Collaborative robotics in ROS

Ferenc Knitlhoffer
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
knitlhoffer.ferenc@gmail.com

Károly Széll
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
szell.karoly@amk.uni-obuda.hu

***Abstract*— The paper deals with the usage of ROS middleware (Robot Operating System) in robotic systems. RT-middleware (Robot Technology Middleware) is a technology that implement elements needed to build and operate spatially distributed complex robot systems. In that manner a smart factory model was created based on full duplex communication using TCP Socket and websocket technology which was later integrated in a ROS-based environment. This paper gives an overview of robotic solutions using ROS showing its main features and benefits.**

Keywords—Collaborative robotics, ROS, embedded systems, RT-middleware

I. Introduction

The spark of today's industrial revolution is the advancement of information technology, which enables wireless connectivity between elements of production, through which channels can flow a lot of data. This revolution called Industry 4.0 includes cyber-physical systems, cloud storage, Internet of Things (IoT) and smart machines and cells (Smart Factories) [1]–[5]. Physically present production involves a system for collecting, processing and streaming data in cyberspace. These so-called cyber-physical systems combine all elements (robots, CNC machines, sensors, cameras, user phones, workpieces, etc.) involved in manufacturing into one network. These elements work together to create a 'path' for the machined part, which can be completed at optimum speed. To stabilize such a complex system, highly reliable elements are needed [6]–[8].

This kind of thinking has replaced the former centralized structure and creates a decentralized process in which each component makes its own decisions. Exceptions to this are error handling or higher priority tasks that are controlled by the central control of the system. This is an integrated approach of Industrial Internet of Things (IIoT). Flowing data is easy to read and speaks to people and machines alike. The reason is that you can make both human and machine decisions in the shortest amount of time. It also allows the operator to identify areas of production that can be improved in the knowledge of large amounts of data. The Big Data environment has been developed to collect and manage these statistics [9]. This facilitates processing large volumes of data that would be difficult to use with earlier tools.

II. Problem description

For our current education and future robotics applications, we have developed a ROS-based cyber physical unit for learning and experimental purposes, which consists of a robot arm, a conveyor, and a shape recognition camera [10]–[12]. Such a unit facilitates the presentation of engineering tasks to potential problems encountered in the engineering work and facilitates communication with our industry partners. The design was divided into two parts: one was the selection of a manufacturing process presented and executed by the cell, and the other was the assembly of the cell to accomplish the task.

Selecting such a task was difficult, as Industry 4.0 can best fit into serial production. The chosen task was to perform a quality control: the dimensional correctness and surface quality control of factory-made or supplied parts.

III. Experimental setup

A. Milestones

The main steps are listed below that we focused during the implementation of the experimental setup:

- Assessing, ordering, and purchasing hardware and software needs
- Layout design
- Motor control by microcontroller
- Server for network communication
- Network communication establishing a client-server connection
- Robot programming
- Creating a Graphical User Interface (GUI)
- Modeling, assembly and control of conveyor
- Implementation of camera image processing
- Implementation of the demonstration task

B. Main units

The smart cell is the sum of several units with decision-making power in the spirit of Industry 4.0. The more of such subsystems we have, the harder the task of coordinating communication becomes, but the higher the level of automation can be achieved. The experimental setup can be divided into three larger units, along with other peripheral elements. For example, we put the conveyor in the latter category because the conveyor does not have a control unit on its own, so it cannot make its own decisions on incoming data.

The first separable logic unit demonstrates a collaborative workspace [13]–[15]. It is a group of one robot and two cameras. The robot is a Universal Robots UR3 collaborative robot. The robot is equipped with an RG-2 gripper that allows the robot to manipulate items weighing less than 2 kg. There is a SICK 2D Vision Inspector PIM60 industrial camera mounted on the robot that can perform tasks ranging from shape recognition to size control. We assigned a Keyence camera and its processing unit also to this group (CV-X152 processing unit and CA-LH16 camera with CV-200C lens). They form a group in terms of network connections.

The second unit consists of a Raspberry Pi Model B and two motors controlled by it. Two low-torque (0.4 Nm) stepper motors are controlled by the Raspberry Pi, through a dedicated Adafruit HAT module. This Adafruit PCB is a

Raspberry compatible controller that can control two stepper motors via two TB6612 drivers.

The third logically separable unit is the server that establishes the communication channels and the GUI.

These elements are supplemented by components that are either used as tools or do not have independent decision-making control. There are 3D printed elements that perform various functions, such as the barrier elements introduced later or the prototype parts that attach the motors to the conveyor.

