Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 51 (2020) 133–139

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
15-18 June 2021, Athens, Greece.

Model-based hardware in the loop control of collaborative robots

Mohammad Safeea[a,b,*], Pedro Neto[a], Richard Béarée[b]

[a]Univ Coimbra, Centre for Mechanical Engineering, Materials and Processes, Department of Mechanical Engineering, Coimbra, Portugal
[b]Arts et Métiers, LISPEN, Lille, France

Abstract

Simulation and model-based design software packages are widely used in many engineering disciplines. When it comes to robotics those tools are very important for robot design, simulation and the development of control algorithms before the implementation on the real robot. Simulink by MathWorks® is an advanced model-based design tool. It is popular in education, industry and research. In addition, Simulink supports several hardware components, facilitating a rapid deployment of the developed programs on the target hardware. In this study, the SimulinkIIWA interface for controlling KUKA iiwa robots from Simulink is presented and compared to the KUKA Sunrise Toolbox (KST). This interface is based on the User Datagram Protocol (UDP) and allows graphical real-time control of iiwa robots from Simulink without a need for writing any code. The interface supports different robot control modes, at the joints level and at the end-effector (EEF) level. Example applications are also provided showing the flexibility and the ease of use of the proposed interface.

Keywords: Collaborative Robots; Model-based design; Simulink; Hardware in the loop control;

1. Introduction

Due to the demand from industry, KUKA has introduced the iiwa series of collaborative manipulators, the result of the joint research efforts between KUKA Roboter and the German Aerospace Center (DLR) [1]. Those robots have seven degrees of freedom (DOF) and are designed to mimic the anatomy of the human arm. KUKA iiwa robots are considered sensitive manipulators due to the integrated torque sensors in their joints. This sensitivity is important to guarantee co-worker's safety in human robot interaction applications. On the other hand, Simulink by MathWorks® is an important engineering tool used for the development of high-tech applications and is widely popular in education, research and industry. For example, in [2] Simulink is used to develop and implement a dynamic walking controller for a human-sized bipedal robot. Thus, combining the ease of use of Simulink on an external computer with the capabilities of KUKA iiwa robot has its merits for many applications. In robotics, it is not uncommon to have an external computer running along with the robot-controller, where the PC can be used for performing computationally expensive operations, for treating data from sensors or even for implementing control algorithms.

The robotics community has presented various software packages related to robot simulation and control. The Robotics Toolbox for MATLAB (RTB) described in [3] is used by many students and researchers worldwide. This toolbox provides packages for the fundamental algorithms in robotics, which are essential for their modeling and control. Roy Featherstone described the algorithms for modeling robot dynamics [4] accompanied by software implementation (including Simulink) [5]. The SpaceDyn toolbox introduced in [6] integrates floating base dynamics as such it can be used for simulating space robots. Another toolbox which can be used for the modeling/simulation of manipulators is described in [7]. This toolbox integrates graphical user interface (GUI), 3D simulator and utilizes symbolic math for producing the physical quantities of the robot. The Machine Vision Toolbox (MVT) introduced in [8], is a MATLAB toolbox which integrates vision and image-processing algorithms, as such the user can extend his/her robotic application with machine vision capabilities [9].

While the previous software packages are used for performing advanced mathematical operations related to robotics, they

* Corresponding author. Tel.: +351 239 790 700 ; fax: +351 239 790 701.
E-mail address: ms@uc.pt (Mohammad Safeea).

2351-9789
10.1016/j.promfg.2020.10.020

do not support direct control of robots from an external computer, consequently other packages have been proposed for this purpose. The Kuka Control Toolbox (KCT) [10] and the JOpenShowVar [11, 12] are used for controlling 6 DOF KUKA robots from MATLAB/Java. However, KCT and JOpenShowVar support the KRC controller only, as such they can not be used for controlling KUKA iiwa robots (provided with the newer Sunrise controller). The Robot Operating System (ROS) is another option for controlling robots from an external computer. Several packages have been proposed for different types of manipulators. In [13] the authors used the ROS interface, iiwa_stack, for controlling iiwa robot to perform autonomous MRI-guided ultrasound, and in [14] the authors presented the KUKA-IIWA-API, an interface for controlling iiwa using ROS. This interface provides data acquisition and point-to-point (PTP) motion features, but it does not support on-the-fly control feature.

