Annals of the Academy of Romanian Scientists
Series on Science and Technology and Information
Online ISSN **2066 - 8562** Volume **16**, Number 1-2/**2023**

EDGE ARCHITECTURE FOR ROBOT DATA COLLECTING IN A DIGITAL TWIN

Lenţoiu IONUŢ[1], Borangiu THEODOR[2], Răileanu SILVIU[3]

Abstract. The paper describes a software system for data collecting in robot digital twins (DT); this system accesses information and data from the robot, the process automated by the robot and the devices connected to the robot (conveyor belt, ASRS, smart meter) via an edge processing structure that includes the robot controller and IoT gateways. The software system includes a data acquisition agent directly connected to the edge processing hard-ware, a database where the collected information is stored and a user interface with multiple data display options. The designed DT software collects robot data in two modes: continu-ously from the robot controller and the IoT gateways using specific software tools available from the robot manufacturer, and discretely from program instructions by messages. Experiments with the DT data collecting system are given for ABB IRC5 robot controllers.

DOI https://doi.org/10.56082/annalsarsciinfo.2023.1-2.21

Keywords: Digital manufacturing, digital twin, industrial robot, edge layer, data acquisition.

1. Introduction

The Factory of the Future (FoF) initiative assumes the digital transformation of manufacturing processes and the strong coupling of shop floor devices, systems and services by means of their digital counterparts interconnected horizontally in Cyber Physical Systems (CPS) frameworks [1, 2] and vertically at enterprise level [3]. In addition to the virtualization of manufacturing entities (resources, products, orders) and the strong coupling of their software complements, data collection and real time processing play an important role in CPS: data acquisition, information aggregation from multiple sources, data storage and analytics in the cloud allow for equipment status monitoring and detecting unexpected events, predicting failures and tailoring maintenance, optimal operations scheduling and assigning to working resources in process supervision and control or simulation with preconfigured device layout (e.g., industrial robots) in the design stage [4]. These monitoring, control, supervi-sion and design tasks are performed in the global perspective of aggregated context (device, process, workplace) by help of the Digital Twin (DT) technology [5].

[1]Ph.D. student, University Politehnica of Bucharest, ionut.lentoiu@stud.acs.upb.ro
[2]Professor, Ph.D., University Politehnica of Bucharest, Academy of Romanian Scientists, theodor.borangiu@upb.ro
[3]Associate Professor, University Politehnica of Bucharest, silviu.raileanu@upb.ro

The complete view of the operating mode, status and real-time capabilities of a manufacturing device (e.g., a robot) comprising its virtual motion model, running context, state history, and changes in time of its working parameters may be captu-red in a digital twin construction defined as a complete digital model of the robot that can be inspected and used even in the absence of the physical robot system.

The general DT construct includes the elements: a) the physical device in the material world; b) a software entity in the virtual space; c) a number of connections between the physical and virtual systems [6]. In the industrial domain, a DT consists of a digital representation of an individual production device (e.g., robot, end-effector) or collection of devices (e.g., robot team) that can execute in different soft-ware environments to orchestrate the two virtual and real systems by help of sensor data according to physics models, mathematical algorithms and data transforma-tions. From the reality modelling perspective there are two classes of DTs: i) DT with a physical component, also known as *data-driven* DT which collects data from the robot, the process automated with the robot and its partner devices (other robots, the conveyor belt and PLC, etc.); in this case the virtual twin is synchronized with the physical twin (the robot system); ii) DT without a material part or *model-driven* DT which is a digital model of the robot and its controller that can be run and used for layout validation, application design, and parameter tuning [7].

There are advantages in using the digital twin technology: a) *Perceptibility*: DTs offer knowledge on how the operations of individual robots and interconnected members of robot teams are executed, such as collaborative welding robots [8]; b) *Prediction*: using modelling techniques the DT predicts future states, behaviours or parameter evolutions, such as robot energy consumption; c) *Interaction* with the physical twin: embedding DT in robot control (tracking and correcting motion, detecting unexpected events, taking ad hoc protective decisions) [9]; d) *Analysis*: activating triggering procedures to clarify and give details about changes in states and performances of robots for health monitoring and maintenance.

The present paper concerns the software design of data-driven DTs for industrial robots. Chapter 2 presents the architecture of the DT embedded in individual robot control with robust deployment, and concurrent task allocation to robot team mem-bers. Chapter 3 describes the DT data stream collecting layer designed as a software system for the edge computing framework of the industrial robot. Chapter 4 offers experimental results obtained for ABB robots with the proposed edge computing DT layer. Conclusions and perspectives of future research are given in Chapter 5.

2. Robot DT embedded in robust control and concurrent task allocation

A Digital Twin model has been designed by aggregating (DTA) a number of Digital Twin Instances (DTI) each of them representing a particular material instan-ce of an individual robot; by aggregating these multiple instances, it becomes possi-ble to query information about clusters of robots that build up composite, re-configurable production structures. The DTA is designed to be embedded (EDT) in individual robot monitoring and maintenance, and in supervision and concurrent assignment of production tasks to multi-robot manufacturing infrastructures. This EDT is scalable and can be software configured according to production needs; it features bidirectional connectivity between the physical shop floor entities (robot manipulator, controller) and its informational model by the acquisition of data from internal robot sensors and specific software operations of the robot language, and by providing support to intelligent decisions for: authorizing / changing / stopping robot motions, reconfiguring working parameters (e.g., speed), selecting best suited team members for batch production tasks - in a private cloud platform.

