International Conference Cybernetics & Informatics 2022
Visegrád, Hungary, September 11th - September 14th, 2022

Monitoring of Discrete-event System Controlled by Revolution Pi Using 3D Engine

Erik Kučera, Oto Haffner and Dominik Janecký
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
Email: erik.kucera@stuba.sk

***Abstract*—In this paper, we will address the issue of digitalisation of various production processes, using modern information and communication technologies. This trend falls under the fourth industrial revolution, Industry 4.0. One of the challenges is the creation of advanced human-machine interfaces (HMIs) for cyber-physical systems. New forms of interfaces can be implemented in augmented reality. The paper therefore deals with the connection of a discrete-event system with 3D engine Unity. 3D engines are used to develop augmented reality applications.**

***Index Terms*—Unity engine, augmented reality, mixed reality, plc, mechatronics**

I. Introduction and Related Works

Augmented reality (AR) could be called one of the biggest technology trends of our time, and it continues to expand [1]. One reason for this is that smartphones and other devices that support augmented reality are much more affordable. The most commonly used applications nowadays make use of a smartphone's camera, which the user can use to view the real world on the screen of a smartphone, or other device that supports this feature, and then rely on an augmented reality application to enhance this real world in a number of ways through various digital overlays, such as [2]:

- Adding routes in real time
- Overlaying images, digital information or 3D models
- Inserting labels
- Changing the appearance with filters

In our perspective, we would focus on the use of augmented reality in industry and specifically in automation [3]. This technology is used by many companies in industry, where data from CAD systems is used to train workers or students and also standardizes workflows and enables collaboration [4].

The first example is Siemens. This company has particularly focused on using augmented reality to support many types of human-centric activities. An example is a maintenance application where technicians are guided through routine transmission maintenance by the Assist AR software product. Once the technician uses LIDAR to scan a realistic model of the transmission, the process begins. Assist AR then provides instructions by highlighting individual elements, such as bolts or faceplates, in the correct order, in a workflow on the computer screen. Also, this application animates the individual elements to show how the parts are removed, replaced or even reassembled (Fig. 1) [5].



Fig. 1. Demonstration of Assist AR when disassembling the gearbox [5]

In its simplest form, augmented reality is mainly used to identify a component, a machine or a piece of equipment or an entire factory and then offer information about that section. More sophisticated applications also include interactive 3D models or holographic animations depicting the various steps that need to be performed.

Augmented reality in industry is used in a number of ways. The first use would be to train new workers, or to guide already experienced workers through various workflow improvements. This style helps the workers to achieve faster and also better skill acquisition and also it adds to the confidence of the workers and users for field use.

As a second way, we could mention the identification of assets. An example might be an operator or technician who needs to quickly find individual assets from many products. Augmented reality can help in this regard and identify these assets quickly, easily and reliably at the same time.

During the Covid-19 pandemic and due to various travel restrictions, another usage of augmented reality developed, namely the transmission of important information or even knowledge at a distance. However, this part has not only expanded due to the pandemic but also due to the lack of experienced personnel. In fact, older and more experienced workers were not able to travel to the field to assist their less experienced colleagues. This is where augmented reality helped a lot. In fact, with this technology, the more experienced workers were able to see what their colleagues or local

978-1-6654-8775-7/22/$31.00 © 2022 IEEE

engineers were seeing and were able to assist them remotely.

Modern forms of visualisation are now realized on the basis of the development of new ICT technologies (e.g. interactive applications made in 3D engine [6], virtual reality or augmented/mixed reality) [7]. Visualisation of process modelling, identification and control of complex mechatronic systems, elements and drives using virtual and augmented/mixed reality allows students to get a much quicker and better understanding of the studied subject compared to conventional education methods.

The ability of AR to assist in remote communication and also to make collaboration more efficient is a step forward for process design or facility layout. Thus, we can talk about a fourth way in which augmented reality is being used in industry. The user himself can place a virtual device in a space in the real world in order to verify the layout of the system or whether the space allocated for this system is sufficient. This prevents miscalculations or major budget cuts before the actual installation of the desired system [8].

Implementing augmented reality is much more difficult than implementing virtual reality itself. This is because virtual reality does not require the exact dimensions of the real world, as is the case with augmented reality. Thus, it needs elements of the different parts of the real world for precise placement and overlapping. Thanks to the gaming industry, many of these problems have been solved, and many developers who have worked for the production sector have borrowed the technology from the gaming industry.

