Implementation of an open-source PROFIBUS interface between ROS and a Panasonic robot

David De Schepper

Master of Energy Engineering Technology

Jorn Geutjens

Master of Energy Engineering Technology

Introduction

Over the last decades, a substantial part of robotics research has focused on path planning algorithms for robotic manipulators.

These novel algorithms make an abstraction of the underlying robotic hardware: they generate robot motion commands and expect robot state information in a specific format that is independent of the proprietary language in which commercial robots are typically programmed.

Therefore, for each commercial robot that needs to be controlled using these algorithms, a driver is required to transform the robot-independent commands to robot-specific commands and vice versa. ROS is an example of an opensource robot software platform in which such drivers can be developed [1].

Objectives

1. Create a robot model of the Panasonic VR-006L robot in ROS (figure 1);
2. Translate ROS commands to commands that can be interpreted by the Panasonic robot controller (unidirectional);
3. Create a robust bidirectional interface.

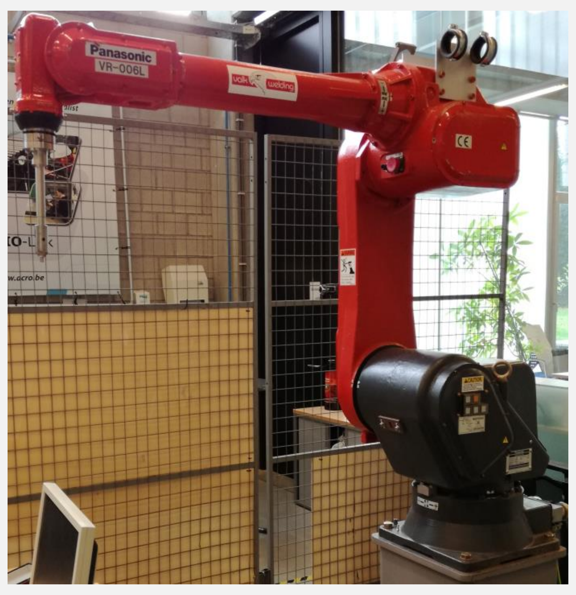
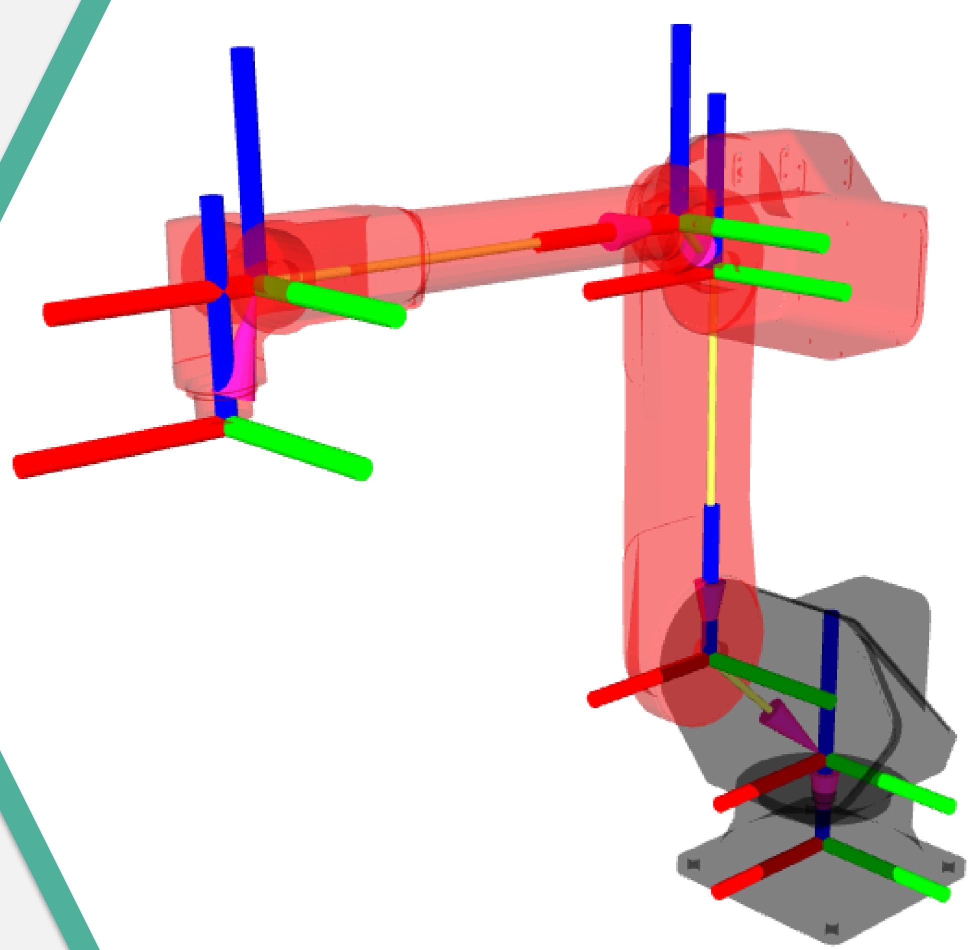


