1. Wojciech KACZMAREK, 2 Szymon BORYS

ORCID: 1. 0000-0003-1620-6145; 2. 0000-0003-3805-9510

DOI: 10.15199/48.2025.06.14

Operator interface for controlling the track of the GoFa collaborative robot

Projekt panelu operatora w celu sterowania torem jezdnym robota kolaboracyjnego GoFa

***Abstract**. The article presents the design of an operator panel for controlling the track of the ABB GoFa collaborative robot. The application was developed in AppStudio and Visual Studio Code. It enables the operator to control the track in manual mode and to activate the robot in automatic mode. The robot station and the track were integrated using the Siemens S7-1200 controller. Communication between the PLC and the robot controller was established via a PROFINET network. The authors conducted functionality tests of the application using both a virtual twin and a real research and teaching workstation.*

***Streszczenie**. W artykule przedstawiono projekt panelu operatora do obsługi toru jezdnego robota kolaboracyjnego GoFa firmy ABB. Aplikacja została wykonana w środowisku AppStudio i Visual Studio Code. Aplikacja umożliwia operatorowi sterowanie torem jezdnym w trybie ręcznym oraz włączanie robota w trybie automatycznym. Stanowisko robota z torem jezdnym zostało zintegrowane z użyciem sterownika S7-1200 firmy SIEMENS. Komunikacja pomiędzy sterownikiem PLC i kontrolerem robota została zrealizowana z użyciem sieci PROFINET. Autorzy wykonali testy funkcjonowania aplikacji używając wirtualnego bliźniaka oraz na rzeczywistym stanowisku badawczo-dydaktycznym.*

Keywords: HMI, collaborative robot, track, operator panel
Słowa kluczowe: interfejs człowiek-maszyna, robot kolaboracyjny, tor jezdny, panel operatora

Introduction

Human-Machine Interfaces (HMI) play a key role in shaping the interaction between people and modern technologies, both machines and processes. HMIs serve as the primary interaction point between operators and technical systems, facilitating communication, control, and information retrieval. Their role is particularly crucial in contemporary technological production lines, where the advent of Industry 4.0 has redefined human-system interaction. The digitalization of data has led to a substantial increase in both the volume and complexity of information, necessitating the development of advanced operator panels capable of structuring and presenting this data in a clear and intuitive manner. Such systems enhance operators' ability to efficiently monitor, supervise, and control machinery while ensuring rapid response to dynamic process conditions.

The future of human-machine collaboration must therefore leverage the strengths of both machines and humans. Research is being conducted worldwide to identify the capabilities and characteristics of humans and machines. The goal of this research is to improve human-machine interaction [1,2,3].

Modern HMIs must support next-generation communication protocols to ensure stability and immediate data transmission between components of production stations [4]. Moreover, the analysis of large data sets is a valued asset when developing a digital transformation strategy, so data acquisition is crucial for production.

Attempts are being made to create universal HMI platforms capable of performing and developing various intelligent functions. An example is an HMI platform whose main components are designed and implemented using component-based development (CBD) [5].

The importance of HMI interfaces in production stations can be considered in several aspects:

- Improved communication and control, which is related to intuitive control of machines, robots, computers or IoT devices. Thanks to this, operators can:
  - monitor processes in real time (e.g. control panels in factories),
  - give commands using buttons, touch screens, voice or gestures,
  - receive immediate feedback (e.g. alerts and warnings).
- Increased efficiency and safety, which should be understood as minimizing the risk of human errors, automating: industrial processes, tasks performed by modern medical devices, and the autonomy of mobile robots.
- Simplification of the principles of using technology. Understandable and easy-to-use interfaces - use of advanced technologies for people without specialistic knowledge (e.g. operators of robotic stations are not required to know the principles of programming robots).
- Increased awareness of operators, which is related to the intuitiveness and aesthetics of interfaces, personalization of settings and adjustment of the amount of information to the needs of a given job position.
- Integration with future technologies. The technologies used to create HMIs should enable easy integration with modern technologies, including virtual and augmented reality, as well as artificial intelligence.
- Data security. The sensitivity of operational data and the necessity for specialized knowledge across various levels of service and management necessitate the implementation of multi-tiered access authorization. This approach safeguards critical information from unauthorized access and mitigates the risk of cyberattacks, ensuring the integrity and security of industrial systems.

HMI interfaces make it easier to perform tasks, but above all, they are a bridge between human needs and the capabilities of machines. The key to their success is the balance between functionality, security and human-centric design [6,7,8]. Research indicates [9] that teaching stations equipped with HMI panels significantly enhance students' competencies and effectively prepare them for work in automated industrial environments.

