INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 5, Issue 2)
Available online at: www.ijariit.com

Implementation of ROS-I on industrial robot simulation environment

Prasanth T.
prasanth.ia17@bitsathy.ac.in
Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu

ABSTRACT

This paper presents the procedure for the implementation of ROS-I architecture into the Robotic simulation offline programming environment. The Implementation is done by developing the packages which are necessary for implementing into ROS-I that are related to modelling and Configuring of the Industrial Robot. The Communication between ROS-I and Roboguide will be established, and motion planning can be performed through Rviz movie! Interfaces.

Keywords— *ROS-I, Rviz, Moveit*

1. INTRODUCTION

Advances in robotics require robots of their medium to integrate into various applications. In the field of cooperation between robots, the questions of simultaneous or sequential action that arise problems and challenges. There are two crucial situations, the cooperation with and without the monitoring of the contact between the actors in this non-contact solution between the active agents.

The precise knowledge of the geometric position of the active agents, such as robots or others, allows the construction of cooperation applications. However, when this knowledge does not exist, it is necessary to determine the relative positioning of the robots in order to cooperate.

1.1 Background

It turns out that the implementation of industrial robots is an expanding market. According to the International Federation of Robotics (IFR) [1] in 2014, all sales of industrial robots reached 229,000 units worldwide. It is estimated that the sale of industrial robots (figure 1) reaches 400,000 units in 2018, 48,000 in America, 275,000 in Asia / Australia, 66,000 in Europe and the remaining not specified by cities. The increasing bet on this equipment is due, primarily, to technological advances that allow the interconnection of manufacturing equipment (industry 4.0), the simplification in the use of robots and the collaboration between humans and robots. Controlling and programming many of these robots is not always a simple process, is often done in low-level programming. Technological knowledge of the interaction between the different components of the robot and of a working cell is also required. It is also evident that programming is specialized equipment, that is, changing the model of the robot used program undergoes profound changes, implying new learning times.

In order to simplify programming, an Application Programming Interface (API). API allows you to use a language that is closer to the logic that translates the behavior of the system, abstracting the user from details. This work theme arose from the need to improve the interface and communication between users and industrial manipulators using Roboguide Software. Before this work, control and communication with robots were ensured by robCOMM [2]. In this dissertation work, the author developed an application server that provided a set of services to remote devices, such as a computer. The robCOMM server was housed in an industrial robot and provided services of movement, manipulation of data, calculations of Kinematic and program execution.

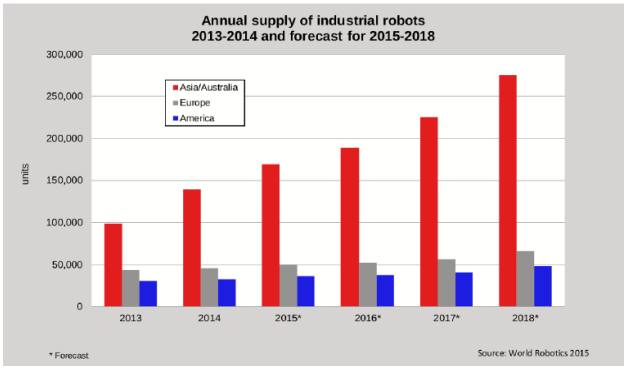

Fig. 1: History and projection of sales of industrial robots

1.2 Objectives

With the present dissertation I intend to reach the main objectives:

- Develop or adapt an API that allows integrating the LRmate Fanuc robot into more complex applications of manipulation and cooperation in the ROS environment.
- Develop ROS-I Moveit Configuration file for Motion Planning of the Robot.
- Implement communication between ROS and Roboguide.

1.3 Offline programming

Offline programming offers a 3D environment that allows you to simulate, validate, create, programming and modeling a work cell

by creating tools and equipment peripherals of robots. One of the main advantages to be highlighted is its use in a complex of the development time. In cells with two or more robots, it became quite expensive and difficult to program the console. Strategies for simulation also become an assertive choice to prevent the occurrence of errors and study improvements. In various processes, such as in the automotive industry, the use of simulators prevents the destruction of cars, which would generate significant costs. In addition to the manufacturers of manipulators have software for programming (Figure 2), there are Computer Aided Design (CAD) applications that have started to incorporate simulators and a programming environment into their software so that to simulate complex cells (Figure 3). This incorporation was based on the Computer Aided Manufacturing (CAM), since the post-processors that generate the programs are general.