Two strategies are deployed in software configurations of the robot DT model:

1) *Robust deployment*: DTIs maintain individual robot control with situation aware-ness. Robot data is fed to the virtual twin (robot model), which allows detecting when the output of the executable robot model deviates (a) or differs significant-ly (b) from the work parameters of the physical robot. Robot health monitoring is realized in two real time stages: i) robot and/or process parameter reconfigu-ring in case (a), and ad hoc robot disabling in case (b) to avoid any damages. This situation awareness permits diagnostics from parameter covariance analysis with separation of concerns: monitoring/diagnostic/predicting maintenance.
2) *Simulation of collaborative work much faster than real-time*: This is the case when the robot DTA acts as concurrent task allocator for multiple robot team members, by help of a meta-level control mechanism which executes in parallel a *simulation of production with software in the loop* (SIL) - much faster than real time (computer time) and the *robust deployment*. At certain moments of time the supervision of the robot team may take certain actions (alter setup parameters, reschedule tasks at robot failure) in the current robust deployment configuration in response to some perturbations. The meta-level will activate in such cases a SIL configuration (fast simulation with software agents acting as virtual twins) that uses the embedded services of robot DTs, which observes much in advance problems and avoid their occurrence. The meta-level will run a high number of such embedded robot work simulations, exactly like the heuristics for job assign-ment. Each virtual robot twin in SIL configuration may change intentions as they bid for manufacturing jobs with

weights changed by the evolution of the robot's performances (e.g., energy consumption). This SIL configuration may also run with prediction of physical twin performances, which permits the meta-level to reschedule tasks on robots at the next update of settings/intentions [10].

The 4-layer DTA model of robot team members in smart manufacturing is given in Fig. 1. The Digital Twin layer DT I collects robot data in an *edge* framework; the data is then processed on the DT II *fog* layer. The higher level DT III and DT IV perform robot health monitoring, supervised control and concurrent task allocation.

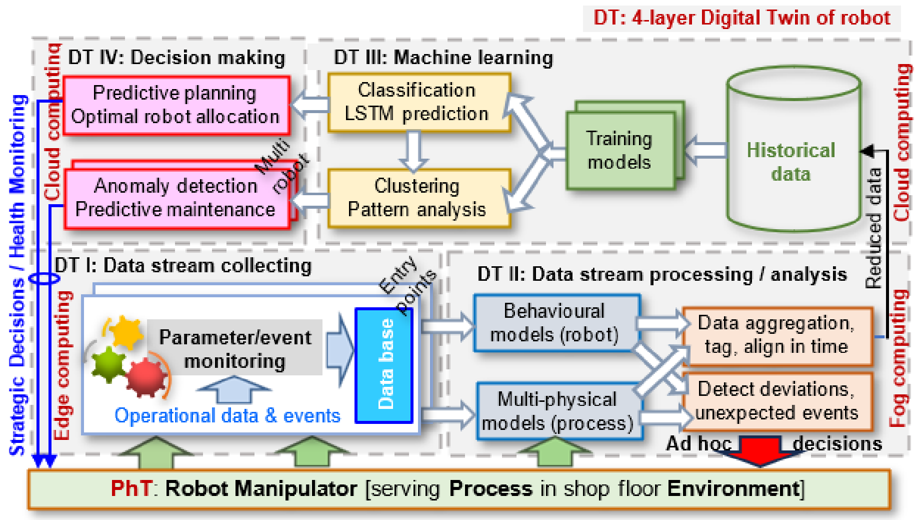

DT: 4-layer Digital Twin of robot
DT IV: Decision making
Predictive planning
Optimal robot allocation
Anomaly detection
Predictive maintenance
Multi robot
DT III: Machine learning
Classification
LSTM prediction
Clustering
Pattern analysis
Training models
Historical data
Cloud computing
Cloud computing
Strategic Decisions / Health Monitoring
Edge computing
DT I: Data stream collecting
Parameter/event monitoring
Data base
Operational data & events
Entry points
DT II: Data stream processing / analysis
Behavioural models (robot)
Multi-physical models (process)
Data aggregation, tag, align in time
Detect deviations, unexpected events
Reduced data
Fog computing
Ad hoc decisions
PhT: Robot Manipulator [serving Process in shop floor Environment]

Figure 1. 4-layer aggregate digital twin embedded in individual reality-aware robot control with health monitoring, and concurrent manufacturing task allocation to robot team members

DT I: Data collecting. This DT part is linked to the physical components of the robot system (manipulator - actuators, end-effector, tool; controller - motor drives, multiprocessor boards; 3D vision - camera, lights; external sensors - energy, vibra-tions), to the process it tends and to the workplace environment (assembly station, welding cabinet) and collects data from multiple sources acting as entry points.

DT II: Data processing and analysis. The role of this DT part is to separate and align the data streams generated by the devices included in DT I in normalized time slices, and to join them by help of covariance methods. This module is composed by four functional elements:

- *Container of robot state and performance models*: it stores robot operating models that reflect with high accuracy the operating modes robot system (e.g., computer, dry run, manual, etc.). This repository includes also references to the robot's working parameters that might be periodically updated at run time function of the current work sequence (e.g., the six components of the robot transformation TOOL, the robot-object grasping model parameters, a.o.).