C. *Collaborative robot arm and image processing*

The robot's job is to signal the arrival of the workpiece and to place it. The states shown in the flowchart (see Fig. 3) are illustrating of the robot's separable functions. Accordingly, this also determines the structure of the robot program. The program is based on a large 'switch case', which means that the robot can be switched between different states. Different inputs cause it to enter a different phase. The process is divided into a total of nine parts (case 0 to case 8), except for the stop button cases. Each is a combination of some kind of movement and network messaging. We have created subroutines for receiving and sending messages, which are called when a robot sends or receives a signal. Programming has revealed that the robot (and the Keyence industrial camera that performs quality control) does not communicate with websocket technology, but with older TCP protocol-based socket communication. Thus, there are two endpoints on the server, one for the websocket and one for the TCP socket connection. The robot program can receive the effects of the stop buttons at any given moment, since we have created a thread to examine the incoming message. If one of the stop buttons is pressed, the robot program switches to the appropriate state within the case structure (case 11 to case 13). The robot program is written in UR Script language. We put the Keyence industrial camera in the logic unit of the robot and the robot-mounted camera.

D. *Raspberry Pi and stepper motors*

The task of the microcontroller is simple, it must control two stepper motors. This is done by a signal converter driver, the unit referred to as Adafruit HAT. Programming is done in Python. The first program is connected to the websocket server and the second program is responsible for making various motions with the motors as a result of the input parameters.

To transfer workpieces from the conveyor to the image processing site, an arm extends onto the assembly line and rotates the workpiece onto a table. As the performance of the motors is relatively small, we encountered problems caused by the friction [16]–[20].

As the critical surface is the cylindrical side of the workpiece, the workpiece must be rotated for lateral image processing so that it can be examined from all directions. The two motors move two arms which are symmetrically located on both sides of the conveyor belt. When closed, the tongue at the end of one arm slides into the gap formed on the other and closes 90 °. Its function is that when the workpiece is sensed by the websocket message, the arms are locked in an open position and hold current in this position by passing current through their coils. The workpiece strikes the arms and positions itself at the apex of the angle defined by the two arms. The robot then takes the component to the image processing site and the levers reopen after a message from the robot indicates that the component has been removed from its hold. The advantage of this solution is that the image processing can be carried out in a larger space designed for it..