Recently, the KUKA Sunrise Toolbox (KST) [15] is introduced. It is a MATLAB toolbox that allows controlling KUKA iiwa robots from an external computer. KST implements Transmission Control Protocol Internet Protocol (TCP/IP) to connect with the robot. It integrates several functionalities including (1) networking functions, (2) point-to-point motion functions, (3) on-the-fly control functions, (4) setters and getters of robot parameters, (5) general purpose control functions and (6) physical interaction functions. KST allows the user to control the robot remotely from MATLAB scripting language. Due to its complex design, it does not implement Simulink integration. As such, we developed the SimulinkIIWA interface for controlling KUKA iiwa robots from Simulink. Unlike the KST and the previously listed ROS-interfaces (all TCP/IP based), the SimulinkIIWA has a different nature, Table I. It is based on the User Datagram Protocol (UDP), it supports on-the-fly control only, and it implements the Desktop Real-Time^{TM} toolbox, which provides real-time kernel for performing real-time simulations or hardware-in-the-loop implementation from Simulink. Since that the Desktop Real-Time^{TM} toolbox is available only for Windows and Mac, the presented interface can only be used under those two operating systems. Simulink comes with a rich library supporting control, digital signal processing, artificial intelligence, interfacing with hardware in addition to others, which gives the ability to develop hardware-in-the-loop control algorithms graphically without requiring written code.

Table 1. Comparison, KST *vs* SimulinkIIWA interface.

Comparison criteria	KST	SimulinkIIWA
Supported OS	Linux, Windows, Mac	Windows and Mac
Programming tool	MATLAB (code-script)	Simulink (graphical)
Communication protocol	TCP/IP	UDP
Messaging format	ASCII	Binary
Running threads	Two	Three
Programming blocks	100+ different methods	Three blocks
Supported functionalities	Seven different categories	Only real-time

2. SimulinkIIWA interface

The KUKA iiwa 7R800 and 14R820 are programmed using the Java based Sunrise.Workbench engineering suite. In addition, various packages for controlling the robot from an external computer have been proposed, by using MATLAB, ROS and C++. However, no Simulink interface is available for these kind of robots (according to our knowledge). Simulink is a popular tool for developing control algorithms, because it implements relevant libraries, including interfaces to a wide range of sensors, actuators and other hardware components. In addition, Simulink allows the development and implementation of powerful control algorithms graphically, without requiring to write any code. Due to those advantages, we present an interface for on-the-fly control of KUKA iiwa manipulators using Simulink, this interface supports several operation modes elaborated in subsection 2.1. The interface consists of two parts, the SimulinkIIWA Sunrise application (runs in the robot controller) demonstrated in subsection 2.2, and the SimulinkIIWA subsystem blocks (in Simulink) described in subsection 2.3. The interface is available in the public repository [16], which includes additional documentation including a web-link to video tutorials.

2.1. Supported operation modes

SimulinkIIWA interface supports different on-the-fly control modes, each operation mode is provided in a different application that the user can run from the teach pendant of the robot. Those applications were written using the Sunrise.Workbench. The control modes provided by the SimulinkIIWA interface are:

- Cartesian/Joints position control mode which is built upon the DirectServo [17]. In this control mode the user can control the robot at EEF or joints level, due to the utilization of the DirectServo this operation mode is very reactive.
- Cartesian/Joints position control mode which is built upon the SmartServo [17]. Due to the utilization of the SmartServo this mode provides smoother motions, but it is less reactive than the first mode.
- Impedance control mode, suitable for applications that involve physical interaction between the manipulator and the surroundings.

All interfaces support feedback from the various sensors integrated in the robot.

2.2. SimulinkIIWA Sunrise application

The SimulinkIIWA Sunrise Application (SSA) is a multi-threaded application which runs on the Sunrise controller. This application is written using the Sunrise.Workbench. The main thread of the application implements the DirectServo and the SmartServo (from KUKA), and is used for on-the-fly control of the manipulator. This application shall be synchronized to the robot, several flavors are provided according to the control

mode required for operating the robot as described previously in subsection 2.1.

