 DOI: 10.3217/978-3-85125-663-5-09

Towards a flexible industrial robot system architecture

Raimund Edlinger[1], Lydia Leimer[2], Michael Zauner[3] and Roman Froschauer[4]

Abstract— **The present work deals with the recording, transmission and presentation of sensor data, which is transmitted by different sensors mounted on or in mobile robots. Complex, heterogeneous, modular robot systems require manufacturer- and user-independent standardized interfaces based on open communication standards and information models to enable interoperability and integration. Cross-system communication and data retrieval from different devices of different manufacturers is complicated by proprietary application programming interfaces (APIs). It is virtually impossible to exchange modules with devices from alternative manufacturers, which makes it difficult to integrate devices that meet the requirements. The OPC-UA communication interface is a platform-independent standard and is widely used in robotics and automation technology to connect compatible devices with different interfaces. In this paper we present the concept and implementation of a standardized communication interface for data exchange and visualization with ROS-based robot systems.**

I. INTRODUCTION

Modern robotic systems often comprise several components from different vendors to accomplish specific tasks. Resulting from this heterogeneous interfaces, communication protocols etc. prevent exchange of components (either hardware or software). Furthermore any kind of reconfiguration or reuse of components is virtually impossible. On a higher level of abstraction ROS (Robot Operating System) has been established as flexible middle-ware dealing with this problem. Unfortunately industrial robots and peripheral devices have no support for ROS and therefore limit the modular software ecosystem provided by ROS. Within the domain of industrial automation systems in the last years OPC-UA has been established as vendor-independent communication protocol. Recent developments such as OPC-UA Publish/Subscribe are dealing with loosely coupled devices at the shop floor level of automation and robot system. Therefore this paper proposes an overall system architecture for implementing flexible and intelligent robot systems featuring ROS on programming or behavioural layer and OPC-UA as core communication layer (see also [8]).

The flexibility of production that will become necessary in the future requires a high degree of cognition and independence from the automation solutions used. Within the framework of Industry 4.0, systems that integrate sensors, actuators and cognition are referred to as cyber-physical systems (CPS), which are regarded as key technologies for the production of the future [7]. The communication at CPS is not only based on pure data exchange between machines, but on the exchange between many functional units from machines to planning software to the integration of human decision makers. The specific implementations of a production system by end users require a simple and adaptable, flexible solution that enables interoperability in the overall system. Exemplary aspects to be considered in an inter-operable production system:

- Consistency of information flows
- Application of suitable methods for modelling, calculation, simulation and optimisation
- Involving people as creative actors in the global value stream
- Design of the man-machine interface

Systems currently on the market that pursue an integrated process chain strategy only consider very specific application areas and/or proprietary product or system technologies. ROS is one of the most popular frameworks for robotics researchers and manufacturers, but it does not provide the necessary security against possible cyber attacks and data theft. [1] present a secure communication channel for ROS which handles the communication between two nodes in a secure manner. [10] et al. introduce a new research tool to facilitate cyber-physical security research.

In the first chapters a short introduction to the topics OPC-UA and ROS [3] is given. Furthermore, an overview of the architecture "from sensor to user interface" and the structure of the ROS topics of a mobile recovery robot is presented. Finally, the research results will be presented.

II. STATE OF THE ART

A. ROS and ROS-Bridge v2.0

Robot Operating System (ROS) has been developed as part of the STAIR project at Stanford University. ROS is an open source software package that can be used for a variety of different applications within robotics. Currently ROS only runs on Unix based platforms. A port to Windows is basically possible, but is in the experimental stage [4]. The following objectives have been set for the development of ROS:

- ROS is designed as a peer-to-peer system: ROS makes it possible to combine several devices into one system. Instead of a central server, a peer-to-peer structure was used. The reason for this is that the entire system can be used in the general consists of devices on a mobile platform and further external computing units. Thus, computation intensive tasks can be transferred to more powerful computers. Since the majority of

*This work was supported by the Austrian Research Promotion Agency within the program "Strategy Innovative Upper Austria 2020" under grant agreement NR. 862013

[1]Raimund Edlinger, [2]Lydia Leimer, [3]Michael Zauner and [4]Roman Froschauer are with the Upper Austrian University of Applied Sciences, 4600 Wels, Austria {raimund.edlinger, lydia.leimer, michael.zauner, roman.froschauer}@fh-wels.at

the mobile robot system is only wirelessly connected to the external computers, unnecessary traffic over the already inefficient connection should be avoided. If the system was set up as/with a the form of a central server, communication between the robot modules would also have to be handled via the possibly external server. The result would be a heavy load on the wireless connection.

