5G DEPLOYMENT FOR SMART INDUSTRY USAGE

CALL: BREITBAND AUSTRIA 2030: GIGAAPP
THEME: DIGITIZATION & BROADBAND
PROJECT TYPE: COOPERATIVE R&D PROJECT
PROJECT START: 1 MAY 2023
PROJECT DURATION: 18 MONTHS

D3.1 DEFINE STANDARDIZED INTERFACES AND TECHNOLOGIES THAT CAN BE USED IN CYBER-PHYSICAL SYSTEMS AND INTEGRATED WITH THE 5G SYSTEM SUCH AS A LOOKUP FOR PROTOCOLS THAT CAN BE EASILY INTEGRATED (CAMERA SYSTEM, LOCALIZATION SYSTEM, ETC.)

5GEARING project is funded under the research and technology development of gigabit applications as part of lighthouse projects Breitband Austria 2030: GigaApp financed by the Austrian Federal Ministry of Finance and by the Austrian Research Promotion Agency (FFG) under the grant agreement n. FO999899772. Breitband Austria 2030: GigaApp was initiated by the Austrian Federal Ministry of Agriculture, Regions, and Tourism.

Document Information	
Project acronym:	5Gearing
Project number:	FO999899772
Deliverable number:	D3.1
Deliverable full title:	Define standardized interfaces and technologies that can be used in cyber-physical systems and integrated with the 5G system such as a lookup for protocols that can be easily integrated (Camera system, localization system, etc.)
Deliverable short title:	Standardized interfaces for robot communication
Submission date:	31.6.2024
Status:	[Draft]
Lead Author(s):	Andreas Gaich

Table of Contents

1 Executive Summary

Communication protocols have always been a big challenge for the robotic industry. Protocols are used to exchange data between two devices. In industrial applications, they are often used to allow a computer to communicate with a robot or an end effector. In order to be able to communicate, the two devices need to understand the selected protocol. Many companies have tried to develop their own protocol over the years. This has resulted in the creation of a lot of different protocols. The first part of this document provides an overview of the most common standardized protocols and interfaces used in industrial robotics and application development. The second part lists and describes the interfaces of the Universal Robots 5e (UR5e) cobot [8] and the Autonomous Mobile Robot (AMR) LD250 from OMRON Global [2] which are used within the scope of this project.

2 Standardized protocols in robot communication

Communication protocols, often called Fieldbus, describe the set of rules to be used in communication between devices. The list of protocols used in today's industrial robotics is quite large. In order for 3rd party suppliers of robotic tools, i.e. ROBOTIQ [5], to be able to develop these tools, they have to be aware of the used protocols by the robot manufacturers. This section lists the most common communication protocols used in industrial robotics.

2.1 TCP/IP and UDP

Transmission Control Protocol/Internet Protocol (TCP/IP) and User Datagram Protocol (UDP) are essential communication protocols robots use to transmit data across networks. TCP/IP requires minimal central management and allows robots to communicate autonomously within a network environment. One of the key strengths is its ability to recover automatically from device failures to ensure continuous operation even if individual components experience issues.
UDP prioritizes speed and efficiency in exchange for potential data loss. While TCP/IP ensures reliable and ordered delivery of data packets, UDP offers a simple, connectionless communication approach. Robots often utilize UDP for applications where immediate transmission is critical, such as video streaming or real-time control systems.

2.2 EtherNet/IP

Ethernet Industrial Protocol (Ethernet/IP) [3] is built on the standard TCP/IP (IEEE 802.3) and communications use existing network infrastructure. Ethernet physical layer technology is used along TCP and UDP ports (44818 and 2222). Its main advantage comes from the progress of physical Ethernet, from 10 Mbits/s to 10/100 Mbits/s to 1 Gbits/s and more. Ethernet/IP also ensures Internet and enterprise connectivity for remote control and offers various network topology options including star or linear with standard Ethernet infrastructure devices. QuickConnect™ functionality allows devices to be exchanged rapidly (e.g., a tool changer on a robot arm) while the network is running.
Like all CIP networks, Ethernet/IP utilizes the Common Industrial Protocol (CIP) for its upper layers. CIP networks follow the Open Systems Interconnection (OSI) model, which defines a framework for implementing network protocols in seven layers: physical, data link, network, transport, session, presentation and application. Networks that follow this model define a complete suite of network functionality from the physical implementation through the application or user interface

layer.