- *Container of physical process models*: this repository stores models of the processes that are automated by the robot; a change in the robot operating parameters induces changes in the parameters of the process models.
- *Data joining and temporal alignment*: raw data streams labelled according to the previous two models are combined in application-specific flows streams (e.g., resource topic, product topic). Typical operations performed on this DT layer are join/merge parallelized on multiple nodes in map-reduce clusters. After assembling the data necessary for an operation in a 'joined' stream the 'reduce' operation follows. Data can be combined in different ways by help of several map-reduce keys. For example, messages can be labelled by their workplace location or robot energy measurement.
- *Detection of anomalies and unforeseen events*: decisions are issued by the software on this DT layer for immediate power disabling and robot stop at robot failure or when unforeseen events occur. Alerts are triggered when some sensed robot parameters deviate significantly from the normal range.

DT III: Machine learning. This higher level cloud-based DT part gets essential information retrieved and pre-processed in real time at shop floor level from the connected robot system and external devices (equipment, process, work area). This DT module uses machine learning (ML) algorithms to supervise and maintain the proper functioning of the robot and to optimize its partaking in individual (energy consumption) and collective manufacturing tasks (throughput, execution time): *prediction* to identify patterns and big deviations from measured signals and on this basis to predict robot behaviours; *classification* to establish classes for feature vectors created; *clustering* to detect similarities in aggregated data and label feature vectors. Historical data stored in the cloud is needed to learn these models.

DT IV: Taking smart decisions. This DT software runs in the cloud ensuring the high level support for intelligent decisions: i) keeping safe robot operating, providing evidence for robot maintenance; ii) optimizing the allocation of manufacturing tasks to robot team members depending on the currently measured and predicted robot state and key performance indicators (e.g., joint wear, accuracy, energy).

The next chapter details the DT I software architecture for robot data collecting.

3. DT software architecture for robot data collecting

In order to collect i) robot sensor data, results of instructions executed in robot application programs, and 2) data generated by external control devices (PLCs, conveyor devices) and sensors with embedded intelligence (smart energy meters), the robot controller and the external measurement and control devices are integrated in a network of IoT gateways for DT edge processing on the DT I layer [13]. The IoT gateways are connected in an aggregation node configuration [14], the hardwa-re kernel of which is a PC workstation or an industrial processing unit.

Fig. 2 shows presents the edge processing architecture of the aggregate DT I layer for a team of n industrial robots (Ind Rob $1, \dots, n$) that are concurrently assigned in manufacturing tasks (welding, assembly). Each resource (Ind Rob $i, i = 1, \dots, n$) offers:

1. *Process data* from Producer 1 - process manager and robot: outcomes of specific robot programming language (e.g., V+, RAPID) instructions indicating the activity of the robot and work performed in the process served by the robot. Examples of such data: robot inactive duration (%); robot run time (%); parts worked per time unit; parts assembled / palletized; parts recognized / located by vision; parts successfully managed by robot vision.

2. *Internal robot data* from Producer 2 - robot controller: multiprocessor bus power and voltage (W, V); DC voltage (V); computer board temperature (°C); joint encoder temperature (°C); motor drive temperature (°C); duty factor of robot (% from limit); harmonic drive usage (%); max. active torque (% from max. allowed torque); max. velocity (RPM); max. trajectory error (%).

3. *External robot and environment data* from Producer 3 … *m* - IoT gateways and smart embedded devices: work-in-process data from intelligent products travel-ling between robot workstations; data from robot-serviced conveyor belts (belt speed [mm/sec]; total part count; parts per minute; deadlocks, diverting faults); inventory data in ASRS; vision data (scene luminosity, no. of lamps switched on, no. of active virtual cameras), energy consumption data from smart meters.

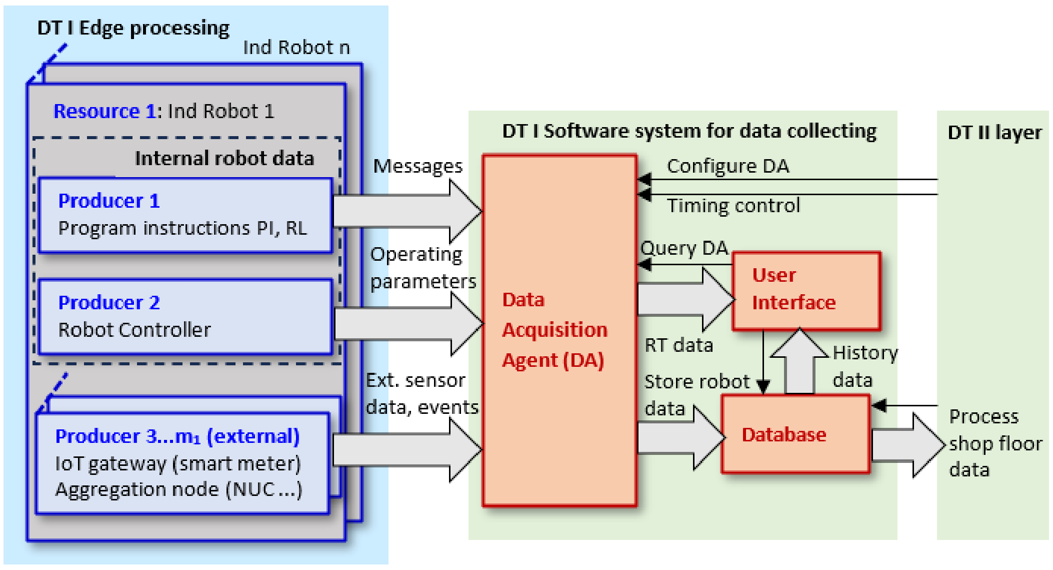


Figure 2. Edge processing and software system for data collecting in DT layer I

The software architecture of the DT I layer system is represented in the right part of Fig. 2, and includes three components: the data acquisition agent (DA), the database and the user interface. The main software component is the DA which performs da-ta collection from the $m_i, 1 \leq i \leq n$ producers of the n industrial robots, in two modes:

