sciendo

Műszaki Tudományos Közlemények vol. 17. (2022) 60–64.
DOI: Hungarian: https://doi.org/10.33895/mtk-2022.17.13
English: https://doi.org/10.33894/mtk-2022.17.13

EME MTSZ MTK

STUDYING THE COMMUNICATION POSSIBILITIES BETWEEN A COLLABORATIVE ROBOT AND AN INDUSTRIAL COMPUTER

Szabolcs Balázs TÓTH,[1] Timotei István ERDEI,[2] Géza HUSI[3]

University of Debrecen Faculty of Engineering, Debrecen, Hungary

[1] *szabolcs978@gmail.com*

[2] *timoteierdei@eng.unideb.hu*

[3] *husigeza@eng.unideb.hu*

Abstract

This project is based on the study of collaborative robots (hereafter: Cobot) and industrial PCs, including the communication possibilities between them. In the course of the project, a number of possibilities were studied in order to solve the problem and then summarised to select a collaborative robot and an industrial PC. After that, the focus of the project was on the communication options between the two systems: communication via I/O ports and net-work communication (TCP/IP, Modbus/TCP). Finally, the possible solutions were analysed from different points of view (complexity, flexibility, technical aspects, etc.) and communication between the chosen devices was implemented. The implementation was mainly carried out in a simulation environment, but as the project progressed a communication solution was tested on real industrial devices.

Keywords: *KR, Cobot, UR, PLC, IPC, Beckhoff, TCP/IP, Modbus/TCP.*

1. Introduction

Robots are becoming more and more important in industry, so a large part of the work is no longer done by humans. However, there are still tasks where human interaction is required. This is why Cobots have been developed [1].

There are many ways to communicate with a robot. Robots are most often controlled by wired, wireless or autonomous methods [2].

The goal of this research was to implement a communication solution between a Cobot and an industrial controller.

2. Selected industrial controller and robot

After examining the main features of collaborative robots and industrial controllers, one of each was selected. For the controller, the Beckhoff CX5140 Embedded PC was chosen, shown in Figure 1. With the CX series of embedded PCs, Beckhoff combines PC technology with modular I/O level. It also provides a range of communication options such as TCP/IP, which fulfils the requirements set at the beginning of the project.

The Cobot chosen is the UR10 Cobot shown in Figure 2, which has several communication options including support for TCP/IP communica-

Figure 1. *CX5140 Embedded PC.* [3]

Figure 2. *UR10 Cobot.* **[4]**

tion protocol, making it an excellent subject for the task. Its programming is also relatively simple and easy to learn. UR10 is designed for larger tasks where accuracy and reliability are a priority. With the UR10 collaborative robotic arm, we can automate larger processes and tasks with loads up to 10 kg. With a reach radius of up to 1300 mm, the UR10 Cobot is designed to be more efficient for tasks over larger areas **[4]**. The programming of both devices is easy to learn, although the industrial controller requires some prior knowledge.

3. Communication possibilities between the selected systems

Three communication options were studied. TCP (transmission control protocol) is the Internet standard that ensures the successful transfer of data packets between devices over a network. TCP works with the Internet Protocol (IP) to determine how data is exchanged over the Internet. The two protocols are collectively referred to as TCP/IP **[5]**.

Modbus TCP is nothing more than sending a Modbus RTU message wrapped in an Ethernet packet, but instead of a serial connection, the message is transmitted over a network. TCP/IP provides the transmission medium for Modbus TCP messages. Simply put, TCP/IP allows the exchange of binary data blocks between automation devices **[6]**.

I/O modules are the simplest way to communicate. The TCP/IP communication protocol, which is server/client based, is best suited to the flexibility of the task.

4. TCP/IP server and client program

In order to implement communication between the two industrial devices in the server/client system, a wired LAN communication was required. The industrial controller was selected as server and the robot as client. The IP addresses of the devices have the same network segments, the server IP address is 192.168.2.100 and the client IP address is 192.168.2.113. This results in a subnet mask of 255.255.255.0, which is in class C of the IPv4 address class.