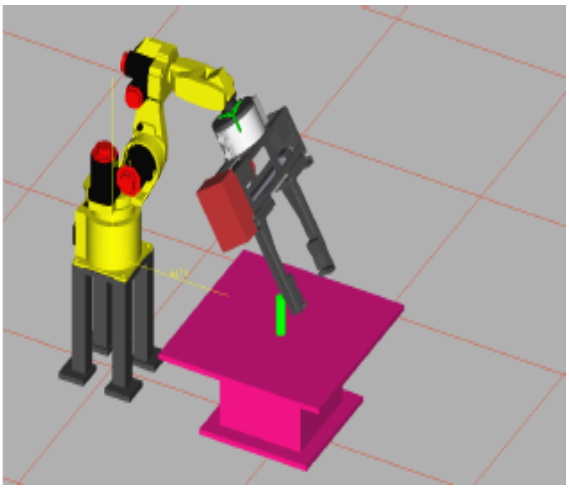

Fig. 2: Robo guide

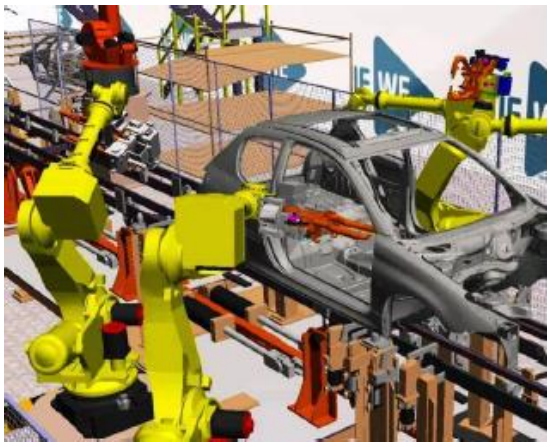

Fig. 3: Robotics spot imulation engineer

Figure 2, figure 3 Fanuc (RoboGuide) manufacturer's offline programming software and Dassault Systems (Robotics Spot Simulation Engineer)

2. EXPERIMENTAL INFRASTRUCTURE AND TOOLS

2.1 Fanuc LR Mate 200iD

In the present work, a Fanuc manipulator is used. The manipulator concerned is compact and has broad applicability in an industrial environment, execution of various operations such as pick and place. The Fanuc LR Mate 200iD (figure 4) is a manipulator composed of 6 axes, with a maximum range of 717 mm, capacity to withstand a maximum load of 7 kg and a repeatability of 0.02 mm, as indicated in figure 6. The workspace, the region within which you can position the end-effector, is conditioned not only by their physical dimensions but also by their limits of joints.

Fig. 4: Fanuc LR Mate 200iD.

2.2 Controller

The controller is the fundamental element that integrates and commands the manipulator hardware. In the present work, the controller is the R-30iB Mate (figure 5), guaranteeing a high level of performance and processing speed. The main differences that are verified at the level of the physical dimension, being smaller and quieter than the previous versions.

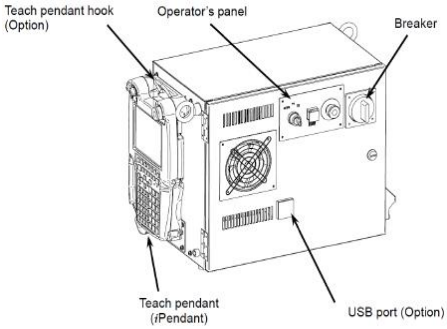

Fig. 5: R-30iB Mate controller [3]

