Remote control of Mitsubishi industrial robot

[1]*Peter PAPCUN (1st year)*, [2]*Matej ČOPÍK (2nd year)*
Supervisor: [3]*Ján JADLOVSKÝ*

[1, 2, 3]Dept. of Cybernetics and Artificial Intelligence, FEI TU of Košice, Slovak Republic

[1]peter.papcun@tuke.sk, [2]matej.copik@tuke.sk, [3]jan.jadlovsky@tuke.sk

***Abstract* — This article discuss about remote control of industrial robotic arm Mitsubishi MELFA RV-2SDB. We are going to control this robot through smart phone with Android operating system. Our system has ability to control robot's endpoint in Cartesian system axes, or control it's joints. Smart phone is used as a joystick, and also it can be used to make or run recorded sequences.**

***Keywords* — Industrial robot, Robot, Mitsubishi, Control, Bluetooth , Wi-Fi, Android, Smart phone.**

I. Introduction

Under the phrase remote control is meant wireless control through Bluetooth, or Wi-Fi. In this document we write about network connections between nodes and methods of their communication. We describe all applications used in our robot control. Our control architecture consists of five network nodes (personal computer, Wi-Fi router, robot controller, robot, Smart phone) and three applications. Finally, we will evaluate remote control advantages

II. Distribution of system

Distributed control system (DCS) is a control system, usually production system, process or any dynamic system in which the system elements are not placed centrally but they are distributed, divided into smaller parts, subsystems that are controlled by one or more control devices. Current industrial information and control systems utilize mainly hierarchic (pyramidal) architectures containing physical and logical distribution elements, integration as a whole, open and scalable. Intelligent features have been applied on a large scale recently whereby direct hierarchic relations are turned into network relations. Emergent trends have also started to appear to a great degree, i.e. merging of originally independent systems, which can result in their new features generation as a whole. [4]

You can see hierarchic architecture of described system on Fig. 1. This architecture does not include all levels of DCS. You can read [4] for more information about DCS. This system pertain a program distribution. Program distribution divides control software to more control units. Software is distributed into robot controller, personal computer and smart phone. Each of them plays specific role in this system.

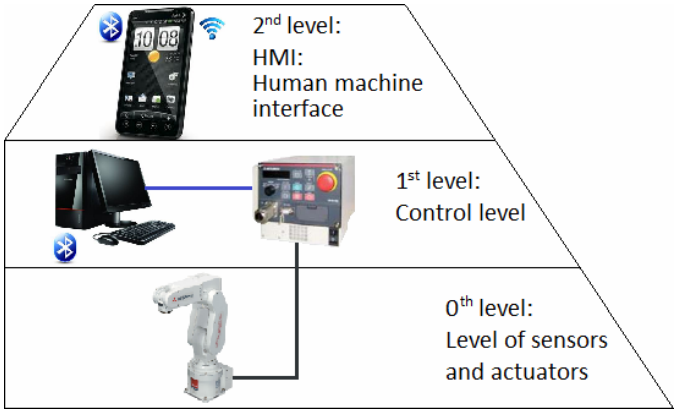

Fig. 1: Hierarchic architecture of described system.

III. Hardware and network conectivity

Diploma thesis [1] present control of robot integrated in flexible assembly company. This control system is also divided into three applications. The first application recognizes images. It operates on personal computer (PC). The second program controls whole production of flexible assembly company. Application operates on programmable logic controller (PLC). The third application controls industrial robot and operates on robot controller (RC). In this article we devote robot separately (robot is not integrated into flexible assembly company here).

We will describe used hardware and network connectivity in this part of document. Robot is connected to RC with machine cables. RC is connected to Wi-Fi router through Ethernet cable. PC uses same router as well. You can see a network connection structure of all hardware parts in Fig. 2.

You can find a more accurate description of robot and RC in [1], [2]. PC has Bluetooth device. In PC is installed operating system (OS) Windows 7. Smart phone (SP) have Bluetooth and Wi-Fi devices integrated with OS Android.

Robot is programmed by integrated development environment RT Toolbox 2 with programming language MELFA V (MELFA IV can be also used). RT ToolBox 2 communicates with RC through USB, TCP/IP, or RS – 232 interfaces. We use TCP/IP interface, in this case. Earlier we connected to RC by USB interface, because we had to change default robot IP address to static network address. Then we disconnect USB and we tried to connect TCP/IP (Ethernet cable). Connection was successful. PC already belongs to network. We configured Wi-Fi router, so that other devices can connect to it using Wi-Fi.