- Continuously, from the robot controller and external IoT gateways, at sampling periods a priori established, by help of a specific software tool available from the robot system manufacturer.
- Event triggered, from the program instructions (PI) of the robot programming language (RL), by messages received from the application program.

Robot data and data from objects connected to the robot (conveyor belt, vision) can be, e.g., directly saved from Omron-Adept systems using the ACE proprietary software tool and stored into a data file [15]. ACE relies on software processes that execute in the reserved V+ robot operating system's kernel space and get knowled-ge about the robot state. ACE on the PC-type robot terminal is connected to the robot controller via Ethernet; the application is launched with a default workspace that integrates the observed robot after which the System Monitor software tool is initiated and arranged to log data into a specified file at the minimum sampling rate of 0.5 seconds. This file is opened by the System Monitor to transmit the saved information to the higher cloud layer of the robot DT by help of Cygwin Linux tools that run on Windows [16].

In the reported research, robot data is directly collected from IRB ABB industrial robots at time intervals of 0.2 seconds using the Software Development Kit ABB PC SDK [17]. PC SDK is a software tool that uses Microsoft .NET and Microsoft Visual Studio and permits developing custom user interfaces for IRC5 robot controllers. The advantage of using such a tool is the ability to access and monitor multiple robot systems from a single location A custom interface was realized as independent application that communicates with the n robot controllers.

Fig. 3 depicts the software architecture of PC SDK.

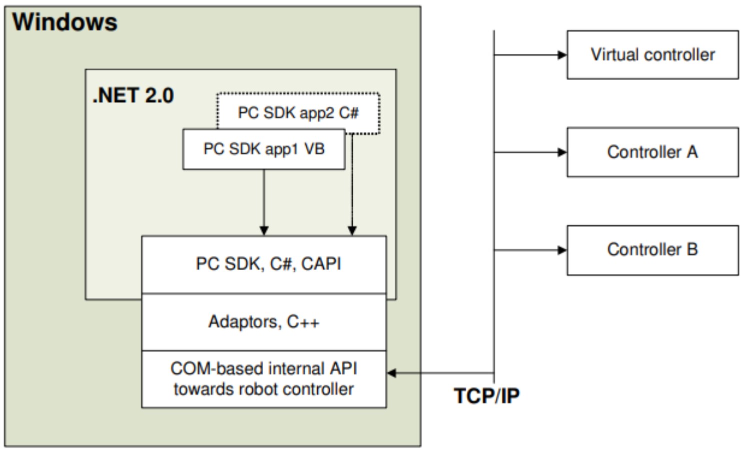


Figure 3. Software architecture of the ABB PC SDK (*ABB Robotics*)

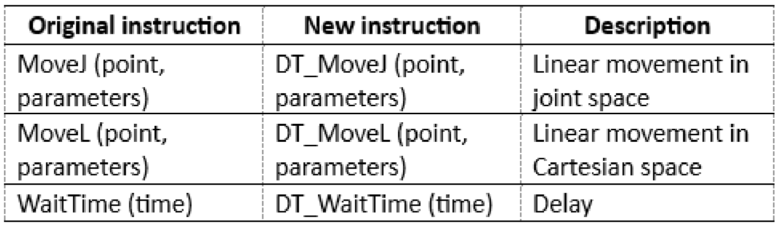
The PC SDK communicate with the industrial robot controller by help of an own Controller API (CAPI) based on the COM technology. This API employs sockets and the local TCP/IP protocol stack to receive and interpret messages from real and virtual controllers. The classes by which the robot controller's functionality can be known constitute the CAPI organised in 11 domains, from which the following are used for robot health monitoring by the digital twin layers DT II and DT IV: Controllers, EventLogDomain, IOSystemDomain, MotionDomain, RapidDomain and Messaging. Event log messages are used to extract knowledge about the con-troller status and run processes, and the execution of RAPID programs.

Data sampling has been established at 200 milliseconds for two main reasons: a) the PC CAPI is unable to ensure hard real-time demands because of the slower network communication rate via TCP/IP and of the robot controller's frequent higher priority tasks (e.g., trajectory generation); b) the minimum response time for IRC5 robot controllers is in the range of 10-100 milliseconds. Using the Controller API the following data can be collected on DT I: status of input / output signals, coordinates of robot points, joint values, information on events, and error messages.

Robot- and process-related data can be collected at discrete time moments, either triggered by events specific to the process served by the robot (e.g., current opera-tion on product finished by the robot, last operation on product finished by the ro-bot) or initiated by the DT to monitor the proper execution of robot operations (e.g., robot motion, end-effector actions, vision-based object recognition and locating); this can be done by the DT's data acquisition agent that processes the messages re-ceived from the robot controller upon execution of particular program instructions that offer the information of interest. For industrial robots, For industrial robots, the transfer of information through PI messages can be done by an auxiliary task that sends the data to the DA without influencing the main program, which assumes that the robot controller has multitasking capabilities.