Model		LR Mate 200iD
Controlled axes		6 axes (J1, J2, J3, J4, J5, J6)
Reach		717mm
Installation (Note 1)		Floor, Upside-down, Angle mount
Motion range (Maximum speed)	J1 axis	340°/ 360° (option) (450°/s) 5.93 rad/6.28 rad (option) (7.85 rad/s)
	J2 axis	245° (380°/s) 4.28 rad (6.63rad/s)
	J3 axis	420° (520°/s) 7.33 rad (9.08rad/s)
	J4 axis	380° (550°/s) 6.63 rad (9.60 rad/s)
	J5 axis	250° (545°/s) 4.36 rad (9.51 rad/s)
	J6 axis	720° (1000°/s) 12.57 rad (17.45 rad/s)
Max. load capacity at wrist		7kg
Allowable load moment at wrist	J4 axis	16.6 N·m
	J5 axis	16.6 N·m
	J6 axis	9.4 N·m
Allowable load inertia at wrist	J4 axis	0.47 kg·m²
	J5 axis	0.47 kg·m²
	J6 axis	0.15 kg·m²
Repeatability		± 0.02 mm
Mass (Note 2)		25 kg
Installation environment		Ambient temperature : 0~45℃ Ambient humidity : Normally 75%RH or less (No dew nor frost allowed), Short term 95%RH or less (within one month) Vibration : 0.5G or less

Fig. 6: Fanuc LR Mate 200iD Specifications. [4]

To establish communication with the controller, several connections are available for communication with the system. In the most type of work were used the mechanical unit, USB connection, a programming console, interface Inputs / Outputs (I/O), Ethernet and AC power supply. It should be noted that this figure includes two types of dashed lines indicating the mechanical and the remaining lines representing the electrical connections. In the mechanical unit the connection with the end-effector that is responsible for the interaction with the environment. On the other hand, pneumatic pressure makes it possible to a given task, such as the picking of environmental objects. The USB connection makes it possible to update operations, system backups or even file transfer. The console is the primary means of interaction with the controller, programming operations, program execution, and editing, errors,

I/O status, among others. The I/O interface is an excellent control option since it is possible to access, control and communicate with peripheral units, such as sensors and actuators. In most field networks, their connectivity is assured by serial ports and Ethernet. Given the advantages, the Ethernet connection was adopted. Finally, the AC power supply which is responsible for feeding the entire system [3]. The controller also has HyperText Transfer Protocol (HTTP), and File Transfer Protocol (FTP) deployed. The platform focused on HTTP was created in [2] allowing the monitoring of some aspects of the manipulator. On the other hand, FTP was used in this work for the transfer of programs created by a post-processor, verifying a fast and versatile method of transferring information.

2.3 Communication

Communication between machines allows a better understanding of each machine, improving not only its efficiency but also the optimization of various processes. The connectivity with industrial controllers can be divided into three categories [2]:

- Connectivity with peripheral equipment, using field networks and I/O digital and analog signals;
- Connectivity with communication hardware for data transfers or programs of the console per serial port, USB, PCMCIA slots, etc.
- Ethernet communication, in which HTTP and FTP servers can be deployed.

It can also serve as a high-speed communication medium with a computer external.

2.3.1 Ethernet: The Ethernet system, proposed by the Xerox company and approved in 1983 by The Electrical and Electronics Engineers (IEEE), defines that each sends your data/signals, you should listen to the transmission medium. When there is no activity, you must submit your data. Thus, this protocol allows access to the transmission medium. Among the several physical layers, the adopted in this work was the medium of transmission10Base-T. There are three types of dialogs, simplex, half and full duplex. When only one of the equipment is limited to sending information, and the other is limited to be received, communication is considered to be simplex. Otherwise, when the two equipment may receive and send data, but not simultaneously, it is considered the half duplex. Finally, the full-duplex dialog is similar to half duplex, differentiating in the possibility of dialoguing simultaneously. The full duplex dialog was adopted in this work.

2.3.2 TCP / IP: The Internet Protocol (IP) protocol, created by the United States Department of Defense allows you to exchange blocks of data between computers. Each piece of equipment has a single address, which allows you to route the data blocks between devices until destination equipment. The protocol also allows the fragmentation of the data to be transmitted so that they can be transmitted through local networks such as Ethernet. The Internet message consists of the header and the data. While the header contains the source and destination addresses of the Internet messages the data are the messages of the transport layer. The IP message is sent to the network, then encapsulated in the message of the logical layer, or directly (MAC-frame) of a local network, usually in the Ethernet message. The Transmission Control Protocol (TCP), as the United States Department of Defense, developed the IP protocol. This protocol provides a reliable service of data transfers, even though its services are provided by the protocol, ensuring that there are no packets lost and that are processed in order correctly.