Computer networks are now used for more than just connecting personal computers. Even at the level of specific sensors and actuators, smaller devices can connect to them. Modern single-board computers and microcontrollers that can be quickly connected to the global Internet are responsible for this trend. The result is a new paradigm - Internet of Things (IoT) as an integral part of the Industry 4.0. Without it, the vision of the fourth industrial revolution would not be possible. In the field of digital factory is a successor of machine-to-machine (M2M) communication. Currently, industrial HMI (human-machine interface) panels, consoles, web or mobile applications are used to control and monitor mechatronic systems in Internet of Things (IoT) networks. Using these conventional control and monitoring methods of mechatronic or cyber-physical systems within IoT networks, this method may be fully satisfactory for smaller rooms. Since the list of devices fits on one screen, we can monitor the status and control these devices almost immediately. However, in the case of several rooms or buildings, which is the case of digital factories, ordinary ways of interacting with mechatronic or cyber-physical systems become troublesome. In such case, there is the possibility to apply advanced digital technologies such as augmented reality. Using these technologies, digital (computer-generated - CGI) objects can be inserted into the real world. The aim of article [9] was to describe design and implementation of a new method for control and monitoring of mechatronic systems connected to the IoT network using augmented reality to create an innovative form of HMI.

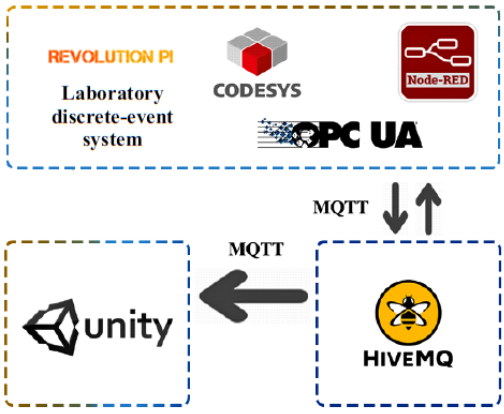


Fig. 2. Scheme of the application system

This article [9], however, was concerned with the creation of an augmented reality HMI for a system that was controlled by a simple microcontroller suitable for hobby projects and prototyping. However, there is a need to take this issue further towards Industry and Industrial IoT. Therefore, the present paper discusses the possibilities of creating an HMI in augmented reality for a system that is controlled by an industrial computer (PLC). As a first step, it was necessary to connect the 3D engine with the PLC and the discrete-event system. This is the subject of the present paper.

II. Monitoring of Discrete-event System using PLC and 3D Engine

Fig. 2 shows a scheme of the system that was implemented to demonstrate the monitoring of a discrete-event system using a graphical user interface (GUI) programmed in Unity 3D engine. The monitored variables are those of the Fischertechnik laboratory discrete-event system. This kit is controlled by an open-source PLC (or sometimes it is called industrial PC - IPC) of the Revolution Pi series. The control program (ladder diagram) is created using the CODESYS development environment. This environment also offers the creation of an OPC UA server, which has also been realized. In addition to the control program and the OPC UA server, the PLC also runs a program implemented using Node-RED. This program reads variables from the OPC UA server and sends them using the MQTT protocol to the HiveMQ broker, which is located in the cloud. The application in Unity 3D engine can use MQTT protocol to receive data from HiveMQ and display it to the user.

A. Connection of laboratory discrete-event system and PLC

In the beginning, it was necessary to connect the individual components of the laboratory system. These were PLC Revolution Pi, RevPi Connect + feat. CODESYS (Fig. 3). Next was the RevPi DIO, i.e. the digital input/output module, the



Fig. 3. Revolution Pi, RevPi connect + feat. CODESYS

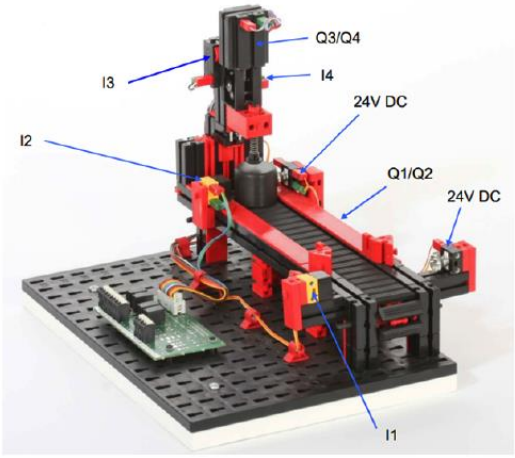


Fig. 4. Punching machine with conveyor Fischertechnik 96785 with sensors and actuators description

MDR-60-24 DIN rail power supply and the Fischertechnik 96785 (Fig. 4). This was a punching machine with a conveyor belt.

