Ain Shams Engineering Journal 15 (2024) 102804

Contents lists available at ScienceDirect

Ain Shams Engineering Journal

journal homepage: www.sciencedirect.com

Task planning and formal control of robotic assembly systems: A Petri net-based approach

Gökhan Gelen [a,*], Yasemin İçmez [b]

[a] Department of Mechatronics Engineering, Faculty of Natural Science, Architecture and Engineering, Bursa Technical University, Bursa, Türkiye
[b] Department of Electronics and Automation, Niksar Vocational School of Technical Sciences, Tokat Gaziosmanpasa University, Niksar, Tokat, Türkiye

ARTICLE INFO

Keywords:
Robotic assembly
Robot control
Robot programming
Petri nets
Formal control
Task planning

ABSTRACT

In modern industrial production, robotic assembly systems play a crucial role. As robots take on more tasks, the need for formal methods arises to define, control, and execute these tasks. This paper introduces a comprehensive approach to designing and generating control code for robotic assembly systems, taking task sequence planning into account. This methodology utilizes Petri nets (PNs) as a formal modeling and synthesis tool for the controller. Initially, the task sequences for assembly operations are represented using PN formalism. Supervisors are then synthesized for task sequence control specifications. Finally, the control code is obtained by the proposed methodology for industrial robots. By implementing this supervisory control structure, real-time control of the robotic assembly system is achieved. Experimental studies were conducted using an assembly cell equipped with an industrial robot. This methodology bridges the gap between the design and implementation of formal controllers for industrial robots. The proposed approach integrates formal methods into robot programming to leverage several advantages, including correctness assurance, complexity handling, improved documentation and clarity, enhanced safety and reliability, property verification, and scalability.

1. Introduction

In today's manufacturing systems, industrial robots are widely used for putting parts together. This operation is called assembly and this type of robot is named assembly robot which moves faster and with greater precision than a human. The synthesis and implementation of control structures for assembly operations are difficult issues involving assembly planning and task planning. To specify a feasible and optimal sequence of required operations to assemble a product is denoted as assembly planning. In addition to this, the translation of assembly plans into robotic operations is called task planning [1,2]. Task planning deals with sensory operations and motion planning of robots. As the number and complexity of tasks performed by robots used in today's assembly processes increase, it is necessary to use formal methods for task planning.

Formal methods provide a systematic approach to specifying, controlling, and implementing robotic tasks [3]. Finite State Automata (FSA) and Petri nets (PN) are popular formal methods to control Discrete Event Systems (DESs)[4]. Petri nets have some advantages over FSA: in a Petri net model, the states of the system can be presented by the number of tokens in places. However, in an automaton model, a different automaton state must be used for each state of the system. Although the number of tokens in PN places has increased, the Petri net model provides more compact and simple models. FSA [5–7] and PN-based methods [8–11] are preferred by the robotic community for the assembly and task planning, trajectory control, code analysis, and verification.

A Petri net-based method for feedback controller design for a robotic assembly cell was proposed by Moody et al [12]. By using this method, the controllers can be easily synthesized by considering constraints and incidence matrix. A framework is introduced for the modeling, analysis, and execution of robot tasks based on Petri nets [3]. The control of an autonomous mobile platform equipped with a manipulator is studied by [13,14]. Petri nets are also used for multi-task, multi-robot control, planning, and programming [15–17]. The mentioned works focus on the design of a Petri net-based controller for solving the robot task scheduling problem but do not consider the realization of the controller. After obtaining the desired controller (supervisor) for the system, a control code realization is required for the physical robot controller. Although there are many ways [18–22] to convert PNs into code for programmable logic controllers (PLCs), which are the main controllers of industrial automation systems, efficient methods to convert PNs into robot

* Corresponding author.
E-mail addresses: gokhan.gelen@btu.edu.tr (G. Gelen), yasemin.icmez@gop.edu.tr (Y. İçmez).

https://doi.org/10.1016/j.asej.2024.102804
Received 20 August 2023; Received in revised form 12 November 2023; Accepted 3 April 2024
Available online 16 April 2024

programming languages have not yet been studied.