Test stand and digital twin

The individual components of the stand were configured using dedicated development environments. RobotStudio allowed the industrial robot to be parameterised and its control programme to be prepared with the aid of digital twin technology [10,11]. The TIA Portal v17 environment was used to configure and develop the control application for the PLC [12,13]. The track configuration was carried out using IndraWorks.

The proposed test stand comprises the following components:

- GoFa 15000 cobot with OmniCore C30 controller [14,15]:
  - payload capacity - 10 kg,
  - reach of the robot wrist - 1.5 m,
  - repeatability - 0.02 mm,
  - TCP speed - 2 m/s,
- a Siemens S7-1200 PLC:
  - model 1214C DC/DC/DC,
  - PROFINET communication,
  - Ethernet switch,
- a Bosch cobot track with IndraDrive controller.
  - track length - 2 m.

A stand diagram is shown in Fig.1.

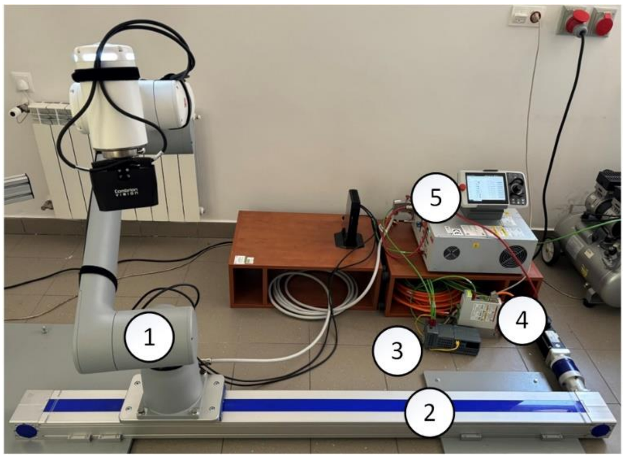


Fig. 1. View of the stand. 1 – GoFa robot, 2 – linear track, 3 – PLC, 4 – track controller, 5 – OmniCore controller with FlexPendant [17]

Integration and control concept

Communication between the individual components of the stand is based on the PROFINET protocol [16,17]. The proposed solution assumes that the PLC is the controller, while the robot and the track are the devices. A schematic of the PROFINET network connection is shown in Fig.2.

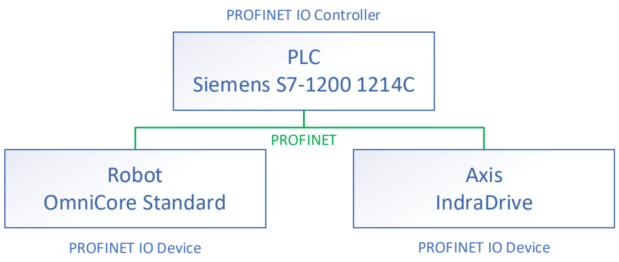


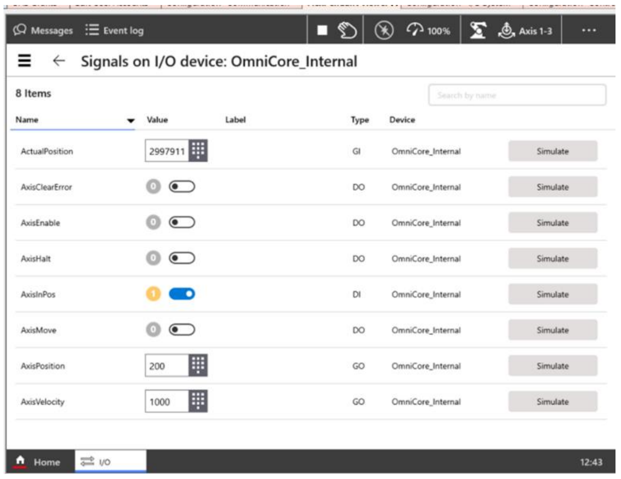
Fig. 2. Block diagram of PROFINET network [17]

The integration of individual workstation components required the acquisition of GSDML files for both the robot and the track, which are essential for the proper configuration of the PROFINET connection. The GSDML files for the OmniCore controller are available within the system on the robot's internal disk. In cases where a physical device is not available, a virtual twin of the system can be downloaded to a PC using the RobotStudio environment. Meanwhile, the GSDML files necessary for track control can be obtained from the manufacturer's website by entering the device's serial number [17].