2.3. SimulinkIIWA subsystem blocks

For controlling the robot from an external computer using Simulink, three different subsystem blocks are provided by the proposed interface, Fig 1. The user can copy the blocks and insert them in his/her customized block diagram. Those blocks communicate with the robot using UDP protocol and are used:

- For controlling the robot at EEF level, the Simulink block `Command IIWA in Cartesian Space` is provided, Fig 1 (left). This block has six inputs representing the target pose (position/orientation) of the EEF. In such a case, the (X,Y,Z) position and the (α, β, γ) rotation angles of the EEF are streamed to the robot using UDP protocol. The (X,Y,Z) coordinates are taken relative to the base frame of the robot, and (α, β, γ) are the fixed rotation angles around axes Z,Y and X (respectively) of the base frame of the robot.
- For controlling the robot at joints level, the Simulink block `Command IIWA in Joint Space` is provided, Fig 1 (middle). This block has seven inputs representing the reference positions of the robot joints. In such a case the joints angles are streamed to the robot using UDP protocol.
- For acquiring feedback about the state of the robot, the Simulink block `IIWA State` is provided, Fig 1 (right). This block has twenty seven outputs, organized: (1) the outputs one to seven represent the joints torques due to external forces acting on the structure of the robot, (2) the outputs eight to fourteen represent the raw torques measurements at the joints as acquired from the integrated torque sensors, (3) the outputs fifteen to twenty one represent the actual joints positions as acquired by the integrated encoders, (4) the outputs twenty two to twenty four represent the components of the external force at the flange, (5) the outputs twenty five to twenty seven represent the components of the external moment at the flange.

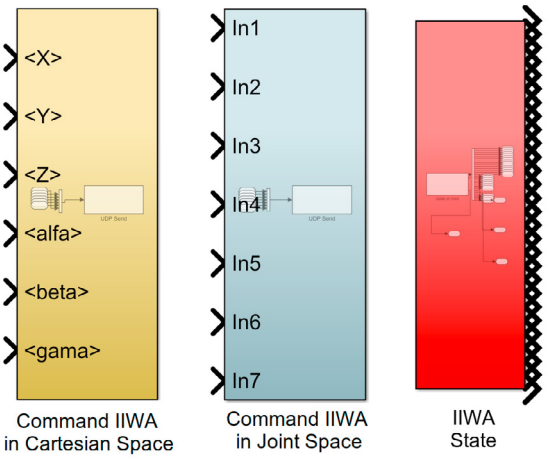


Fig. 1. Simulink block for controlling the robot at EEF level (left), controlling the robot at joints level (middle), acquiring feedback about the state of the robot from its controller (right).

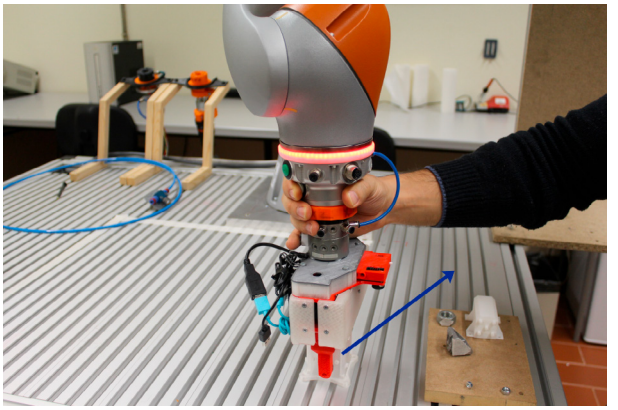


Fig. 2. Human-robot interaction example implemented in Simulink using the proposed interface, the user is moving the EEF by applying a force at the robot, the arrow shows the direction of the motion (orientation is kept fixed in this example).