In Fig. 5 we can see the final wiring of the laboratory system.

During the solution, it was necessary to work with the operating system that resides on the Revolution Pi. This is a modified Raspbian, with which it was possible to verify the individual functions of our PLC. The Revolution Pi could be connected to a monitor using a micro-HDMI connector and as our PLC also had 2 USB ports, it was possible to plug in a mouse and keyboard for additional control.

After powering on the operating system, a login was required in the Raspbian command prompt. The login credentials were provided by the manufacturer. After a successful login,

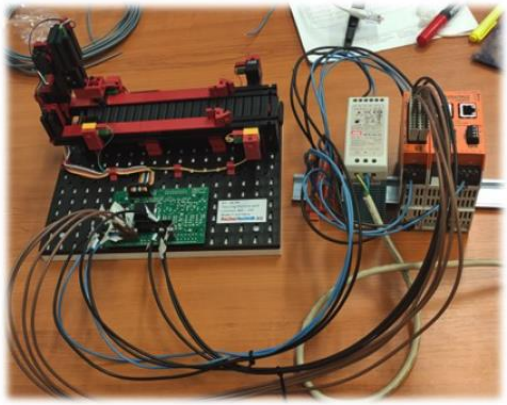


Fig. 5. Laboratory system wiring

the option was to continue in the command line style, or in the so-called Raspbian graphical interface. We chose the graphical interface option, which was entered using the *startx* command in the command line.

For further tasks, it was necessary to provide an internet connection to our operating system in the Revolution Pi. Since the PLC itself had two RJ45 Ethernet interfaces, we connected our PLC using an RJ45 wire - first to the router and then to the computer itself, which we used to program the PLC. For further work, we found out the IP address to which our system was connected, and proceeded with other tasks.

B. Communication and installation

After successful connection of the whole system and correct connection to the internet network, we were able to continue working with our system. Using the IP address we found in the previous step, we moved on to the PiCtory configuration (Fig. 6). PiCtory is a web browser application that aids in the communication and configuration of the PLC. After logging on to the localhost of our PLC, we also composed our system in virtual form using PiCtory. Thus, we have a PLC with a DIO module connected to it. Using this application, it was possible to set up and annotate the individual pins that we used in our project.

We used 4 input and 4 output pins. Output pins were under numbers 1-4, which we used to control the conveyor motor forward and backward and the punching machine motor up and down. Input pins controlled the two photosensors on the edges of the conveyor and the buttons on the punching machine. We can see the specific input/output PLC pins (left side of the list) controlling specific parts of the machine (right side of the list):

- `I_11` - Button I3 (bottom)
- `I_12` - Button I4 (top)
- `I_13` - Photosensor at the end of the belt (I1)



Fig. 6. PiCtory

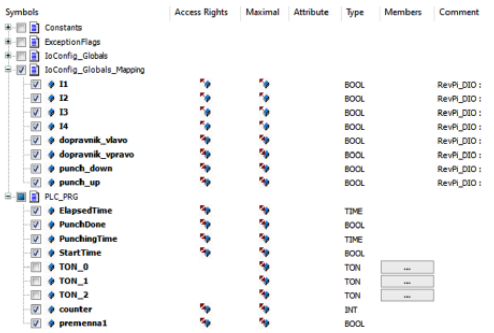


Fig. 7. Symbol configuration

- `I_14` - Photosensor under the punching machine (I2)
- `O_1` - Conveyor left
- `O_2` - Punching machine up
- `O_3` - Conveyor right
- `O_4` - Punching machine down

C. PLC programming

For programming the PLC system itself we used the CODESYS program, with which our Revolution Pi was compatible. This was CODESYS V3.5 SP17 Patch 2, which we used on a separate computer running Windows. It was then necessary to include our RevPi and RevPi DIO module in the program.