The following digital signals were used to control the external axis:

- AxisEnable – motion enabled
- AxisHalt – stop axis
- AxisMove – execute movement
- AxisClearError – clearing active errors
- AxisPosition – group signal – external axis position
- AxisVelocity – group signal – external axis velocity

Access to signals is possible both from RobotStudio, through the IO Engineering tab and on the FlexPendant (Fig. 3)



Signals on I/O device: OmniCore_Internal

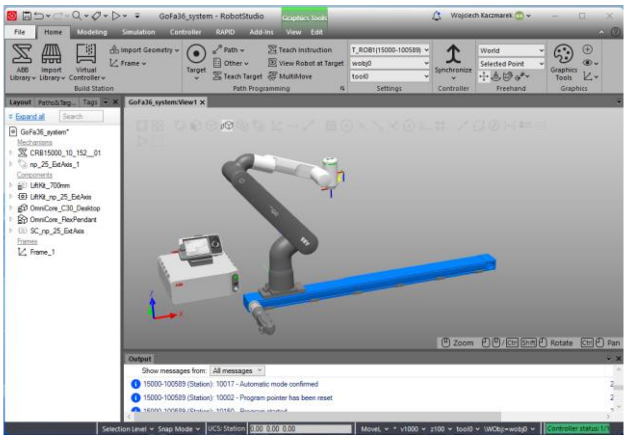
8 Items

Name	Value	Label	Type	Device	
ActualPosition	2997911		GI	OmniCore_Internal	Simulate
AxisClearError	0		DO	OmniCore_Internal	Simulate
AxisEnable	0		DO	OmniCore_Internal	Simulate
AxisHalt	0		DO	OmniCore_Internal	Simulate
AxisInPos	1		DI	OmniCore_Internal	Simulate
AxisMove	0		DO	OmniCore_Internal	Simulate
AxisPosition	200		GO	OmniCore_Internal	Simulate
AxisVelocity	1000		GO	OmniCore_Internal	Simulate

Fig. 3. I/O device view on FlexPendant

HMI interface

As part of the ongoing work, a digital twin was prepared in the RobotStudio environment (
Fig.4), modelled on the real robot's operating system [17].



```
15000-100589 (Station): 10017 - Automatic mode confirmed
15000-100589 (Station): 10002 - Program pointer has been reset
```

Fig. 4. General view of the digital twin in RobotStudio

Correct representation of the movement and control of the linear track required the development of a SmartComponent object reflecting the operation of an additional axis (SC_np_25_ExtAxis – Fig.5). SC_np_25_ExtAxis is a logical description of the actual operation and consists of the following blocks:

- Logic – track operation logic and signal scaling. The Logic program block calculates the current and target position to generate control signals.
- JointMover – control of the movement of the mechanism joints. Based on the received set position, the block executes the movement and informs about its completion.
- JointSensor – monitoring the mechanism joints during simulation. Allows for current reading of the mechanism position.
- RapidVariable – handling variables in the Rapid code. Sets or gets the values of a RAPID variables.

The exchange of control signals between the robot controller and SmartComponent is defined in the Station Logic tab of the RobotStudio environment.

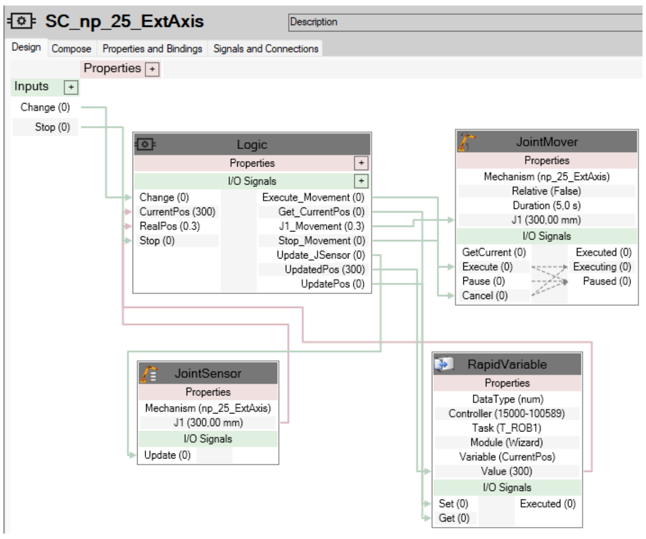


Fig. 5. Design view of a smart component SC_np_25_ExtAxis

The diagram of the data exchange between the virtual robot controller and SmartComponent is shown in Figure 6. The robot controller controls the SmartComponent using two digital signals, where:
- the Change signal triggers movement to the desired position,
- the Stop signal stops the robot movement.

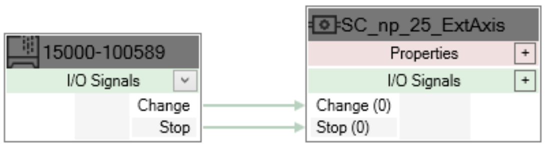


Fig. 6. Station logic diagram.

An Application tab has been added to the main FlexPendant window (fig.7), providing easy access to the developed functionality.

The application (Fig. 8) facilitates the execution of the robot's control program (Fig. 9) and enables the management of the GoFa robot's track and gripper (Figs. 10-11). Figures 9-11 present the corresponding application interfaces [18].

