JOpenShowVar: an Open-Source Cross-Platform Communication Interface to *Kuka* Robots

F. Sanfilippo [1], L. I. Hatledal [1], H. Zhang [1], M. Fago [2] and K. Y. Pettersen [3]

[1]Department of Maritime Technology and Operations, Aalesund University College, Postboks 1517, 6025 Aalesund, Norway,
`[fisa, laht, hozh]@hials.no`

[2]IMTS S.r.L. Company, Taranto, Italy, `massimiliano.fago@gmail.com`

[3]Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway,
`kristin.y.pettersen@itk.ntnu.no`

This work is partly supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254 and the Innovation Programme for Maritime Activities and Offshore Operations, project number 217768

IEEE ICIA 2014

Summary

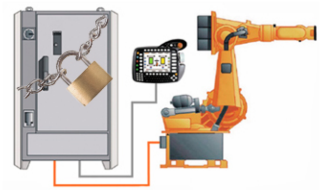
Current robot control interfaces

While the hardware and mechanical requirements of developed robots are often similar for both industry and research, scientific software requirements are quite different and even contradictory in many aspects [1].

The goal of scientists is to try to gain as much control over the robot as possible:

- it is difficult to find interfaces that are applicable for research purposes;
- the disclosure of the internal control architecture is very hard to come by;
- many manufacturers are unwilling to publish internal details regarding system architecture due to the high levels of competition in the robot market.

Consequently, it is not possible to fully exploit many robotic platforms in a scientific context.



[1] G. Schreiber et al., "The fast research interface for the kuka lightweight robot", ICRA, 2010

Kuka robots

The *Kuka Robot Language* (KRL) offers the possibility of declaring data types, specifying simple motions, and interacting with tools and sensors via I/O operations [2].

While the KRL offers an easy to use interface for industrial applications, it is very limited when it comes to research purposes:

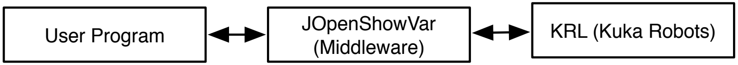
- a KRL program only runs on the *KUKA Robot Controller* (KRC);
- controller-specific set of instructions;
- no support for advanced mathematical tools;
- no mechanism for including third party libraries;
- no external input devices can be directly used.

A common workaround:

- supplementary software packages provided by *Kuka* such as *KUKA.RobotSensorInterface KUKA.Ethernet KRL XML*;
- drawbacks such as limited I/O, a narrow set of functions and often require major capital investments.

[2] M. Schopfer et al., "Open source real-time control software for the Kuka light weight robot", WCICA, 2010

Underlying idea

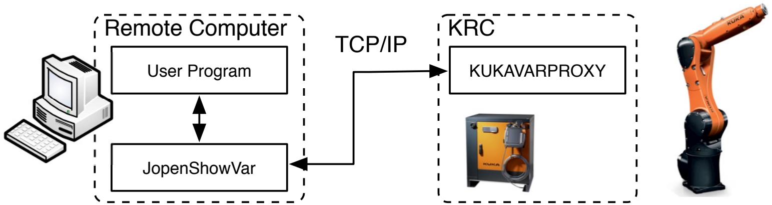


A Java open-source cross-platform communication interface that makes it possible to read and write all of the controlled manipulator variables:

- different input devices;
- sensors;
- alternative control methods.

JOpenShowVar works as a *middleware* between the user program and the KRL. This software interface is an open-source project and it is available on the Internet at `https://github.com/aauc-mechlab/jopenshowvar`, along with several detailed class diagrams, documentation and demo videos.

JOpenShowVar architecture



- It is a client-server architecture with *JOpenShowVar* running as a client on a remote computer and *KUKAVARPROXY* acting as a server on the KRC. *JOpenShowVar* locally interacts with the user program and remotely communicates with the *KUKAVARPROXY* server via *TCP/IP*.
- *KUKAVARPROXY* is a multi-client server (Visual Basic 6.0) and can serve up to 10 clients simultaneously. It implements the *Kuka CrossComm* class.
- *Kuka CrossComm* allows for the interaction with the real-time control process of the robot and makes it possible to perform several operations. *KUKAVARPROXY* implements the reading and writing methods.

JOpenShowVar architecture

- All the variables have to be declared as global variables in the predefined global system data list $CONFIG.DAT.
- All kinds of variables can be declared in this file from basic types such as INT, BOOL and REAL to more complex structures like E6POS and E6AXIS that allow for storing the robot configuration.
- Several system variables can be accessed provided there are no restrictions due to the type of data such as for $PRO_IP, $POS_ACT, $AXIS_ACT or $AXIS_INC.

Kuka CrossComm class does not provide a real-time access to the robot's data:

- it takes a non-deterministic time to access a specific variable. According to our experiments the average access time is about 5 ms;
- this time interval is not affected by the kind of access to be performed (whether it is a reading or a writing operation) or by the length of the message;
- by using structures of data it is possible to simultaneously access several variables thereby minimising the access time.