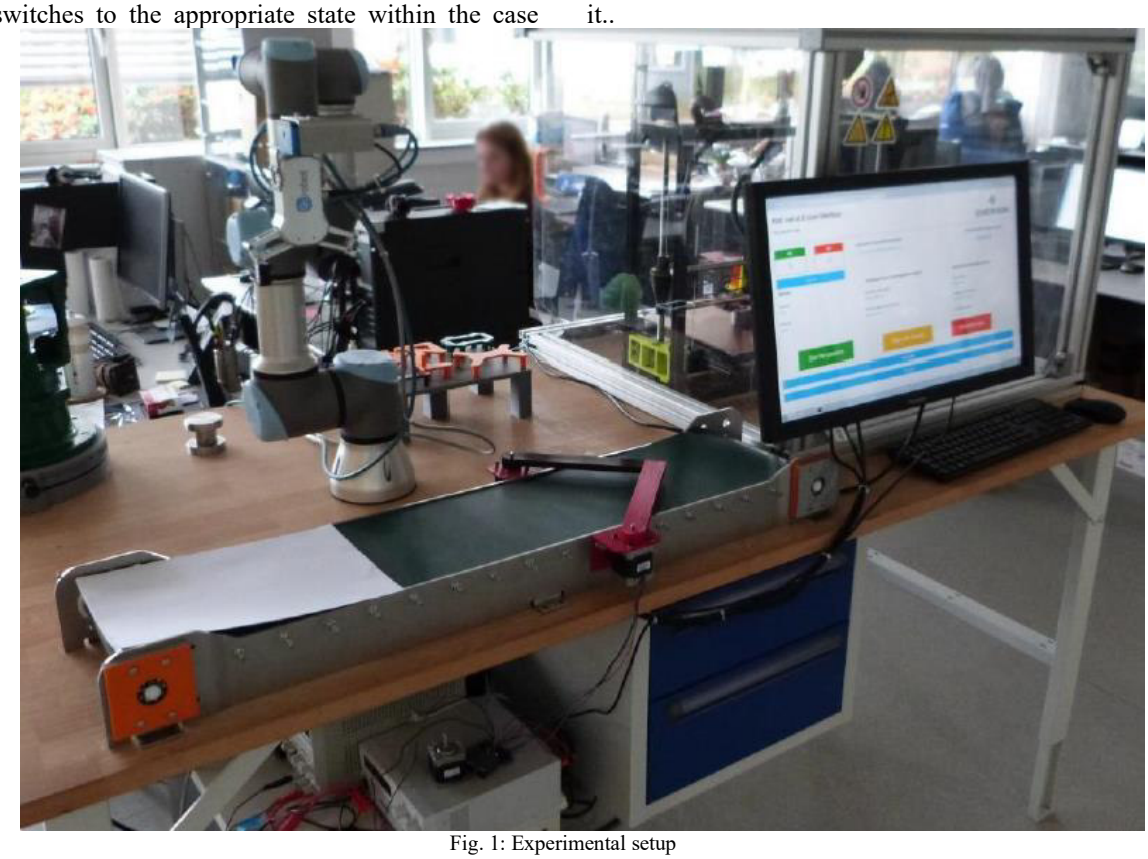

Fig. 1: Experimental setup

E. *Network communication*

One of the most important parts of Industry 4.0 is the communication platform where cell components can send and receive messages. A diagram of network elements (see Fig. 2) shows the color-coded connections. The server for the communication is shown on the right of the diagram. The figure shows two wires of different colors connected to the PC, but in reality, one Ethernet cable connects to the switch. The reason for this is that the robot and the Keyence camera are not capable of websocket technology, but only communicate as TCP sockets. Because of this, the server has two endpoints, which means that the same single server is capable of websocket and TCP socket communication. Since most of the network is capable of websocket messaging, the problem of two types of messaging is solved by converting an incoming TCP socket message into a websocket message and sending it straight away. This is fast and simplifies communication, with no need for two interfaces on each client.

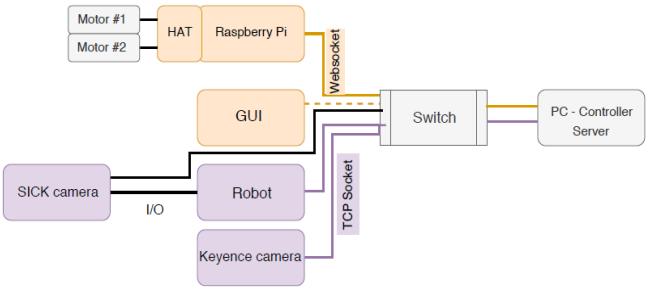

Fig. 2: Communication diagram

The function of the switch is to establish connection between the components. Beside the PC, the robot, Keyence camera, Raspberry and SICK camera are connected to the switch. Indirectly, the graphical user interface is attached to it as a client, but it is directly connected to the server on the PC, but if someone opens the interface in another browser, this connection is made via a switch. The Keyence camera is is not directly connected to the PC. Keyence provides a software option for creating a TCP socket connection point so that it can be connected directly to a robot.

The messages exchanged by the elements are JSON text messages (string). JSON is a language that can be easily transformed into an object in any major programming language, and its structure is easy to read by the human eye. At the beginning, the sender and the nodes to which a given message is intended are recorded. Then there is the message itself, broken down into components. The 'operation' section is a one-word short description that gives the message content to the GUI. In each section, the data to be processed by them. At the end, a 'connection' is a number that indicates whether the message conveys the connection of an element up or down.

The server is designed in a minimal way, only performing tasks that are necessary for the cell to function. An important part, however, is the termination of improperly closed connections, as this may provide false data to the operator. Periodically, the system checks all connected devices and kills the connection if there is no feedback.

Only the robot is connected to the server's TCP socket communication endpoint. The robot has the ability to establish multiple socket connections and has been used to connect to the Keyence camera. The server is also responsible for logging messages. Every message exchanged is also sent to the server and written to a text file using a separate program. By inserting the date, the operator can see at what time the messages were exchanged on the websocket channel. Similarly, the system records the current good and bad workpiece numbers by an interface button.

