Use case 6:

Virtual reality programming of a manufacturing cell

1. Introduction:

Welcome to this comprehensive training module, designed to guide you through the process of understanding, installing, and implementing a specific module within your system.

In this tutorial, you will find the following sections:

2. System Requirements: Before diving into the training, it's crucial to ensure that your system meets all technical requirements. This section will provide you with information on the required software, hardware tools, operating systems, libraries, and programming languages. Additionally, you will find useful links to documentation and tutorials, as well as sequence and architecture diagrams to help you better understand the module.
3. System Architecture: This section delves deeper into the structure and functionality of the module. You will learn more about sequence and architecture diagrams, and any other information deemed valuable for your specific module. We will also provide code snippets and links to facilitate hands-on learning.
4. Conclusion: In closing, we will express our gratitude for your participation and encourage you to explore the use case lectures to further enhance your understanding of the module and its applications.

By the end of this training, you will be well-equipped to tackle any challenges related to the module and integrate it into your projects.

2. System Requirements:

The digitalization of a production environment is the focus of this demonstrator, which illustrates how robots and other machines can be connected and programmed using a standard industrial information server. In this section, we detail the module's technical requirements and the necessary hardware and software tools.

Hardware and Software Infrastructure:
The system necessitates an industrial robot arm, mobile robot, or other machines that can be controlled with an external computer. Each robot requires an external computer for control and monitoring, with either a Raspberry Pi or Windows machine being appropriate options based on the robot and specific requirements. A separate computer hosts the OPC UA server to collect data from the external computers, and all these computers connect through a dedicated router.

The OPC UA standard connects the robots and other machines. The Python library (https://github.com/FreeOpcUa/python-opcua) is employed for hosting the OPC UA server and connecting to it. Each external computer in the system runs an OPC UA client to connect the machine to the server, using the same code for monitoring and controlling the robots.

This module requires a VR headset compatible with SteamVR. Any VR headset supporting SteamVR will work; however, the Oculus Quest is utilized in this example because it functions wirelessly and doesn't require extra cameras.

Visual Components Premium 4.2 serves as the platform for the digital twin model of the system. This industrial simulation software is suitable for simulating production systems and has built-in connectivity for the OPC UA server. Additionally, Visual Components Experience is required, as it connects Visual Components Premium to the VR headset, as shown in Figure 2. The simulation can be streamed to the local computer or another computer on the same network.

Cybersecurity:
The cybersecurity measures for this system and the Wire Arc Additive Manufacturing with Industrial Robots are similar:

- Connect robots to a separate Wi-Fi router.
- Implement endpoint protection on computers and shield ports.
- Restrict computer access via Jump server or Team Viewer, with two-factor authentication.
- Use SecurityOnion to monitor computers, including the OPC UA server host.
- Employ encryption and signature for OPC UA server communication. Further cybersecurity measures for the OPC UA server are in progress.

3. System Architecture:

This module employs VR technology for programming various robot arms and mobile robots within a production system. All robots in the system are connected to a single industrial information server, which utilizes the OPC UA standard in this case.

The module requires a digital twin of the production system in the Visual Components module: Rapid Mapping of a Production System in a Virtual Environment, and communication between the robots and the Visual Components module: Connecting Virtual Model with the Physical Model.

Part 1: Connecting the Robots

The OPC UA server facilitates communication between the simulation software and the physical robots. An additional folder has been added to the OPC UA server (Figure 1) with subfolders for each robot in the production system and a system-program subfolder. These subfolders contain data on robot movement positions and variables for starting and stopping robot execution. The implementation and use of the OPC UA server with robots can be found in (BEIBEI, PLEASE ADD A LINK HERE TO YOUR VIDEOS).

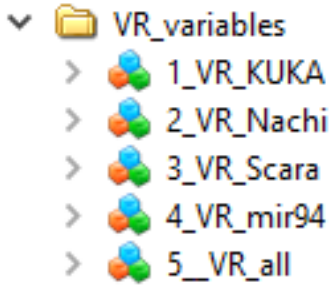

Figure 1: The OPC UA folder with VR variables used to store data from Visual Components

A Python-based code translator has been developed to enable the execution of programs created in VR. This translator connects to the OPC UA server as a client, retrieves data from the server's VR folder, and converts it into executable code for the robots. The same translator is compatible with different robots, as they are all controlled by the OPC UA server in a similar manner.

Part 2: Setting up the Virtual Environment

The second part involves setting up and creating a virtual environment where it is possible to interact with and program the robots. However, it is a prerequisite to already have a digital twin of the environment to implement this module. For more information on building the digital twin, see (BEIBEI, PLEASE ADD A LINK HERE TO USE CASE 6 MODULE 1 TUTORIAL!!).