3. Tests and Results

In this section two different tests and a use case are proposed. In Test 1 an example application is provided demonstrating the capabilities of the proposed interface. The SimulinkIIWA is used in developing a control algorithm for manually guiding iiwa robot at the EEF level, Fig. 2. In this example a feedback about the forces at the flange of the robot is acquired and used as input for the control algorithm. Based on this feedback the motion commands are calculated and streamed to the robot. Virtue to the graphical user interface in Fig. 3, the user can change the stiffness along each axis separately (X, Y or Z), or even constrain the motion to only one or two dimensions. Meanwhile, in Test 2 a detailed comparison between the network communication performance for the SimulinkIIWA interface and the KST is provided. A use case is also provided showing the use of the SimulinkIIWA interface for performing a pick and place operation. More examples can be found in the web-page of the presented interface, including drawing geometrical shapes using the robot, and for commanding the joints in impedance mode.

3.1. Test 1

In this example, the user can guide the robot at EEF level while keeping the orientation of the robot fixed, Fig 2. A video demonstration of the test is available in [18]. For the sake of simplicity, this example is made different to our previous work in [19], in which the user can move the robot manually, including orientation control, at one direction at a time (implementing force and position feedback). Figure 3 shows the Simulink diagram used for the control scheme in Test 1. In the presented

block diagram the subsystem `IIWA State` is used to acquire a feedback about the external force at the flange frame of the robot, the orientation of the EEF is kept fixed using the block `Orientation Value`. Then, the applied forces along the X, Y and Z directions of the base frame of the robot are used as inputs to the following motion controller subsystems:

- Subsystem `X motion control` is used to calculate the X reference coordinates, or the motion command along the X axis according to the X component of the force-feedback at the flange of the robot. The building blocks of this subsystem are shown in Fig 4. From the figure the subsystem consists of (a) a dead zone filter (b) a gain (c) an integrator (d) a constant bias and (e) a summing block. The dead zone filter is used to cut off the noise from the measurement and also to add an initial resistance to the motion. The gain is used to change the resistance of the motion according to the filtered force. The integrator is used to calculate the motion command, represented by the displacement along the X axis from the initial position. The constant is used to specify the initial position. The summing block is used to sum the initial position with the integrated motion command, the result is used for the X input of the subsystem `Command IIWA in Cartesian Space`, which streams the reference X coordinate to the robot.
- Subsystem `Y motion control` is used to calculate the Y reference coordinates, or the motion command along the Y axis according to the Y component of the force-feedback at the flange of the robot. Similar to subsystem `X motion control`.
- Subsystem `Z motion control` is used to calculate the Z reference coordinates, or the motion command along the Z axis according to the Z component of the force-feedback at the flange of the robot. Similar to subsystem `X motion control`.

The reference coordinates calculated from the previous subsystems are used as inputs to the block `Command IIWA in Cartesian Space` provided by the SimulinkIIWA interface. This block streams motion reference positions using UDP protocol to the robot, which are used by the Sunrise controller as motion commands that drive the robot. The block diagram in Fig 3 also includes the Real-Time Sync block from Simulink, this block utilizes the real-time kernel for executing Simulink models in real-time on a personal computer (laptop or desktop) with Windows or Mac operating systems. In addition, the presented Simulink block diagram includes graphical potentiometers which are used to change the values of the gain blocks, accordingly changing the resistance of the motion with the applied force for each axis separately.

Tests using KUKA iiwa 7R800 were performed and the force/coordinates data at EEF were collected from Simulink and then plotted as shown in Fig 5.