3. ROS-INDUSTRIAL API INTEGRATION

Since in the related projects there was an API available in the open source code in a ROS environment and with broad applicability, it was integration in the industrial robot present in the LAR. Like ROS-Industrial's libraries were in an advanced stage of development, learning of the tools associated with ROS and all its tools need to be gradual.

ROS-Industrial has associated a set of meta-packages from robot manufacturers which contains not only the organized structure of folders and files of a meta package but also the generic models of robot geometry. Among the packages of manufacturers of robots, the one referring to the ROS Fanuc was selected. In addition to this work, use an environment that serves as the basis of all software (the ROS), a second environment that, among many features, plan trajectories (the MoveIt!). The structure of this chapter begins with the explanation of the ROS followed by the ROS-Industrial and ROS Fanuc. Finally, we explain the libraries that make Move It.

3.1 ROS

ROS is a development platform that emerged in 2007, strongly driven by Willow Garage. Integrates application-driven libraries and tools in robots, simplifying complex and robust tasks. Software from various companies increasingly being developed in the ROS environment, particularly in the industrial robotics [5].

ROS is an excellent choice for a robotic design, such as the ability to integrate stand-alone navigation to plan motions, for example, MoveIt! It also offers tools debugging, visualization and simulation like rqt_gui, RViz, and Gazebo, respectively. Supports interfaces for robotic sensors and actuators and the ROS structure can establish communication between several nodes with different programming languages, such for example C, C ++ and Python [6].

The ROS is divided into three groups: the file system level, the Computation Graph level and the Community level (figure 7). The first level, the file system level, defines the internal operation, the storage, file structure and the minimum requirements for its operation. It is at this level that details of messages, services, and folders are specified.

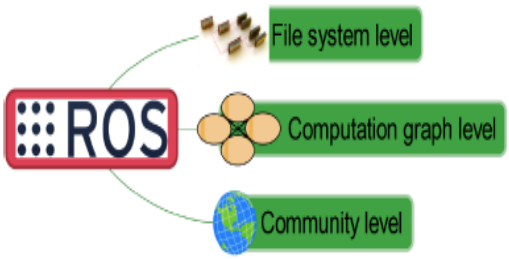

Fig. 7: General ROS structure

Next, the Computation Graph level describes the communication between processes and the system. It is at this level that the definitions and explanations of the relations between all constituents of ROS. Finally, the Community level represents the online that contains the algorithms and code that are intended for sharing. Will not be described the content of this topic, since it refers to online sites that integrate packages and metapackages [7].

3.2 File system level

The packages are the core of ROS and every meta-package. The met packages are used to generate a group of packages, created to process and execute a task. In order to maintain an

organization, both the folders and the content of the information shared with the ROS community, and a structure was defined typical for each package, as shown in figure 8: Structure and organization of the folders of each package.

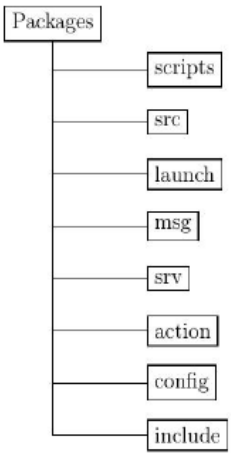

Fig. 8: Structure and organization of the folders

The standard folders that make up a package in ROS have the following structure [6]

- All code developed in C ++ is in the src folder, while the code developed Python is present in the scripts folder.
- When it is intended to start more than one program and library, at the same time, launch files are used. These files are in the launch folder.
- The msg folder contains the definition of all messages developed by the programmer.
- In turn, the srv folder has all the messages used in the template client/server in ROS.
- The action folder includes the files to create an action. In the present study, this feature was used, since there was never a need to create this paste.
- The configuration files for each package are present in the config folder.
- The header files used in each node are stored in the include folder.

