NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

MASTER'S THESIS

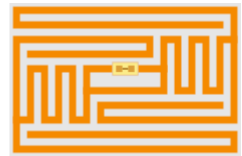
ROS-Lidar-RFID technologies for navigation and item detection in a Warehouse

Paris A. Ioakeimidis
A.M.: M1397

Supervisor
Hadjiefthymiades Stathes, Associate Professor

Thesis outline

- Analysis of used technologies
  - ROS – Robotic Operating System
  - Turtlebot
  - Lidar
  - RFID
  - Etc.
- Study of above technologies
- Integration of above technologies
- Navigation algorithm implementation
- RFID and Position data collection
  - Hash
  - Apache Kafka
  - Etc.
- Experiments : Simulated and Physical worlds



Intro

- Current Situation
  - **Amazon's**
  - **Alibaba**
  - Etc.
- Logistics, e-commerce growth, etc.
- Robots and automated solutions bring efficiency , accuracy and speed to the services provided.

- Questions
  - How to automatically detect goods in a warehouses?
  - How to navigate a robot in a predefined root?
  - What are the available technologies?
  - Why these technologies?
  - How to integrate these technologies?
  - Etc.

Intro

- **Amazon(USA)** robots is to move 24 hours per day a great amount of goods (100K robots)
- **Alibaba(China)** perform approximately 70 percent of the work by carrying goods (>100kg)
- **Ocado(UK)** online grocer they use automatic arm to store and retrieve products.
- **Swisslog(Swiss)** is a company that designs, develops and delivers automation solutions in domains like health, warehouses and distribution centers. Swisslog together with KUKA have introduced CarryPick

Previous Work

- Michail Chatzidakis, Michail Loukeris, Kostis Gerakos, Stathes Hadjiefthymiades. 2016. E-Pres: **Monitoring and Evaluation of Natural Hazard Preparedness At School Community**. Pervasive Computing Research Group. National And Kapodistrian University of Athens. Department of Informatics and Telecomunications
- Kostas Kolomvatsos, Michael Tsiroukis, Stathes Hadjiefthymiades. 2017. **An Experiment Description Language for Supporting Mobile Iot Applications. National And Kapodistrian University of Athens**. Department of Informatics and Telecomunications
- Kshitija Deshmukh, Ashitha Ann Santhosh, Yogesh Mane, Saurabh Verma, Sdhana Pai. Nov 2015. **Robotic navigation and inventory management in warehouses**. International Journal of Soft Computing and Artificial Intelligence. ISSN. 3(2): 75-79
- Kaiyu Zheng. Sep 2016, **ROS Navigation Tuning Guide**
- **Warehouse logistics chain through tracking, identifying, and detecting objects**, such works are: (a)Wamba and Chatfield(2010), (b)Hassan et al.(2012), (c)Rodrigues(2009), (d)Pacciarelli et al.(2011), (e)Yan et al.(2008), and (f)Wang et al.(2015), others use **RFID technology for indoor location sensing**.

Problem Description

Autonomous robot navigation and object detection in a warehouse environment

- Proper technology selection
- Technology selection and configuration
- Map Creation of the environment
- Route identification
- Autonomous navigation of the robot
- Detection and avoidance of obstacle
- Object detection
- Exact Object position identification
- Handling the collected data
- Etc...

Proposed Solution

- Turtlebot 2 robot and ROS – Robotic Operating System for navigation in warehouse.
- Lidar called RPLIDAR 360 degrees laser scanner for map creation and navigation.
- EDL – Experimental Descriptive Language to produce JSON file containing the path(mission) for the Turtlebot autonomous navigation.
- Implementation of an algorithm in Python and ROS for autonomous navigation that will send goals positions to Turtlebot.
- RFID technology (MTI RU 861-010 reader/module, RFID tags of different types, and RFID Antennas) for detecting objects located in warehouse.
- Data collected is published to Kafka topic, ROS topic, saved in a Hash structure, in a log file.

Concepts and Technologies

- Turtlebot 2
- RPLIDAR A2
- RFID - MTI RFID RF Module RU 861-010, tags & antennas

- Linux
- ROS
- Python
- Kafka
- EDL

Robotic frameworks

1. **ROS : Robotic Operating System**
2. **Player** : Player Project is a Free Software tools for robot and sensor applications.
3. **MOOS** : MOOS is a Cross Platform Software for Robotics Reasearch
4. **YARP** : YARP stands for Yet Another Robot Platform
5. **Orocos** : Orocos stands for Open Robot Control System. Orocos is free and open, licensed as LGPL
6. **Carmen** : Carmen stands for Carnegie Mellon Robot Navigation Toolkit.
7. **Orca** : Orca is a project with software reusability in mind
8. **Microsoft Robotics Studio** : Microsoft Robotics Studio is Windows-based and .NET-based programming environment
9. **Tekkotsu :** Robotics framework thats name means 'iron bones in Japanese'
10. **OpenRDK** : Modular framework is another robotic framework which is open-source as well

ROS – Robotic Operating System

- **ROS stands for Robotic Operating System**
- Development of ROS started back in 2007 by the Standford Artificial Intelligence Laboratory (SAIL)
- ROS isn't really an operating system it is a framework (meta-operating system)
- Needs a real operating system Linux Ubuntu
- Collection of all necessary tools, packages, and services, needed for robotics project implementation

Figure 11: Indigo ROS logo, source: http://wiki.ros.org/Distributions

ROS – Why?

Positive Aspects of ROS

- ROS runs on top of Turtlebot 2
- **Free**
- **Open source**
- Real robots out there are running ROS
- All the necessary libraries and tools for robotics development
- All the necessary packages already implemented
- **Reusability** of already implemented code
- ROS has an **distributed** internal approach - more **stable** and hard to fail
- Any **component** can be **integrated easily** into ROS due to it messaging system
- **Sensors** can be mounted on top of ROS robots
- ROS follows a **modular** approach
- **Large Community** – accepted and supported by the robotics community
- **Widely used** today by researchers and companies
- Development of multiple components of any robotic system accessible and easy
- Theoretically **language independent**
- **Scalable**
- Etc...

Negative

- Learning Curve

ROS – Basics

Main standard operating system services provided

- Hardware abstraction
- Low-level device control
- Implementation of commonly-used functionality
- Message passing between processes
- Package management

ROS Graph Concepts

- **Nodes** : Simple executable that uses ROS to exchange messages
- **Messages** : Nodes communication , publishing messages to topics.
- **Topics** : ROS nodes publish or subscribe to topics
- **Services :** Send requests and receive responses
- **Master :** Name service for ROS. ROS nodes locate each other
- **Rosout :** Similar to stdour & stderr
- **Roscore** : ROS Master, ROS Parameter Server and rosout logging node

ROS filesystem

- **Packages**: which contain libraries, tools, executable, etc. (lowest level)
- Package Manifest files: contain the description of the ROS packages and the dependencies of the packages.
- **Stacks**: Collection of packages (higher level library)
- Stack Manifest files: contain the description of the Stacks and the dependencies.