The main contribution of this paper is to propose a methodology for converting PN models into control code for industrial robots. This method uses Petri nets and supervisory control theory as formal tools for controller modeling and synthesis. Initially, the task sequence for the assembly process is modeled using Petri net formalism. A supervisor is then synthesized using a place invariant-based method considering task control specifications. The obtained PN-based supervisory structure is implemented to perform real-time control of robotic assembly cells. An assembly cell consisting of a Mitsubishi Robot manipulator is used to perform experimental studies.

The rest of the paper is organized as follows. The notations and basic concepts about Petri nets and place invariant-based supervisor synthesis methods are briefly reviewed in Section 2. In Section 3, the robotic assembly cell is introduced. PN model of the task sequence and supervisor synthesis is presented in Section 4. The implementation methodology is proposed in Section 5. In Section 6, the test applications carried out to verify the obtained code are explained. Finally, some conclusions are given in Section 6.

2. Background

In this work, Petri nets are used to model assembly system operation and task sequence control specifications. A brief introduction about Petri nets is presented here. For a detailed introduction to Petri nets in the context of DES, see [4,23]. An ordinary Petri net comprises four components denoted $N = (P, T, F, W)$, where P and T are finite, nonempty, and distinct sets that represents the set of places and the set of transitions, respectively. The flow relation of the net, denoted by F is depicted by arcs with arrows connecting places to transitions or transitions to places. W is a mapping that assigns a weight to each arc. Ordinary Petri nets do not incorporate actuators or sensors. Consequently, it becomes essential to define a controller based on Petri nets that can encompass both actuators and sensors within an extended Petri net framework known as Automation Petri Net (APN). In APN, sensor readings can serve as firing conditions for transitions. The presence or absence of sensor readings can be combined with extended Petri net preconditions to trigger transitions. Formally, an APN can be described as follows:

$$APN = (N, X, Q, M_0) \tag{1}$$

where,

$N = (P,\ T,\ F,\ W)$ is a Petri net,

$X = \{\chi_1, \chi_2, ..., \chi_m\}$ is the set of firing conditions associated with the transitions,

$Q = \{q_1, q_2, ..., q_n\}$ is the set of actions that might be assigned to the places,

$M_0 : P \rightarrow N$ is the initial marking.

An APN is graphically represented by using circles for places, squares for transitions, and black dots for tokens as shown in Fig. 1. The number of tokens present in places reflects the current system state, while transitions represent events. Each transition has a set of input and output places, representing the preconditions and post-conditions of the transition. In the APN, firing conditions (represented by the variable χ) are considered external events, such as sensor readings. A firing condition is a Boolean variable that can take the value 1 or 0 indicating that the transition should or should not be triggered. The APN's marking illustrates how tokens are distributed in each place. The progress of the APN is characterized by the transfer of tokens between places, which happens when enabled transitions are triggered. In this paper, APN is used to model the task sequence of a robot for assembly operation in the system.

The concept of place invariants (PIs) based supervisor computation method is briefly reviewed here. For more information, consult a standard reference such as [24]. The system to be controlled is modeled by a Petri net N with n places and m transitions. Let $[N]$ be the incidence matrix of the plant net N. The supervisor consists of the transitions of the

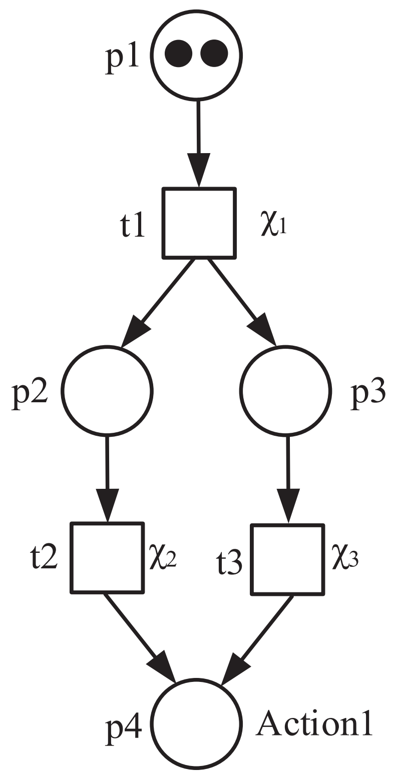


Fig. 1. An automation Petri net (APN) model.