3.3 Computation graph level

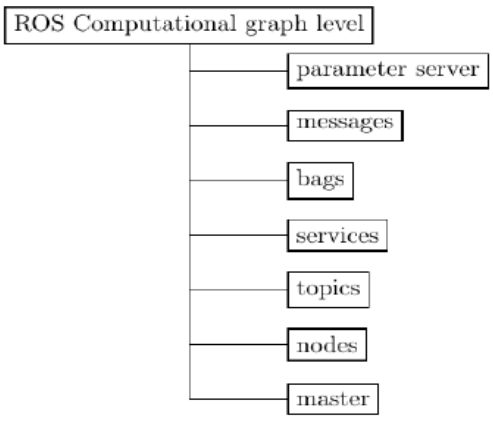

Fig. 9: ROS Graph level structure

The ROS creates a network that is responsible for maintaining all the interconnected architecture. Provides each ROS node with access to the network, being able to interact, access and transmit information to other nodes. The concepts of ROS computational graph level are present in figure 9.

In the ROS nodes, where the entire computational process is performed, in roscpp (C ++) or rospy (Python). Each robot can simultaneously process different nodes and tasks, establishing communications among themselves, using topics, services, and parameters. With the methodologies available in the ROS, it becomes possible to implement different forms of communication between nodes, making possible the exchange of information. Processes simpler and computationally lighter are the primary objective of the nodes, avoiding overloads and high computational times. If for some reason a node block, all other functionalism remains assured. The ROS master has a crucial role in managing the entire architecture. Without the triggering of this functionality, the nodes could not communicate between them. ROS master has the identity of all the nodes present in the system, establishing the exchange of information between each ROS node. The parameter server allows the user to change and configure the parameters of the nodes while they are running. This section belongs to the master ROS. ROS nodes communicate with each other using messages. Messages have a well-defined and well-known data structure between each ROS node. It is a standard structure or a manually created structure for exchange applications more specific information.

Each message is transported using topics. When a ROS node sends a message through a topic, the node is considered to be a publisher. On the other hand, when a ROS node receives a message from a topic, is defined as being a subscriber node. A subscriber publisher template is a form of unidirectional data transmission; that is, it is a form of data transmission that only transmits information to the network without feedback.

Bags is a format that stores all messages of topics and services, giving the possibility of playback from a defined period. In the present study, this feature was useful. In some applications, it is imperative to place orders and wait for a response, something that the publisher/subscriber model does not guarantee — using the client /server structure itself a service that is divided into two parts, requests, and responses. The client node sends a message with a request to a server node. While the server node processes the message and performs the service, the client waits by the result. After completing, the server sends a response to the client. In this model, communication is established only between two nodes [7].

3.4 ROS Industrial

So far, the ROS functionalities, applicable to didactic robots and for research. However, and in the face of all industrialization, it is intended to develop application at the industrial level. The ROS-Industrial project began in January 2012 with great collaboration Yaskawa Motoman Robotics, Willow Garage, and Southwest Research Institute. As ROS-Industrial, it is possible to combine the advantages provided by ROS to the technologies and safety in order to exploit the large capacities in manufacturing processes. Robust and reliable software for industrial applications, supported by a community of researchers and professionals are the main objectives of ROS-Industrial [8].

The use of ROS-Industrial provides numerous advantages, especially:

- Exploitation of resources: the packages are linked to the ROS library, enabling their use in different industrial robots. The ROS platform also is rich in tools such as RViz and Gazebo for visualization and simulation.
- Out-of-the-box applications: the ROS interface enables advanced work in applications of, for example, pick and place of complex objects.
- Economic and straightforward: given the ease in calculating trajectories between positions and free of collisions, the ROS-

Industrial allows to realize robust applications. Being this an open source software enables commercial use without any restrictions or even licenses so that the entity that owns the robot does not depend exclusively from the manufacturer's software.

The diagram in figure 10 shows a representation of the packages that are organized on top of the ROS.