Figure 1: The Panasonic VR-006L robot at ACRO



Implementation

Create a model (figure 2) of the robot in ROS. This is performed by making a URDF (Unified Robot Description Format) file, which describes the kinematic and geometric relations between the links and the joints of a robot. ROS needs this for visualisation, path planning and collision detection.

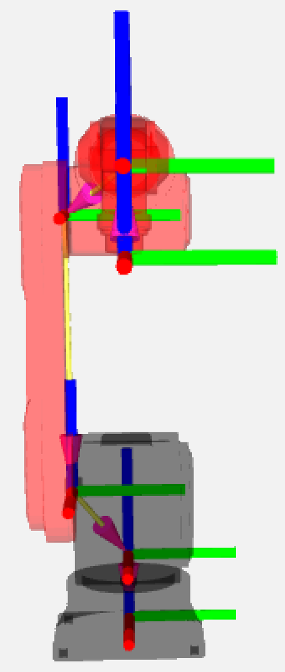


Figure 2: Robot model of the Panasonic VR-006L visualised with the joint frames in RViz

Make a physical connection between ROS and the controller. Figure 3 shows a schematic representation of the bidirectional setup. Since the Panasonic robot controller has got a PROFIBUS card, a connection using this fieldbus is obvious. To connect ROS with the robot, an open implementation of the PROFIBUS-DP protocol in Python is used [2],[3]. This makes it possible to run our driver in ROS and send commands to the inputs of the controller. Then, a Panasonic program on the controller reacts to the inputs accordingly. Finally, a USB to RS485 adapter makes it possible to send data from ROS to the robot controller using differential voltages over the PROFIBUS-DP network.