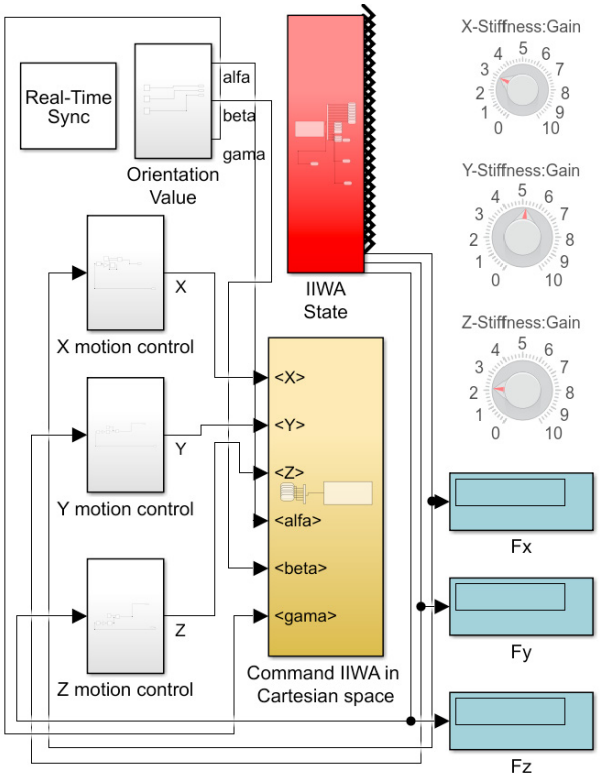


Fig. 3. Simulink block diagram for the example of hand-guiding the EEF in Cartesian space in Test 1.

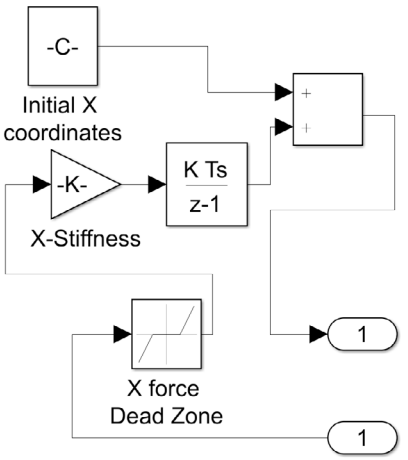


Fig. 4. Simulink block diagram for the X motion control block.

3.2. Test 2

In this test, technical comparison between the SimulinkIIWA and the KST is presented in terms of the communication architecture and performance. Given that Simulink is used for real-time hardware-in-the-loop control, the SimulinkIIWA interface is designed to support only the real-time control of KUKA iiwa

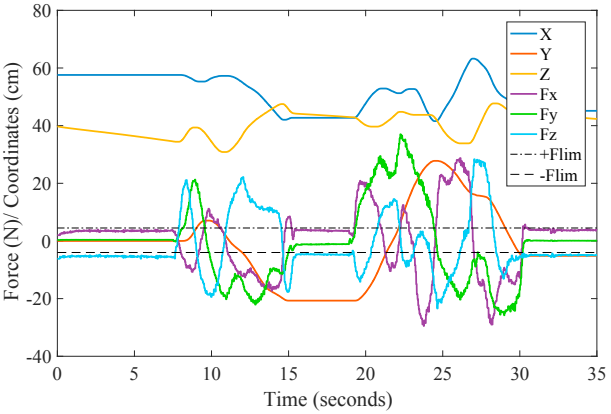


Fig. 5. Coordinates and force-feedback at EEF with time for the example guiding the EEF in Cartesian space.

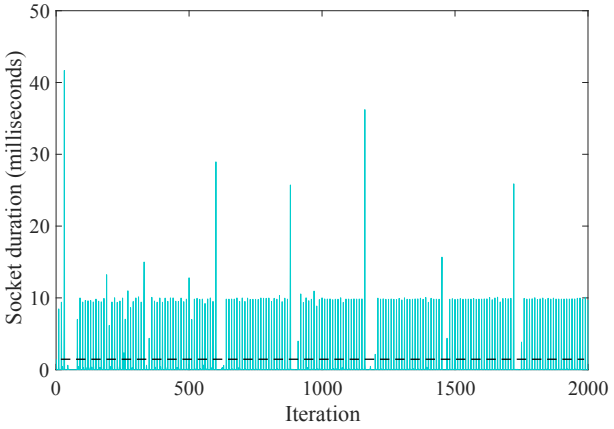


Fig. 6. Time interval between two consecutive motion commands (UDP sockets) received by the robot from Simulink (using SimulinkIIWA interface). The dashed line represents the average value.