- ROS is open to all languages: ROS supports the programming languages C++, Python, LISP and Octave. In order to be able to use ROS on a par with other programming languages, data within ROS is represented in a neutral format.
- ROS is an open source project: All ROS source code is freely available under the BSD license. This also allows the development of commercial products with ROS without the obligation to make the developed code freely available.
- ROS has a modular structure: The functionalities of ROS are strictly packed into individual modules. This may lead to a loss of efficiency, but the stability of the system is improved. In case of faulty software fragments, the affected modules can be identified and deactivated if necessary.
- Encourage the generation of reusable code: Too strong system-specific dependencies make it difficult to extract and reuse code that has already been written. Through the implemented build system, ROS tries to encourage the reuse of existing code packages. ROS itself uses code from other open source projects [12].
- The rosbridge v2.0 server implementation makes it easy to add and modify protocol operations and decouples JSON-handling from the websockets server [2]. This allows users to arbitrarily change the specific websockets server implementation they are using.

B. OPC Unified Architecture (UA)

Based on OPC, one of the first standards established for inter-device communication in the 90s, OPC-UA was developed in order to provide platform independent functionality, supporting more complex data and systems than its predecessor [9]. The new standard, which promotes reliable, scalable and flexible communication across systems, can be used in automation as well as other areas for data transport. Using a self descriptive, extendable abstract base model, OPC-UA supports object orientation and provides security features [9]. The service oriented architecture is based on TCP/IP and covers layers 5, 6 and 7 of the OSI-Model. For use in non time-critical systems across internet and firewalls, XML-based variants have been specified. In order to support easy access for developers, the OPC Foundation provides the basic implementation as an OPC-UA Stack for different platforms, currently supporting C, C# and Java. Toolkits for C, C++, C# and Java are provided by independent vendors [5]. This makes OPC-UA particularly attractive for use in robot systems.

C. Other Communication Protocols

To support connectivity for various communication protocols and standards common interfaces should include interface implementations of a WebSocket API, OPC-UA Server and Client and an MQTT broker. Schel et.al. developed the concept and implementation of Manufacturing Service Bus (MSB) [13]. Another communication protocol is DDS (Data Distribution Service), which has its applications mainly in government and military uses. Like OPC-UA's server-client architecture, it provides data transport on a publish-subscribe basis. Due to its similar transport style, the ROS community uses DDS as communication standard for ROS 2. Unlike OPC-UA, DDS has been implemented over UDP, although some vendors provide support for TCP as well [11]. Also designed for machine to machine transport of telemetric data is MQTT (Message Queuing Telemetry Transport)[1]. It is aimed at usage scenarios with distant and/or mobile devices, where an efficient use of bandwidth is a requirement. Wireless Sensor Networks (WSNs) have been gaining increasing attention, where MQTT is used as an extension of the open publish/subscribe protocol [6].

III. Approach ROS - OPC-UA Stack

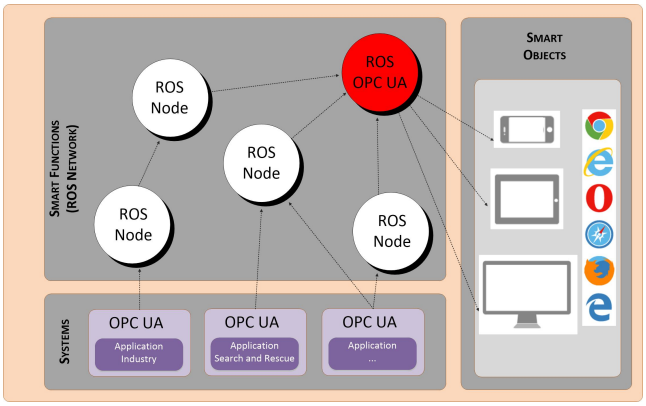

Fig. 1. ROS networks using OPC-UA Pub/Sub

Heterogeneous robot systems are robot systems that either combine several robots of the same or a different design or robots with complex machines, such as injection molding machines. Typically if mobile robots are equiped with external sensors, measuring systems, a connection to higher-level control systems and a control system are also used.