A brief description of the current packages present in the ROSIndustrial

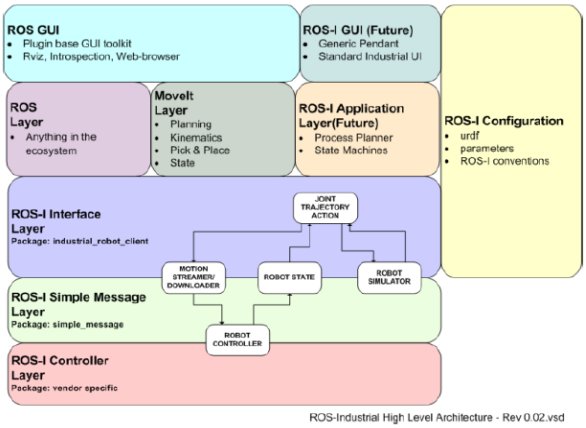

Fig. 10: Block diagram of the ROS-Industrial high-level architecture [8]

- ROS GUI includes all graphical interface tools based on plugins ROS, such as the RViz and Gazebo.
- ROS Layer is the base layer where all the communications.
- MoveIt! Layer offers solutions to plan trajectories, calculate kinematics inverse and direct and pick and place solutions in the RViz interface.
- ROS-I interface layer consists of the robot's client. Establish a connection to the manipulator controller, using the simple message protocol.
- ROS-I simple message is the communication layer with the industrial robot. It is a standard protocol that exchanges data from the client to the controller and vice versa.
- ROS-I controller is a dedicated and proprietary unit of manufacturer information of the robot.
- ROS-I configuration defines not only building parameters but also design and the model of the robot in question.

4. COMMUNICATION BETWEEN ROS-I AND ROBOGUIDE

4.1 Installing the server

In order to install the ROS-Industrial server on the Fanuc controller, the tutorials available at ROS-Industrial [9] were used as an aid. The information exchanged between the server, and the client is in the space of meetings, that is, information is exchanged of each of the robot's six points. In the same way, the robot is also in the space of meetings. Before starting any change in the driver, it is suggested that you create a backup as well as verify that the driver supports KAREL and User Socket Messaging (USM) files. The procedure is divided into three groups: installation, configuration, and execution. In summary, the procedure implemented.

Initially, a new, empty work cell was created in Roboguide, software supplied by Fanuc. In this cell, they are imported (figure 11) and compiled all programs provided in the fanuc_driver [10] folder. There are two methods for transferring the compiled files to the driver. The first method is to use the FTP connection. How the implementation server was one of the first tasks performed at work if so deep transfer of programs to the controller via FTP. The second method is the transfer of files to the controller via USB. This method is called a binary install and was adopted in the transfer of data to the controller.

Fig. 11: Fanuc driver files, imported and compiled, in Roboguide

Then, the server and downloaded files were setups. A configuration of the server requires two FTP files and two USM files. At work of Cancela [2], the four files previously explained had already been created. Because the communication protocol was the same as the one implemented in Cancela's work, the two FTP files have been running, which serve as the basis for the connection TCP / IP. On the other hand, the execution of existing USM files was interrupted, so two new and specific ROS server specifics must be created.

4.2 Starting the server

The startup of the ROS server is performed based on a program that has been transferred to the controller. The program is called ROS (figure 12) and consists of three lines of code (figure 13). The first two have the content explained in section previous, while the latter waits for the connection of a client. The possession running the program, the server is active and waiting for the connection of a customer (figure 14). When the client application is executed, the connection between both (yellow rectangle) and waits for the receipt of messages (figure 15).

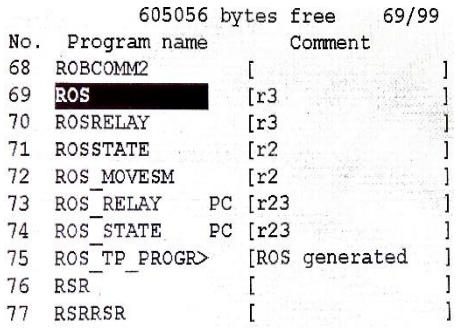

Fig. 12: List of programs in the controller

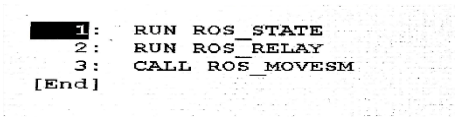

Fig. 13: Contents of the ROS program

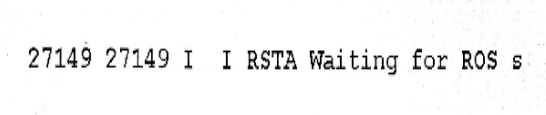

Fig. 14: Starting the server


```
27149 27149 I  I RSTA Waiting for ROS s
27175 I RSTA Connected
27175 I RREL Connected
```

Fig. 15: Connection established

4.3 Creating and Sending a TP Program

As an example, a program (Figure 16) was created in a programming language C ++. The first three lines of the program concern typical configurations of a node in ROS. After the blank space begins the code to generate a program TP.