If this condition is not met, like in the case of IRC5 ABB robot controllers, a solution has been developed that allows simultaneously sending IP information to DT I's data acquisition agent and performing the robot operation. The solution con-sists in redefining those PI of the RL that are of interest for the DT to generate and send messages to the DA. The PIs redefined in Table 1 first send information about the instruction type, prescribed parameters (e.g., desired joint values and estimated execution time, then execute the respective instruction and at PI completion send again information concerning the execution of the instruction (real joint displace-ments and execution time).

Table 1. Redefined RL instructions for message transmission to the DA



Original instruction	New instruction	Description
MoveJ (point, parameters)	DT_MoveJ (point, parameters)	Linear movement in joint space
MoveL (point, parameters)	DT_MoveL (point, parameters)	Linear movement in Cartesian space
WaitTime (time)	DT_WaitTime (time)	Delay

Other PI of interest for DT are related to visual global recognition of the scene foreground, object recognition and locating, visual measurements and calibration.

The DTA's principal functionality consists in monitor the operating mode of the physical team robots and to create their history, which needs a database on the DT I layer to save information about work parameters, execution times, and elements describing robot motion (trajectory execution and speed profile), robot operations, sequences, experiments and applications performed.

The designed database, implemented in MariaDB and MySQL with the XAMPP Apache distribution, is composed of the following tables: 1) Programs; 2) Available Programs; 3) Program Instructions; 4) Available Instructions; 5) Equipment (robot) parameters; 6) Event description; 7) Experiments, see Fig. 4.

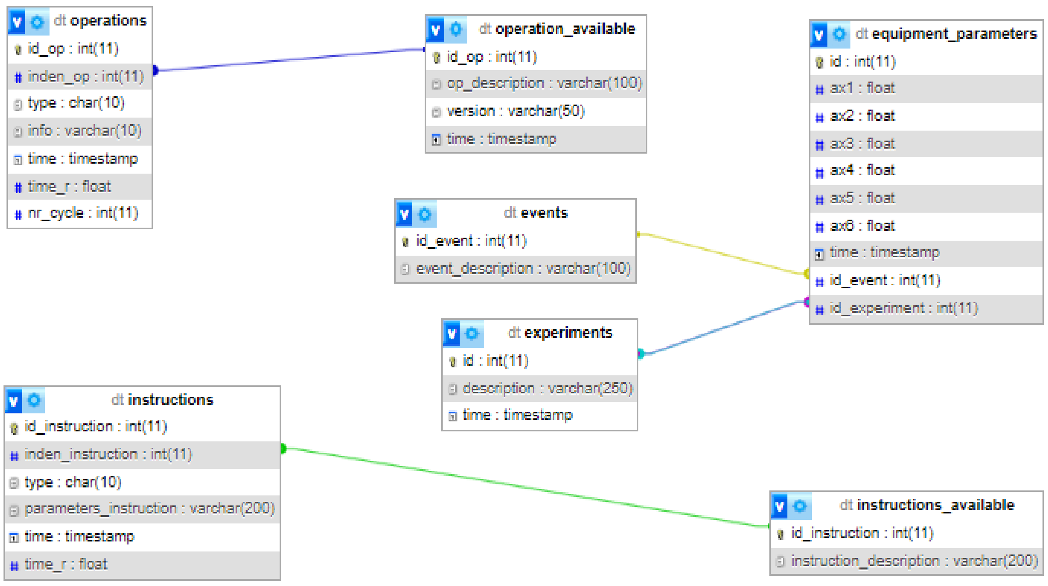


Figure 4. Database structure of the DT I software system for robot data collecting

Table 1 *Programs* stores information about the programs/procedures performed: program identifier; information (parameters, description, auxiliary information); date and time; duration of the current procedure; cycle number (for procedures that is executed several times, like palletizing, sorting, assembling).

Table 1 is in a many-to-one relationship with Tables 2 - *Available Programs* which contain a brief description of the program/procedure; version; date and time.

Table 3 *Instructions* saves information concerning the instructions: instruction identifier; type of instruction; parameters of the instruction; date and time; time at which the instruction is executed relative to the beginning of the current procedure.

This table is in a many-to-one relationship with Table 4 - *Available Instructions* which contain a brief description of the existing instructions.

Table 5 *Equipment parameters* contains information regarding the parameters of an equipment (e.g., 6-axis industrial robot): joint values; date and time; event identi-fier (if an event occurs); identifier of the current experiment.

This table is in a many-to-one relationship with Tables 6 - *Event description* and Table 7 - *Experiments* which contain a description of the possible events respective-ly a description of the experiments performed along with the date and time of the experiment.

A user interface was developed in the DT I software system to present the robot data collected in real-time and the robot, process and workplace environment infor-mation saved in the database. This user interface is composed of three data presenta-tion windows:

- **Monitor** is the window which includes the server management elements (start/stop, status, etc.), the elements for message display, the values of various parameters received in real-time (e.g., joint values, no. of parts recognised, located and palletised) and the initialisation elements for the current experiment.
- **Info** is the window in which the general information about the connection with clients (e.g., robot controllers, robot-served devices) and database infor-mation display modes are presented.
- **Info2** is the window in which the work parameters (e.g., joint values, scene foreground) recorded and saved from previous experiments are visualised, see Fig. 5.