plant net and a set of control places, whose incidence matrix is denoted as $[Ns]$. The controlled net with the incidence matrix $[Nc]$ consists of both the original plant net and the supervisor. The control goal is to enforce the plant net to satisfy the following constraint

$$\sum_{i=1}^{n} l_i.\mu_i \leqslant \beta \tag{2}$$

where μ_i denotes the marking of place p_i, and li and β are non-negative integer constants. After the introduction of a non-negative slack variable μ_c, the above constraint can be transformed as follows:

$$\sum_{i=1}^{n} l_i.\mu_i + \mu_c = \beta \tag{3}$$

where μ_i denotes the marking of control place pc. All constraints can be grouped as follows,

$$[L].\mu_p \leqslant b \tag{4}$$

where μ_p is the marking vector of the plant net, $[L]$ is an $nc \times n$ integer matrix, b is an $nc \times 1$ integer vector, and nc is the number of the constraints. All PIs can be rewritten as follows:

$$[L].\mu_p + \mu_s = b \tag{5}$$

where μ_s is an $nc \times 1$ integer vector, representing the marking of the control places. Finally, given a plant Petri net $[N]$ and the constraints $[L]$ and b, the supervisor $[Ns]$ can be computed as follows:

$$[N_s] = -[L].[N] \tag{6}$$

The initial marking of the supervisor μ_{s0} is calculated as follows:

$$\mu_{s0} = b - [L].\mu_{p0} \tag{7}$$

where μ_{p0} is the initial marking of the plant net N.

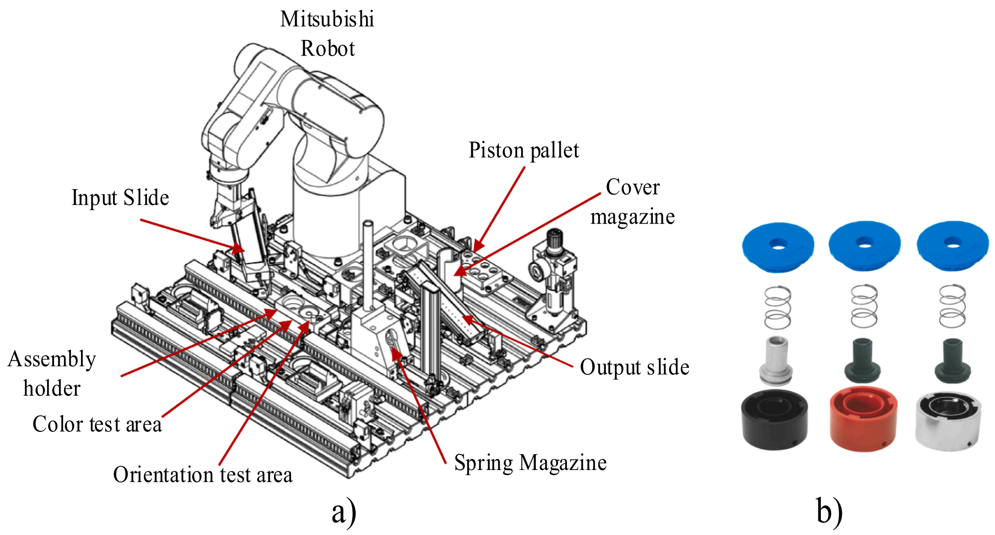


Fig. 2. A) block diagram of the assembly station. b) assembly sequence of the cylinder's parts.

3. Robotic assembly system

A robotic assembly system [25] produced by FESTO Didactic consisting of a Mitsubishi RV-2SDB Robot is considered in experimental studies. This cell produces a small pneumatic cylinder as an assembled good consisting of a body, piston, spring, and cover. The block diagram of the robotic assembly cell is illustrated in Fig. 2.a. In this setup, cylinder bodies can have one of three colors: black, red, or metallic, and, pistons can be two types black or metal. If the body color is red or metallic, then the piston must be black; on the opposite, if the body is black, the piston must be metallic as shown in Fig. 2.b.

The bodies of the pneumatic cylinders are fed to the robot via an input slide. The availability of a body in the input slide is sensed by an infrared sensor. The robot carries the body from the input slide to color test area and determines the color via color sensor mounted to the gripper. Then, the body is moved to the orientation test point. After the orientation test, body is placed in the assembly holder in the correct orientation. The robot takes the piston from the pallet and placed it into the body. The piston springs and the cylinder caps are fed to the robot from controlled magazines. The fully assembled pneumatic cylinder is then placed on the output slide. The control of the robot and other peripherals is performed by a CR1-571 robot controller located in the assembly station. Melfa Basic language and RT Toolbox program is used to code robot. Fig. 3 shows the all system connections.