ROS basic commands

- Roscd
- Roscore
- Rosed
- Rosls
- Roslaunch
- Rosrun
- Rosmake
- Rostopic
- Rosmsg
- Rosservice
- Rossrv
- Rosnode
- rqt_tools
- catkin_create_package
- catkin_make

ROS – Navigation Stack

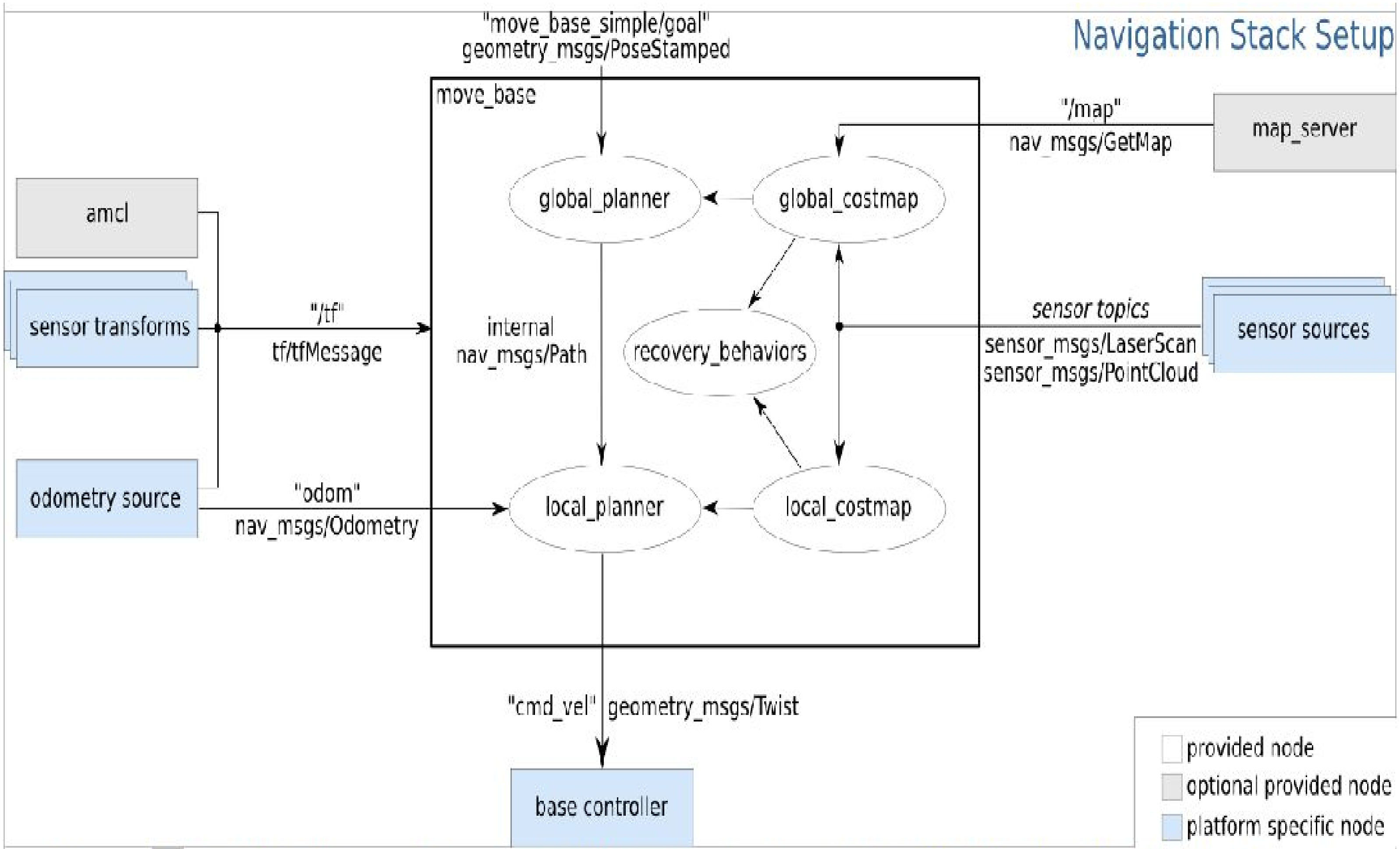


Figure 14: In this diagram is presented the organization of the ROS Navigation Stack. Source: http://wiki.ros.org/navigation/Tutorials/RobotSetup

- Map Creation
- Path Planning
- Localizing

ROS – Move_base Recovery

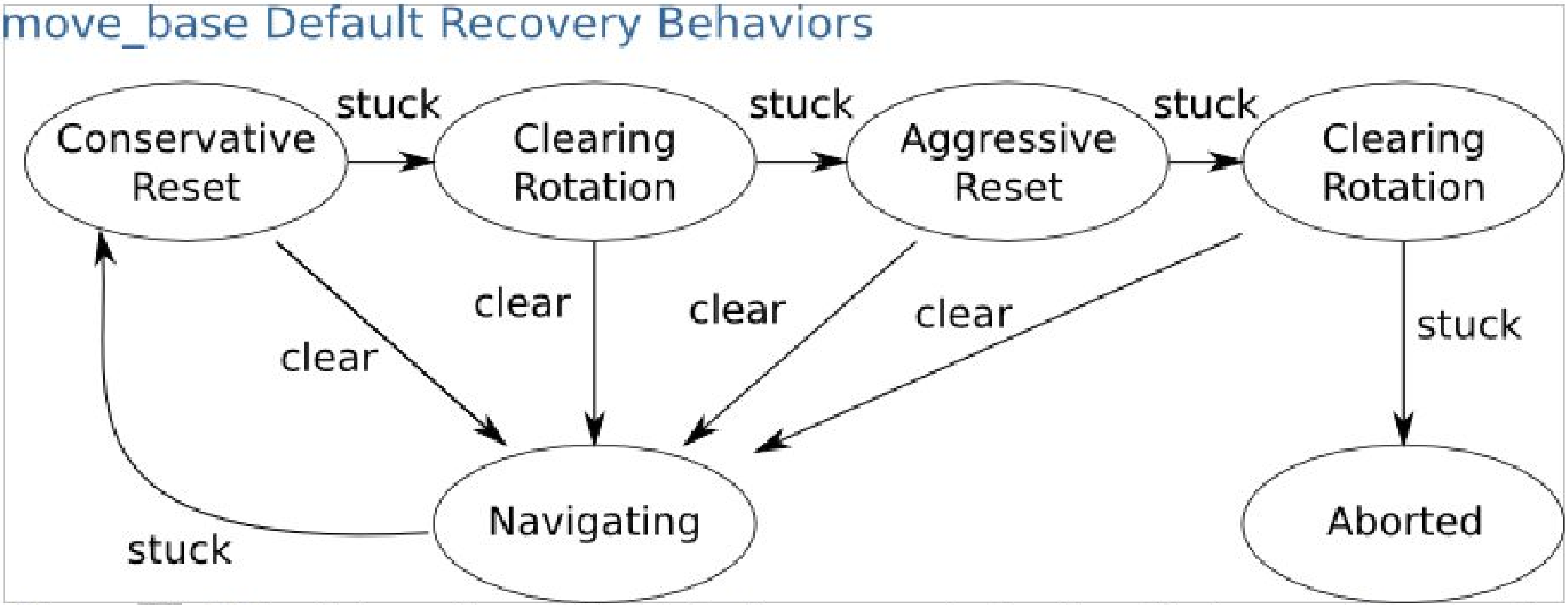


Figure 17: This picture shows the default recovery behaviors that move_base apply, source: http://wiki.ros.org/move_base

ROS – Navigation Stack

- **AMCL** : Adaptive Monte Carlo Localization approach. Probabilistic localization.
- **Map Server** & **Map Saver**.
- **Move_base** : Global and a local planner which are attached to interfaces of nav_core, provides configuration and the interaction with the navigation stack of the robot. (costmap_2d, nav_core, base_local_planner, navfn, clear_costmap_recovery, and rotate_recovery)
- **Actionlib**