Fig. 2: Network connectivity.

IV. SOFTWARE

Robot control consists of 3 separate applications:

- Program in RC,
- Program in PC,
- Program in SP.

Program in RC

The simplest algorithm is applied in RC:

```
Open "ENET:192.168.0.2" As #1
Mxt 1,1,50
Hlt
End.
```

Command MXT is one of loop commands. This command is specific. It is expecting User Datagram Protocol (UPD) packet from specific IP address. You can see three MXT arguments. The first argument is communication channel number. The second argument is position data type (Cartesian system, joint, motor pulse). The last argument is filter time constant. Flowchart for previous algorithm looks is on Fig. 3.

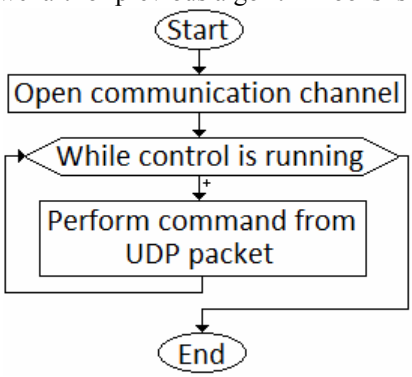

Fig. 3: RC flowchart.

Data of UDP packet (without Ethernet, IP, UDP headers and Ethernet Trailer) consists for example of: command, transmission and reply data type designation, position data, timeout time counter value, etc. Structure of UDP packet (its data part) is determined by Mitsubishi. When RC receive UDP packet, robot will move accordingly to packet instruction. In other words, an MXT is command for real time control of robot.

PC application

We used C# programming language. The following figure (Fig. 4.) shows the high level app diagram:

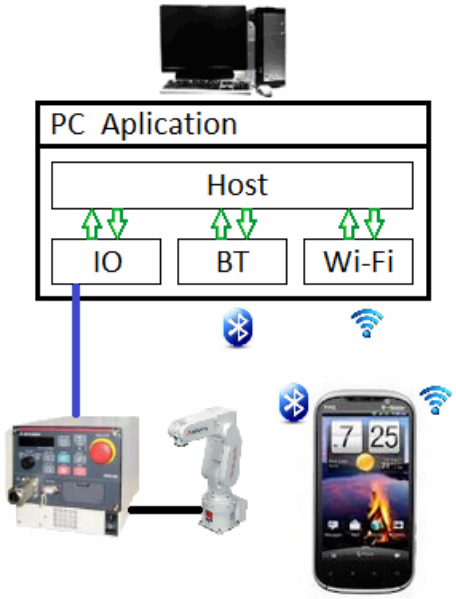

Fig. 4: Program diagram.

Main and basic application module is Host. You can load plug-in modules to host. IO, BT and WiFi are plug-in modules. Plug-ins implements custom plug-in interface in order to be "plugged into" host module.

The main task of Host is to maintain plug-ins in memory and inform them about events, which user invokes in main application (language change, request plug-in information, etc.). Another task of Host is switching between displayed plug-ins, as required by user. When user has switched between plug-ins, instances of plug-ins run in background and their functionality is not compromised in any way.

IO is next plug-in module. Its task is communication with robot. The IO module obtains information about robot (arm position, joint rotation, etc.) and this information is passed over through host to special global class. This class maintains current information about robot. Main task is to execute orders from command queue (type FIFO – first in first out) which is also part of global class. Other plug-ins write command to command queue and IO takes care of sending and performing those commands in robot.

Plug-ins BT and WiFi communicate between SP and plug-in IO. BT communicates with Bluetooth interface. We are using standard communication protocol RFCOMM. WiFi communicates through TCP/IP interface. We are using UDP in WiFi. This plug-ins behaves as server, which are waiting for client to connect through Bluetooth interface or WiFi. Application obtains connection information about remote client, after accepting connection. Then communication is running between SP and PC. You can control robot directly. Robot coordinates are sent to SP (by IO plug-in) and SP is

sending new coordinates. You can record and activate command sequence from SP also. Mobile phone sends command for sequence to module BT or WiFi. Or you can teach new sequences. By mobile you confirm some points, these points are saved to file in computer and then you can activate those new sequences from file.