4. Modelling of assembly task sequences and synthesis of supervisors

In this section, the APN model of assembly task sequences and synthesis of supervisors are presented. The required operations for assembling are classified into seven different tasks. These assembly tasks are labeled as TASKi (i = 1,2, …, 7) and tabulated in Table 1. The APN model of assembly task sequence is presented in Fig. 4. Tasks are assigned to places of the APN as actions. If there is a token in the action assigned place, this means that the assigned task will be performed. When the task is completed, the token is removed from the place. The meaning of places and firing conditions of transitions are provided in Table 2 and Table 3. In the APN model, there are 12 places labeled as p1, p2, …, p12 and 10 transitions labeled as t1, t2, …, t10. Initially, there are four black and four metal pistons, eight springs, and eight covers in the system. The number of tokens in p9, p10, p11, and p12 places represents the number of black pistons, metal pistons, and capacities of spring and cover magazines, respectively.

If a cylinder body is detected, i.e. $\chi 1 = \text{part\_av} = 1$, the transition t1 fires, and a token is deposited into p1. While a token is placed in p1, the assigned action, i.e. TASK1, starts to run. In TASK1, robot picks a body from the input slide and places in the color test area. If there is a token in p1 and then the TASK1 is finished, i.e. $\chi 2 = \text{tsk1\_fnsh} = 1$, transition t2 fires by removing the token from p1 and by depositing a token in p2. This means that there is a token in p2 and the assigned action TASK2 starts to run. The color and orientation tests are performed and the body is placed in the assembly holder as TASK2, If the color of the body is metal or red the variable CYLTYPE is set to zero, if the color of the cylinder body is black the variable CYLTYPE is set to one. If there is a token in p2 and then the TASK2 is finished, transition t3 fires by removing the token from p2 and by depositing a token in p3. This means that there is a body in the assembly holder and it is ready for assembling. When there is a token in place p3, transitions t4 or t5 can be fired by considering firing conditions and the number of tokens in the places p9 and p10. If the transition t4 fires, the token in p3 and a token from p9 is removed and a token is added to p4. A token in p4 means that TASK3 starts to run. By TASK3, a black piston is taken from the pallet and placed in the body; on the contrary, if the transition t5 fires, a token is transferred to p5 and TASK4 starts to run, i.e. a metal piston is taken from the pallet and placed in the body. For the rest of the model, token flow occurs similarly and related robotic tasks are operated sequentially. For the remainder of the model, token flow occurs similarly to the number of tokens and trigger conditions at the entry sites. Several constraints must be imposed by the controller to ensure physical limitations are obeyed. In this work, five constrained called control specifications are considered and tabulated in Table 4.

The computation of control places for the above specifications is performed by using the place invariant-based method that is explained in Section 2. A separate supervisory structure (control place and related arcs) will be calculated for each of the control specifications. The incidence matrix and initial marking used in each calculation are the same for the APN model shown in Fig. 4. The incidence matrix *[N]* of any Petri net is a matrix where its rows represent places, and its columns represent transitions within the Petri net. The entry at position *(i, j)* of the matrix indicates the relationship between place *i* and transition *j*. A value of *1* signifies an arc from transition *j* to place *i* ($t_j \rightarrow p_i$) and -1 signifies an arc from place *i* to transition *j* ($p_i \rightarrow t_j$). If there is no direct connection between place *i* and transition *j*, the entry at position *(i, j)* is equal to 0. The incidence matrix *[N]* of the plant APN model is as follows:

$$[N]=\begin{bmatrix} +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & +1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