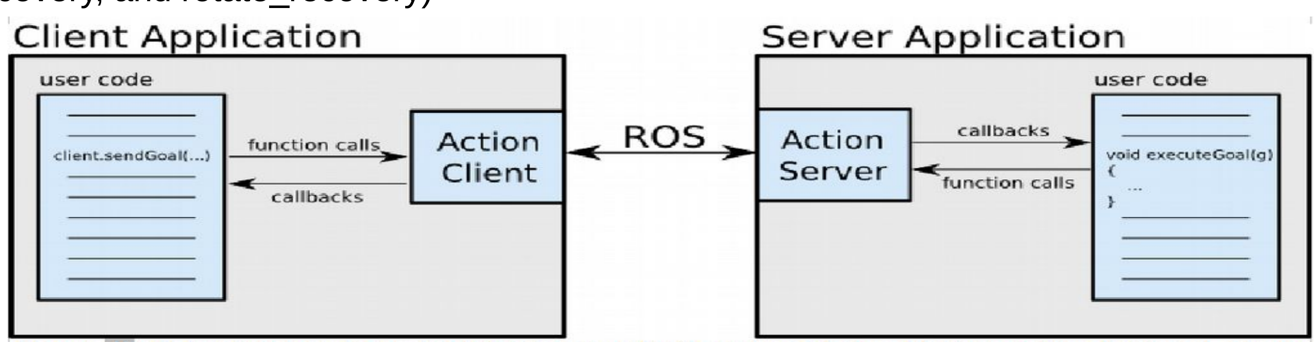


Figure 13: This pictures shows how the communications is performed between the client and server application, source: http://wiki.ros.org/actionlib

Turtlebot

- Created at Willow Garage by Melonee Wise and Tully Foote in November 2010

Hardware

The main hardware includes:

- Kobuki Base
- Asus Xion Pro Live
- Netbook (ROS Compatible)
- Kinect Mounting Hardware
- TurtleBot Structure
- TurtleBot Module Plate with 1 inch Spacing Hole Pattern

Software

The robotic software development environment includes:

- An SDK for the TurtleBot
- A development environment for the desktop
- Libraries for visualization, planning, and perception, control and error handling.
- Demo applications

Open Source

TurtleBot is an open source hardware project as described by the **Open Source Hardware Statement of Principles and Definition v1.0.**

It is released under the **FreeBSD Documentation License**. See the **documentation** page to download the designs.

Turtlebot – Why?

- ROS
- Low cost
- Open Source
- Implements a lot of out of the box functionalities
- Turtlebot SDK
- Large community
- Comes with per-installed sensors for navigation
- Modular

RFID

- Electromagnetic fields to identify and keep track of special tags (radio waves)
- Used in many fields
- Composed of four main parts : RFID **reader/module**, RFID **antennas**, RFID **tags**, and some software is also needed.
- Active and Passive RFID tags

RFID – Why?

- Popular and publicly accepted
- Mature enough, it exists for quite a long time.
- Small size of RFID tags, attachable on any surface.
- Fast bootstrap, doesn't require much time for learning
- Great market share which is continuously growing.
- Possible RFID tags in combination with barcode technology.
- RFID technology is used in many applications world wide.
- Acceptability.
- Affordability.
- High distance coverage.
- High data rates.
- Durability

- Other technologies **Barcord**(small distance, durability), **NFC**(small distance, high price) or even **Bluetooth**(price)

RFID Measurements

Power Levels	Plastic card	Square sticker	**Rectangle sticker**
100 - 10dBm	0 ~ +-1m	0 ~ +-0.7m	0 ~ +-0.5m
150 - 15dBm	0 ~ +-1.5m	0 ~ +-1/1.2m	0 ~ +-1/1.2m
200 - 20dBm	0 ~ +-2-2.3m	0 ~ +-1.2/1.5m	0 ~ +-1.2/1.5m
250 - 25dBm	0 ~ +-2.5m	0 ~ +-2m	0 ~ +-1.5/2m
300 - 30dBm	0 ~ +-4/4.5m	0 ~ +-3.5/4m	0 ~ +-3.5m

- Detect all the RFID tags around Turtlebot in a specified radius.
- Avoid noise from distant RFID tags.
- Reduce the power consumption(powered by a mobile device).

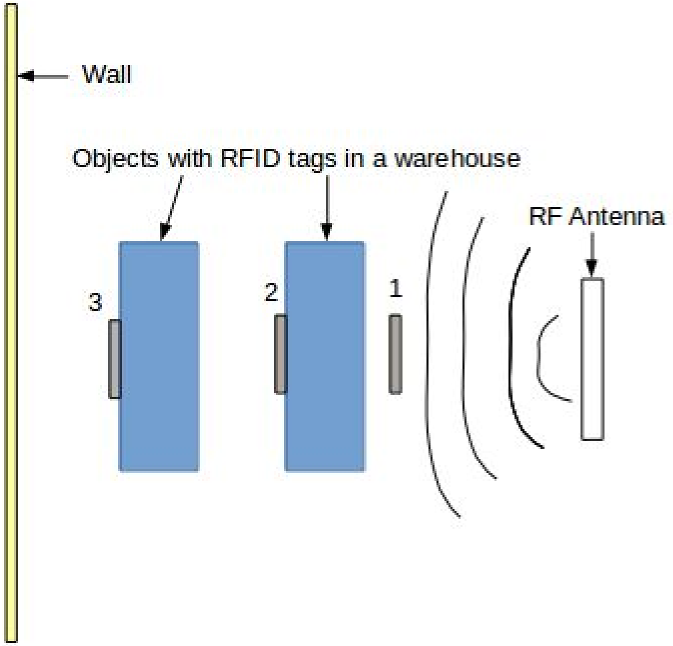
- Power Level - dBs
- Distance from the RFID tag
- Type of RFID tag

Figure 28: Kathrein Wide Range 70 degrees Antennas, source: https://www.kathrein-solutions.com/products/hardware/rfid-antennas/wide-range-70-antennas

- Wide Range UHF RFID Antennas 70 degrees 865 – 868MHz. (UHF – Ultra High Frequency), circular polarized antennas

RFID Measurements

- Materials interference
  - Liquids
  - High percentage of water
  - Carbon
  - Other competitive frequencies
  - Etc.
- Multiple Interposed Objects
- Antennas Position and Angle

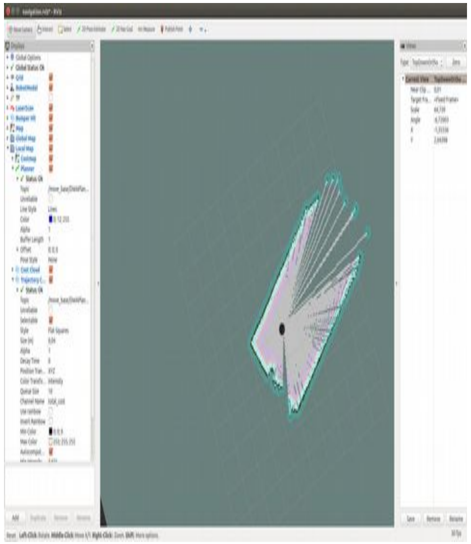


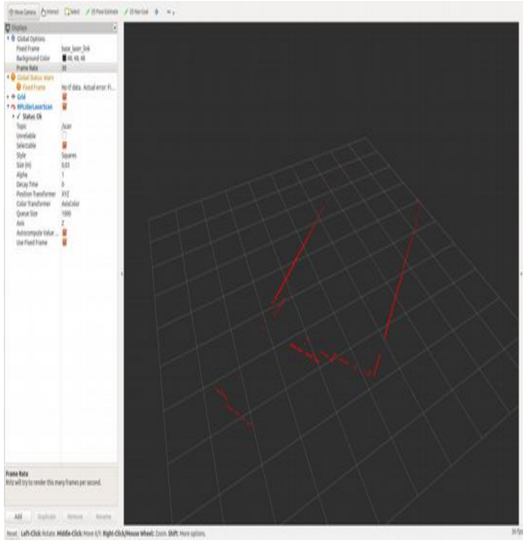
Lidar

- LIDAR is an acronym for Light Detection and Ranging
- LIDAR pulsed light laser + sensor = distances to the scanned surface.
- Representation of scanned environment 2D or 3D
- Creates a point cloud of elevation points.
- In project RPLIDAR A2 is used

Lidar - RPLIDAR

- Low cost
- Scan Rate: 5 – 15Hz (scan frequency) (typical 10Hz)
- Sample rate: 2000-8000 times (measurement frequency) (typical – 4000)
- Ultra thin: 4cm
- Range radius: 0.14m - 12m/18m (6 meters on other sources)
- Low power infrared laser light
- ROS packages “rplicar_ros and rplicar_python” are available for RPLIDAR A2
- Map building, slam(Simultaneous Localization and Mapping) and 3D object/environment model construction





RPLIDAR POSITION

Different positions on Turtlebot

- On top of Turtlebot's top plate.
- Under the Turtlebot's top plate turned 180 degrees.
- Over the RFID antennas installed on top of Turtlebots Top plate.
- On top of Turtlebot's middle plate.