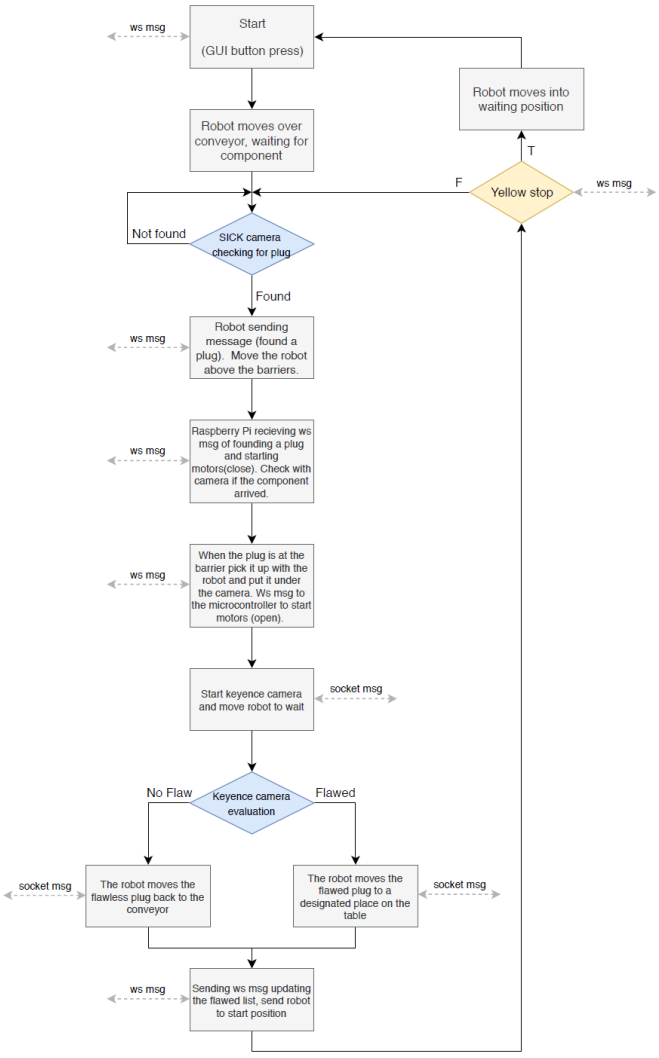

Fig. 3: Flowchart of the experimental setup

F. *Graphical user interface*

The communication interface between the operator and the cell is implemented by the GUI, which is a graphical user interface on the touch screen. The plans for implementing the GUI were minimal enough, the key points we wanted to achieve were start and stop buttons, ease of use, and feedback on message processing.

The web page has HTML structure with CSS and JavaScript elements. The GUI consists of two pages and a third "live" page showing a live image of the camera mounted on the robot. The main page has a brief introduction to the cell. The generic start stop buttons are also located in the

middle of the screen and the navigation buttons for the other two pages.

The second page is a surveillance page ('Data observation page'), where you can find detailed information about the cell. Highlighted and color coded in the upper left corner, the result of the quality control so far, passed and failed. In addition, a brief description of the type of defect in the case of a defective component. This is a development option, and further analysis of the camera image can automatically recommend how to handle a defective component. Highlighted in a slightly larger font at the top right, a description of the type of message processed by the GUI on the last websocket channel. This was a useful feedback during the design process, and the imaginary operator can see what processes are taking place in the background. Below the good and bad counters is a 'Save Data' button, which the operator can use to save the countdown so far. This is to allow multiple users to log their activity. It currently saves the data in a text file with a '.txt' extension but can be expanded with a login system. In the middle of the screen there is general live information about each component. Next to these are the buttons and a status block at the bottom of the screen. The latter is a brief, one-on-one feedback of the current state of the system.

The purpose of the interface is to provide general information and to start and stop the process. In the first plans, we designed two stop buttons. The yellow 'Stop when finished' button only stops the system after the quality-controlled part has finished. The red 'Emergency stop' button, as its name implies, immediately stops the process. This button is split into two parts and after pressing it, one of the following options can be selected: the process resumes from the moment of stopping or the process will cancel the advance of this part so far and will restart.

IV. Experiment

The main task is to demonstrate a surface quality control of a component with an industrial camera. Starting the process on the interface, the robot moves over the conveyor belt and the SICK camera takes a picture at short intervals. In the program that comes with the camera, it runs a round detection to detect the cylindrical part (which, viewed from above, is a circle) passing through the conveyor. The arrival of the workpiece is indicated to the other units and the robot moves over another position on the conveyor. Parallel to this, Raspberry will start the two motors connected to it, with two levers attached. They close in a barrier manner and then hold them in this position, guiding the current flowing through their coils until the component is positioned in the designated position. This is recognized by the SICK camera using another circular recognition algorithm. The component is then moved to the image processing site. After image processing, depending on whether the part was defective, the part is moved to different locations. The good part is returned to the conveyor, while the bad part is placed in a designated location. The interface then updates the number of good and bad parts and restarts the process from the beginning. In the GUI, the operator can stop the process in several ways.

V. Conclusion

In this paper a robot, an industrial camera, a conveyor belt, a microcontroller and a computer have been integrated into one system to perform a quality control demonstration task. The purpose of the cell is not to integrate it into direct production but demonstrate the advancement of information and operational technology. After studying the cell, one becomes familiar with the spirit of Industry 4.0 and the clear benefits of using Smart Factories and RT-middlewares.