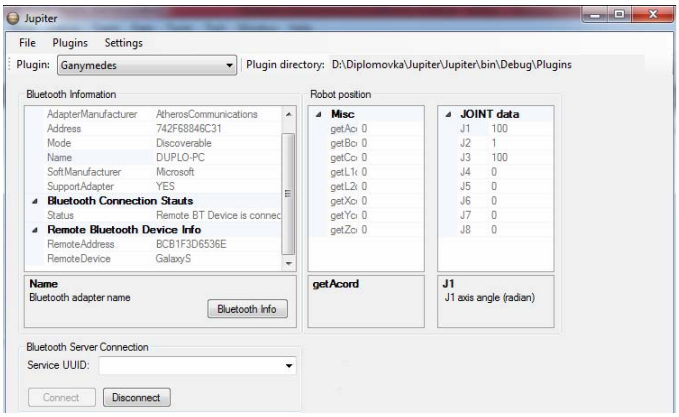

Fig. 5: PC application.

Application in SP

App is programmed in Java programming language using integrated development environment (IDE) Eclipse Galileo 3.5.2. Our app is then sent to SP through this IDE. You can see basic and the first screen of application on figure (Fig. 6).

Fig. 6: Basic screen of program in SP.

RobotControl Activity (Fig. 7) is main activity. User can receive information from PC about robot position through this activity. User can define every coordinate independently. He can choose position data type (Cartesian system, joint, motor pulse), which he sends to PC. PC (module IO) sends this information to RC. Manual control is next option of by arrow buttons (Fig. 7).

Fig. 7: RobotControl Activit (SP screen).

User activates sequences from next part of app. List of sequences is on screen called with same name. Those sequences are saved in PC (in files). User has option to teach robot new sequences, with other part of application. User confirms some points (in Cartesian system), this points are saved to file in computer. Then users can active new sequence through Sequences activity.

V. Conclusion

We tried this type of control. Every part of applications are working. SP replace Teaching pendant, but not completely. Because user can program robot with Teaching pendant.

With SP we can:

- control robot with arrow buttons,
- send robot to a chosen coordinate,
- rotate joints to a chosen axis,
- active sequences,
- teach robot new sequences.

With SP we can not:

- program robot,
- teach robot a sequences with define type of movement.
- set RC parameters.

We can program application, which can do mentioned things (without set RC parameters). Purpose was not to create remote Teaching pendant in SP. We are only trying option of controlling robot in C# with real time control (command MXT in RC). For example one of diploma thesis in this year (in our department) is about real time control of robot through cameras system with infra sensor. You can control robot with your body movement in application of this diploma thesis.

ACKNOWLEDGMENT

This work is the result of the project implementation: Development of the Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120030) supported by the Research & Development Operational Program funded by the ERDF.

This work has been supported by the Scientific Grant Agency of Slovak Republic under project Vega No.1/0286/11 Dynamic Hybrid Architectures of the Multiagent Network Control Systems.

REFERENCES

[1] PAPCUN, P. 2011. Control of robot integrated in flexible production line, diploma thesis, Košice, Slovakia, 2011.
[2] Instruction manual, CRnQ/CRnD Controller, Mitsunishi Electric, Ratingen, Germany, 2010.
[3] Product leaflets, RV-2SDB, Mitsubishi Electric, Ratingen, Germany, 2010.
[4] JADLOVSKÝ, J. – LACIŇÁK S. – CHOVAŇÁK J. - ILKOVIČ J. 2010. Proposal for distributed control system of flexible production line, Journal of Cybernetics and Informatics, vol. 11, Košice, Slovakia, ISSN: 1336-4774
[5] ILKOVIČ, J. – ČOPÍK, M. – JADLOVSKÝ, J. – LACIŇÁK, S. 2011. Technological level of flexible manufacturing system control, Acta Electrotechnica et Informatica, vol. 11, no. 1, pages 20 – 24, Košice, Slovakia, ISSN: 1338-3957
[6] ILKOVIČ, J. – ČOPÍK, M. 2011. The assembly line model at Department of Cybernetics and Artificial Intelligence, SCYR 2011: 11th Scientific Conference of Young Researchers of Faculty of Electrical Engineering and Informatics Technical University of Košice: proc. - Košice : FEI TU, 2011, pages 373-376, Košice, Slovakia, ISBN 978-80-553-0644-5