A column of incidence matrix indicates that firing of j-th transition. For example, the second column of matrix *[N]*, indicates that firing of transition *t2* consists of removing a token from place *p1* and adding a token to place *p2*. The initial marking in a Petri net represents the number of tokens in each place at the beginning of the system's operation. The initial marking is typically depicted as a vector or a set of values that correspond to each place in the Petri net. Each value in the vector represents the number of tokens in the corresponding place when the system begins its operation. For APN model depicted in Fig. 4, the number of tokens in places are $\mu_1=0, \mu_2=0, \mu_3=0, \mu_4=0, \mu_5=0, \mu_6=0, \mu_7=0, \mu_8=0, \mu_9=4, \mu_{10}=4, \mu_{11}=8$, and $\mu_{12}=8$ where μ_i is the number of tokens in place *pi*. The initial marking vector μ_{p0} μ_{p0} of the plant APN model is as follows:

$$\mu_{p0}=\begin{bmatrix}\mu_1\\ \mu_2\\ \mu_3\\ \mu_4\\ \mu_5\\ \mu_6\\ \mu_7\\ \mu_8\\ \mu_9\\ \mu_{10}\\ \mu_{11}\\ \mu_{12}\end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\4\\4\\8\\8\end{bmatrix}$$

For the first specification, there is only one robot available for the assembly operation, thus only one task can be performed at any given time. Robotic tasks are assigned to places p1, p2, p4, p5, p6, p7, and p8 of the APN model shown in Fig. 4. If the related places are considered, the total number of tokens in these places should be less than or equal to one token. This place invariant can be written in the inequality form as $\mu_1+\mu_2+\mu_4+\mu_5+\mu_6+\mu_7+\mu_8\leqslant 1$. Considering equation (4), the inequality belonging to first control specification $\mu_1+\mu_2+\mu_4+\mu_5+\mu_6+\mu_7+\mu_8\leqslant 1$ can be converted to L_1 vector as follows and b is *1* for this constraint.

$$L_1=[1\quad 1\quad 0\quad 1\quad 1\quad 1\quad 1\quad 1\quad 0\quad 0\quad 0\quad 0]$$

The incidence matrix of control place C1 that provides the first control specification can be calculated by using Equations (6). According to Equation (6),

$$[N_s]=-[L].[N]$$

By substituting the vector *[L]* and the incidence matrix *[N]* into equation (6)

$$[N_s]=-[1\quad 1\quad 0\quad 1\quad 1\quad 1\quad 1\quad 1\quad 0\quad 0\quad 0\quad 0].\begin{bmatrix} +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & +1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

the incidence matrix of the control place C1 is obtained as follows

$$[N_s]=[-1\quad 0\quad +1\quad -1\quad -1\quad 0\quad 0\quad 0\quad 0\quad +1]$$

The initial marking of control place C1 can be calculated using Equations (7). According to Equation (7),

$$\mu_{s0}=b-[L].\mu_{p0}$$

By substituting the vector *[L]* and the initial marking vector μ_{p0}, initial marking of C1 place is computed as follows

$$\mu_{s0}=1-[1\quad 1\quad 0\quad 1\quad 1\quad 1\quad 1\quad 1\quad 0\quad 0\quad 0\quad 0].\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\4\\4\\8\\8\end{bmatrix}=1$$

The *[N_s]* vector is the incidence matrix of the *C1* control place. As can be seen from the vector, t3, and t10 are the input transition of *C1,* and t4 and t5 are the output transitions of *C1. C1* has one token as initial marking. The PN representation of this control place is depicted in Fig. 5.

The computation of control places for the other control specifications can be performed as computation of the first CP by following the above steps. All control specifications are expressed as place invariants, and these invariants are written in the form of inequalities. The created place invariants for each specification are converted to vectors, and control places are calculated using Equation (6) and Equation (7). Place Invariants for specification, related vectors for computations, obtained Incidence matrix, initial markings, and control places are presented in Table 5. The final closed-loop (controlled) model for the task sequence