To program the control system, it was necessary to determine the variables. There are two ways of specifying variables, either globally or locally in the program. In our program we have specified variables in both ways. The variables controlling sensors *I1* to *I4* and the variables controlling the motors of the laboratory system *O_1* to *O_4* were determined as global variables, which we register directly at the inputs and outputs of the DIO module. Variables such as counter, time or auxiliary variables for controlling the system, we have determined as local variables directly in the program.

After determining all the variables, we wrote a short code that controls our laboratory system. The program is written in ladder logic style. We chose this language for its clarity as the system will be used for educational purposes.

The program waits for the product to be inserted on sensor *I1*, which is located at the beginning of our production line. When the object is inserted at the beginning of the line, the program starts a loop where the product goes under the punching system where sensor *I2* is located. Then, when the conveyor stops, the punching system starts moving downward until sensor *I3*, which is in the form of a button, is activated at the bottom of the rails on which the punching system moves. The system punches for five minutes and then returns to its original position at the top of the rails. Sensor *I4* is used to detect the upper limit of the punching system displacement. After this cycle, the product is moved by the conveyor to the right side of the laboratory system, where the program is terminated and the next cycle begins.

D. OPC UA server

To acquire and manipulate the information produced by our laboratory system, we needed to use the Node-RED programming tool, which is located directly in our Revolution Pi. To continue in this direction, we needed certain variables

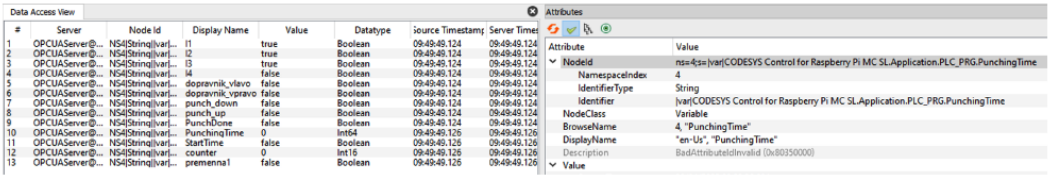


Fig. 8. Variables in OPC UA client - UAExpert

that we were interested in monitoring. These variables are needed to be published to the OPC UA server.

In CODESYS we have created a so-called Symbol configuration, where we have chosen the individual variables using a simple option, as we can see in Fig. 7.

To create the OPC UA server itself, it was necessary to set the certificate for OPC UA. In the Security screen section of CODESYS itself we searched for individual certificates that had not been created yet and created the necessary certificate.

Subsequently, it was necessary to verify the functionality of this server, but during various testing of the OPC UA server using the client in the UAExpert application, we found that there were problems with the Revolution Pi setup. We found an error in the security of the PLC itself in CODESYS. This was an additional password security on our system and also an anonymous login to the system. Once the bug was eliminated, we moved on to the actual tracking of the individual variables. For proper functionality of the entire system, it was also necessary to connect our PLC to the actual source of the internet connection and based on this fact, adjust the settings in CODESYS such as the IP address and so on. After verifying the functionality of the connection again, we connected to the OPC UA server of our PLC system and found all the variables that were needed to continue with the UAExpert program. In Fig. 8 we can see the interface of this program with the different variables.

On the right side of Fig. 8, we can see the value of our nodes that were needed to identify the observed variables in Node-RED. As an example, we can take the variable that we have shown in Fig. 8. We can see that it is the Punching time variable that we have set in CODESYS for a certain time during which our punching system creates a hole in the product. Since the PLC was not punching at this point, the value was zero. For us, the important data from this program was *NodeId* and *DataType*. *NodeId* in this case was *ns=4;s=|var|CODESYS Control for Raspberry Pi MC SL.Application.PLC_PRG.PunchingTime*, where we can see that for each variable. The majority of the ID value did not change and the last part of ID changed based on the name of the variable used. We used *DataType* for the data type of the variable so that we could use it in the Node-RED program afterwards.