The embedded PC program was written in Beckhoff's own development environment in TwinCat 3 in ST. The program for UR10 was written in URSim Offline Simulator in the programming language of URScript.

Figure 3 shows the program part of the server states. State 0 is the listener state, when the server is running and waiting for incoming connections. State 1 is set up when a successful connection is established, and from there it enters state 2, i.e. message exchange. After sending a message, the program enters state 3 and then state 4 after the server is closed.

Figure 4 shows a part of the client program. When writing the client, a 'connection' and a 'server' Boolean variable was selected in the BeforeStart segment. The 'server' variable is set to TRUE as soon as the connection is established with the server. Then after a successful connection, the client sends a message to the server and enters the message send/receive state. Based on predefined messages, the UR client can perform various tasks and due to the peculiarity of TCP/IP, messages are delivered in the order they are sent.

5. Operation of server/client connection

The flowchart in **Figure 5** shows how the connection works.

The server is first set up and started. Then, when the server is waiting for the client to connect, the client program is started and within a few seconds the connection is established (assuming all settings are correct).

Figure 6 shows a TCP/IP connection implemented in a simulation environment. Once the connection is established, the server can send and receive messages from the client. This is true back and forth. **Figure 7** shows the tasks that the robot client can perform based on the instructions from the server.

```
CASE nState OF
0: (* Listening State *)
    bReceive := FALSE;
    IF eState = eSOCKET_SUSPENDED THEN
    sMessage := 'Kapcsolatra vár...';
    MEMSET(ADR(sReceivedData),0,SIZEOF(sReceivedData)); (* Delete Received data *)
    ELSE IF eState = eSOCKET_CONNECTED THEN
        nState := 1;
    END_IF
    END_IF
1: (* Connection State *)
    IF NOT fbServer.bBusy THEN
            IF NOT fbServer.bError THEN
                IF eState = eSOCKET_CONNECTED THEN(* Connected *)
                    sMessage := 'Kapcsolat létrejött!';
                    bReceive := TRUE;
                    nState := 2;
                END_IF
            END_IF
    END_IF
2:(* Exchange data State *)
            (* ---------------------- Receive data ---------------------------------- *)

        IF NOT fbSocketReceive.bBusy AND NOT fbSocketReceive.bError THEN
            IF fbSocketReceive.nRecBytes = 0 THEN
                bDisableSendButton := TRUE; //Disable Send Button
                //IF there's no data received, reset the fbSocketReceive
                bReceive := TRUE;
            ELSE
                bDisableSendButton := FALSE; //Unlock Send Button
            END_IF
        END_IF
        (* ---------------------- Send data ---------------------------------- *)
        IF bSEND THEN
            bReceive := FALSE;
            fbSocketSend.bExecute := TRUE;
                MEMSET(ADR(sReceivedData),0,SIZEOF(sReceivedData)); (* Delete Received data *)
            nstate := 3;
            bSEND := FALSE;
        END_IF
3:(* Send data State *)
            MEMSET(ADR(sSentData),0,SIZEOF(sSentData)); (* Delete Sent data *)
            fbSocketSend.bExecute := FALSE;
            IF NOT fbSocketSend.bBusy AND NOT fbSocketSend.bError THEN
                fbSocketReceive.bExecute := TRUE;
                nstate := 1;
        END_IF
4: (* Disconnect *)
    IF bEnable THEN
```