The appropriate equipment and safety systems are needed to implement the desired functions. With an increasing number of system components and possible actions, higher demands are required on the control of the overall system. Robot and component manufacturers often integrate UGV, robot arm systems, safety devices and peripherals into proprietary overall solutions. By using OPC-UA as a uniform communication standard between the system components, it is possible to make hardware and manufacturer independent components easier to integrate, exchange and thus more

[1] http://mqtt.org/

flexible in their application, in order to be able to engineer heterogeneous robot systems more quickly.

A. Data transmission from sensor to user interface

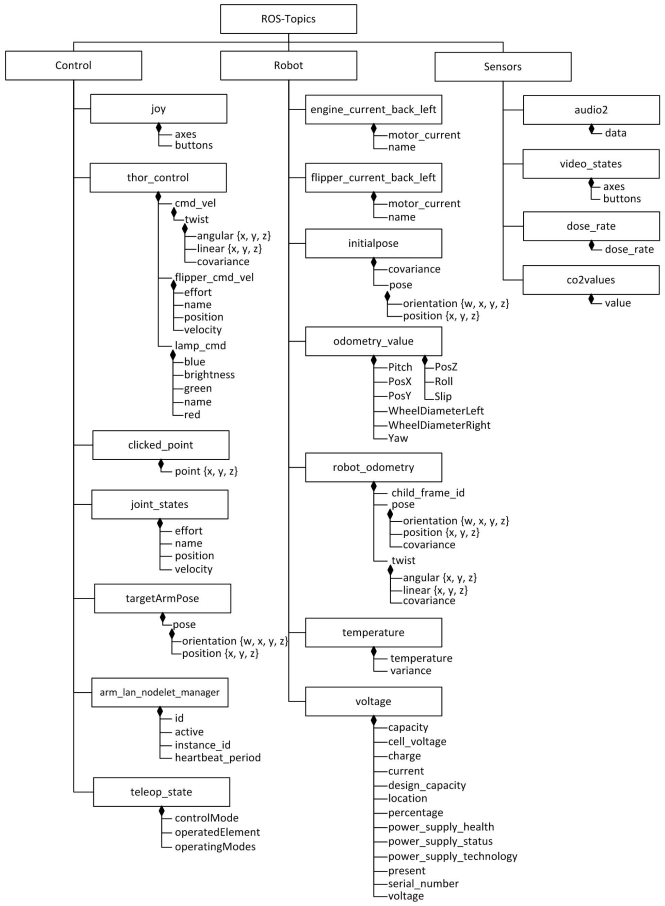

Fig. 2. ROS topics of a mobile rescue robot

In ROS operated robot systems, sensor data is forwarded from the operating system to ROS, where they can be made accessible to other applications. Sensor data can be read, visualized for users or processed, for example to create maps of the robot surroundings. Utilizing appropriate extensions like the ros_opcua_communication package, data can be provided to other applications and computers across the network. In this approach it is not the sensor itself which publishes its data and meta-data via OPC-UA, but ROS which publishes the information independently as ROS topics. These ROS topics include attributes common to all OPC-UA nodes as well as those only applicable to variable type nodes. Depending on its size and structure, data from several sensors can be merged before publishing. This is as much part of the freedom in implementation design as the naming of topics and variables. Fig. 2 shows an example of ROS topics of a mobile rescue robot. The separation into control, robot and sensors is content related and not reflected by the interface. ROS also publishes several topics required for the communication and by the software, such as for logfiles. Based on the existing ROS OPC-UA communication stack, the package was extended by a parameterization, where it is then possible to publish selected ROS topics and ROS services. This enables a robot-specific adaptation of the data to be published in relation to the wireless connection and limited bandwidth to mobile robot systems. Due to flexibility and the need for information in today's industrial world is constantly increasing and more and more process data must be processed and visualized. At the process control level, dashboards are required to give a quick overview of the most important indicators of a process. This is particularly important in order to react quickly to changes in a process.