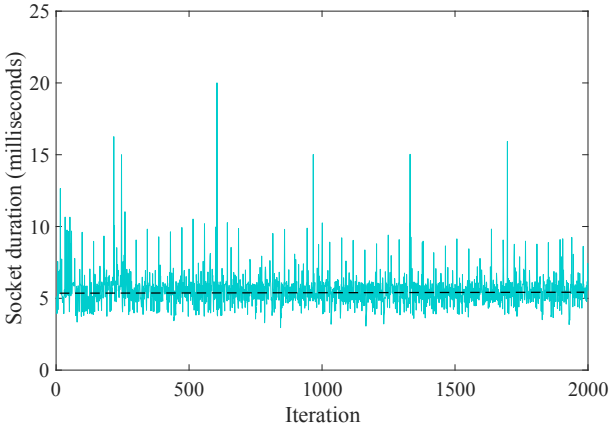
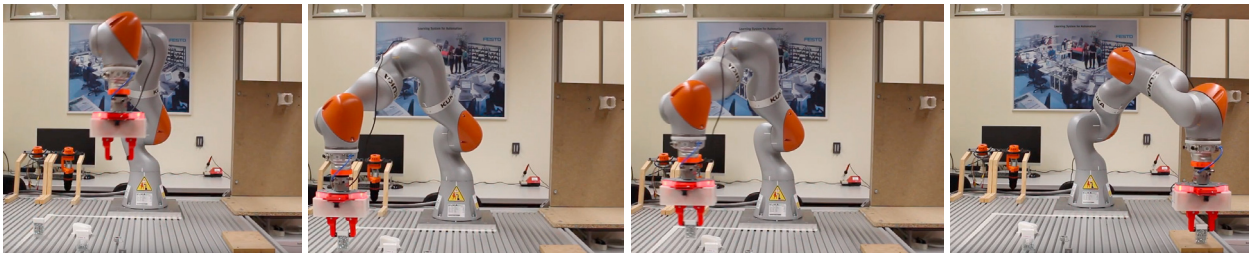


Fig. 7. Time interval between two consecutive feedback data messages (UDP sockets) sent from the robot to Simulink (using SimulinkIIWA interface). The dashed line represents the average value.

robot. Consequently, we opted to build the interface upon UDP communication protocol. UDP is better suited for data streaming than the TCP/IP protocol. However, using UDP does not guarantee the arrival of every message to the server. This is why the KST and other ROS interfaces for KUKA iiwa implement a TCP/IP protocol, where losing a message in particular type of commands (onetime-execution command) will lead to execution errors (for example in point-to-point motion). Because of its ACK-NACK (acknowledgment, negative acknowledgment) design, the server application of KST implements two separate threads, the main thread and the communication thread. The main thread is used for the main control loop. The communication thread is used for receiving and transmitting data. Only one socket is used, a response message is sent for each command received. On the other hand, the SSA implements three different threads, one is used for the control loop, another is used for receiving position command stream from Simulink and the third is used for streaming the state feedback from the robot to Simulink. The SimulinkIIWA interface does not implement ACK-NACK. Two different sockets are running at the same time, one socket is used to stream feedback data, another socket is used to receive position command stream. This fact allows the SimulinkIIWA interface to operate at higher network rates. In our tests the KST achieves 275 HZ rate for motion commands transmission [15]. On the other hand, Fig. 6, shows the timing interval between each two consecutive UDP motion commands received by the robot from Simulink, where the timing data was collected at the controller side. Using the Sunrsie.OS a timestamp is recorded when each motion command message is received. The results show that the SimulinkIIWA interface is able to achieve 1KHz rate for motion commands stream capacity. The figure also shows that the timing intervals are not regular, there are sporadic spikes with a maximum of 42 milliseconds. However, the average time interval is 1 millisecond. At the same time, the interface achieved an average 177 Hz rate for the feedback data stream, Fig. 7, where the maximum time interval between two consecutive feedback data measurements sent from the robot to Simulink is around 20 milliseconds, the average time interval is 5.6 milliseconds. The slow rate for feedback data streaming is caused by the fact that the SimulinkIIWA interface is designed to feedback the state data for (1) actual joints positions, (2) the external torques, (3) the measured torques of all the joints and (4) the force and the moment acting at the EEF, all together, leading to considerable processing time by the robot controller to acquire the various measurements. Finally, the timestamps shown in the figures 6 and 7 are calculated at the robot side, where we took advantage of the deterministic nature of the Sunrise.OS to achieve higher accuracy in timing measurements.