Figure 3. *Part of the server program.*

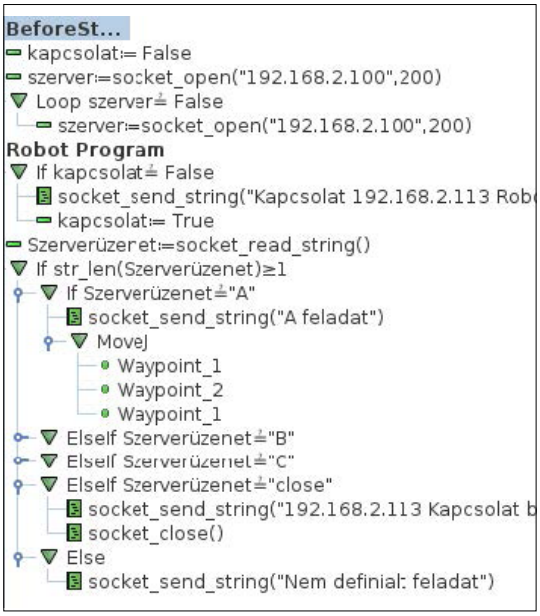

Figure 4. *Part of the client program.*

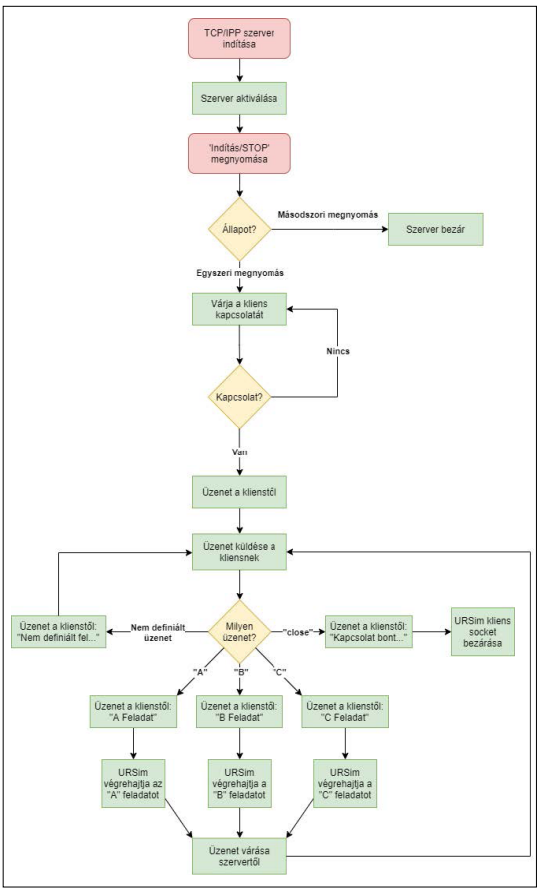

Figure 5. *Server/client connection flowchart.*

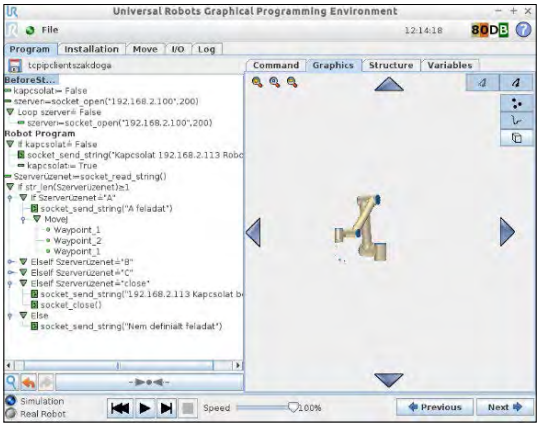

Figure 6. *Connection in simulation environment.*

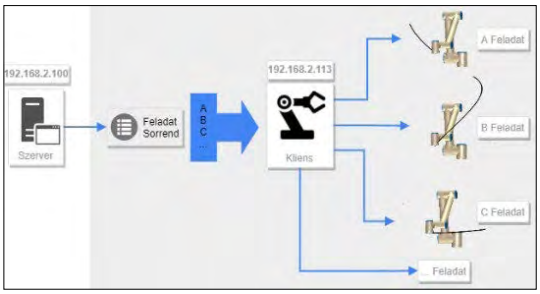

Figure 7. *Robot tasks.*

The client program can now perform 3 different tasks besides closing the socket and sending the "undefined task" message. It can receive an "A", "B" or "C" message and execute the task accordingly.

This configuration is created by directly connecting the two devices. However, in some cases, the communication can be improved by adding a router if connection of multiple clients to the same server is required.

The implemented communication was tested in real life with a UR10 Cobot and a CX5140 industrial controller at Vitesco's Debrecen premises.