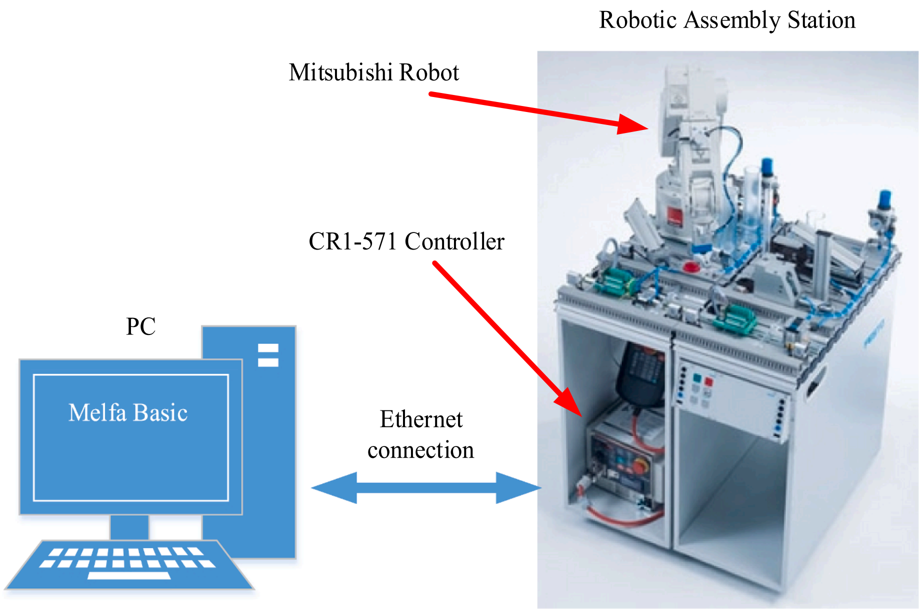


Fig. 3. Schematic of the system.

Table 1
List of robotic tasks.

Task	Explanations
TASK 1	A body is picked from the input slide and placed in the color test area
TASK 2	The body is placed in the assembly holder after color and orientation tests.
TASK 3	A black piston is taken from the pallet and placed in the body
TASK 4	A metal piston is taken from the pallet and placed into the body
TASK 5	A spring is taken from the spring magazine and placed into a body
TASK 6	A cover is taken from the cover magazine and assembled to the body after the orientation test.
TASK 7	A finished part is placed on the output slide.

of the robotic assembly system is obtained by coupling all computed CPs to the APN model as shown in Fig. 6.

5. The implementation methodology

The implementation methodology employed in this approach utilizes the concept of tokens from Petri nets as the primary mechanism for controlling the flow of the control logic. Each place in the Petri net corresponds to a variable in the robot language. To emulate the behavior of tokens in the Petri net, the implementation maps actions in the Petri net places to conditional subroutine calls in the robot program. Whenever a transition in the Petri net involves the movement of tokens, a simulated movement of tokens between variables is performed. To achieve this simulated token movement, separate variables are assigned to each place in the robot program. During the execution of the robot program, these variables are incremented or decremented to simulate the flow of tokens within the Petri net.

The state of these variables reflects the current state of the control logic and helps guide the robot's actions and decisions. By adopting this approach, the control logic of the robot system is structured using Petri net concepts, enabling a clear mapping between Petri net elements and robot programming elements. The use of simulated token movement through dedicated variables facilitates the coordination of actions, decisions, and transitions within the control logic, enhancing the control capabilities of the robot. By changing the values of these variables, the program can emulate the movement of tokens between different states of the system. The firing conditions of transitions are considered as sensor readings from inputs of controllers and some events that are used for detecting the end of the related task subroutine. It is essential to arrange the MELFA BASIC code for the robot in the following order to ensure proper functioning: first, the variable definitions and initial marking is written; next, the actions are implemented by calling related task subroutines. After this, the transition mechanism of APN is coded by considering related firing conditions, and finally, the task-finished data are reset. As mentioned before, the task-finished data are used as firing conditions and are generated in the related subroutine by setting tsk_fns variables to "1″. The task-finished data must be reset at the bottom of the main program to re-detection of task finishing. The program organization for the proposed implementation methodology is presented in Fig. 7.

A simple APN model shown in Fig. 8 is considered to explain the proposed implementation methodology. In this APN, there are a place p1 and two transitions t1 and t2. It is assumed that a sample robotic task is assigned to place p1. The firing conditions of transitions t1 and t2 are $\chi 1$ and $\chi 2$, respectively. It is assumed that $\chi 1$ is an external event (sensor information) and $\chi 2$ is an event related to task finish data of the task subroutine. Initially, it is assumed that there is no token in the place p1. If the firing condition $\chi 1$ occurs, t1 can be fired and a token is deposited to place p1. While there is a token in place p1 and the firing conditions $\chi 2$ occurs, in this case, the transition t2 can be fired and the token is removed from place p1.