Reading variables

- To access a variable, the client must specify two parameters in the message: the desired type of function and the variable name. To read a specific variable, the type of function must be identified by the character “0”.
- For instance, if the variable to be read is the system variable $OV_PRO, which is used to override the speed of the robot, the message that the client has to send to the server will have the following format:

Table: Reading variables

Field	Description
00	message ID
09	length of the next segment
0	type of desired function
07	length of the next segment
$OV_PRO	Variable to be read

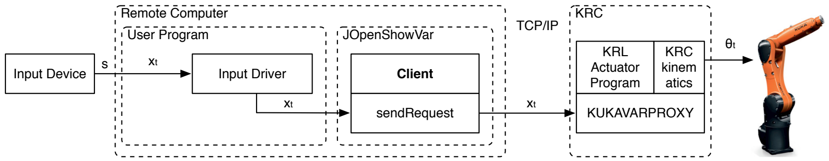
Writing variables

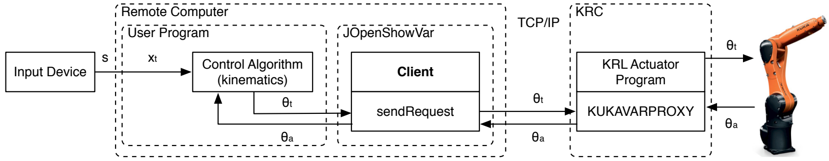
- To write a specific variable, three parameters must be specified: the type of function, the name of the desired variable and the value to be assigned. The writing function is specified by the character “1”.
- For instance, if the variable to be written is the system variable $OV_PRO with a value of 50 (50% override speed), the message that the client has to send to the server will have the following format:

Table: Writing variables

Field	Description
00	message ID
0b	length of the next segment
1	type of desired function
09	length of the next segment
$OV_PRO	Variable to be written
50	value to be written

Control approach





Control approach

It should be noted that KRL does not provide a native way to obtain velocity control. When using the KRC kinematics, this limitation can be overcome by expressing the target position as:

$$\mathbf{x}_t = \mathbf{x}_d, \tag{1}$$

if operating in position control mode, or by:

$$\mathbf{x}_t = \mathbf{x}_a + \dot{\mathbf{x}}_d \Delta t, \tag{2}$$

if operating in velocity control mode, where Δt is the time interval between two successive iterations. Alternatively, when a custom control algorithm is needed, the target joint configuration is given by:

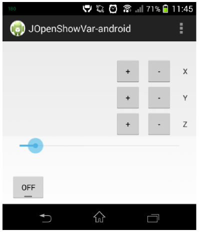
$$\theta_t = \theta_d, \tag{3}$$

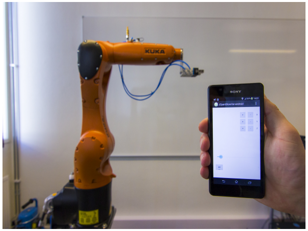
if operating in position control mode, or by:

$$\theta_t = \theta_a + \dot{\theta}_d \Delta t, \tag{4}$$

if operating in velocity control mode.

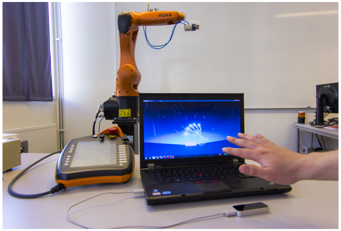
Controlling the KR 6 R900 SIXX manipulator with an Android mobile device





```
DEF ACTUATOR()
  INI
  PTP HOME Vel = 100 % DEFAULT
  $ADVANCE=1
  LOOP
    PTP_REL MYPOS C_PTP
  ENDLOOP
  PTP HOME Vel = 100 % DEFAULT
END
```

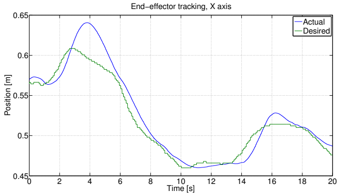
Controlling the KR 6 R900 SIXX manipulator with a Leap Motion controller

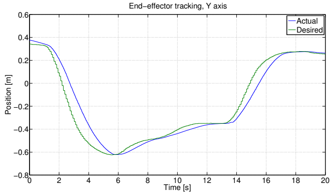


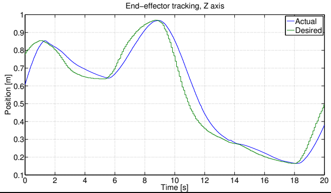
```
DEF EXT_MOVE_AXIS()
  DECL AXIS LOCAL
  INI
  PTP HOME Vel = 100 % DEFAULT
  $ADVANCE=1
  LOCAL.A1 = $AXIS_ACT.A1
  ...
  LOCAL.A6 = $AXIS_ACT.A6
  LOOP
    LOCAL.A1 = LOCAL.A1 + MYAXIS.A1
    ...
    LOCAL.A6 = LOCAL.A6 + MYAXIS.A6
    PTP LOCAL C_PTP
  ENDLOOP
  PTP HOME Vel = 100 % DEFAULT
END
```

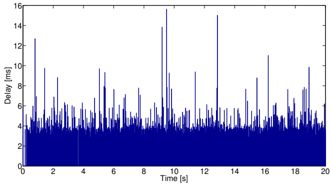
Experimental results

Experimental results









Conclusion and future work

JOpenShowVar opens up to a variety of possible applications:

- Even though *JOpenShowVar* only provides a soft real-time access to the manipulator to be controlled, this *middleware* package opens up to a variety of possible applications making it feasible to use different input devices, sensors and to develop alternative control methods.

Different control approach and standardisation:

- In the future, different control algorithms may be tested as alternatives to the standard KRC.
- Finally, some effort should be put in the standardisation process of *JOpenShowVar* to make it even more reliable for both the industrial and the academic practice.
- In the author's opinion, the key to maximising the long-term, macroeconomic benefits for the robotics industry and for academic robotics research relies on the closely integrated development of open content, open standards, and open source.

Thank you for your attention