causes shifts on odometry of robot.

5. After mapping is done, map server should be used in order to make robot know the map. Map server loads map to robot. This can be done with this console command: rosrun map_server map_server mymap.yaml (my_map is the name of map that is created by gmmapping.)

6. AMCL is used for localization of robot and getting coordinates of this location on the map.

7. Navigation of robot is provided by action server. In order to start action server, first navigation stack must be The TurtleBot navigation is ruled (as in almost any other ROS robot) by a combination of launch and yaml files.

   The Navigation Stack is fairly simple on a conceptual level. It takes in information from odometry and sensor streams and outputs velocity commands to send to a mobile base. Use of the Navigation Stack on an arbitrary robot, however, is a bit more complicated. As a pre-requisite for navigation stack use, the robot must be running ROS, have a tf transform tree in place, and publish sensor data using the correct ROS Message types. Also, the Navigation Stack needs to be configured for the shape and dynamics of a robot to perform at a high level. [25]

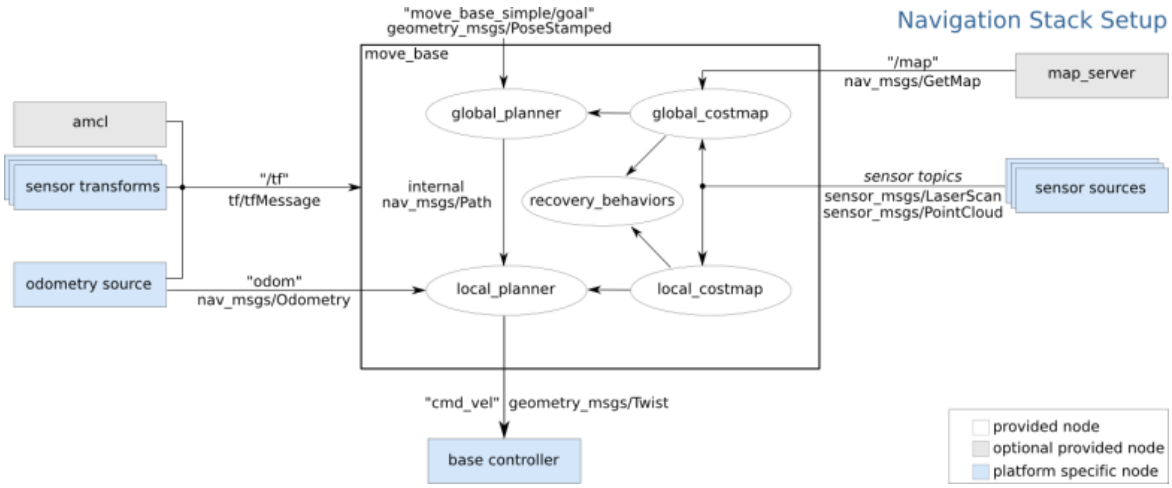

Figure 12: Navigation Stack Setup [26]

The robot must satisfy some requirements before it uses the navigation stack:

• The navigation stack can only handle a differential drive and holonomicwheeled robots. The shape of the robot must be either a square or a rectangle. However, it can also do certain things with biped robots, such as robot localization, as long as the robot does not move sideways.

• It requires that the robot publishes information about the relationships between all the joints and sensors' position.

• The robot must send messages with linear and angular velocities.

• A planar laser must be on the robot to create the map and localization.

Alternatively, you can generate something equivalent to several lasers or a sonar or you can project the values to the ground if they are mounted in another place on the robot. [27]

10. Test Results and Evaluation

First mapping process is done. First maps were not very clear at initial tries. Many maps are made until a proper map achieved. Tests are done at Social Facilities, desks' coordinates are used as inputs. There are problems about wheels, so when robot makes rotational movements there happened hatches. Sending goal process is repeated a lot of times to test if it goes to correct destination. Turtlebot's starting point is crucial.

11. Conclusion

In conclusion, the introducer robot project is about using the robot to guide the candidate students for university in introducing days. The robot performs its job in same building and same floor. This robot will be able to get the student candidate who need some information about department of university. The candidate will be able to easily follow it for going to information desk of university.