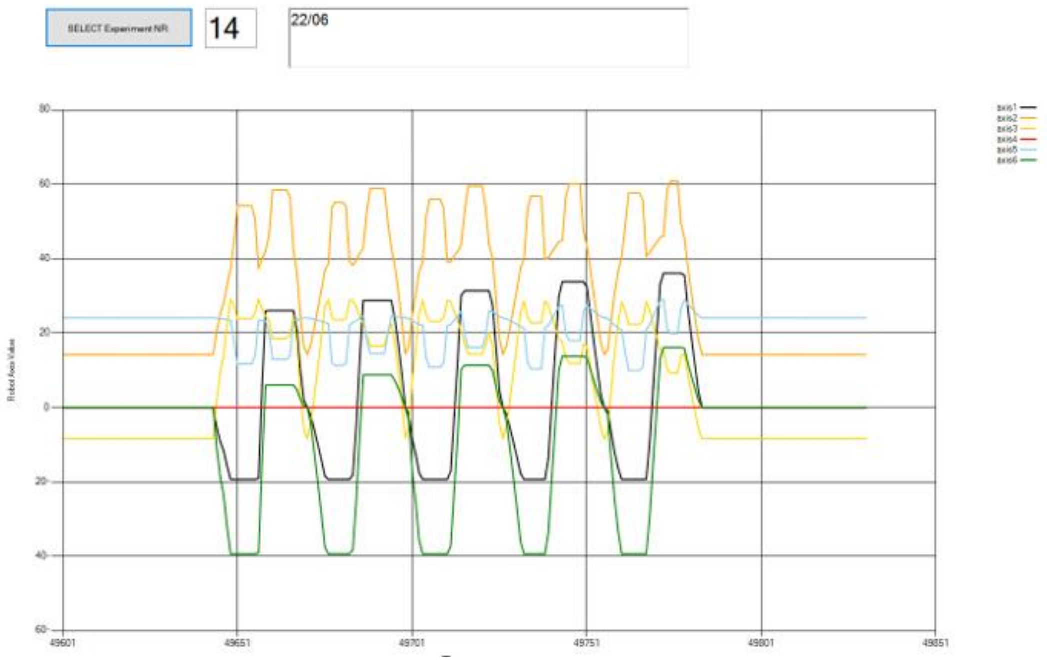


Figure 5. Values of 6-d.o.f. robot joint variables for CNC workplace tending point recorded in previous Experiment no. 14 and displayed in the Info2 window of the DT I user interface

4. Experimental results

The Digital Twin software developed for robot data collecting was tested with a 6-d.o.f. vertical articulated industrial robot ABB IRB140. The test program implements a simple part palletising procedure in a 1D part stack, as shown in Fig. 6, a screen capture of the corresponding model-driven RobotStudio application [18].

Using the GetPosition() function of the CAPI PC SDK 5.13 class library motion domain objController.MotionSystem.ActiveMechanicalUnit.GetPosition(), the set of joint values are saved in the C# program in a variable of aJointTarget type as working parameters of interest, accessed then with AJointTarget.RobAx.Rax_no PI

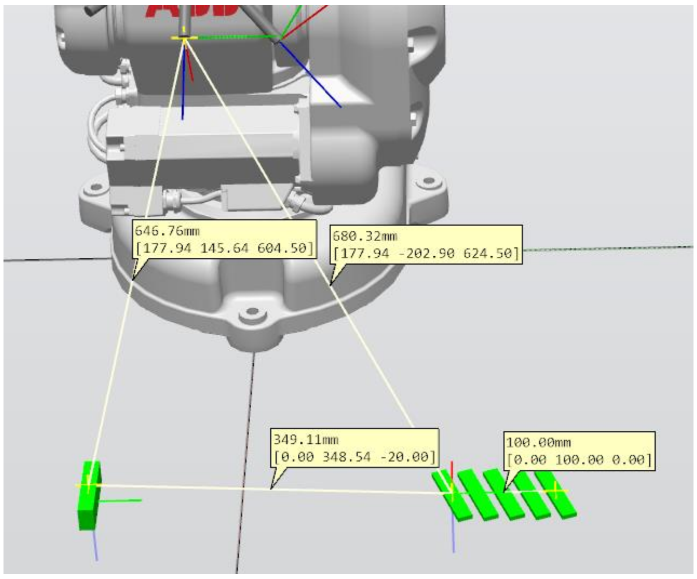
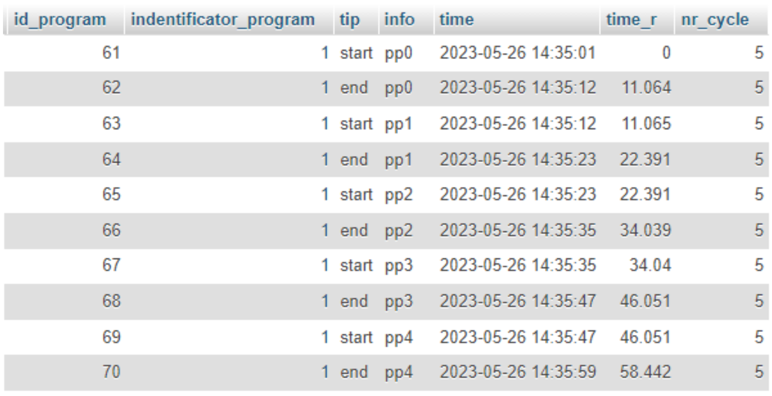