The developed application for track control provides the following functionalities:
- Production Control (Fig. 9) – comprises two sections:
  - *Production*: Enables management of the production process, including starting, pausing, and stopping the program, setting the program pointer, and moving the robot to the home position in both manual and automatic modes.
  - *States*: Displays the system's current states, including robot mode and motor status.

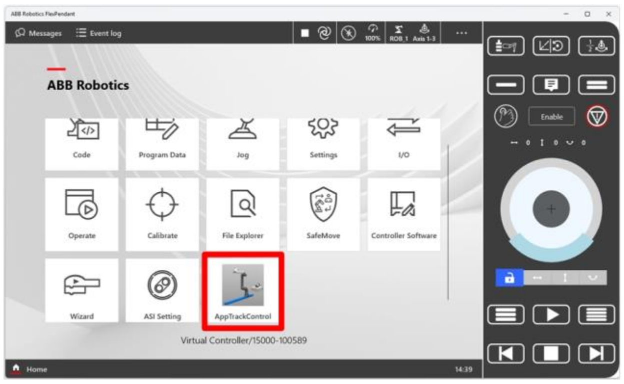


Fig. 7. Main FlexPendant window

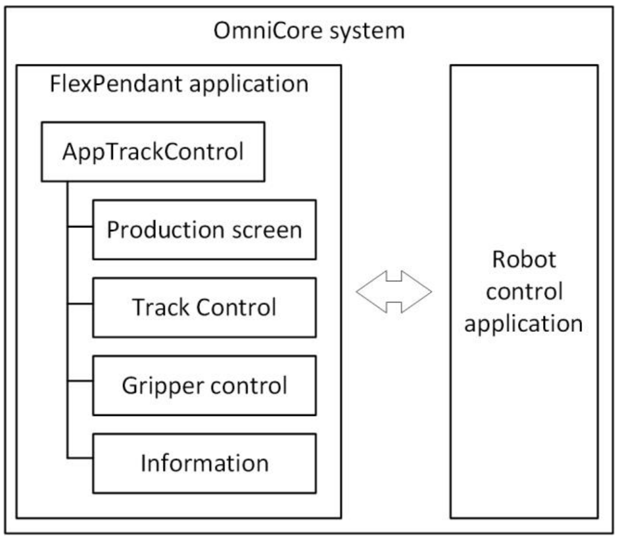


Fig. 8. Developed application structure

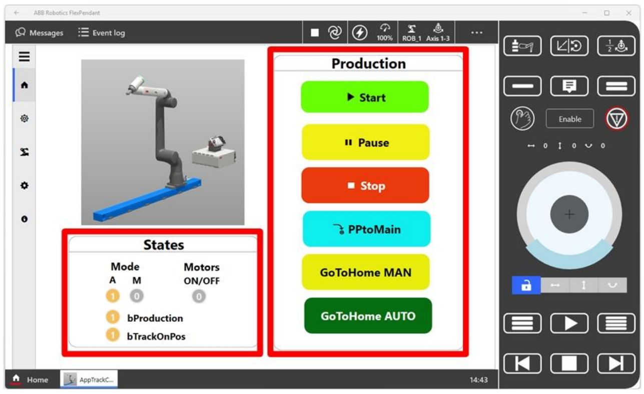
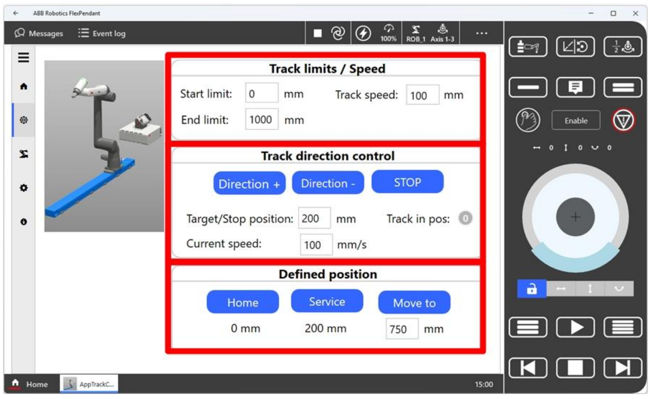


Fig.9. Production window.

- Track Control (Fig. 10):
  - Configuring movement constraints (track limits, track velocity).
  - Controlling the track movement direction within the specified limits.
  - Monitoring the robot's position along the track.
  - Sending the robot to predefined positions (Home, Service) or user-specified positions (Move To) within the defined limits.