B. Data visualization

Software for the visualization of robot data is available especially for the field of automation technology in industry. In most cases, the hardware manufacturers also develop and/or distribute the appropriate software. For service robots the offer is clearly smaller. In the following, visualization possibilities with the open source software ROS are presented. By using the OPC-UA technology, CERTEC EDV GmbH offers a universally applicable solution. Fig. 5 shows an example for a visualization of live stream, sensor and robot data. The interfaces created with Atvise[2] are made available via a web server, so that they can be accessed from any end device via a web browser. This approach also makes the interfaces useful for displaying sensor data on mobile devices.

IV. Show case

For demonstrating the approach the architecture of an existing mobile robot system has been adopted as shown in 3.

The architecture for the mobile robot documented in Fig. 3 utilizes CAN bus, USB and Ethernet for the transmission of sensor data. The sensor data of the motor controllers are transmitted to ROS via a main controller. Sensors that detect the robot's environment are often connected via USB or RJ45 to the onboard PC. Video data can also be transmitted to a ROS based system via Ethernet, which makes the live stream available via HTTP. For simple and versatile user interaction a web-based visualization tool-kit featuring OPC-UA communication has been used.

A ROS-controlled robot, see fig. 4 provides the sensor data via OPC-UA and is therefore suitable for OPC-UA capable visualization software. With the help of the visualization software Atvise, the data can be grouped and formatted on graphical user interfaces. Depending on the application and user, interfaces can be designed for different terminal devices and screen sizes. Figure 5 shows a graphical user interface which has been designed for different terminal devices and screen sizes and is designed for any browser-enabled devices such as desktop, laptop or tablet computers, PDAs or smartphones.

[2]https://www.atvise.com/en/

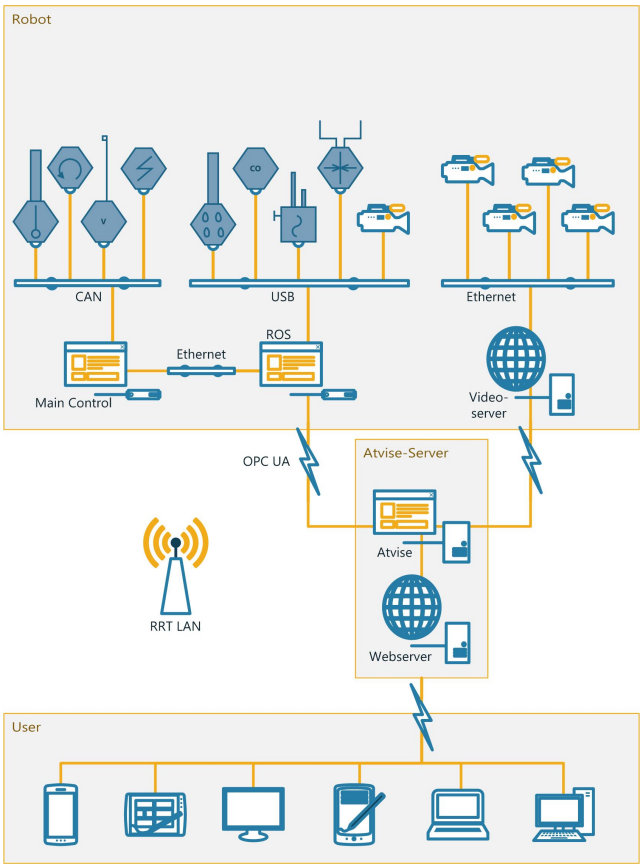

Fig. 3. Architecture: from sensor to user interface

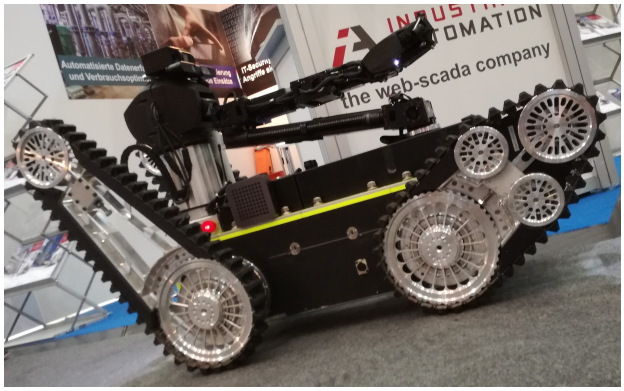

Fig. 4. Rescue Robot

V. Conclusion

The basic research focuses on OPC-UA, a platform-independent and object-oriented standard for communication between machines that has been developed [3]. OPC-UA covers the upper three layers of the OSI model. An overview graphic was created for the data transmission in which it is schematically shown that the sensors on the robot can be connected via different systems, the data is passed on via servers and prepared for the end application. The rough structure of the controller, robot and sensor data is also graphically documented, the fine structure of the so-called