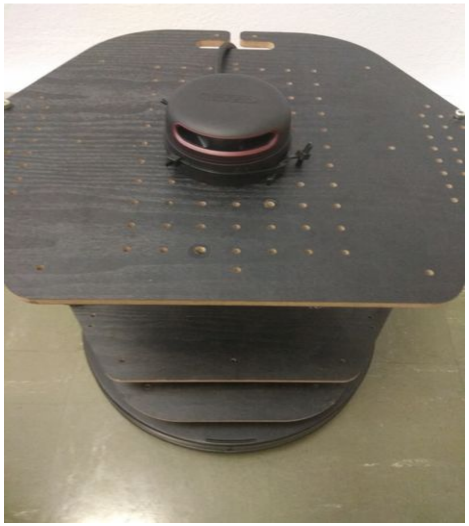
TF transform

- After mounting RPLIDAR on top of Turtlebot it is important to create a ROS TF transform.

Problems

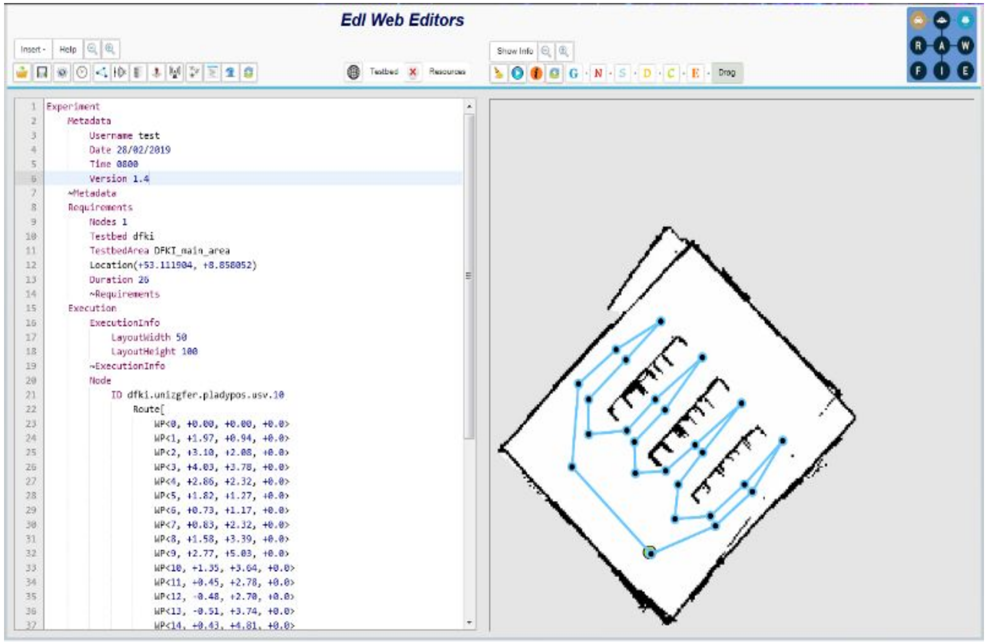
- Small objects (cables, etc)
- RPLIDAR data is received by Turtlebot only when it moves.