The MELFA BASIC code implementation of the sample APN model is shown in Appendix A. In the obtained code, variables are defined in lines between 1 and 4. In lines 6 and 7, initial values are assigned to related variables. A loop is created between lines 8 and 23 by using the "Start" label and "GoTo Start" command. In the model, there is only one action assigned to place p1. In line 11, the TASK1 subroutine is called by using

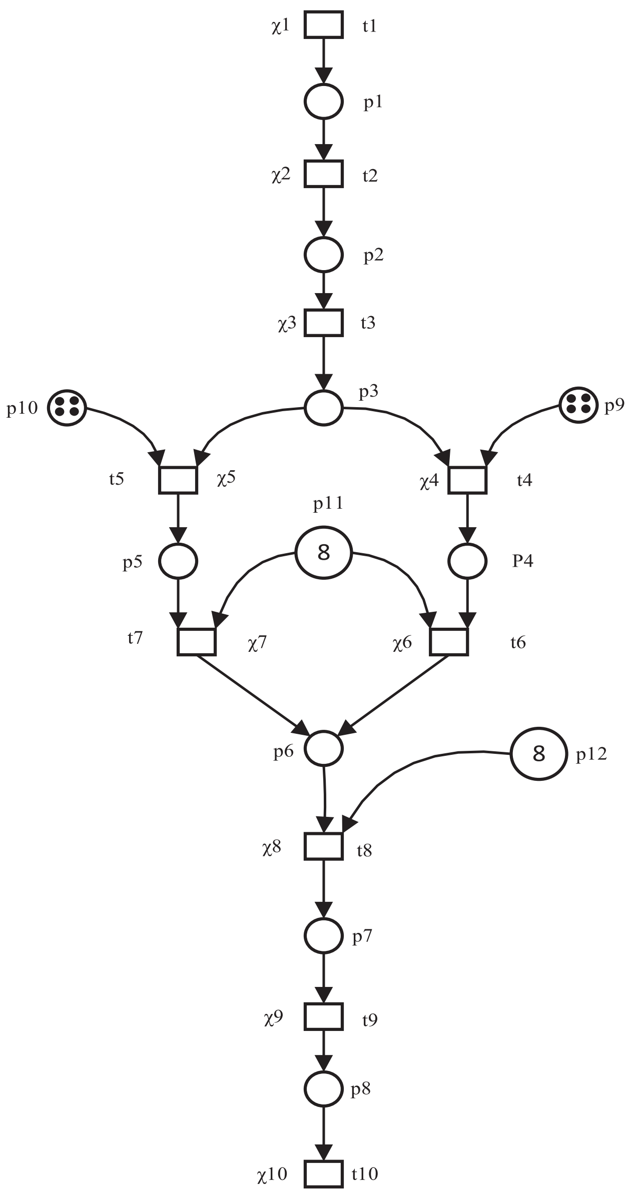


Fig. 4. APN model of the robotic task sequence.

the "GoSub" code if there is a token in p1 i.e. P1 = 1. The transitions of APN are implemented between lines 11 and 19. The "T1_FNS" data are reset, i.e. T1_FNS = 0 in line 21 for re-detection of the task finished of the TASK1 subroutine. After this code block, the TASK1 subroutine should be implemented. As shown in the code, in the task subroutine firstly the

Table 2
Meaning of places and assigned tasks.

Places	Assigned action (task) or meaning
p1	TASK1
p2	TASK2
p3	A base is in the assembly holder and ready for assembling
p4	TASK3
p5	TASK4
p6	TASK5
p7	TASK6
p8	TASK7
p9	The number of black pistons in the pallet (max. cap. 4)
p10	The number of metal pistons in the pallet (max. cap 4)
p11	The number of springs in the magazine (max. cap. 8)
p12	The number of covers in the magazine (max. cap. 8)

Table 3
Transitions and firing conditions.

Trans.	Firing conditions	Explanations
t1	χ1={Part_Av = 1}	A cylinder is available at the input.
t2	χ2 = tsk1_fnsh	TASK1 is finished.
t3	χ3 = tsk2_fnsh	TASK2 is finished.
t4	χ4={CYLTYPE = 0}	The color of the cylinder is metal or red.
t5	χ5={CYLTYPE = 1}	The color of the cylinder is black.
t6	χ6 = tsk3_fnsh	TASK3 is finished.
t7	χ7 = tsk4_fnsh	TASK4 is finished.
t8	χ8 = tsk5_fnsh	TASK5 is finished.
t9	χ9 = tsk6_fnsh	TASK6 is finished.
t10	χ10 = tsk7_fnsh	TASK7 is finished.