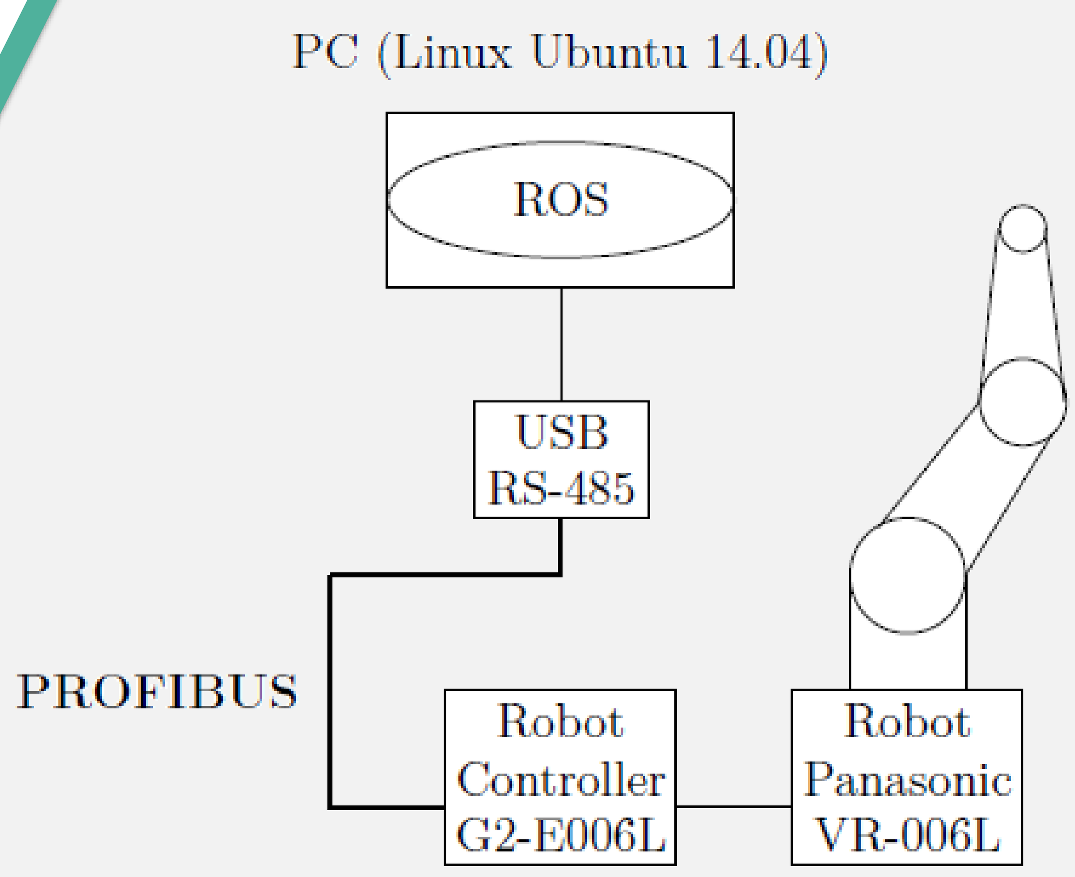


Figure 3: Bidirectional setup

Results

1. A robot model of the Panasonic VR-006L manipulator in ROS has been made.
2. An offline program for making csr files (Panasonic-specific file containing all the information of the robot program) from ROS commands has been established.
3. An online ROS driver using an open PROFIBUS implementation has been realised. Figure 4 shows an execution of a real movement visualised in RViz. The driver is based on Panasonic's G2 robot controller, of which the biggest limitations are that it cannot read in joint positions and sending these back to ROS. Therefore, an upgrade to a G3 controller with an Ethernet connection is suggested for future work to make accurate and reliable path planning possible.

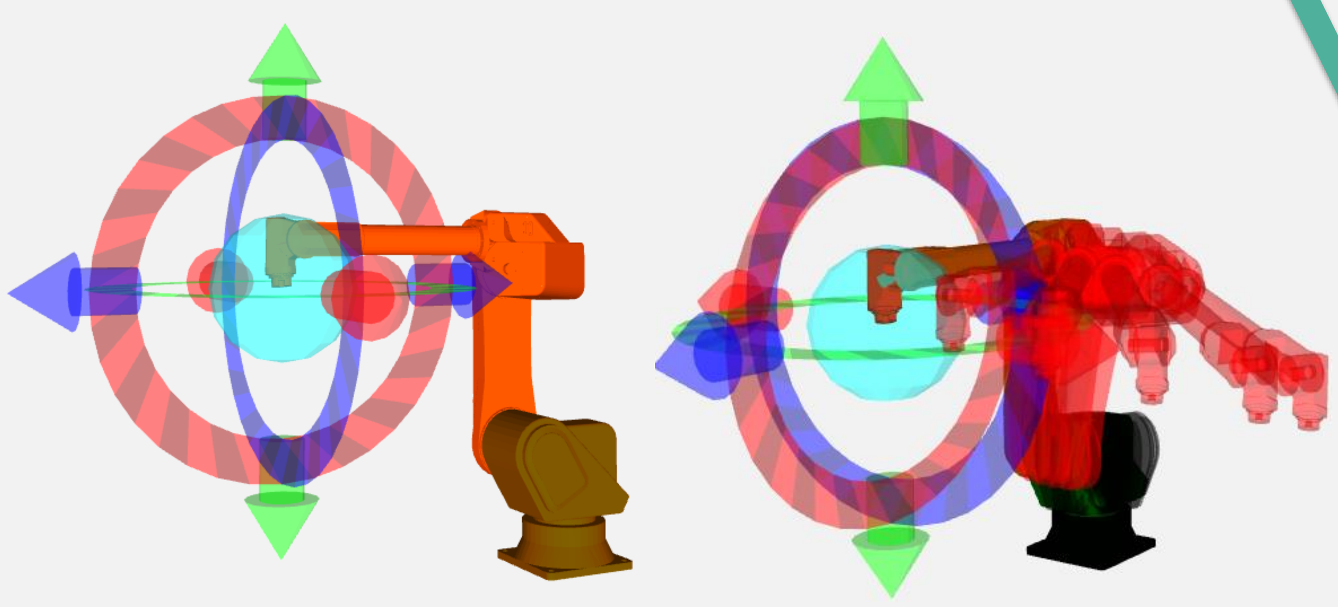


Figure 4: Generated movement visualised in RViz

Supervisors / Cosupervisors: Prof. Dr. Ir. Eric Demeester
Ir. Jeroen De Maeyer

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," Icra, vol. 3, p. 5, 2009.
[2] M. Büsch, "PROFIBUS software stack," 2016, last visited on 20-05-2018. [Online]. Available: https://bues.ch/cms/automation/profibus.html
[3] ACROMAG, "Introduction to PROFIBUS-DP," ACROMAG, Tech. Rep., 2002.

ACRO UHASSELT KU LEUVEN