RPLIDAR POSITION

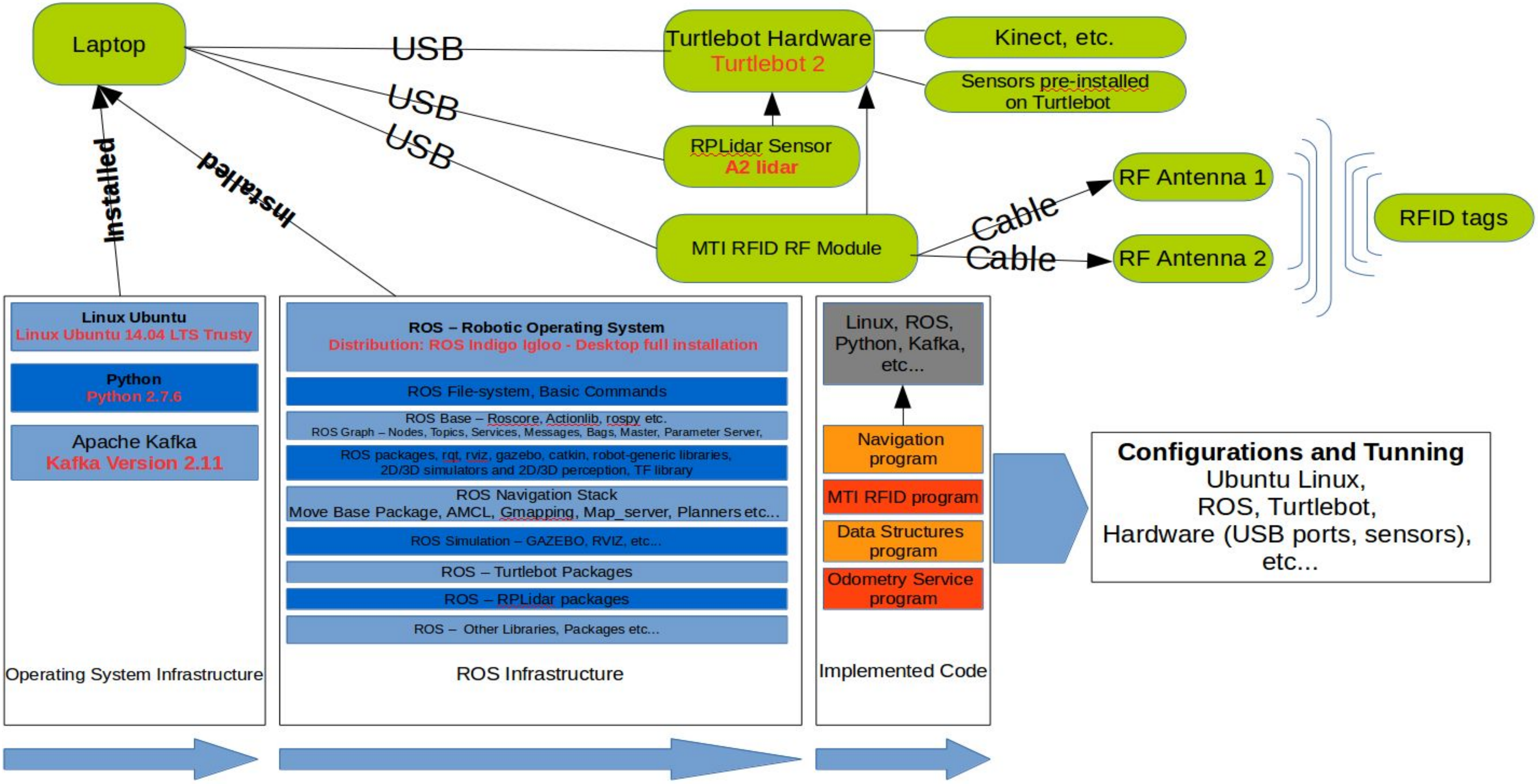


EDL

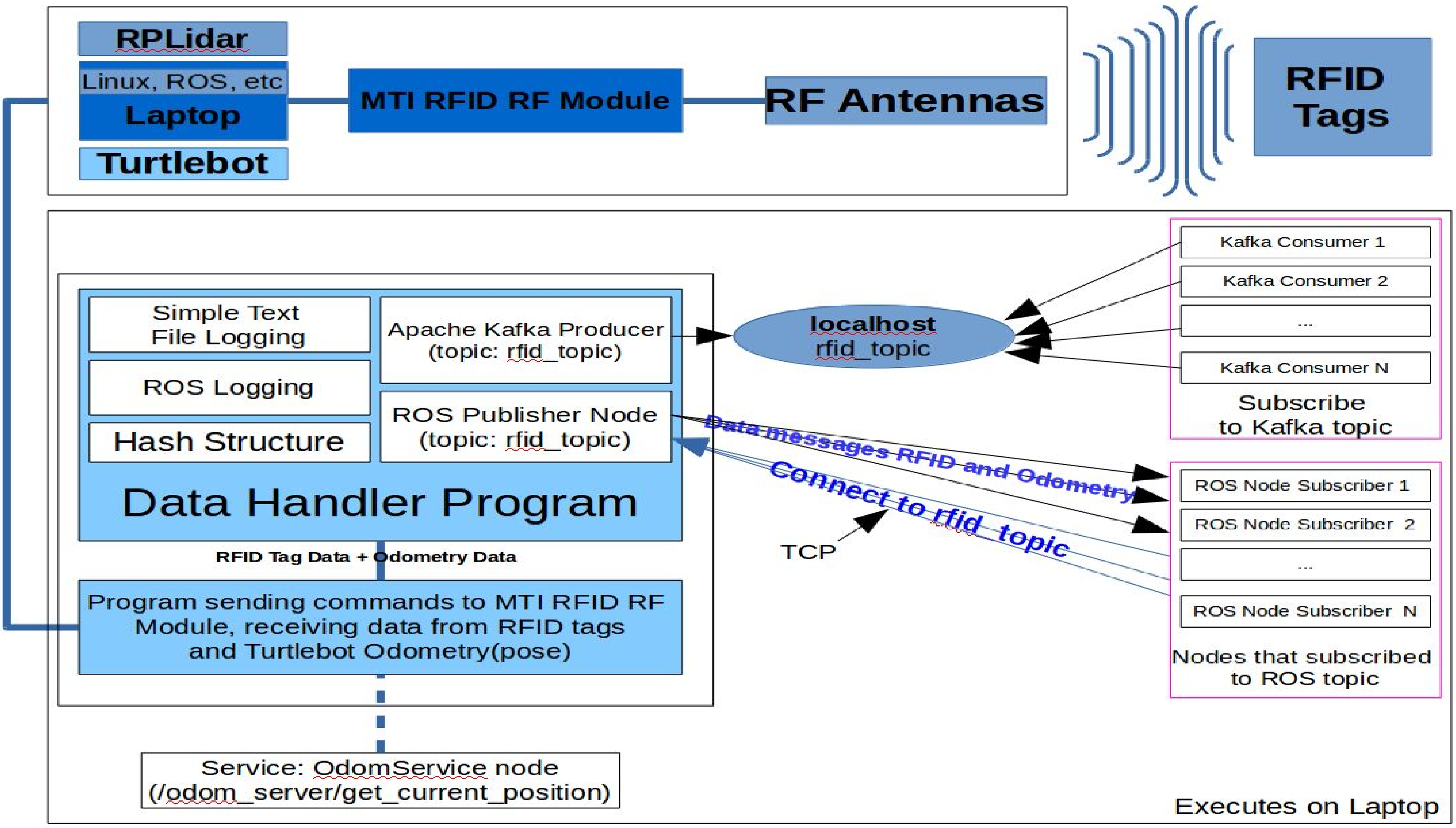
- EDL stands for Experiment Description Language
- EDL is a DSL - Domain Specific Language
- With the help of EDL we create a mission for our Turtlebot
- JSON file of waypoints