Figure 6. Robot Studio screen capture for 1D palletising task in DT I software test experiment

The program information for the 5 pick-and-place (pp) robot cycles in included in Fig. 7.



id_program	indentificator_program	tip	info	time	time_r	nr_cycle
61	1	start	pp0	2023-05-26 14:35:01	0	5
62	1	end	pp0	2023-05-26 14:35:12	11.064	5
63	1	start	pp1	2023-05-26 14:35:12	11.065	5
64	1	end	pp1	2023-05-26 14:35:23	22.391	5
65	1	start	pp2	2023-05-26 14:35:23	22.391	5
66	1	end	pp2	2023-05-26 14:35:35	34.039	5
67	1	start	pp3	2023-05-26 14:35:35	34.04	5
68	1	end	pp3	2023-05-26 14:35:47	46.051	5
69	1	start	pp4	2023-05-26 14:35:47	46.051	5
70	1	end	pp4	2023-05-26 14:35:59	58.442	5

Figure 7. List of robot working programs for the 5-cycle palletising application (Experiment 2)

The execution of the robot program starts at 0.5 seconds from launching the ABB PC SDK tool; this time period is needed to create the client and realize the connec-tion for data transmission. In order to collect variables, signals and robot working parameters, a controller-type object was created using the builder Controller(object ControllerInfoCollection).

The total processing time for the 5-pp cycle palletising application with robot speed limited at 50% of its maximum value is 58.442 seconds. Fig. 8 presents the Info2 user interface display for the evolution of the joint displacements in the test palletising application (Experiment 2).

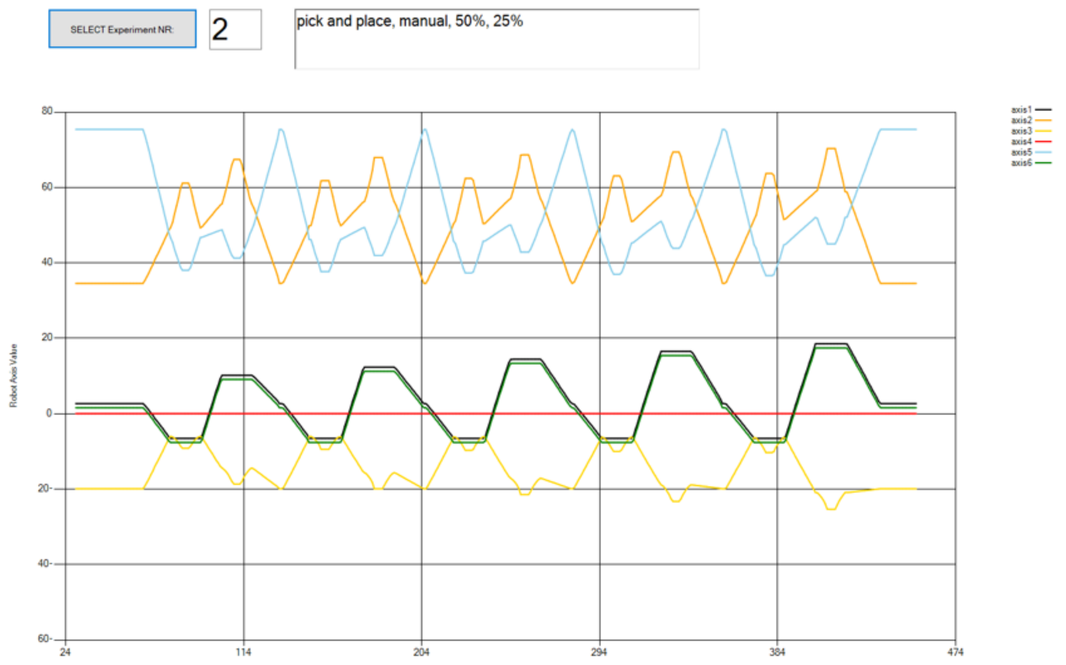


Figure 8. Evolution of robot joint values collected in real time during Experiment 2

This variation in real time of the six robot joint displacements characterizes the operation of the robot and can be associated to a certain characterisation of the robot in terms of working parameters (programmed speed and acceleration, number of motion cycles, load, cumulated work time), robot status (wear, time from last main-tenance) and environment (energy consumption). The evolution of joint values tag-ged with these factors is saved and stored by the DT as a digital signature, compared with future evolutions and used as support to decision for weighting the participa-tion of the physical robot in batch product operations, customized maintenance, a.o.

5. Conclusions

The paper describes a software system for data collecting in robot digital twins.

This software system accesses information and data from the robot, the process automated by the robot and external devices interfaced to the robot (conveyor belt, ASRS, smart meters) via an edge processing structure that includes the robot controller and IoT gateways. The software system includes a data acquisition agent directly connected to the edge processing hardware, a database where the collected information is stored and a user interface with multiple data display options.