Track limits / Speed

Start limit: 0 mm   Track speed: 100 mm

End limit: 1000 mm

Track direction control

Direction +   Direction -   STOP

Target/Stop position: 200 mm   Track in pos:

Current speed: 100 mm/s

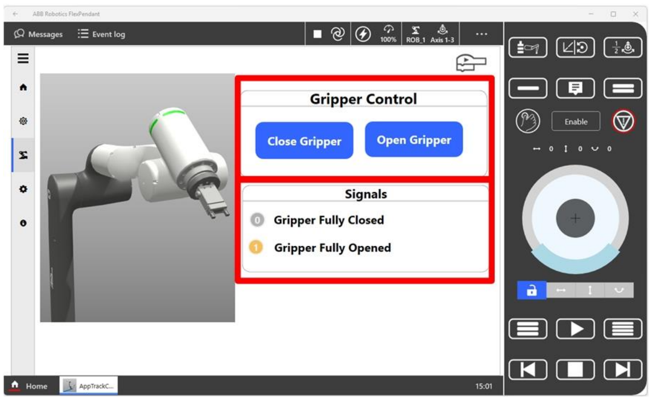
Defined position

Home   Service   Move to

0 mm   200 mm   750 mm

Fig. 10. Track control window.

- Gripper Control (Fig. 11) – Allows manual operation of the gripper and provides real-time feedback on the gripper's status.



Gripper Control

Close Gripper   Open Gripper

Signals

Gripper Fully Closed

Gripper Fully Opened

Fig. 11. Gripper control window.

The AppTrackControl application was developed using ABB AppStudio and Visual Studio Code, featuring the following components:

- Graphical User Interface (GUI): Designed and implemented using AppStudio.
- Scripts: Written in JavaScript to define the displayed graphical elements and facilitate communication with the robot controller.

Below is an example of a function handling the MoveTo button. Lines 1-10 collect the values entered by the operator on the FlexPendant that determine the movement of the robot on the track. The movement position (MoveTo) must be within the permitted movement range (StartLimit, EndLimit). Additionally, the movement speed value (TrackSpeed) is collected. In lines 13-28, a new position and speed of the track movement are set (the CurrentPos and CurrentTrackSpeed variables in the RAPID program are overwritten). After the motors are switched on, the program is executed.

The developed application interfaces with the routines executed on the robot controller. For example, the control of the speed and position of the track is performed using the routine shown in figure 13. In the first part of the code (Fig.13), numeric data of the persistent type storing movement parameters were declared. The second part presents the procedure for implementing the group signal control sequence allowing for the execution of the desired movement. The movement of the track is realized only when its current position differs from the set position. The implemented logic sequence results from the specific nature of the track control.

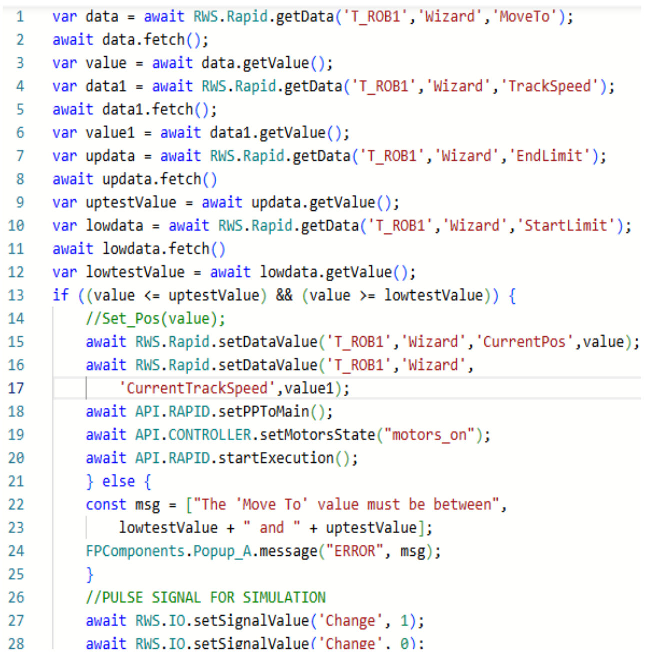

```
1   var data = await RWS.Rapid.getData('T_ROB1','Wizard','MoveTo');
2   await data.fetch();
3   var value = await data.getValue();
4   var data1 = await RWS.Rapid.getData('T_ROB1','Wizard','TrackSpeed');
5   await data1.fetch();
6   var value1 = await data1.getValue();
7   var updata = await RWS.Rapid.getData('T_ROB1','Wizard','EndLimit');
8   await updata.fetch()
9   var uptestValue = await updata.getValue();
10  var lowdata = await RWS.Rapid.getData('T_ROB1','Wizard','StartLimit');
11  await lowdata.fetch()
12  var lowtestValue = await lowdata.getValue();
13  if ((value <= uptestValue) && (value >= lowtestValue)) {
14      //Set_Pos(value);
15      await RWS.Rapid.setDataValue('T_ROB1','Wizard','CurrentPos',value);
16      await RWS.Rapid.setDataValue('T_ROB1','Wizard',
17          'CurrentTrackSpeed',value1);
18      await API.RAPID.setPPToMain();
19      await API.CONTROLLER.setMotorsState("motors_on");
20      await API.RAPID.startExecution();
21      } else {
22      const msg = ["The 'Move To' value must be between",
23          lowtestValue + " and " + uptestValue];
24      FPComponents.Popup_A.message("ERROR", msg);
25      }
26      //PULSE SIGNAL FOR SIMULATION
27      await RWS.IO.setSignalValue('Change', 1);
28      await RWS.IO.setSignalValue('Change', 0);
```

Fig. 12. JavaScript – Move To button.