3.3. Use case

To further test the proposed interface, a KUKA iiwa robot is controlled from an external computer using Simulink to manipulate a box in a pick and place operation as shown in the snapshots of Fig. 8. First the robot moves from the home position towards the box Fig. 8 (a). When the robot reaches an



(a) (b) (c) (d)

Fig. 8. Snapshots of KUKA iiwa robot performing a pick and place operation, the robot is controlled from an external computer using SimulinkIIWA interface. (a) Robot moves from home position towards the box. (b) Robot reaches the box location and closes its gripper. (c) Robot starts manipulating the box. (d) Robot places the box in the target position.

appropriate picking location of the box, the gripper is closed Fig. 8 (b). Afterwards, the robot starts moving the box Fig. 8 (c) towards the target location. When the release position is reached the robot leaves the box Fig. 8 (d). The block diagram of the Simulink program for this example is shown in Fig. 9, A MATLAB Function block (named Cartesian Path) is used to program the path for the pick and place motion, the input to this block is the simulation time (from Simulink's Clock block), while the outputs are the Cartesian coordinates (calculated as a function of time). Those Coordinates are fed to the block Command IIWA in Cartesian space of the SimulinkIIWA for controlling the robot. In this use case the control is open loop at Simulink side, where we rely on the robot's internal servo closed loop control (of the direct servo motion) to track the reference coordinates. The gripper used is custom made in our laboratory and it is also controlled from Simulink. The control command for the gripper is also a function of time (calculated in the block Gripper Command), and streamed to the gripper using the block Gripper Ctl. From this example, it is shown that the presented interface is a useful tool in education, allowing students to have a hands-on experience by prototyping and testing their Simulink algorithms on the real robot.

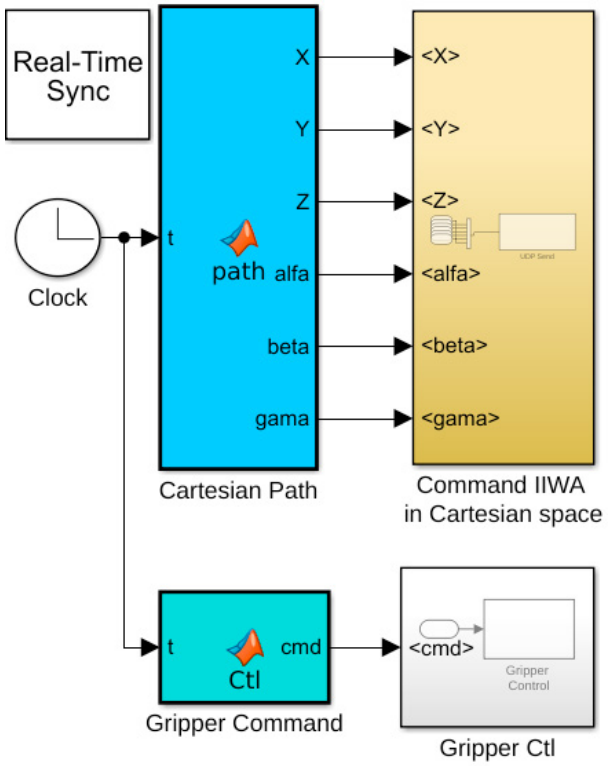


Fig. 9. Simulink block diagram for the pick and place use case example.

4. Conclusion

In this study we presented the SimulinkIIWA interface and compared it with the KST for controlling KUKA iiwa robots from an external PC. Using this interface, the user can design and run complex control algorithms in Simulink for iiwa manipulators graphically and without a need to write any code. The interface supports different control modes, spanning Cartesian and joint space, either in impedance or in position control modes. It also supports feedback from the robot sensors including the joint torques/positions, in addition to the forces and moments acting at the EEF. The SimulinkIIWA interface is based on UDP protocol which is advantageous for the on-the-fly control applications. To show the flexibility and the ease of use of the proposed interface various examples are provided.

Acknowledgements

This research was partially supported by Portugal 2020 project DM4Manufacturing POCI-01-0145-FEDER-016418 by UE/FEDER through the program COMPETE 2020, and the Fundação para a Ciência e a Tecnologia (FCT) SFRH/BD/131091/2017 and COBOTIS (PTDC/EME-EME/32595/2017). This research is also sponsored by FEDER funds through the program COMPETE Programa Operacional Factores de Competitividade, and by national funds through FCT Fundação para a Ciência e a Tecnologia under the project UIDB/00285/2020.