The DT data collecting software is able to monitor the working robot and retrieve real-time data expressing the current operating mode, working parameters and per-formances of the robot. The DT can thus identify different operating patterns of the industrial robot, compare them using the history saved in the database, and provide support to decision to optimize motion trajectories (avoid jerks and collisions, keep the imposed speed profile, minimize time and energy consumption), production processes (reduce operation cycles) and maintain robot health (predict unexpected events, detect anomalies, customize maintenance).

The designed DT software collects robot data in two modes: continuously from the robot controller and external IoT gateways by help of specific software tools available from the robot system manufacturer, and discretely, event triggered, from the program instructions of the robot programming language by messages received from the application program.

The design solution and experiments for the DT robot data collecting are descri-bed in detail for ABB industrial robots equipped with IRC5 controllers.

Future research will be directed toward defining energy consumption patterns for classes of robotic processes and predicting energy consumption of robot team members to weight their participation in shared batch tasks for cost optimization.

R E F E R E N C E S

[1] Peres, R.S., Rocha, A.D., Leitão, P., Barata, J., IDARTS - Towards intelligent data analysis and real-time supervision for Industry 4.0, Computers in Industry, Vol. 101, pp. 138-146, doi: 10.1016/j.compind.2018.07.004, 2018

[2] Redelinghuys, A.J.H., Kruger, K., Basson, A.H., A Six-Layer Architecture for Digital Twins with Aggregation, Proceedings SOHOMA'18, Studies in Computational Intelligence, Vol. 803, Chapter 32, pp. 412-423, Springer, 2019

[3] Soldatos, J., Cavadini, F.A., Lazaro, O., The Digital Shopfloor: Industrial Automation in the Industry 4.0 Era, River Publishers Series in Automation, Control and Robotics, doi: 10.13052/rp-9788770220408, 2020

[4] Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W., Digital Twin in manufacturing: A categorical literature review and classification, IFAC-PapersOnLine, Vol. 51, Issue 11, pp. 1016-1022, doi: 10.1016/j.ifacol.2018.08.474, Elsevier, 2018

[5] Negri, E., Fumagalli, L., Macchi, M., A Review of the Roles of Digital Twin in CPS-based Production Systems, Procedia Manufacturing, Vol. 11, pp. 939–948, ScienceDirect, doi: 10.1016/j.promfg.2017.07.198, 2017

[6] Grieves, M., Vickers, J., Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In book Transdisciplinary Perspectives on Complex Systems, pp. 85-113, doi: 10.1007/978-3-319-38756-7_4, Springer, 2017

[7] Stan, M., Borangiu, T., Raileanu, S., Data- and model-driven digital twins for design and logistics control of product distribution, Proceedings of 23rd Int. Conference on Control Systems and Computer Science (CSCS), pp. 33-40, doi: 10.1109/CSCS52396.2021.00013, IEEE Catalog no. CFP2172U-ART, 2021

[8] Melesse, T. Y., Di Pasquale, V., Riemma, S., Digital twin models in industrial operations: A systematic literature review. Procedia Manufacturing, 42, pp. 267–272, 2020

[9] Da Cunha, C., Cardin, O., Gallot, G., Viaud, J., Designing the Digital Twins of Reconfigurable Manufacturing Systems: application on a smart factory, IFAC-PapersOnLine, Vol. 54, Issue 1, pp. 874-879, doi: 10.1016/j.ifacol.2021.08.103, 2021

[10] Valckenaers, P., Perspective on holonic manufacturing systems: PROSA becomes ARTI, J. Computers in Industry, vol. 120, 103226, doi: 10.1016/j.compind.2020.103226, 2020

[11] Morariu, C., Răileanu, S., Borangiu, T., A Distributed Approach for Machine Learning in Large Scale Manufacturing Systems, Service Orientation in Holonic and Multi-Agent Manufactur-ing, Studies in Computational Intelligence, Vol. 803, pp. 41–52, doi: 10.1007/978-3-030-03003-2_3, Springer, 2019

[12] ABB Robotics, Technical reference manual RAPID Instructions, Functions and Data types. RobotWare 5.13, Document ID: 3HAC 16581-1 Rev. J, ABB AB Robotics, accessed 2.03.2023

[13] Gloria, A., Cercasa, F., Souto, N., Design and implementation of an IoT gateway to create smart environments, Proceedings of the 8th Int. Conference on Ambient Systems, Networks and Technologies ANT'17, doi: 10.1016/j.procs.2017.05.343, 2017

[14] Anton, F., Borangiu, T., Răileanu, S., Anton, S., Cloud-Based Digital Twin for Robot Integra-tion in Intelligent Manufacturing Systems, Advances in Service and Industrial Robotics. Proc. of RAAD 2020, Mechanisms and Machine Science, vol 84, pp. 565-573, doi: 10.1007/978-3-030-48989-2_60, Springer, Cham, 2020

[15] Omron, Automation Control Environment (ACE). Version 4, User's Manual, Omron Robotics and Safety Technologies Inc., 2020

[16] Cygwin, Cygwin homepage, https://www.cygwin.com/, accessed 22.05.2023

[17] ABB Robotics, Application manual PC SDK. RobotWare 5.14, Doc. ID: 3HAC036957-001, https://library.e.abb.com/public/124d6b59313ed85fc125793400410c5b/3HAC036957-en.pdf, ABB AB, accessed 12.05.2023

[18] ABB, RobotStudio® Suite, https://new.abb.com/products/robotics/robotstudio, ABB AB, accessed 07.05.2023