First, open Visual Components and load the digital twin model. In the Visual Components library, there is an "Interactive VR HMI", as seen on the left side of Figure 2. Additionally, it is possible to edit the configuration and layout of the buttons, as shown on the right side of Figure 2.

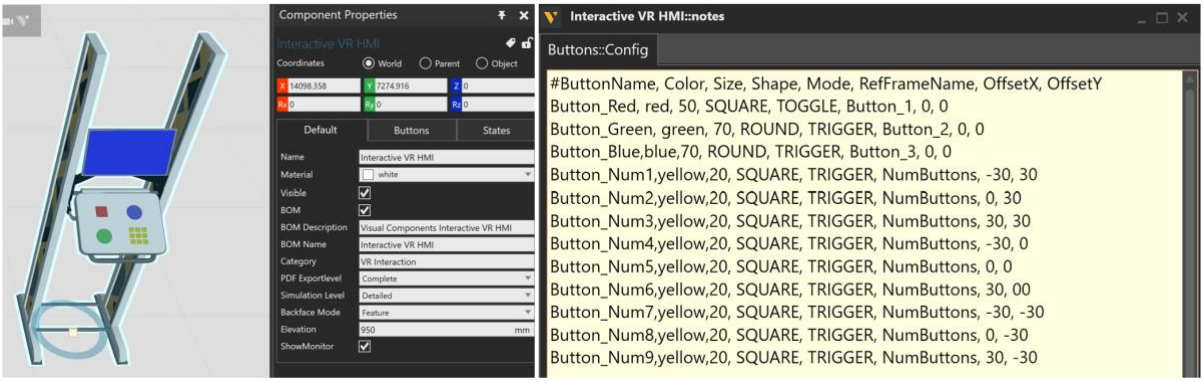

Figure 2: The Interactive VR HMI in Visual Components

To add custom functions for the HMI, first click on the HMI itself and then on modeling. Here, there are Python scripts that detect interactions with the buttons, and custom functions can be created depending on which button is pressed. Note that Python 2 is used to write and create functions. It is also possible to create custom scripts that run in parallel when the simulation is running.