2.3 ControlNet

Built on its own physical and data link layer, ControlNet [3] uses a single media link with RG6 coaxial cables. It features 5Mbits/s of speed, upload/download of data, peer-to-peer (P2P) communication, and up to 99 nodes. ControlNet provides users with the tools to achieve deterministic, high-speed transport of time-critical I/O. ControlNet offers a choice of topology options including trunkline-dropline, star or tree. Hardware options are also offered for applications requiring intrinsically safe hardware. Redundant network communication is available as well. ControlNet is part of the CIP family.

2.4 DeviceNet

DeviceNet [3] is a digital, multi-drop fieldbus network that connects and serves as a communication network between industrial controllers and I/O devices, providing users with a cost-effective network to distribute and manage simple devices throughout the architecture. DeviceNet utilizes Controller Area Network (CAN) for its data link layer, the same network technology used in automotive vehicles for communication between smart devices.
DeviceNet uses a trunkline-dropline topology and has DC power available on the network cable to simplify installations by providing a single connection point for network communications and device power up to 24 Vdc, 8 Amps. QuickConnect™ functionality allows devices to be exchanged while the network is running. In addition, DeviceNet operates in a controller-device or a distributed control architecture using P2P communication, and it supports both I/O and explicit messaging for a single point of connection for configuration and control. DeviceNet is part of the CIP family.

2.5 Modbus

The Modbus [1] protocol is based on a master/slave communication. The master device can choose to broadcast messages or directly address devices on the network, and the slave devices can respond to directly addressed messages only. The slave devices can also perform actions based on the messages received from the master device.
Each message will have an address, a function code, a data register, and a cyclic redundancy check (CRC) register. The slave device will compare the message address, and if matches the assigned address, the device will execute the function

code. The simplicity of Modbus function codes and addresses allows it to be used for both digital and analog inputs and outputs, in addition to data exchange. Since it can be used on top of common protocols like RS-232, RS-485, and Ethernet, it has remained highly capable and flexible, while being basic enough to be used in countless devices.

2.6 Profibus

Process Field Bus (Profibus) [4] is a standard for fieldbus communication in automation technology. Profibus employs a master/slave communication architecture. In this setup, one or more devices act as masters, controlling communication on the bus, while other devices act as slaves, responding to requests from the master. The physical layer can be a twisted pair cable delivering power to the device and providing 9.6 kbit/s to 12 Mbit/s or an optical fiber cable. Several network topologies are supported, i.e. line, ring and star structures, providing flexibility in system design. A single Profibus network can support up to 126 nodes.

2.7 ROS-Industrial

Robot Operating System (ROS) [6] is an open source software development kit for robotics applications. ROS is heavily utilized by the research community for service robotics applications, but its technology can be applied to other application areas, including industrial robotics. Since it is a multi-platform framework ROS 2 is supported on Linux, Windows and macOS, as well as various embedded operating sytems. ROS is mainly composed of two things: A core (middleware) with communication tools and a set of plug & play libraries.
In ROS you program nodes, which are small programs that do some processing. The nodes then communicate with each other using topics, services and actions. Topics are mainly used for sending data streams between nodes. Services allow to create a simple synchronous client/server communication between nodes. Actions are based on topics. They exist to provide an asynchronous client/server architecture, where the client can send a request that takes a long time, i.e. asking to move the robot to a new location. The client can asynchronously monitor the state of the server, and cancel the request anytime. An advantage of ROS is that its language is agnostic and therefore nodes can be written in any programming language. ROS uses standard TCP/IP sockets to communicate between nodes.
ROS-Industrial [7] is an open-source project that extends the advanced capabilities of ROS to manufacturing, automation and robotics. The ROS-Industrial repository includes interfaces for common industrial manipulators, grippers, sensors, and device networks.