E. Node-RED

Since all the necessary information was collected, we proceeded using Node-RED. The first task was to activate this service in the PLC system using the PiCtory interface, which then guided us to the correct IP address and port number to use our Node-RED. After successfully navigating to this IP address, it was necessary to test the connection to the OPC UA server located at our desired address. We did this by first installing all the necessary add-ons in Node-RED, specifically this was *node-red-contrib-opcua*. This add-on offered us many additional nodes to work with the OPC UA server and client. Then, using the OPC UA client node in Node-RED, we connected to the server.

Since our task was not only to get data from the PLC but also to send it to Unity, we chose the MQTT broker HiveMQ for this task. It is a freely available cloud broker for various uses and allows connecting IoT devices. We created functions for each variable, which we used to split the variables into different MQTT topics.

In Node-RED, to send data to the MQTT broker, we need an *mqtt out* node. This node is used to forward individual data to topics on our broker's server.

In Fig. 9 we can see the final version of the Node-RED we have set up, which contains all the necessary elements and nodes to continue with the project.

F. Creating an interactive application in Unity engine

Since all the previous steps have been completed, we started to look at the topic of the interactive application itself. For the interactive application, we decided to choose the Unity engine.

The first thing to do was to connect Unity to the MQTT broker.

The next step was the creation of the application itself. For functionality we created two scripts. The first one is called *mqttReceiver*. This script was then assigned to a newly created game object in Unity called *MQTT_Receiver* (Fig. 11). Here, the broker address and port needed to be set. It also needed to subscribe to a topic, here specifically *Diplomova_Praca_I1*.

The second script that needed to be created was *mqttController*. Again as before, we needed to insert this script into a new game object in Unity, which we called *ControlledObject*. We also inserted the previously created object into the Event Sender entry, for communication with each other.

After successfully testing the functionality of the data extraction from HiveMQ, we displayed all the observed variables in the application by duplicating all the game objects that were needed. Each variable was assigned a text describing which variable it is in that row, as well as the value of the variable

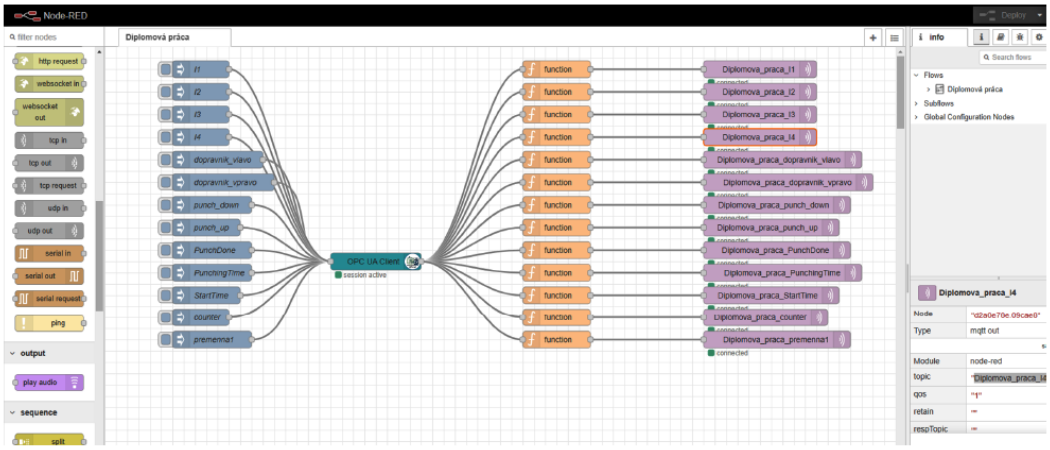


Fig. 9. Final version of the Node-RED program for transferring data to the MQTT broker