The program starts with the necessary settings for a TP program. In this, if a name was given to the TP program, a comment to describe the program and the number of rows. By default, a program is generated with permission to write and read. After all, settings are completed, the program body is built. The program body, in the example case, starts with a comment. Posteriorly, the registration status is changed, half a second is expected and the status of the registration. After waiting another nine seconds, a blank line is added and a point in space for the robot to move. Finally, a TP program is executed existing on the controller.

After programming the post-processor, the generated program is compiled and, in a terminal, the ROS node of the created program is executed. A file is generated that contains the TP program to be sent to the controller (figure 17). The content of the generated TP program has the same information as the program body in C ++.

The last step is to upload the program from the computer to the controller via FTP. After the program is transferred to the controller (figure 18), this needs to be consoled.

```
int main(int argc, char **argv)
{
  std::string package = "fanuc_post_processor_application";
  ros::init(argc, argv, package);
  ros::NodeHandle node;

  FanucPostProcessor fanuc_pp;
  fanuc_pp.setProgramName("ros_tp_example_program");
  fanuc_pp.setProgramComment("ROS generated");
  fanuc_pp.useLineNumbers(false);
  fanuc_pp.appendComment("This is a ROS generated TP program");
  fanuc_pp.appendSetRegister(7, false);
  fanuc_pp.appendWait(0.5);
  fanuc_pp.appendSetRegister(7, true);
  fanuc_pp.appendWait((unsigned) 9);
  fanuc_pp.appendEmptyLine();
  fanuc_pp.appendPoseCNT(FanucPostProcessor::JOINT,
Eigen::Isometry3d::Identity(), 2, 20, FanucPostProcessor::PERCENTAGE, 100);
  fanuc_pp.appendRun("MY_OTHER_TP_PROGRAM");

  std::string program;
  fanuc_pp.generateProgram(program);
  ROS_WARN_STREAM("This is the generated program:\n\n" << program);

  if (!fanuc_pp.uploadToFtp("192.168.0.231"))
    return -1;
  return 0;
}
```

Fig. 16: Post-processor C ++ programming

```
ROS_TP_EXAMPLE_PROGRAM.ls  ×
/PROG ROS_TP_EXAMPLE_PROGRAM
/ATTR
COMMENT = "ROS generated";
PROTECT = READ_WRITE;
DEFAULT_GROUP = 1,*,*,*,*;
/MN
:  !This is a ROS generated TP program;
:  R[7]=0;
:  WAIT  0.5(sec);
:  R[7]=1;
:  WAIT  9(sec);
:  ;
:J P[2] 20% CNT100 ;
:  RUN MY_OTHER_TP_PROGRAM;
/POS
P[2]{
   GP1:
     UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
     X = 0.000000 mm, Y = 0.000000 mm, Z = 0.000000 mm,
     W = -0.000000 deg, P = 0.000000 deg, R = -0.000000 deg
};
/END
```

Fig. 17: TP language, generated by the post-processor, present on the computer

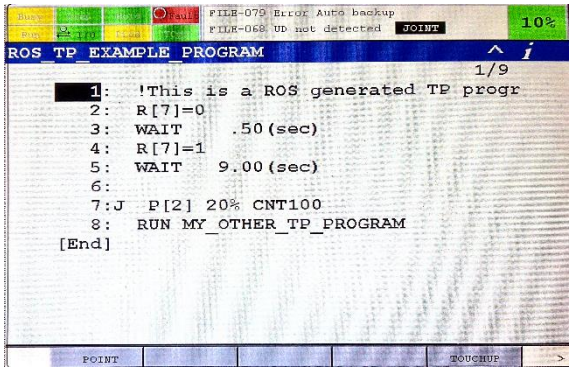

Fig. 18: Presence of the program generated by the post-processor in the controller

