Use case 5:

Simulation welding

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:

This is a robotic TIG welding system, used for automatic welding and incorporates a KUKA robot and rotary table. This section will outline the module's technical requirements and the necessary software and hardware tools for successful operation. It should be noted that use case 5 module 2 (Additive TiG welding), has the same hardware and software requirements.

Hardware Infrastructure:
The hardware infrastructure comprises a six-axis KUKA 30-3 robot arm and a DPK400 rotary table, both connected to a KUKA controller version 2 (KR C2). The system is also compatible with KUKA controller version 4 (KR C4). A Fronius MagicWave 5000 welding equipment connects to the PLC, which in turn links to the KUKA controller for laser fence and welding equipment control. The system is managed by an external computer for KUKA system control and another for OPC UA server hosting. These computers can run any operating system (Windows, Linux, Mac) supporting the OPC UA standard. For enhanced security, it is recommended to connect the system to a separate Wi-Fi router, though this is not mandatory. The hardware setup is illustrated in Figure 1.

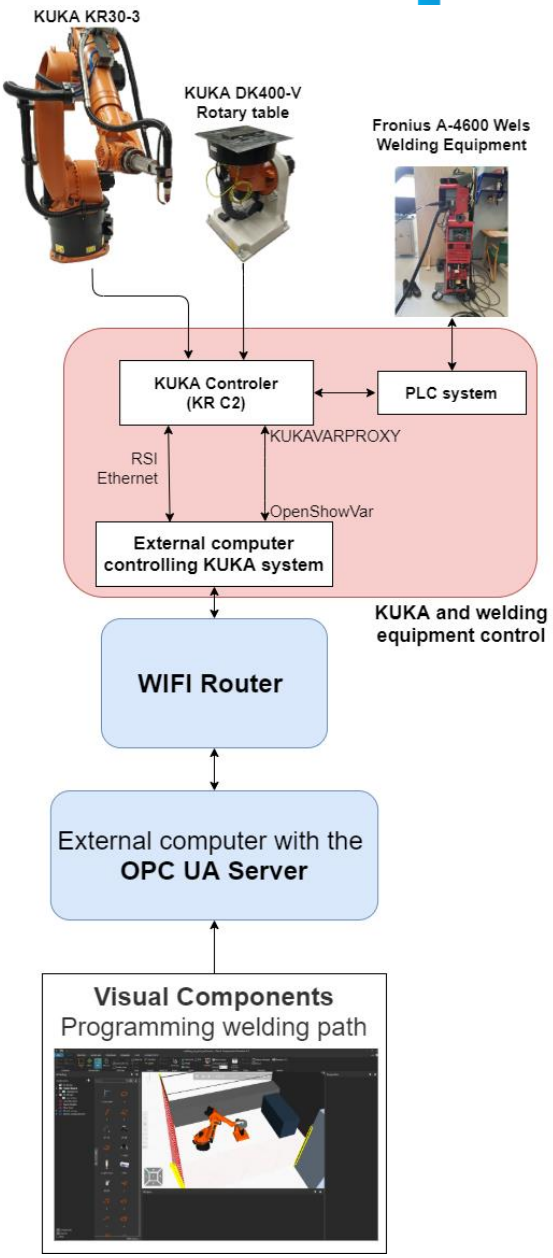

Figure 1. How the welding system is setup and connected.

Software Infrastructure:
Visual Components Premium 4.2+ is used for programming and testing (https://www.visualcomponents.com/products/visual-components/premium/). This manufacturing simulation software features an open Python API for customization and built-in OPC UA connectivity for communication between Visual Components and the KUKA system. A Python version of the OPC UA standard is utilized (https://github.com/FreeOpcUa/python-opcua).

Data from Visual Components is sent to the OPC UA server, which is then translated by a Python program into a format executable on the KUKA controller. The translation program is written in Python 3. Additional software is needed for KUKA robot and rotary table control from an external computer:

Robot Sensor Interface (RSI) for joint and TCP control with a 12ms update rate.
OpenShowVar-KUKAVARPROXY for communication between an external computer and the robot controller (https://github.com/ImtsSrl/KUKAVARPROXY and https://github.com/linuxsand/py_openshowvar).

Cybersecurity:Similar cybersecurity measures apply to this system and the Production flow simulation/supervision system:

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.
Refer to Figure 2 for an overview of the different cybersecurity levels.