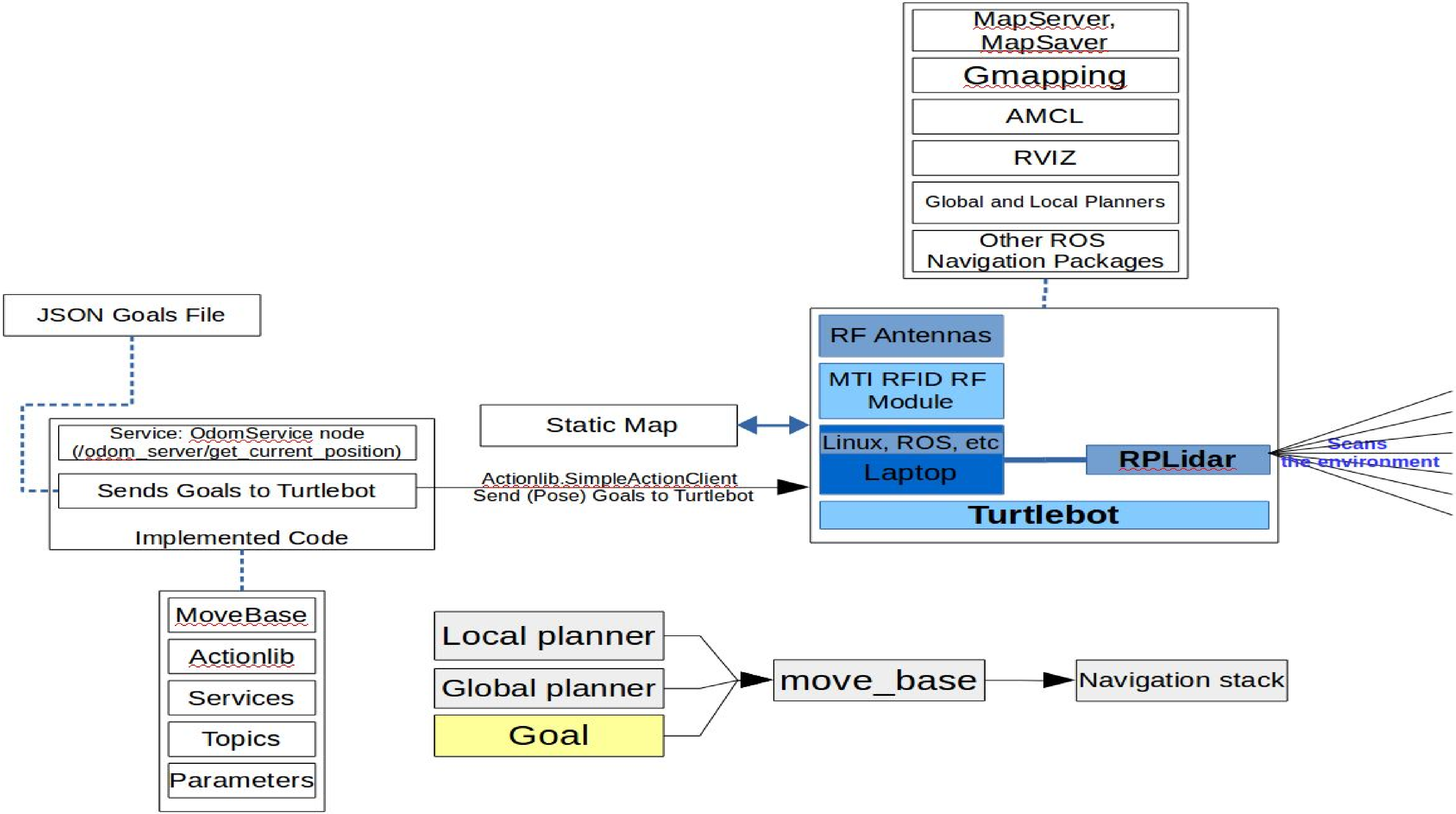
Integration of all Technologies



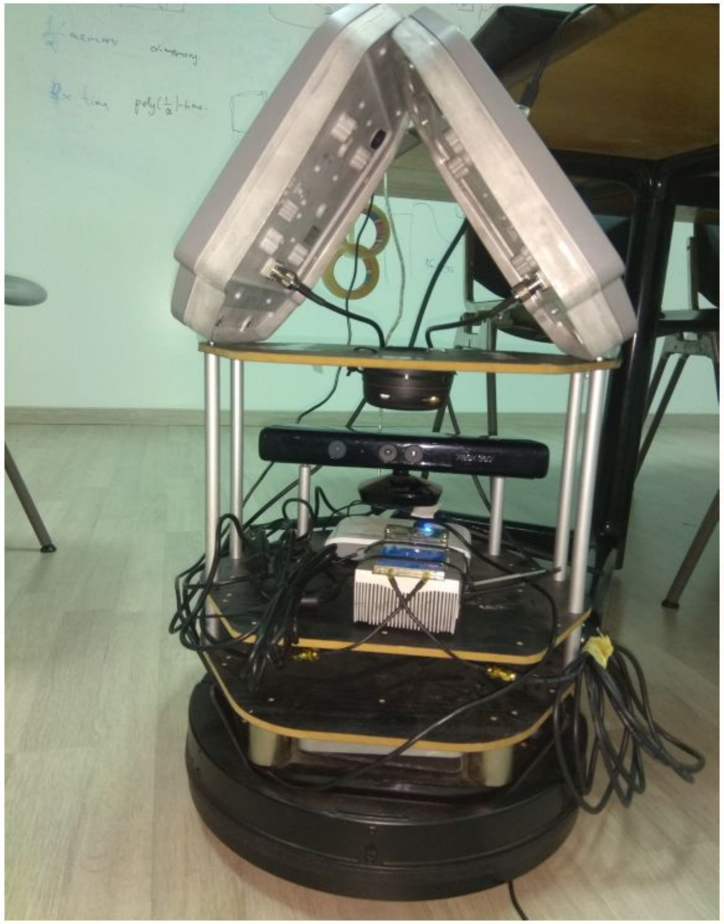
Integration of all Technologies



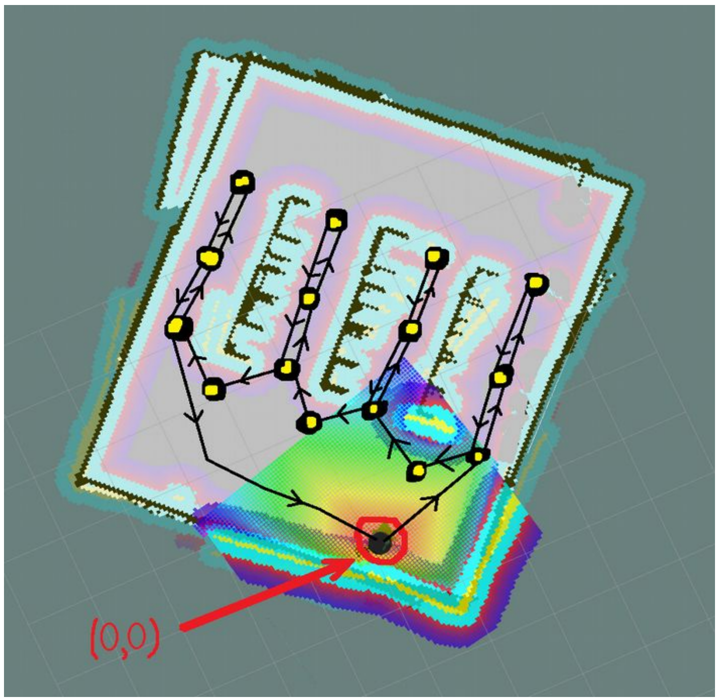
Integration of all Technologies

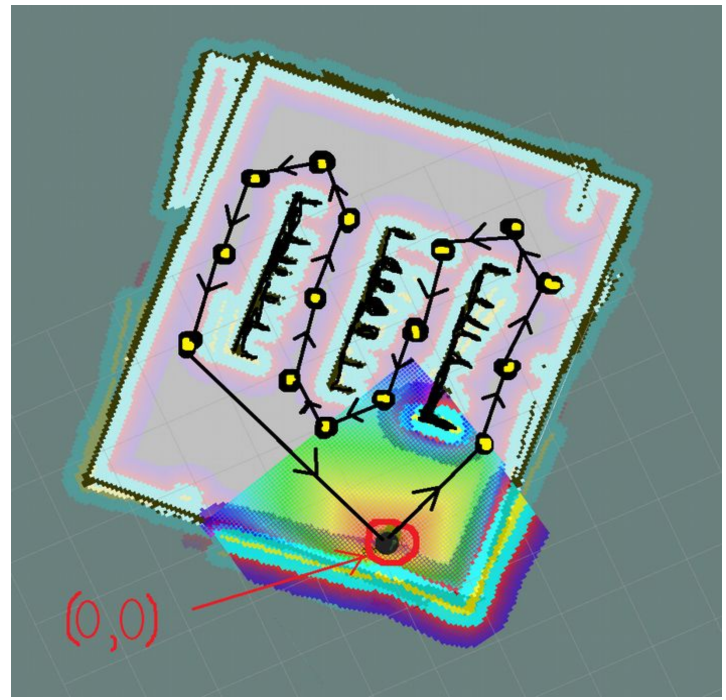


Integration of all Technologies



Implemented Navigation Algorithm- Movement





Different approaches of the Robot movement

Implemented Navigation Algorithm

1 Read all predefined goals from the JSON file which form a path.

2 During the robot navigation perform validation of completed goals within the "radius".

3 For each goal (position and quaternion) on the path.

3.1 Construct goal to pass to the robot.

3.2 Repeat for "goal re-execution max limit" times if goal not completed successfully

3.2.1 Send constructed goal to Action Server to be executed by the robot.

3.2.2 Wait for the robot to finish the execution of the goal for duration specified "goal wait duration"

3.2.3 If the robot is still executing the goal wait again for duration specified in "goal wait duration" for "goal repeat wait max" times.

3.2.4 - If goal finished within the allocated time with goal status "SUCCEEDED" move to the next goal, set goal as completed and move to the next goal.