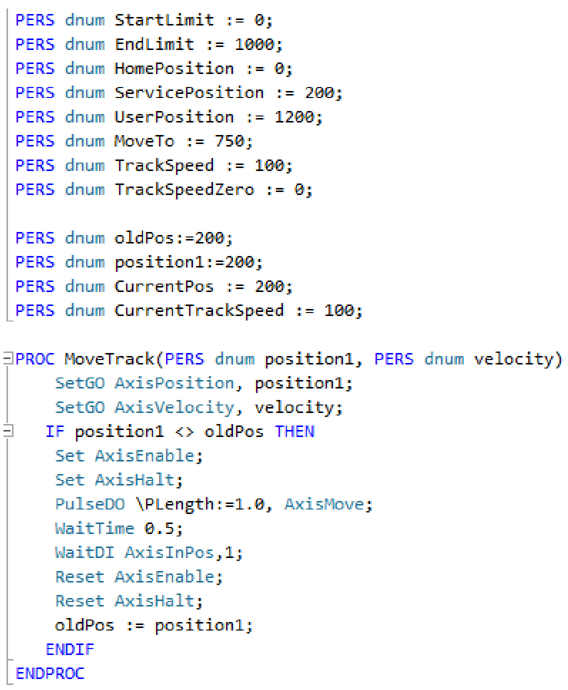
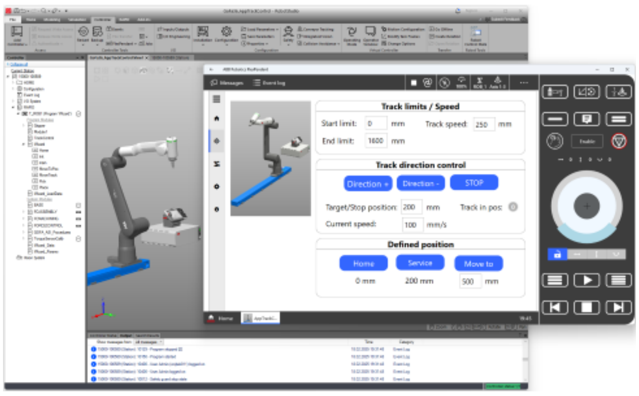

```
PERS dnum StartLimit := 0;
PERS dnum EndLimit := 1000;
PERS dnum HomePosition := 0;
PERS dnum ServicePosition := 200;
PERS dnum UserPosition := 1200;
PERS dnum MoveTo := 750;
PERS dnum TrackSpeed := 100;
PERS dnum TrackSpeedZero := 0;

PERS dnum oldPos:=200;
PERS dnum position1:=200;
PERS dnum CurrentPos := 200;
PERS dnum CurrentTrackSpeed := 100;

PROC MoveTrack(PERS dnum position1, PERS dnum velocity)
    SetGO AxisPosition, position1;
    SetGO AxisVelocity, velocity;
   IF position1 <> oldPos THEN
    Set AxisEnable;
    Set AxisHalt;
    PulseDO \PLength:=1.0, AxisMove;
    WaitTime 0.5;
    WaitDI AxisInPos,1;
    Reset AxisEnable;
    Reset AxisHalt;
    oldPos := position1;
   ENDIF
ENDPROC
```

Fig. 13. RAPID – track control routine.

The input data of the routine are position and velocity. These values are converted into group signals and sent to the PLC controller. The program then checks if the target position is different from the position occupied by the robot. If the positions are different, a signal is given to run the axis until the target and current positions are equal.

Interface performance tests were performed in RobotStudio using a digital twin (Fig.14,15).

Figure 15 shows selected test cases in RobotStudio. In the presented case, the parameters listed in Table 1 were set.