3 Interfaces of the UR5e and Omron LD250

3.1 UR5e

The UR robot can interact with external devices by different types of communication interfaces as shown in Figure 1.

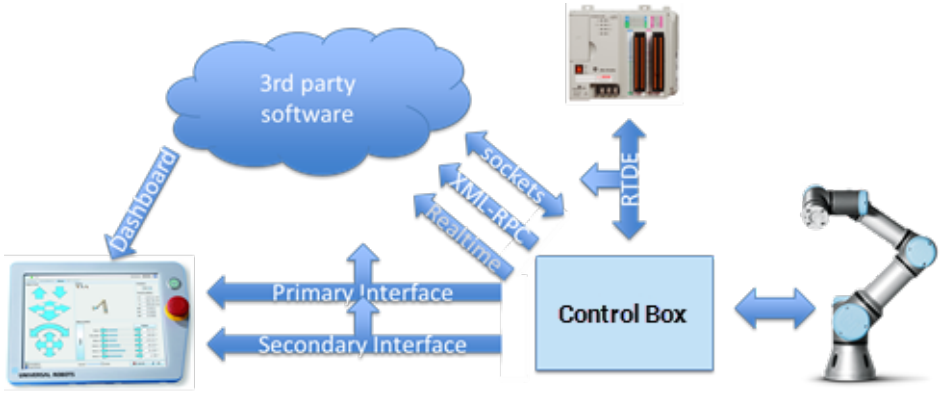

Figure 1: Overview of UR5e client interfaces (Source: https://www.universal-robots.com)

The UR5e can operate in local and remote mode. In the remote mode it can receive and process commands from external sources through several interfaces shortly described in the following.

3.1.1 Primary/Secondary Interfaces

The UR controller provide servers to send robot state data and receive URScript commands. The primary interface transmits robot state data, such as positions, temperatures, etc. and additional messages. The secondary interface transmits robot state data only. The data is mainly used for communication between a Graphical User Interface (GUI) and the controller. Both accept URScript commands with 10 Hz update rate, what makes it possible to control the robot remotely without a local robot program running.

3.1.2 Dashboard Server

The cobot can be controlled from remote by sending simple commands to the GUI over a TCP/IP socket. This interface is called the Dashboard Server. Main functions of the server are to load, play, pause, and stop a robot program, set user access level, and receive feedback about robot state.

3.1.3 Socket Communication

The UR robot can communicate with external equipment through the TCP/IP protocol. Data can be transferred via socket communication between the robot and an other device. In the socket communication, the robot acts as client and the other device play a role as server. URScript provides commands which open and close sockets and send/receive different data formats.

3.1.4 XML-RPC

XML-RPC is a Remote Procedure Call method that uses Extensible Markup Language (XML) to transfer data between programs over sockets. With it, the UR controller can call methods/functions (with parameters) on a remote program/server and get back structured data. By using it, a complex calculation which is not available in the command set of the robot can be performed. In addition, other software packages can be combined with URScript.

3.1.5 RTDE

The Real-Time Data Exchange (RTDE) interface provides a way to synchronize external applications with the UR controller over a standard TCP/IP connection, without breaking any real-time properties of the UR controller. This functionality is among others useful for interacting with fieldbus drivers (e.g. Ethernet/IP), manipulating robot I/O and plotting robot status (e.g. robot trajectories).
The RTDE functionality is split in two stages: a setup procedure and a synchronization loop. On connection to the RTDE interface, the client is responsible for setting up the variables to be synchronized. Any combination of input and output registers that the client needs to write and read, respectively, can be specified. To achieve this the client sends a setup list of named input and output fields that should be contained in the actual data synchronization packages. The definition of a synchronization data package format is referred to as a recipe. When the setup is complete the data synchronization can be started and paused. When the synchronization loop is started, the RTDE interface sends the client the requested data in the same order it was requested by the client. Furthermore the client is expected to send updated inputs to the RTDE interface on a change of value. The data synchronization uses serialized binary data and runs at up to 500MHz.