6. Related developments

One of the big advantages of TCP/IP communication protocol is that the server program can easily be extended, so it can be written to connect to multiple clients. However, only a two client solution has been developed here for now. This is an important development because it allows 2 robots to work collaboratively in a workspace at the same time on a given workstation. Furthermore, the position of each robot can be read and sent to the other robot. Taking this development further,

a collaboration is created in which not only the human works with the robot, but also the robot works with another robot by enabling the two devices to communicate with each other through an intermediate TCP/IP server. In this way, industrial work can be done more securely.

In the simple Pick and Place task shown in **Figure 8** two Cobots work together with the operator. The TCP/IP communication solution allows a fast exchange of data between the two robots and the operator can track the task via the server HMI..

The HMI of the server shown in **Figure 9** allows the operator to give various instructions to the client.

From the user interface of client 2 shown in **Figure 10** we can retrieve the robot's position with commands and later return the robot to that fixed state. This solution gives room for further possibilities which this research plans to implement in future studies.

7. Conclusion

The goals set for the project have been met and are explained in the project report. The communication solution has been implemented in a simulation environment and tested on real industrial equipment. As a further development, the implementation of a more secure collaborative space was realised.

Acknowledgements

I would like to take this opportunity to thank Ádám Kamrás Beckhoff, software engineer at Beckhoff, for helping me with my work. Ádám Kamrás's professional knowledge was a great help for me to understand how Beckhoff IPCs work.

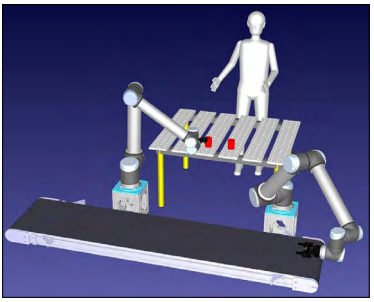

Figure 8. *Collaboration between human and robot.*

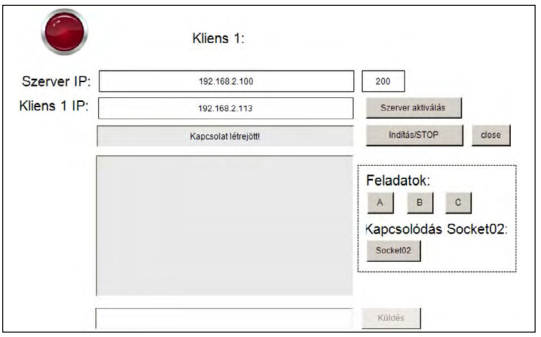

Figure 9. *Server HMI, client 1.*

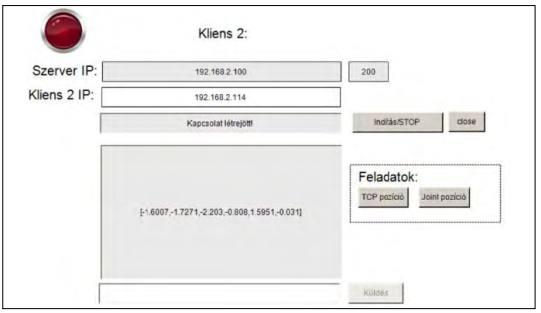

Figure 10. *Server HMI, client 2.*

References

[1] Kollaboratív robotok (2021.09.20) https://www.mmk.hu/informaciok/hirek/kollaborativ-robotok

[2] Kulcsár B.: *Robottechnika*. LSI Oktatóközpont, 1998.

[3] Beckhoff Automation GmbH (2021.09.28) https://www.beckhoff.com/hu-hu/products/ipc/embedded-pcs/cx5100-intel-atom/cx5140.html

[4] Universal Robots (2021.09.28) https://www.universal-robots.com/cb3

[5] Casad J.: *TCP/IP in 24 Hours*. Sams Teach Yourself, Pearson Education (US), 2017.

[6] Real Time Automation (2021.10.04) https://www.rtautomation.com/technologies/modbus-tcpip