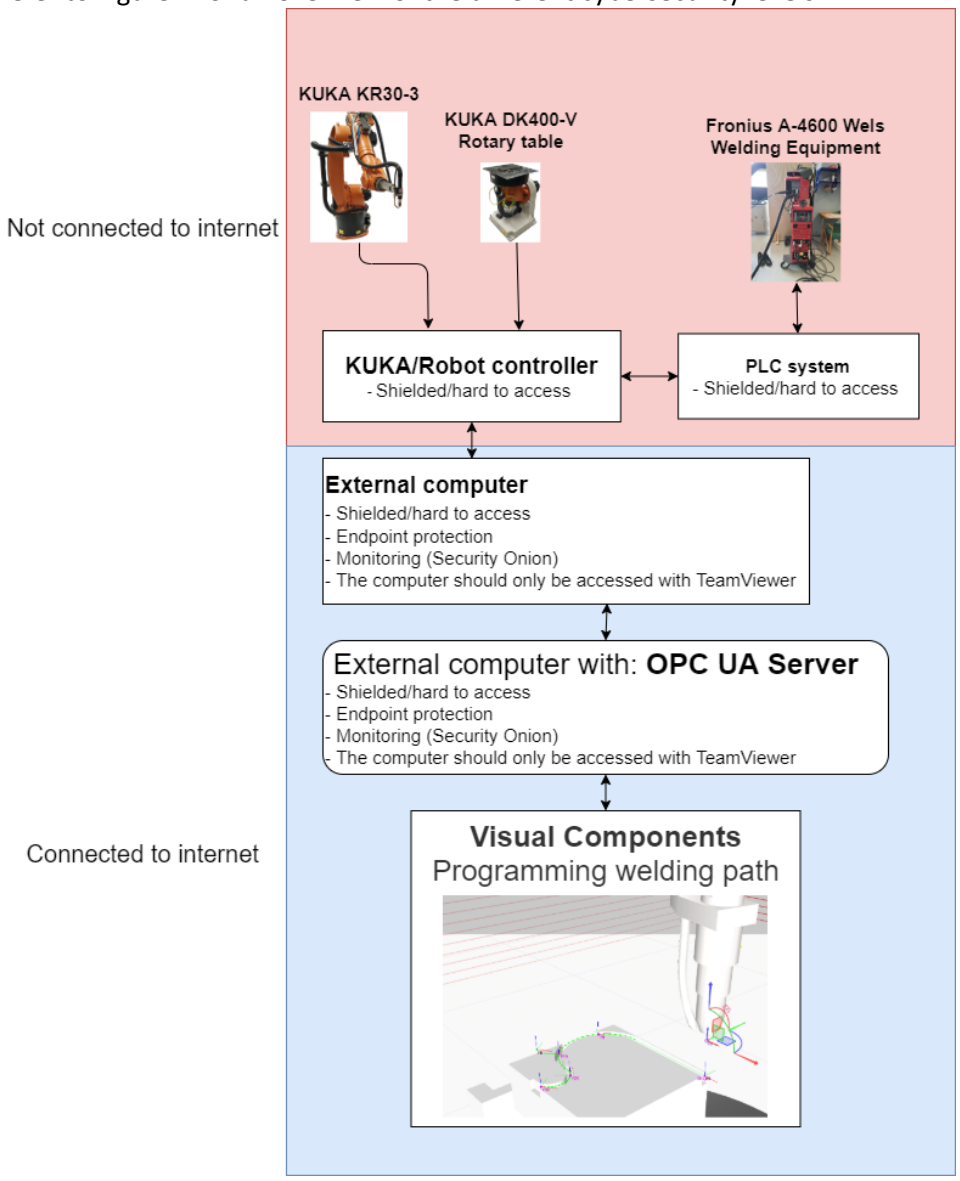

Figure 2. Cybersecurity setup of the welding system

2. System Architecture:

The primary objective of this module is to leverage Visual Components simulation software for programming a welding path. The software enables programming of a KUKA robot arm and a rotary table, which are integrated within Visual Components. This facilitates programming the robot arm and rotary table as a single, cohesive unit with a consistent tool center point (TCP), using either point-to-point or linear motion.

****Part 1: Programming the Robot****

First, a digital twin model of the welding cell is set up. A CAD model of the desired welding object is imported into Visual Components. The "PROGRAM" tab, depicted in Figure 3, is then used for programming.