Track limits / Speed

Start limit: 0 mm Track speed: 250 mm

End limit: 1600 mm

Track direction control

Direction + | Direction - | STOP

Target/Stop position: 200 mm Track in pos:

Current speed: 100 mm/s

Defined position

Home 0 mm | Service 200 mm | Move to 500 mm

Fig. 14. RobotStudio and virtual FlexPendant

a) Home: pos = 0 mm b) Service: pos = 100 mm

c) User: pos = 500 mm d) End limit: pos = 1600 mm

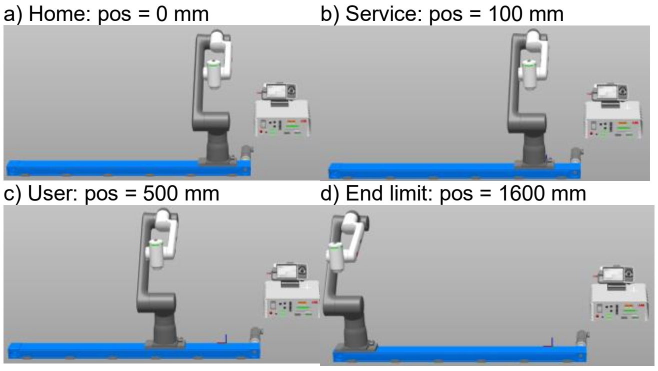


Fig.15. RobotStudio – robot positions during testing - example.

Table. 1. Test in RobotStudio/real station - parameters

Parametr	Value [mm]
Start limit	0
End limit	1600
Home position	0
Service position	100
User position	500
End position	1600

After conducting tests in RobotStudio on the station with the virtual twin and confirming the correct operation of all the implemented functionalities, the application was uploaded to the real robot (Fig. 16). No errors in the operation of the application were observed. The application is flexible and can be easily expanded with new functionalities.

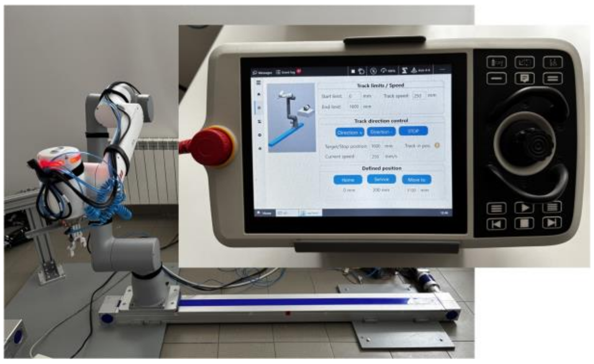


Fig. 16. Real robot

Figure 17 shows exemplary test results on a real test stand. In the presented case, the parameters set in Table 1 were used.

a) Home: pos = 0 mm b) Service: pos = 100 mm

c) User: pos = 500 mm d) End limit: pos = 1600 mm

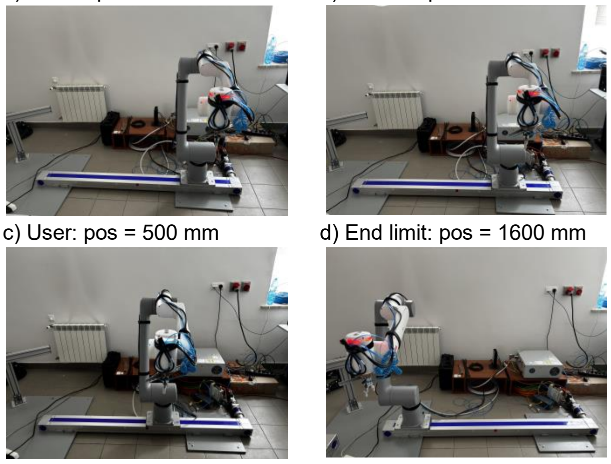


Fig. 17. Real robot – robot positions during testing - example

Conclusions

Data digitization has caused the amount of information and data complexity to increase significantly, which makes it necessary to create advanced operator panels that present the flood of this knowledge in the form of simple and clear messages that are easily digestible by operators. This allows for quick response to emerging process events and easy monitoring, supervision and control of machines.

The primary objective of the authors was to develop an operator panel for controlling the track of the ABB GoFa collaborative robot. The application was created using the AppStudio and Visual Studio Code environments. It enables the operator to manage the track and gripper in manual mode and to control the production process. The robot station, including the track, was integrated using the SIEMENS S7-1200 controller [17]. Communication between the PLC and the robot controller was established via the PROFINET network. The method of device integration and the operational testing of the system were detailed in [17].

In this article, the authors focus on the development of the application and its integration with both the digital twin and the physical robot. This integration facilitates software development without requiring the operation of the physical robot. Tests conducted in RobotStudio and on the physical station confirmed the correct functionality of the application. The dedicated application significantly simplifies the operation of the robotic workstation, both for individuals with expertise in industrial robot operation and programming, and especially for personnel without such knowledge.

In the next stage, the application will be expanded to include a screen for creating recipes related to the repackaging of courier parcels. This enhancement will enable the automation of the sorting process for parcels of various sizes.