3.2.5 - Else cancel goal

4 Re-validate if any of passed positions are within the radius of any of goals in the path.

5 Print successfully completed goals.

Implemented Data Handler

Data Collected

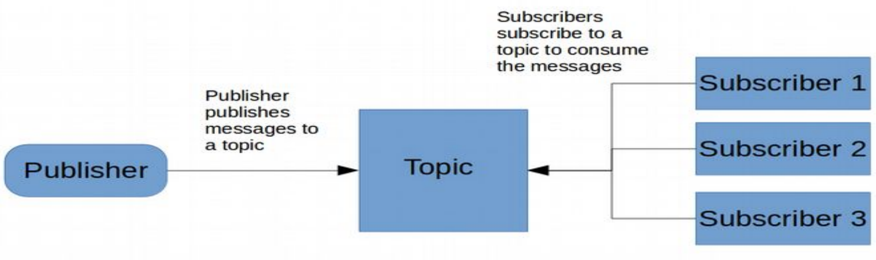
- Turtlebot's navigation data
  - Current Position of the TurtleBot
- RFID tag data
  - RFID value
  - Antenna (0 or 1) located the current RFID
  - RSSI value

Data Handling

- Apache Kafka (topic = rfid_topic)
- ROS topic (topic = rfid_topic)
- ROS logging (terminal or rqt_console)
- TXT file log (simple reliable solution)
- Hash (unique RFID with highest RSSI)

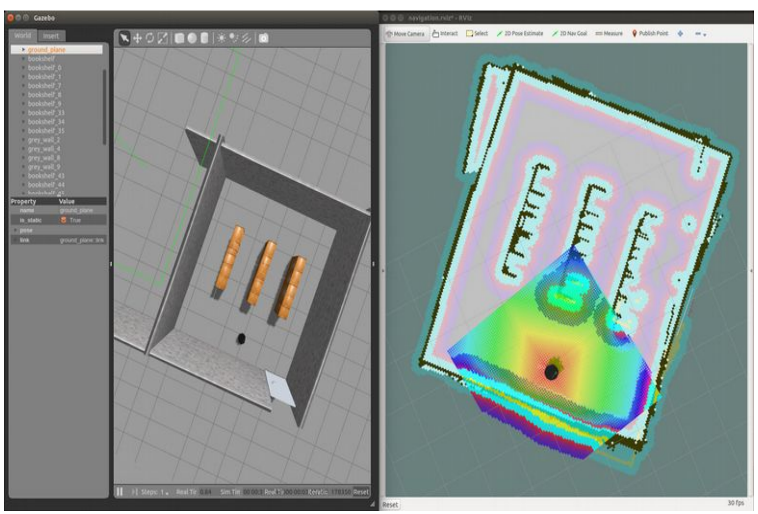
Kafka

- Open-Source
- Free
- Used for industrial applications
- Scalable
- Fault tolerant (distribution & replication)
- **Publish – Subscribe messaging system**
- StackOverflow, Google, and Kafka GitHub Apache Kafka related question has grown up exponentially



Simulated Experiments

- ROS
- Gazebo
- AMCL
- Rviz
- Implemented algorithm



- Build a simulated world with Gazebo.
- Create a map of the built world (.yaml, .png).
- Set the path of the Robot - waypoint (json)
- Run Gazebo, RVIZ, AMCL.
- Run Navigation algorithm.

Simulated Experiments

1. No obstacle appearance

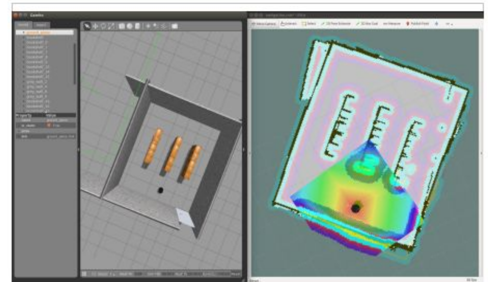


Figure 51: In this picture we can observe a simulation where no obstacles appear lying in the corridors of the warehouse.

2. Medium size obstacle

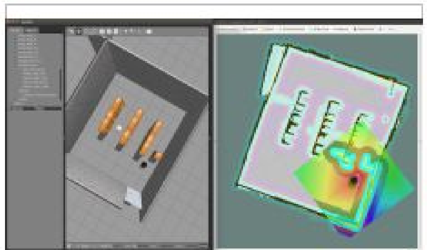


Figure 54: An obstacle appears on the Turtlebots path

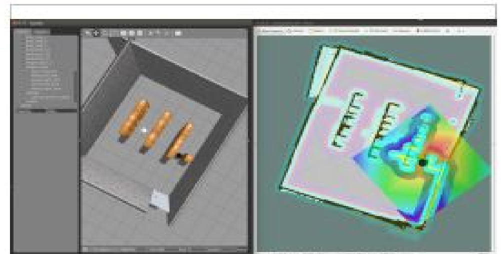


Figure 52: Turtlebot avoid the obstacle by moving around the left side of the object

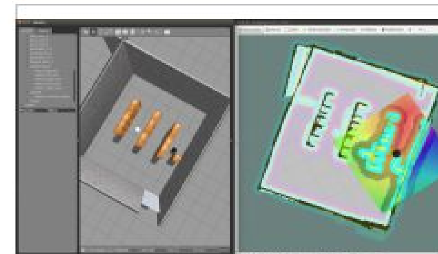


Figure 53: Turtlebot continues moving towards the specified goal

3. Unexpected Obstacle
4. Small Object
5. Large Obstacle

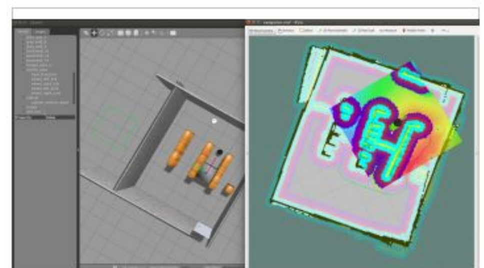
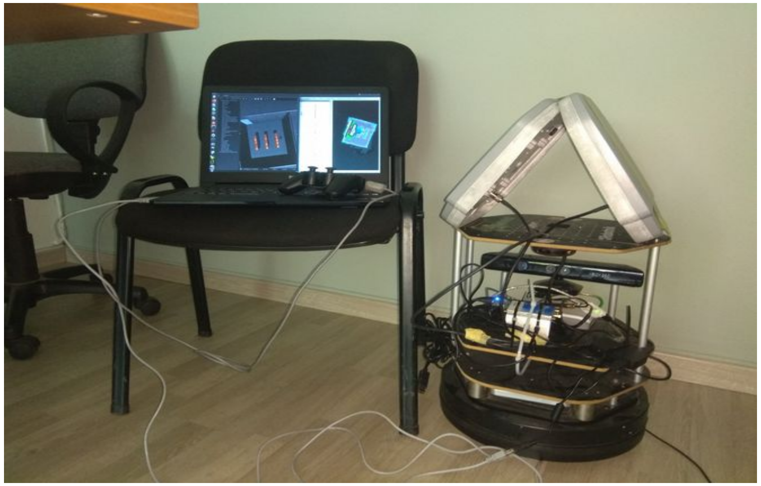


Figure 55: Large object the whole corridor

Physical World Experiments

- First Steps similar to simulated world experiment
- Start Inventory algorithm for RFID tags
- Check the collected RFID tags and the total number of tags