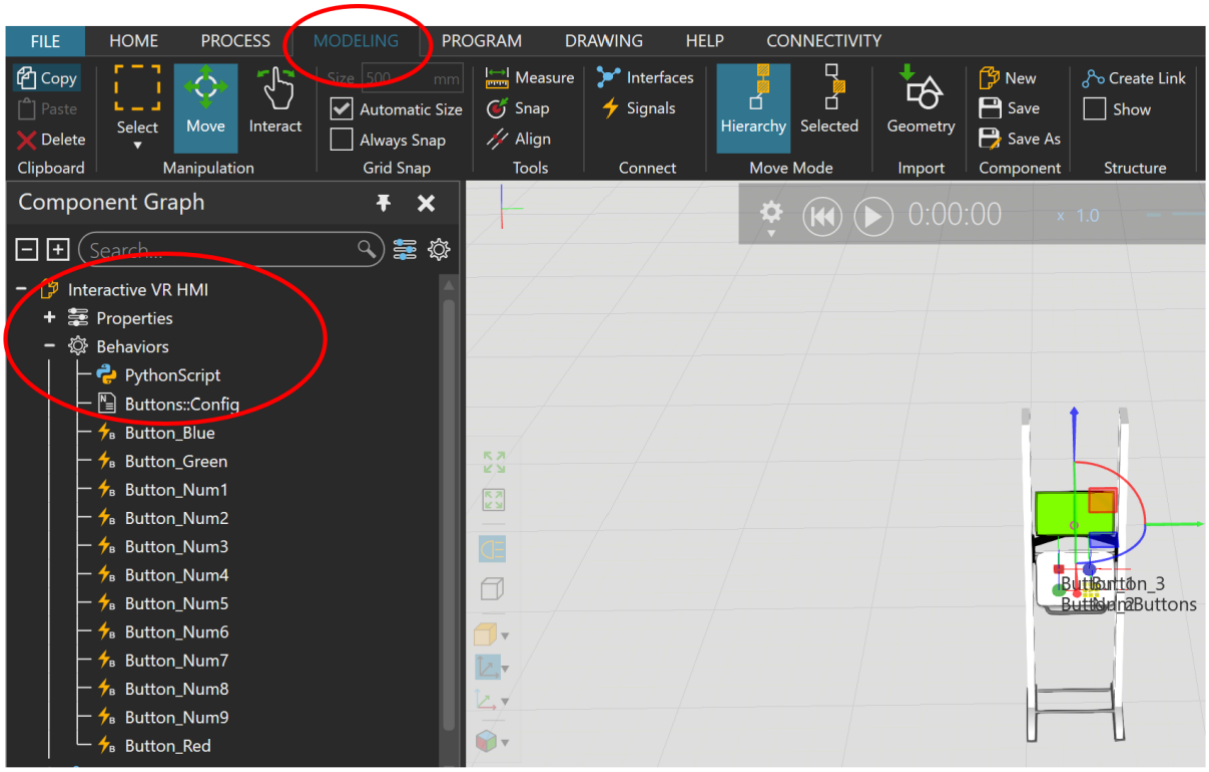

Figure 3: Where the functions and code for the Interactive HMI are located

The HMI used in this module can be seen in Figure 4. The blue button reads and saves the position of the robot, the green button runs through the positions that have been added to run the program, the red button deletes positions, and the yellow button checks and changes positions that have already been added. There is also a button for gripping objects, and the black button activates a function that sends the program to the OPC UA server.

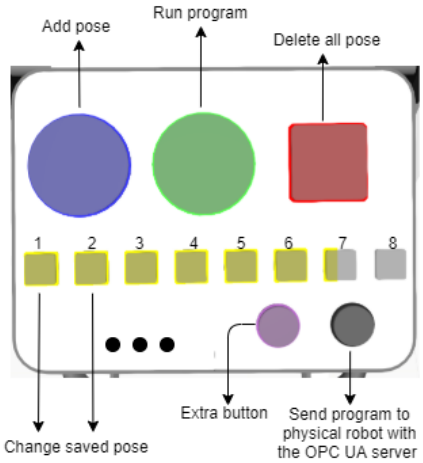

Figure 4: HMI table used in Visual Components for programming and the functionality of the buttons

An example of the functions in the HMI can be seen in Figure 5.

```
# Green buttons -----------------------------------
if(button_green.Value == True and add_num != 0): # drive robot
  for k in range(0, add_num):
    data_run = data_positions[k]
    nachi_control.driveJoints(data_run[0], data_run[1], data_run[2], data_run[3], data_run[4], data_run[5], move_speed[0])

  while(button_green.Value == True):
    delay(0.05)
```

Figure 5: The Python function in the HMI used to execute robot movements by pressing the green button

Part 3: Running the Digital Twin in VR

The final part is running the system using VR. Visual Components Experience is used to connect Visual Component Premium with the VR headset, as shown in Figure 6. The simulation can be streamed to a local computer or another computer on the same network.

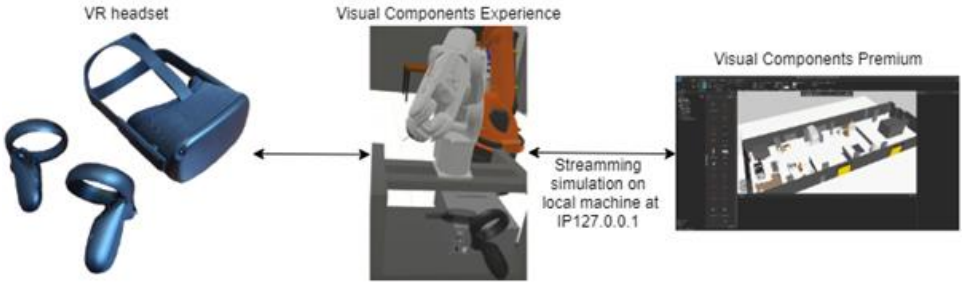

Figure 6: Connection between the VR headset and the virtual world in Visual Components

When the robots are connected to the same OPC UA server as Visual Components, it is possible to directly program the robots in VR. Press the black button to transfer the programs to the OPC UA server, and then run the program on the physical robot. By integrating VR technology and the OPC UA server, this module enables seamless programming and control of robot arms and mobile robots in a production system.

3. Conclusion:

In this training module, we have explored the process of virtual reality programming for a manufacturing cell. Through the integration of VR technology and the OPC UA server, the module allows for seamless programming and control of robot arms and mobile robots within a production system. By following the outlined system requirements and the three-part system architecture, users can effectively connect robots, set up the virtual environment, and run the digital twin model in VR.

Throughout the module, we have examined the necessary hardware and software infrastructure, as well as crucial cybersecurity measures. We have also provided a step-by-step guide to connecting the

robots, setting up the virtual environment using Visual Components Premium 4.2, and running the digital twin in VR using Visual Components Experience.