***Authors**: Wojciech Kaczmarek, Szymon Borys, Military University of Technology, Faculty of Mechatronics, Armament and Aerospace*

REFERENCES

[1] Yang, J, Liu, T, Liu, Y, & Morgan, P. "Review of Human-Machine Interaction Towards Industry 5.0: Human-Centric Smart Manufacturing." *Proceedings of the ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. *Volume 2: 42nd Computers and Information in Engineering Conference (CIE)*. St. Louis, Missouri, USA. August 14–17, 2022. V002T02A060. ASME. https://doi.org/10.1115/DETC2022-89711.

[2] Singh, H. P., & Kumar, P. (2021). Developments in the human machine interface technologies and their applications: review. *Journal of Medical Engineering & Technology*, *45*(7), 552–573. https://doi.org/10.1080/03091902.2021.1936237.
[3] Mourtzis, D., Angelopoulos, J., & Panopoulos, N. (2023, May 1). The Future of the Human–Machine Interface (HMI) in Society 5.0. Future Internet. MDPI. https://doi.org/10.3390/fi15050162.
[4] Prusaczyk P., Kaczmarek W., Panasiuk J., Besseghieur K., Integration of robotic arm and vision system with processing software using TCP/IP protocol in industrial sorting application. *AIP Conf. Proc. 1* March 2019; 2078 (1): 020032. https://doi.org/10.1063/1.5092035.
[5] Park, IH., Yoon, J.S., Sohn, J.H. *et al.* Platform Supporting Intelligent Human–Machine Interface (HMI) Applications for Smart Machine Tools. *Int. J. Precis. Eng. Manuf.* **25**, 1073–1086 (2024). https://doi.org/10.1007/s12541-024-00960-6.
[6] Selwin Mich Priyadharson, A., Vinson Joshua, S., & Thilip Kumar, C. (2015). PLC - HMI and Ethernet based monitoring and control of MIMO system in a petrochemical industry. *Indian Journal of Science and Technology*, *8*(27). https://doi.org/10.17485/ijst/2015/v8i27/72100.
[7] Normanyo, E., Husinu, F., & Agyare, O. R. (2014). Developing a Human Machine Interface (HMI) for Industrial Automated Systems using Siemens Simatic WinCC Flexible Advanced Software. *Journal of Emerging Trends in Computing and Information Sciences*, *5*(2), 134–144. Retrieved from http://www.cisjournal.org.
[8] Ardanza, A., Moreno, A., Segura, Á., de la Cruz, M., & Aguinaga, D. (2019). Sustainable and flexible industrial human machine interfaces to support adaptable applications in the industry 4.0 paradigm. *International Journal of Production Research*, *57*(12), 4045–4059. https://doi.org/10.1080/00207543.2019.1572932.
[9] Aswardi, A., Yanto, D. T. P., Dewi, C., Zaswita, H., Kabatiah, M., & Kurani, R. (2023). Human Machine Interface-Based Control Training Kit as Innovative Learning Media to Enhance Students' Automation Control Skills in the Industry 4.0 Era. *TEM* Journal, 12(4), 2157–2165. https://doi.org/10.18421/TEM124-26.
[10] Borys Sz., Kaczmarek W., Laskowski D., Selection and Optimization of the Parameters of the Robotized Packaging Process of One Type of Product. *Sensors* (ISSN 1424-8220), 2020, 20, 5378. doi: 10.3390/s20185378.
[11] Kaczmarek, W., Lotys, B., Borys, S., Laskowski, D., & Lubkowski, P. (2021). Controlling an industrial robot using a graphic tablet in offline and online mode. Sensors, 21(7). https://doi.org/10.3390/s21072439.
[12] Balla, M.; Haffner, O.; Kučera, E.; Cigánek, J. Educational Case Studies: Creating a Digital Twin of the Production Line in TIA Portal, Unity, and Game4Automation Framework. *Sensors* **2023**, *23*, 4977. https://doi.org/10.3390/s23104977.
[13] Alaameri, K. J., Ramadhan, A. J., Fatlawi, A., & Idan, Z. S. (2024). Design of a new sorting colors system based on PLC, TIA portal, and factory I/O programs. *Open Engineering*, *14*(1). https://doi.org/10.1515/eng-2022-0547.
[14] Product manual OmniCore C30, Document ID: 3HAC060860-001, Revision: U, ABB.
[15] Product manual CRB 15000, OmniCore, Document ID: 3HAC077389-001, Revision: L, ABB.
[16] Application manual PROFINET Controller/Device, RobotWare 7.16 Document ID: 3HAC066558-001 Revision: P, ABB.
[17] Borys, S., Kaczmarek, W., Integration of the GoFa collaborative robot with a linear track using PROFINET. Przegląd Elektrotechniczny, 2024, (12), ISSN 0033-2097, R. 100 NR 12/2024, pp. 32–36, DOI: 10.15199/48.2024.12.08.
[18] ROBOTICS Tutorial, How to create an OmniCore™ Web App, ABB.