It uses wireless connection for guiding. Because of this, wireless connection is important for the robot. This final report provides a complete description of Introducer Robot, as a final design project of TOBB ETU Computer Engineering Department, including all design issues.

In the introduction part, it defines the problem that our team intends to solve, and then explains the purpose of this document. Immediately after that, the scope of the project is explained in an exhaustive way, the content of the document is summarized, and important terms, acronyms and abbreviations relevant to the project is clarified. In the system overview part, a general description of the software system including its functionality and matters related to the overall system and its design is provided.

Second part of the document is the information about similar projects and how they differs from this project. Moreover, user and literature survey of this part explains the similar projects done before is declared.

In the third part, specification requirements of system are explained clearly.
In the fourth part, data design is elaborated. The way that our team transforms information domain into data structures, the major data and system entities that are stored is explained.

Subsequently, a data dictionary is provided in order to provide a detailed description of the system entities and major data, including data objects, their attributes and methods. Libraries and tools are explained.

The fifth part covers the system implementation.

The sixth part includes user's manual that shows how user can send goals to robot. In addition how to system runs in other computers are meaned in deployment manual part.

In the seventh part, test that be run on system and their results are explained.

At least line of code table is given.

12. References

[1] http://www.cc.gatech.edu/~acosgun3/papers/trevor_map_annotation.pdf

[2] https://smartech.gatech.edu/bitstream/handle/1853/37966/04059204.pdf

[3] https://smartech.gatech.edu/bitstream/handle/1853/37967/04058720.pdf

[4] http://biorobotics.ri.cmu.edu/papers/paperUploads/00928558.pdf

[5] https://en.wikipedia.org/wiki/Roomba

[6] http://www.gizmag.com/savioke-savione-service-robot/33369/

[7] https://uhdspace.uhasselt.be/dspace/bitstream/1942/19513/1/12352512014H54.pdf

[8] https://www.cs.cmu.edu/~sboucher/turtlebot.pdf

[9] http://wiki.ros.org/ROS/Introduction

[10] https://openclipart.org/image/2400px/svg_to_png/173914/Robot-Head-Even.png

[11] https://i.ytimg.com/vi/xCRsszVAP1E/maxresdefault.jpg

[12] http://shop.ciseco.co.uk/ciseco-wireless-introduction/

[13] https://en.wikipedia.org/wiki/Guake

[14] https://en.wikipedia.org/wiki/OpenSSH

[15] http://www.sublimetext.com

[16] http://www.bawiki.com/wp-content/uploads/2013/11/Widipedia_Waterfall.png

[17] https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

[18] http://wiki.ros.org/amcl

[19] http://wiki.ros.org/move_base_msgs

[20] http://wiki.ros.org/tf

[21] http://wiki.ros.org/actionlib

[22] http://wiki.ros.org/actionlib/DetailedDescription

[23] http://wiki.ros.org/geometry_msgs

[24] https://en.wikipedia.org/wiki/Kidnapped_robot_problem

[25] http://wiki.ros.org/navigation

[26] http://wiki.ros.org/navigation/Tutorials/RobotSetup

[27] MARTINEZ, Aaron; FERNÁNDEZ, Enrique. Learning ROS for Robotics Programming. Packt Publishing Ltd, 2013

[28] https://github.com/markwsilliman/turtlebot/blob/master/go_to_specific_point_on_map.py

13. Lines-of-code Table

Module/Package	Program	Programmer(s) or the sources borrowed from	Purpose of the Program	Lines of Code Borrowed	Lines of Code Developed
	app.py	Alper-Fırat	Sends goal to robot		267
	launch.py	Fırat Top	Runnig introbot.launch		3
	introbot.sh	Alper Taday	Running launch.py & app.py		7
	introbot.launch	Alper-Fırat	Activating the robot & loading the map of area		7
	go_to_specific_point_on_map.py	Mark Silliman [28]	Sends the robot to specific point on map	60	
			TOTAL	**60**	**284**