References

[1] Bischoff, R., Kurth, J., Schreiber, G., Koeppe, R., Albu-Schffer, A., Beyer, A., Eiberger, O., Haddadin, S., Stemmer, A., Grunwald, G. and Hirzinger, G., 2010, June. The KUKA-DLR Lightweight Robot arm-a new reference platform for robotics research and manufacturing. In ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), 1–8.

[2] Hubicki, C., Abate, A., Clary, P., Rezazadeh, S., Jones, M., Peekema, A., Van Why, J., Domres, R., Wu, A., Martin, W. and Geyer, H., 2018. Walking and Running with Passive Compliance: Lessons from Engineering: A Live Demonstration of the ATRIAS Biped. IEEE Robotics & Automation Magazine, 23–39.

[3] Corke, P., 1996. A robotics toolbox for MATLAB. IEEE Robotics & Automation Magazine, 24–32.

[4] Featherstone, R., 2014.Rigid body dynamics algorithms. Springer.

[5] Featherstone, R., Spatial Vectors and Rigid-Body Dynamics software packages [On-line]. Available: http://www.royfeatherstone.org/spatial/index.html#spatial-software/.

[6] Yoshida, K., 1999. The SpaceDyn: a MATLAB toolbox for space and mobile robots. International Conference on Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ , 1633–1638.

[7] Dean-Leon, E., Nair, S. and Knoll, A., 2012, December. User friendly Matlab-toolbox for symbolic robot dynamic modeling used for control design. IEEE International Conference on Robotics and Biomimetics (ROBIO), 2181–2188.

[8] Corke, P., 2005. The Machine Vision Toolbox: a MATLAB toolbox for vision and vision-based control. IEEE Robotics & Automation Magazine, 16–25.

[9] Corke, P., 2007. MATLAB toolboxes: robotics and vision for students and teachers. IEEE Robotics & Automation Magazine, 16–17.

[10] F. Chinello and S. Scheggi and F. Morbidi and D. Prattichizzo., 2011. KUKA Control Toolbox. IEEE Robotics & Automation Magazine, 69–79.

[11] Sanfilippo, F., Hatledal, L.I., Zhang, H., Fago, M. and Pettersen, K.Y., 2014, July. JOpenShowVar: an open-source cross-platform communication interface to kuka robots. Proc. of the IEEE International Conference on Information and Automation (ICIA), Hailar, China, 1154–1159.

[12] Sanfilippo, F., Hatledal, L.I., Zhang, H., Fago, M. and Pettersen, K.Y., 2015. Controlling Kuka industrial robots: Flexible communication interface JOpenShowVar. IEEE robotics & automation magazine, 96–109.

[13] Hennersperger, C., Fuerst, B., Virga, S., Zettinig, O., Frisch, B., Neff, T. and Navab, N., 2017. Towards MRI-based autonomous robotic US acquisitions: a first feasibility study. IEEE transactions on medical imaging, 538–548.

[14] Mokaram, S., Aitken, J.M., Martinez-Hernandez, U., Eimontaite, I., Cameron, D., Rolph, J., Gwilt, I., McAree, O. and Law, J., 2017. A ROS-integrated API for the KUKA LBR iiwa collaborative robot. IFAC-PapersOnLine, 15859–15864.

[15] Safeea, M. and Neto, P., 2019. KUKA Sunrise Toolbox: Interfacing Collaborative Robots With MATLAB. IEEE Robotics & Automation Magazine, 91–96.

[16] SimulinkIIW interface package [On-line]. Available: https://github.com/Modi1987/Simulink-iiwa-interface/.

[17] KUKA Sunrise.Connectivity Servoing 1.7. KUKA Roboter GmbH, 2015.

[18] Video demonstration of Test 1 [On-line]. Available: https://youtu.be/PLOWiyzmVqA/.

[19] Safeea, M., Béarée, R. and Neto, P., 2017. End-effector precise hand-guiding for collaborative robots. in Iberian Robotics conference, Springer, 595–605.