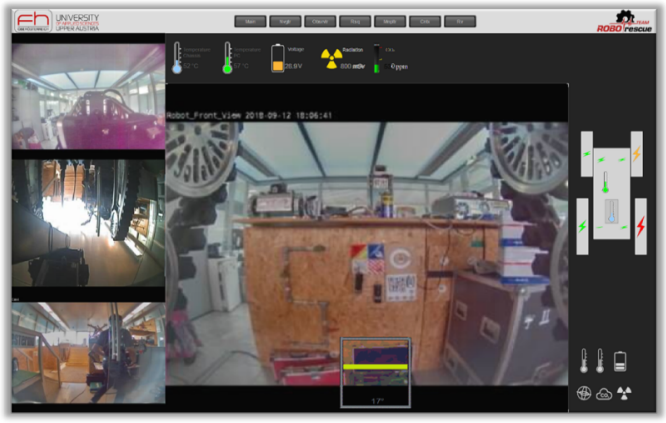

Fig. 5. Graphical user interface for prototype

nodes is recorded in the OPC-UA fundamentals. In order to provide an overview of the possible approach to visualization, three visualization tools are presented: the open source framework ROS for mobile robots, the universally applicable Atvise, and Visual Components for industrial automation. Sensor data can be visualized in different ways, there are hardly any limits to creativity. For example, distances can be plotted directly in camera images, physical quantities displayed using slide controls or dial gauges, or camera images are distorted/corrected, and can be superimposed. The good overview of the components and technologies used the graphics and explanations provided are suitable for the training of future Employees. The documentation can also be used as a basis for further developments.

References

[1] B. Breiling, B. Dieber, and P. Schartner, "Secure communication for the robot operating system," in *2017 Annual IEEE International Systems Conference (SysCon)*. IEEE, 2017, pp. 1–6.

[2] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Robotics Research*. Springer, 2017, pp. 493–504.

[3] O. Foundation. (Accessed: 2015-11-25) Opc foundation. opc ua in the reference architecture model rami 4.0. [Online]. Available: https://opcconnect.opcfoundation.org/2015/06/opc-ua-in-the-reference-architecture-model-rami-4-0/

[4] O. S. R. Foundation. (Accessed: 2017-04-03) Ros introduction. [Online]. Available: http://wiki.ros.org/ros/Introduction

[5] A. GmbH. (Accessed: 2018-07-18) Opc unified architecture. [Online]. Available: http://www.ascolab.com/de/unified-architecture/

[6] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-sa publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, 2008, pp. 791–798.

[7] N. Jazdi, "Cyber physical systems in the context of industry 4.0," in *2014 IEEE international conference on automation, quality and testing, robotics*. IEEE, 2014, pp. 1–4.

[8] C. Lalancette. (Accessed: 2017-04-24) Robot operating system tutorial. [Online]. Available: http://wiki.ros.org/ROS/Tutorials/ExaminingPublisherSubscriber

[9] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[10] J. McClean, C. Stull, C. Farrar, and D. Mascareñas, "A preliminary cyber-physical security assessment of the robot operating system (ros)," in *Unmanned Systems Technology XV*, vol. 8741. International Society for Optics and Photonics, 2013, p. 874110.

[11] G. Pardo-Castellote, "Omg data-distribution service: Architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.* IEEE, 2003, pp. 200–206.
[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
[13] D. Schel, C. Henkel, D. Stock, O. Meyer, G. Rauhöft, P. Einberger, M. Stöhr, M. A. Daxer, and J. Seidelmann, "Manufacturing service bus: an implementation," in *11th CIRP Conf. Intell. Comput. Manuf. Eng.*, vol. 67, 2017, p. 6.