4.4 RViz Interface for MoveIt!

The RViz interface for MoveIt! (figure 19) Is composed of two large sections, the planning, and plugin. This interface also includes the position of each robot joint represented in the robot model. Of the available tabs, the planning tab was the one used at work, since it is the only one that plans trajectories. The rest tabs allow actions like manipulating, adding, and interacting with objects in the scene surrounding the robot. On the Planning tab, you define the start, end, and plan the robotic arm will have to travel. By default, when the viewer starts, the final and the initial Positions coincide. The orientation and position of the robot can be altered with the interaction of the interactive mark, coincident with the end of the effector. Note that this interface prevents situations of collision with the links themselves or in positions where the manipulator does not reach the intended position. The plugin focuses on visualization. The first defines the presence of the robot in the

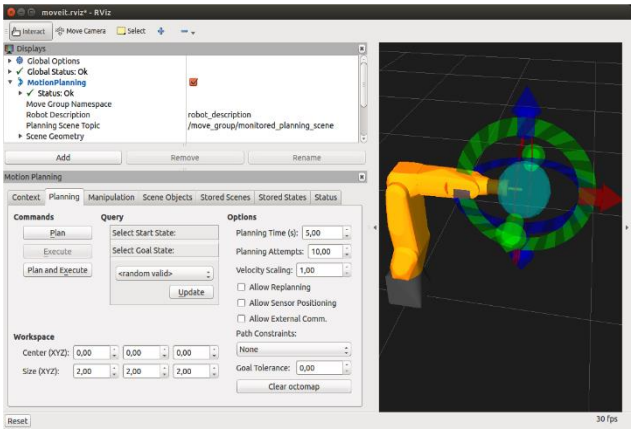

Fig. 19: RViz Interface for Fanuc LR Mate 200iD

viewer, while the second allows the visualization of the workload of the robot, the start and end positions and colors for specific situations. Finally, the third region subscribes to the topic of movement of the trajectory to be able to visualize the regular course.

5. CONCLUSION

In the context of this work, the ROS-Industrial server was adapted and implemented for the R-30iB Mate controller of the Fanuc LR Mate 200iD industrial robot and the ROSIndustrial API. The information that the server exchanges with the client and all the performed in the joint space, at a constant speed. Such as speed control, force control is also not supported by the server. The ROS-Industrial and in particular the ROS Fanuc proved modern alternative complex applications with a high level of physical and programming aspects. The ROS-Industrial combined with the advantages provided by ROS and MoveIt! Between different possibilities, the interaction with the robot in order to:

- Planning movements.
- Create virtual robot models.
- Include objects present in the surroundings of the robot in the calculation of trajectories.
- Obtain data, such as kinematics and Jacobian.
- Transfer TP programs developed on the computer to the controller.

6. REFERENCES

[1] I. F. of Robotics, "World Robotics 2015 Survey - Executive Summary", World Robotic Report - Executive Summary, pp. 10–21, 2015.

[2] R. Cancela, "Extensão e flexibilização da interface de controlo de um manipulador robótico FANUC", Dissertação de Mestrado, Universidade de Aveiro, 2007.

[3] "R-30iB Mate Controller - Maintenance Manual", rel. téc., 2011.

[4] Fanuc Corporation, "Features Application system Load / unload from ROBODRILL Operating space," Fanuc, Yamanashi, Japan, rel. téc., 2012.

[5] ROS, Ros history, 2016. address: http://www.ros.org/history/

[6] L. Joseph, Mastering ROS for Robotics Programming, 1st Ed., P. Publishing, ed. Packt Publishing, 2015, vol. 1, p. 689.

[7] A. Martinez, E. Fernandez, L. Sanchez Crespo e A. Mahtani, Learning ROS for Robotics Programming, 2st Ed., P. Publishing, ed. Packt Publishing, 2015, vol. 1, p. 458.

[8] ROS, Industrial, 2016. Address: http://wiki.ros.org/Industrial

[9] Fanuc tutorials, 2014. address: http://wiki.ros.org/fanuc/Tutorials/

[10] G. vd. Hoorn, Fanuc_driver, 2015. https://github.com/ros-industrial/ fanuc / tree / indigo-devel / fanuc_driver.