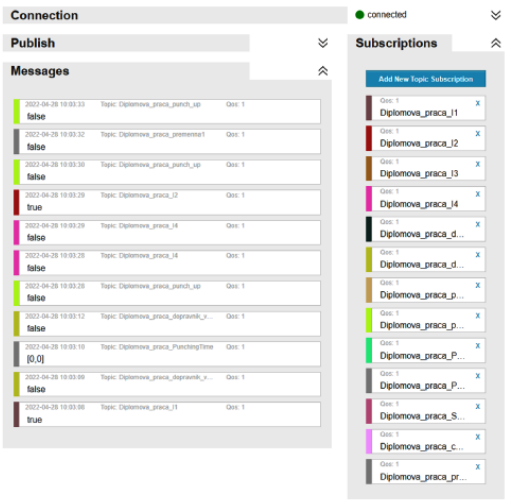


Fig. 10. Insight into broker and correctness of transmitted variable data

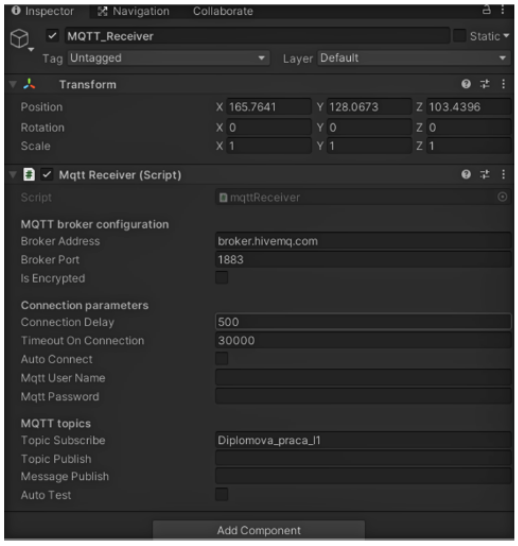


Fig. 11. Modification of MQTT_Receiver

itself. In Fig. 13, we can see the hierarchy that makes up our application.

It was then necessary to run the interactive application and make sure that the data from the PLC system was correctly sent via Node-RED and HiveMQ to Unity. In the following figures (Fig. 14-16) we can see the finished and functional application with the individual descriptions at the beginning of the whole process. We can notice that instead of specific values, a dash is placed to represent a process not yet started.

III. Conclusion

Currently, during the ongoing Industry 4.0 revolution, new forms of human-machine interfaces (HMIs) are being sought for monitoring and control of mechatronic (cyber-physical) systems. Such HMIs can be implemented using augmented reality. In order to realize such a concept, it is necessary to connect a 3D engine with the controlled system. Therefore, the paper dealt with the interfacing of open-source PLC (Revolu-

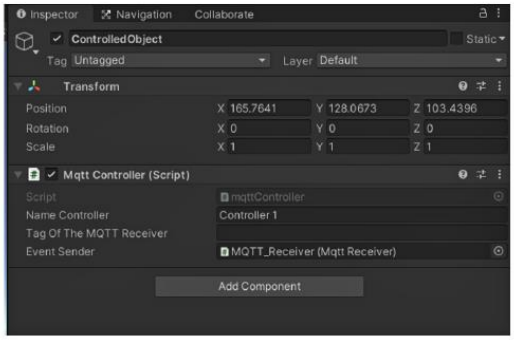


Fig. 12. ControlledObject settings

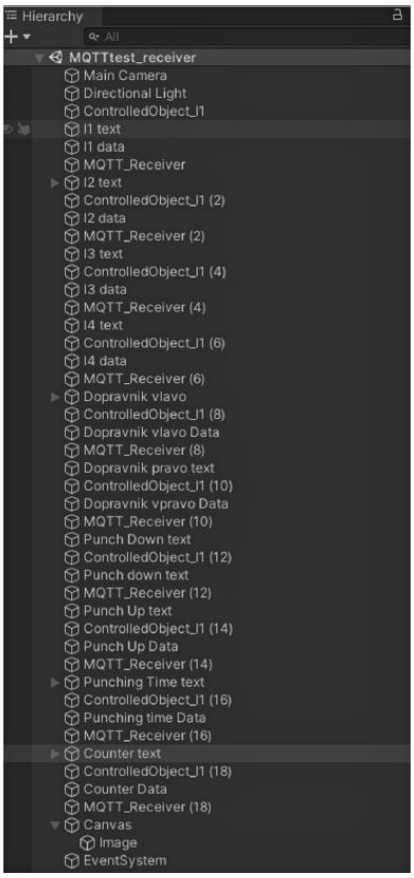


Fig. 13. Interactive application hierarchy in Unity

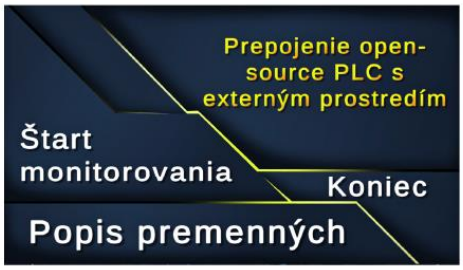


Fig. 14. Application menu



Fig. 15. Variable labels and descriptions

tion Pi series) with the application implemented in 3D engine. OPC UA server, Node-RED middleware and MQTT broker were used for their interconnection. The future challenge will be the implementation an augmented reality application in the 3D engine Unity. The current application only works on the computer screen. However, it is an important start for testing the possibilities of interfacing discrete-event PLC systems with Unity 3D engine.

