TECHNISCHE UNIVERSITÄT WIEN

ACIN AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Rockly: A graphical tool for programming ROS based robots

MASTER THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Ao.Univ.-Prof. Dipl.-Ing. Dr. techn. Markus Vincze

submitted at the

TU Wien

Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by

Alexander Semeliker, BSc
Georg Humbhandl Gasse 1
2362 Biedermannsdorf
Österreich

Wien, im September 2018

Preamble

<Hier bitte Vorwort schreiben.>
Wien, im September 2018

Abstract

<Please write an abstract here.>

Summary

Contents

List of Figures

List of Tables

1 Introduction

2 Related Work

2.1 Choregraphe

2.2 evablockly_ros - Inovasyon Muhendislik

2.3 robot_blockly - erlerobotics

2.4 Open Roberta Lab

2.5 RobotC

2.6 Ardublock

3 Architecture

This chapter describes the architecture and implementation of Rockly (portmanteau of ROS and Blockly). First the purpose and need of it are explained, followed by a architectural overview of HOBBIT, the used robot. Based on this constraints the options implemtenting the tool are presented as well as a explanation of the decision. Then a short description of the robot operating system ROS and the fundamental JavaScript frameworks, Node.js and Express, are given and finally the necessary details of the implementation are documented - for both, the frontend and the backend.

3.1 Requirements

In the field of software engineering constraints are the basic design parameters. Therefore it is necessary to provide them as detailed as possible. In the given case the basic constraints are given by the purpose of the tool and the architecture of the robot.

3.1.1 HOBBIT - The Mutal Care Robot

The HOBBIT PT2 (prototype 2) platform was developed within the EU project of the same name. The robot was developed to enable independent living for older adults in their own homes instead of a care facility. The main focus is on fall prevention and detection. PT2 is based on a mobile platform provided by Metralabs. It has an arm to enable picking up objects and learning objects. The head, developed by Blue Danube Robotics, combines the sensor set-up for detecting objects, gestures, and obstacles during navigation. Moreover, the head serves as emotional display and attention center for the user. Human-robot interaction with Hobbit can be done via three input modalities: Speech, gesture, and a touchscreen. [1]

In terms of technology HOBBIT (see Figure 3.1) is based on the robot operating system ROS (Section 3.3.1), which allows easy communication between all components. The system is set up to be used on Ubuntu 16.04 together with the ROS distribution *Kinetic*. All ROS nodes are implemented in either

Figure 3.1: HOBBIT - The Mutal Care Robot

Name	Type	Message type	Description
/cmd_vel	Topic	geometry_msgs/Twist	move HOBBIT
/head/move	Topic	std_msgs/String	move HOBBIT's head
/head/emo	Topic	std_msgs/String	control HOBBIT's eyes
/MMUI	Service	hobbit_msgs/Request	control UI interface
hobbit_arm	Action	hobbit_msgs/ArmServer	control HOBBIT's arm
move_base_simple	Action	geometry_msgs/PoseStamped	navigate HOBBIT

Table 3.1: Common commands used by HOBBIT

Python or C++. In order to provide a fast and simple way to implement new behaviours several commands should be pre-implemented. These commands are performed either by publishing messages to topics or services, or executing callbacks defined in the corresponding action's client. The common commands and their description are listed in Table 3.1.

3.1.2 Purpose of the tool

Since the above mentioned EU project HOBBIT became very popular and demos of it's behaviours has been presenting at large number of fairs. Unfortunally only the following show cases are currently implemented on the robot:

follow me HOBBIT follows the user

learn object	HOBBIT learns object
call SOS	HOBBIT starts an emergency call
pick up	HOBBIT picks up an object

All of the demos can be started via the UI running on HOBBIT's tablet. Re-writing new demos would need a detailed knowledge of the robot's setup. In order to implement new behaviours and demos more easily it is necessary to provide an programming interface, which provides both a powerful, generic base to cover a wide range of HOBBIT's features and intuitive handling.

Furthermore the Automation and Control Institute of the TU Wien is part of the Educational Robotics for STEM (ER4STEM) project, which aims to turn curious young children into young adults passionate about science and technology with hands-on workshops on robotics. The ER4STEM framework will coherently offer students aged 7 to 18 as well as their educators different perspectives and approaches to find their interests and strengths in robotics to pursue STEM careers through robotics and semi-autonomous smart devices. [0] Providing an intuitive programming tool would allow the integration of HOBBIT into the project, which would be an extra input evaluation parameter.

At last the framework should be implement to be re-used for other ROS based robots. This means, that it should not only provide an interface to the mentioned commands for HOBBIT, but an open, adpatable framework. It should be able to allow a flexible configuration and assembly of the provided functions.

3.2 Options

There are several approaches to fulfill the mentioned requirements. In the following subsections three different options are presented by a simple example: the implementation of picking up an object from the floor an putting it on the table. This should give a rough overview in terms of complexity of the usability as well as the implementation of the corresponding approach.

3.2.1 Python/C++API

The most obvious way to fulfill the requirements is to provide an application programming interface (API) for the desired programming languages (Python, C++). An API is a set of commands, functions, protocols, and objects that

programmers can use to create software or interact with an external system. It provides developers with standard commands for performing common operations so they do not have to write the code from scratch. In the present case such a API could consists of the following components:

Initialization setting up communication and intial states - e.g. creating ROS nodes, starting the arm referencing or undocking from charger

Topic management managing the messages published to ROS topics and creating subscriber nodes if applicable

Service management managing the ROS services of HOBBIT - e.g. the tablet user interface

Action management creating ROS action clients for e.g. navigation or arm movement

It should be noted, that the components doesn't re-implement ROS functionality, but extend it and prvovide a simpler use of it. Depending of how generic the API is implemented it is possible that the user can control the robot without any detailed knowledge of the technical setup of it. Nevertheless this approach would presuppose the user to have knowledge of the programming language the API is desigend for.

3.2.2 SMACH

3.2.3 Blockly

3.2.4 Decision

3.3 Frameworks

3.3.1 ROS

3.3.2 Node.js

3.3.3 Express

3.4 Server implementation

3.5 Frontend implementation

4 Evaluation

4.1 Methods

4.2 Results

4.3 Discussion

5 Conclusion

Bibliography

[1] Automation and T. W. Control Institute, *HOBBIT The Mutual Care Robot*. [Online]. Available: `https://www.acin.tuwien.ac.at/vision-for-robotics/roboter/hobbit/` (visited on 08/17/2018).

[0] ——, *ER4STEM Educational Robotics for STEM*. [Online]. Available: `https://www.acin.tuwien.ac.at/project/er4stem/` (visited on 08/17/2018).