VI. Acknowledgement

The authors wish to thank the support to the Arconic Foundation.

VII. References

[1] M. Hermann, T. Pentek, and B. Otto, “Design Principles for Industrie 4.0 Scenarios,” in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 3928–3937.

[2] D. Vuksanović D. ,. Ugarak, J. ,. Korčok, “Industry 4.0: the Future Concepts and New Visions of Factory of the Future Development,” pp. 293–298, 2016.

[3] D. R. Sjödin, V. Parida, M. Leksell, and A. Petrovic, “Smart Factory Implementation and Process Innovation,” *Res.-Technol. Manag.*, vol. 61, no. 5, pp. 22–31, Sep. 2018.

[4] Y. Liu, M. Kashef, K. B. Lee, L. Benmohamed, and R. Candell, “Wireless Network Design for Emerging IIoT Applications: Reference Framework and Use Cases,” *Proc. IEEE*, vol. 107, no. 6, pp. 1166–1192, Jun. 2019.

[5] A. Selmeci and T. Orosz, “Usage of SOA and BPM changes the roles and the way of thinking in development,” in *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics*, 2012, pp. 265–271.

[6] G. Györök and B. Beszédes, “Adaptive Optocoupler Degradation Compensation in Isolated Feedback Loops,” *2018 IEEE 12th Int. Symp. Appl. Comput. Intell. Inform. SACI*, pp. 167–172, 2018.

[7] G. Györök and B. Beszédes, “Fault tolerant power supply systems,” in *11th International Symposium on Applied Informatics and Related Areas (AIS 2016)*, 2018, pp. 68–73.

[8] G. Györök and B. Beszédes, “Highly reliable data logging in embedded systems,” in *2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2018, pp. 49–54.

[9] É. Hajnal, “Big Data Overview and Connected Research at Óbuda University Alba Regia Technical Faculty,” in *AIS 2018 - 13th International Symposium on Applied Informatics and Related Area*, 2018, pp. 1–4.

[10] S. Mokaram *et al.*, “A ROS-integrated API for the KUKA LBR iiwa collaborative robot,” *IFAC-Pap.*, vol. 50, no. 1, pp. 15859–15864, Jul. 2017.

[11] Z. GUO, W. YANG, M. LI, X. YI, Z. CAI, and Y. WANG, “ALLIANCE-ROS: A Software Framework on ROS for Fault-Tolerant and Cooperative Mobile Robots,” *Chin. J. Electron.*, vol. 27, no. 3, pp. 467–475, 2018.

[12] S. Cousins, “Welcome to ROS Topics,” *IEEE Robot. Autom. Mag.*, vol. 17, no. 1, pp. 13–14, Mar. 2010.

[13] G. Anand, E. S. Rahul, and R. R. Bhavani, "A sensor framework for human-robot collaboration in industrial robot work-cell," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 2017, pp. 715–720.
[14] G. Dumonteil, G. Manfredi, M. Devy, A. Confetti, and D. Sidobre, "Reactive planning on a collaborative robot for industrial applications," in *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2015, vol. 02, pp. 450–457.
[15] J. Van den Bergh, F. Cuenca Lucero, K. Luyten, and K. Coninx, "Toward specifying Human-Robot Collaboration with composite events," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016, pp. 896–901.
[16] C. Budai, L. L. Kovács, J. Kövecses, and G. Stépán, "Effect of dry friction on vibrations of sampled-data mechatronic systems," *Nonlinear Dyn.*, vol. 88, no. 1, pp. 349–361, Apr. 2017.
[17] C. Budai, L. L. Kovács, J. Kövecses, and G. Stépán, "Combined effects of sampling and dry friction on position control," *Nonlinear Dyn.*, Jul. 2019.
[18] C. Budai, L. L. Kovács, and J. Kövecses, "Combined Effect of Sampling and Coulomb Friction on Haptic Systems Dynamics," *J. Comput. Nonlinear Dyn.*, vol. 13, no. 6, Jun. 2018.
[19] C. Budai and L. L. Kovács, "On the Stability of Digital Position Control with Viscous Damping and Coulomb Friction," *Period. Polytech. Mech. Eng.*, vol. 61, no. 4, pp. 266–271, Sep. 2017.
[20] C. Budai and L. L. Kovács, "Friction Effects on Stability of a Digitally Controlled Pendulum," *Period. Polytech. Mech. Eng.*, vol. 59, no. 4, pp. 176–181, Oct. 2015.