Physical World Experiments

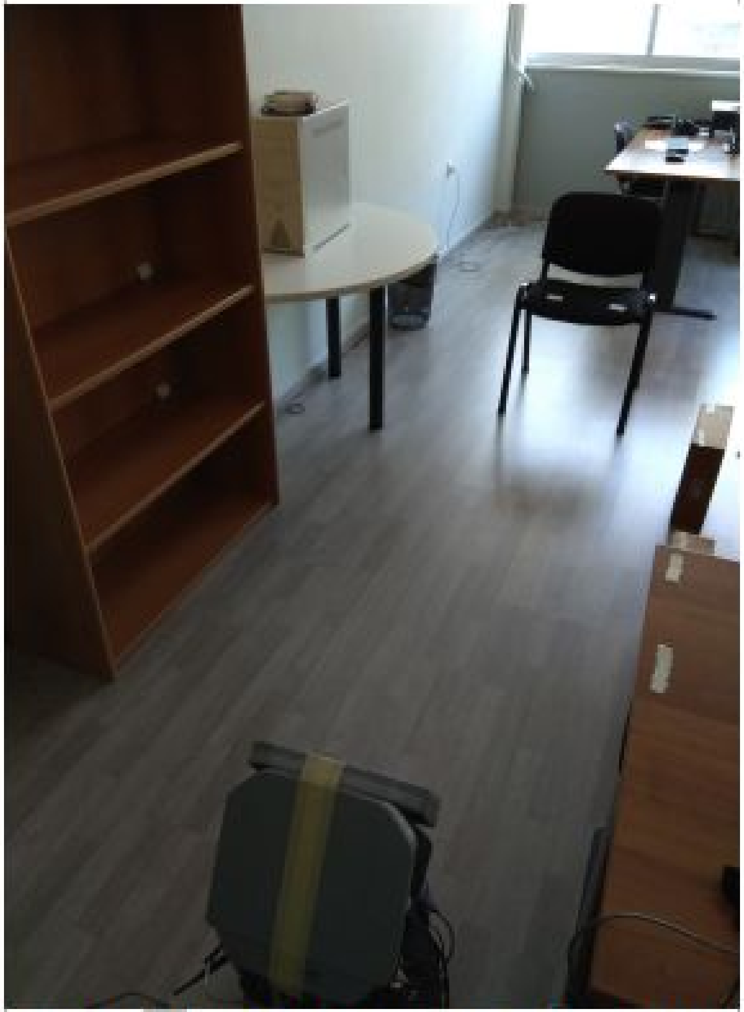


Figure 57: Room where the experiment was performed

Figure 58: Right side of the Room were the boxes where positioned

Figure 59: RFIDs position on different hights

Physical World Experiments

- 15 RFID tags placed around the Turtlebot
- All 15 were detected

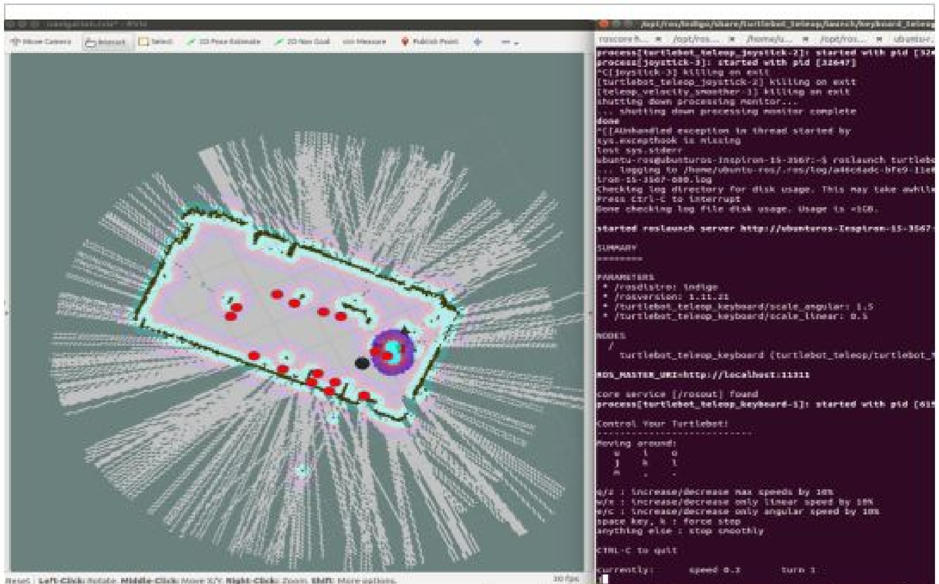


Figure 60: Created map of the room where the experiment was performed

- 7 goals positions on the map (waypoints)

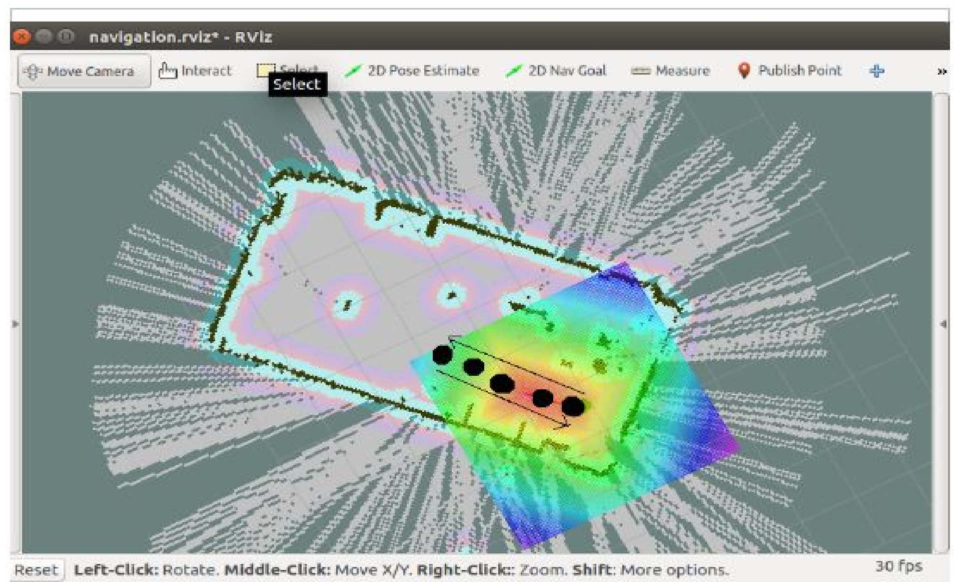


Figure 61: Turtlebot's path is marked with black dots. This picture shows the performed movement.

Future Work

- Experiments with other ROS like systems.
- Experiments in different environments.
- Implementation of different Path Planners and Recovery technicals.
- Adding fire sensors.
- Intruders detection.
- RFID tags, GPS, and Wifi for more position accuracy.
- Interface Implementation.
- Create an RFID ROS package.
- Automated algorithm for robot direction and RFID tag side detection, independent of the direction robots moves.
- Automated antenna activation and deactivation in areas where robot already passed.

References

[1] Kostas Kolomvatsos, Michael Tsiroukis, Stathes Hadjiefthymiades. 2017. “An Experiment Description Language for Supporting Mobile Iot Applications”. National And Kapodistrian University of Athens. Department of Informatics and Telecomunications

[2] Michail Chatzidakis, Michail Loukeris, Kostis Gerakos, Stathes Hadjiefthymiades. 2016. “E-Pres: Monitoring and Evaluation of Natural Hazard Preparedness At School Community”. Pervasive Computing Research Group. National And Kapodistrian University of Athens. Department of Informatics and Telecomunications.

[3] Kshitija Deshmukh, Ashitha Ann Santhosh, Yogesh Mane, Saurabh Verma, Sdhana Pai. Nov 2015. “Robotic navigation and inventory management in warehouses”. International Journal of Soft Computing and Artificial Intelligence. ISSN. 3(2): 75-79

[4] Kaiyu Zheng. Sep 2016, “ROS Navigation Tuning Guide”

[5] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng, 2009, ROSQ: an open-source Robot Operating System”, ICRA Workshop on Open Source Software, Kobe, Japan, 2009

[6] “Effective Robotics Programming with ROS” Third Edition by Anil Mahtani, Luis Sanchez, Enrique Fernandez, Aaron Martinez, Publisher Packt, 2016

[7] http://www.ros.org/

[8] http://wiki.ros.org/

[9] http://learn.turtlebot.com/

[10] https://en.wikipedia.org/

[11] https://www.turtlebot.com

[12] https://www.mtigroup.com/

[13] https://www.slamtec.com/en/Lidar/A2

Special Thanks to

Supervisor : Stathes Hadjiefthymiades

Michael Loukeris
Kakia Panagidi
Michail Tsiroukis
Kostas Kolomvatsos

Thank you!!!

Questions???