3.1.6 ROS/ROS2 Driver

The ROS driver is developed on top of the Universal_Robots_Client_Library and supports key cobot functionalities such as: pause at emergency stop, safeguard stop, automatic speed scaling to avoid violating safety settings, and manual speed

scaling from the teach pendant. In addition, the externalControl URCap allows the integration of ROS2 behaviors into the robot program.

3.2 Omron LD250

The LD250 is an AMR that operates on his own or is integrated within a fleet which is then managed by a so-called fleet manager. All the sensor data necessary for localization, path planning, collision avoidance, etc. is processed locally on the AMR and not available externally. However, the LD250 can be instructed to do certain tasks as well as to send status data via the Advanced Robotics Command Language (ARCL). The ARCL is a simple, text-based, command-and-response operating language for integrating the AMR with an external automation system. ARCL allows you to operate and monitor the AMR, its accessories and its payload devices over the network. Connection is established via Teletype Network (Telnet), a client/server application protocol that provides access to virtual terminals of remote systems on local area networks or the Internet. It is used as a standard TCP/IP protocol for virtual terminal service. The Telnet client, which is a Raspberry Pi 5 mounted on top of the LD250 and connected via Ethernet port, initiates the connection by sending requests to the Telnet server, which is the LD250. Once the connection is established, the client can send commands to the LD250. The server (LD250) processes these commands and responds accordingly.

4 Conclusions

There are many standards available for robot communication in industrial robotics. If robots need to collaborate together you have to be aware of these standards. Running a fleet of robots within the ecosystem of one robot manufacturer is often supported by own management software, i.e. a fleet manager that takes care about path planning of the AMRs to guarantee a smooth traffic of the AMRs within a factory floor. However, integrating robots from other manufacturers into these systems is often not possible or at high integration cost. There is effort going in standardizing communication between robots such as the ROS-Industrial Community, that provide software libraries and interfaces for common industrial manipulators, grippers, senors and device networks as part of the ROS framework. But still there is no real standard supported by all robots and peripherals by default, as it is e.g. in the computer industry.

Acronyms

AMR Autonomous Mobile Robot. 2, 8

ARCL Advanced Robotics Command Language. 8

CAN Controller Area Network. 4

CIP Common Industrial Protocol. 3, 4

CRC cyclic redundancy check. 4

Ethernet/IP Ethernet Industrial Protocol. 3, 7

GUI Graphical User Interface. 6

OSI Open Systems Interconnection. 3

P2P peer-to-peer. 4

Profibus Process Field Bus. 5

ROS Robot Operating System. 5, 7, 8

RTDE Real-Time Data Exchange. 7

TCP/IP Transmission Control Protocol/Internet Protocol. 3, 6, 7

Telnet Teletype Network. 8

UDP User Datagram Protocol. 3

UR5e Universal Robots 5e. 2, 6

XML Extensible Markup Language. 7

References

[1] *Modbus Organization Webpage*. URL: https://modbus.org/.

[2] *Omron LD 250*. URL: https://www.omron-ap.com/solutions/robotic-solutions/robotics/mobile-robot/ld250/overview/.

[3] *Open DeviceNet Vendor Association (ODVA) Webpage*. URL: https://www.odva.org/.

[4] *PROFIBUS Webpage*. URL: https://www.profibus.com/technologies/profibus.

[5] *ROBOTIQ Webpage*. URL: https://robotiq.com/.

[6] *ROS - Robot Operating System Webpage*. URL: https://www.ros.org/.

[7] *ROS-Industrial Webpage*. URL: https://rosindustrial.org/.

[8] *Universal Robots UR5e Cobot*. URL: https://www.universal-robots.com/products/ur5e/.