ACKNOWLEDGMENT

This research was funded by the Slovak research and Development Agency under the contract no. APVV-21-0125, by the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic

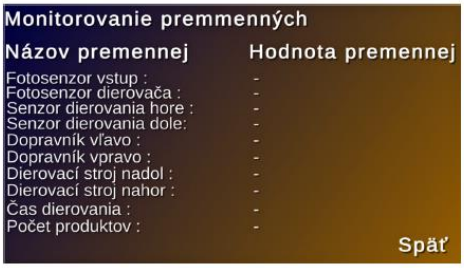


Fig. 16. Variable monitoring

KEGA 016STU-4/2020 and 039STU-4/2021, by the Scientific Grant Agency of the Ministry of Education, Research and Sport of the Slovak Republic No. 1/0107/22. Special thanks for the help with initial analysis of current applications and software implementation to Katarína Gajdošíková.

References

[1] P. A. Rauschnabel, A. Brem, and Y. Ro, "Augmented reality smart glasses: definition, conceptual insights, and managerial importance," *Working paper, The University of Michigan-Dearborn, Tech. Rep.*, 2015.
[2] B. Houston. (2020) What is augmented reality (ar)? a practical overview. [Online]. Available: https://www.threekit.com/blog/what-is-augmented-reality
[3] K. Sloan. (2016) Rotor brings toyota showroom 360 to life with unreal engine. [Online]. Available: https://www.unrealengine.com/showcase/rotor-brings-toyota-showroom-360-to-life-with-unreal-engine
[4] D. Parvati, W. L. Heinrichs, and Y. Patricia, "Clinispace: a multiperson 3d online immersive training environment accessible through a browser," *Medicine Meets Virtual Reality 18: NextMed*, vol. 163, p. 173, 2011.
[5] Siemens. (2022) Manufacturing work instructions. [Online]. Available: https://www.plm.automation.siemens.com/global/en/products/manufacturing-planning/manufacturing-work-instructions.html
[6] Triseum. (2017) Variant: Limits. [Online]. Available: https://triseum.com/calculus/variant/
[7] J. Majernik, M. Madar, and J. Mojzisova, "Integration of virtual patients in education of veterinary medicine," in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2017. doi: 10.15439/2017F134
[8] J. R. Koelsch. (2021) How augmented reality became a serious tool for manufacturing. [Online]. Available: https://www.automationworld.com/process/iiot/article/21259479/how-augmented-reality-became-a-serious-tool-for-manufacturing
[9] E. Stark, E. Kučera, O. Haffner, P. Drahoš, and R. Leskovský, "Using augmented reality and internet of things for control and monitoring of mechatronic devices," *Electronics*, vol. 9, no. 8, 2020. doi: 10.3390/electronics9081272. [Online]. Available: https://www.mdpi.com/2079-9292/9/8/1272

Authors



doc. Ing. Erik Kučera, PhD.
Institute of Automotive Mechatronics,
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
erik.kucera@stuba.sk
His focus is mainly on modern information and communication technologies and their use in the context of fourth industrial revolution Industry 4.0. This icludes e.g. internet of things, virtual and mixed reality, cloud computing and new microcontrollers.



doc. Ing. Oto Haffner, PhD.
Institute of Automotive Mechatronics,
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
oto.haffner@stuba.sk
Research activities in applications of computer and machine vision in mechatronic systems using modern ICT such as microcomputers, cloud, computational intelligence, etc. Current focus also on the use of deep learning in practical applications for Industry 4.0 such as visual quality control. He is also involved in modelling and simulation of production processes in manufacturing.



Ing. Dominik Janecký
Institute of Automotive Mechatronics,
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
He studies PhD. progamme Mechatronic Systems. His focus is mainly on virtual and mixed reality and their use in the context of new human-machine interfaces.