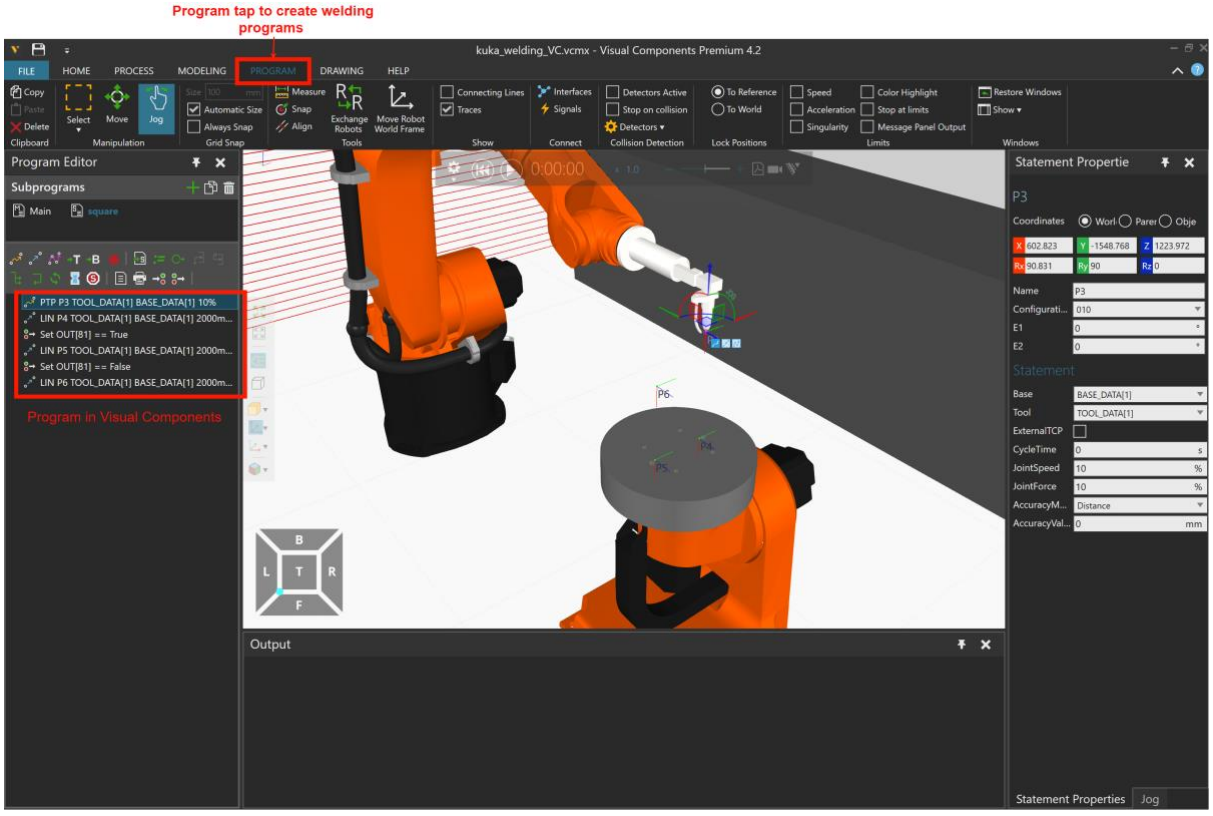

Figure 3: Programming interface in Visual Components

Visual Components features a path function that automatically generates paths based on an object within the software. CAD models can be imported and attached to the rotary table, allowing the robot

path to be automatically generated from the model. Once the program is complete, the simulation software verifies the robot's movement and visualizes the weld. The “PROGRAM” function offers a user-friendly approach to programming the welding cell.

To transfer the program from Visual Components to the physical robot, a function is added to the robot arm. By selecting the robot arm and then clicking on models, Python scripts can be added, as illustrated in Figure 4. Python 2 is utilized for adding functions.

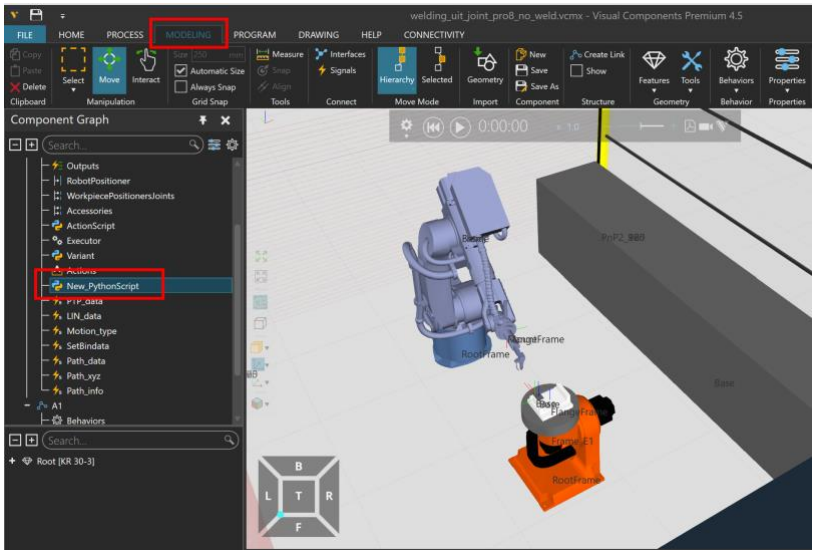

Figure 4: Location for adding Python scripts to the robot arm

In this module, the Python script is executed whenever the simulation runs. The script contains a function that reads the points and data from the “PROGRAM” window in Visual Components, translates the robot program into a string, and inputs the program into an OPC UA server. A snippet of the code used for this process is displayed in Figure 5.

```
#Add the types of movement: ------------------------------------------
if (i.Type == "PtpMotion" or i.Type == "LinMotion" or i.Type == "Path"):

  if(i.Type == "PtpMotion"):
    type_motion = type_motion + "PtpMotion" + ","
  if(i.Type == "LinMotion"):
    type_motion = type_motion + "LinMotion" + ","
  if(i.Type == "Path"):
    type_motion = type_motion + "Path" + ","

#Add the movement values to the
if (i.Type == "PtpMotion" or i.Type == "LinMotion"):
  point = i.Positions[0]

  kuka_joint = i.Positions[0].InternalJointValues
  rotary_joint = i.Positions[0].ExternalJointValues

  for k in range(len(kuka_joint)):
    joint_data = joint_data + str(kuka_joint[k]) + ","

  for k in range(len(rotary_joint)):
    joint_data = joint_data + str(rotary_joint[k]) + ","

  joint_data = joint_data + "d"
```

Figure 5: Code snippet for reading the “PROGRAM” in Visual Components and translating it into a string

Part 2: OPC UA Setup

A flexible method for sending robot programs from Visual Components to the physical robot is achieved through the OPC UA standard. The built-in OPC UA connectivity function in Visual Components is used to connect the software to an OPC UA server as a client. For information on creating the server and connecting the robot to the server, refer to (BEIBEI ADD YOUR MODULE HERE, USE CASE 6 MODULE 3!!!).

As mentioned earlier, when the simulation runs, a Python script takes data from the “PROGRAM” window, translates it into a string, and sends it to a variable in the OPC UA server.

Part 3: Robot Control

The final aspect involves controlling the robot arm, which varies depending on the brand and type of robot. This module utilizes a KUKA robot and rotary table connected to a KUKA KR C2 controller, as well as the RSI real-time control interface add-on. As shown in Figure 6, a control computer connects to the KUKA controller and the OPC UA server as a client, retrieves data from the server, and converts it into executable code for the physical KUKA robot. RSI communication controls the KUKA robot and rotary table, while KUKAVARPROXY activates the welding equipment.

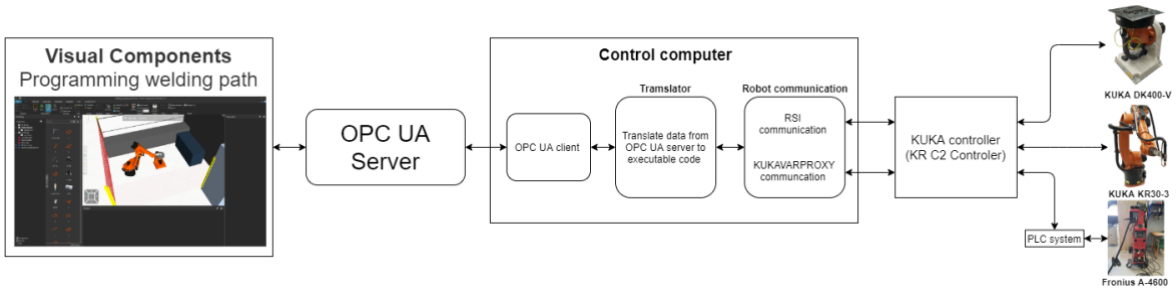

Figure 6: The system's code structure

RSI communication sets up a TCP/IP connection between the KUKA controller and an external computer, featuring a 12-millisecond update interval that can be increased to 2 milliseconds if required. KUKAVARPROXY functions similarly to RSI communication, creating a server and running in the background on the KUKA controller. An external computer connected via Ethernet can communicate with the KUKA controller through a second program called "OpenShowVar" (Figure 6).

2. Conclusion:

Thank you for completing this training module on simulation welding using a robotic TIG welding system. Throughout this tutorial, you have gained valuable insights into the technical requirements, hardware and software tools, system architecture, and cybersecurity measures essential for successfully implementing the module. By understanding the role of Visual Components, OPC UA, and control interfaces like RSI and KUKAVARPROXY, you are now better equipped to tackle challenges related to the module and integrate it into your projects.

We encourage you to explore other use case lectures to further enhance your knowledge of the module and its potential applications. As you progress, you will discover that the skills and concepts you have acquired in this tutorial are invaluable for optimizing the performance of your robotic welding systems and ensuring their seamless integration into your workflow.