Table 4
Control specifications for assembly.

#SPEC	Explanation
1	There is only one robot available for the assembly operation, thus only one task can be performed at any given time.
2	The capacity of the assembly holder is one-cylinder body. Therefore, it can be only one part can be assembled at any time
3	It can be used only one type of piston (metal or black) in the same assembly operation because of assembly holder capacity.
4	If there is no available black piston, a new cylinder body should not be put into operation.
5	If there is no available metallic-colored piston, a new cylinder body should not be put into operation.

T1_FNS data should reset, i.e. T1_FNS = 0, and at the end of this subroutine, this data should be set i.e. T1_FNS = 1. The validation of this sample code is tested by using sample input–output assignments. By using a PC software called "RT ToolBox2'', this code was programmed on a robot controller. Test results show that the obtained code fully reflects the functions of the APN model.

6. Applications and tests

The controlled APN model of the robotic assembly cell shown in Fig. 6 is implemented on the robot controller using the above procedures. The obtained robot code can be verified by timing diagrams showing the occurrence of tasks. To perform the verification process, some assembly scenarios that take into account control specifications are created and the related data in the robotic assembly system is accused to draw a timing diagram. In the graphs, the vertical axis displays the status of task execution, while the horizontal axis represents time in seconds. In the first scenario, the bodies of red, black, and metallic colors are sequentially placed into the system. The addition of each body is performed while waiting for the assembly of the other bodies to be completed. The timing diagram belongs to the part available sensor that detects a body at the input and activities of the tasks during the experiment is presented in Fig. 9. As a result of this experiment, it is observed that all the bodies are successfully assembled. In addition, it can be seen from the timing diagram that there is no active task at the same time, so only one task occurs at any given time during assembly. This verifies that the computed controller structure satisfies

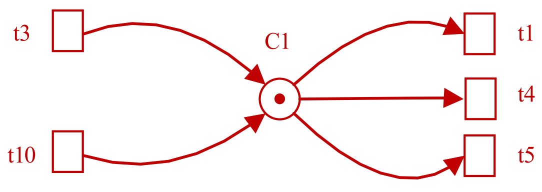
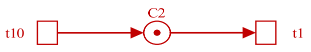
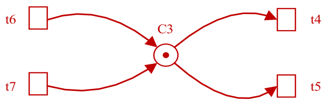
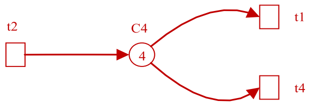
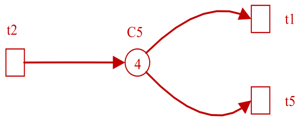


Fig. 5. PN representation of this control place *C1*.

Table 5
Place invariants and computed control places (CP).

CP	Place Invariants and related vectors for computation	Obtained Incidence matrix $[N_s]$, initial markings μ_{s0}, and Control Places
C2	$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_5 + \mu_6 + \mu_7 + \mu_8 \leqslant 1 L_2 = [1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0] b_2 = 1$	$[N_s] = [-1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ +1]$ $\mu_{s0} = b_2 - [L_2].\mu_{p0} = 1$ 
C3	$\mu_4 + \mu_5 \leqslant 1 L_3 = [0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0] b_3 = 1$	$[N_s] = [0\ \ 0\ \ 0\ \ -1\ \ -1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0]$ $\mu_{s0} = b_3 - [L_3].\mu_{p0} = 1$ 
C4	$\mu_1 - \mu_9 \leqslant 0 L_4 = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ -1\ \ 0\ \ 0\ \ 0] b_4 = 0$	$[N_s] = [-1\ \ 1\ \ 0\ \ -1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$ $\mu_{s0} = b_4 - [L_4].\mu_{p0} = 4$ 
C5	$\mu_1 - \mu_{10} \leqslant 0 L_5 = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ -1\ \ 0\ \ 0] b_5 = 0$	$[N_s] = [-1\ \ 1\ \ 0\ \ 0\ \ -1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$ $\mu_{s0} = b_5 - [L_5].\mu_{p0} = 4$ 

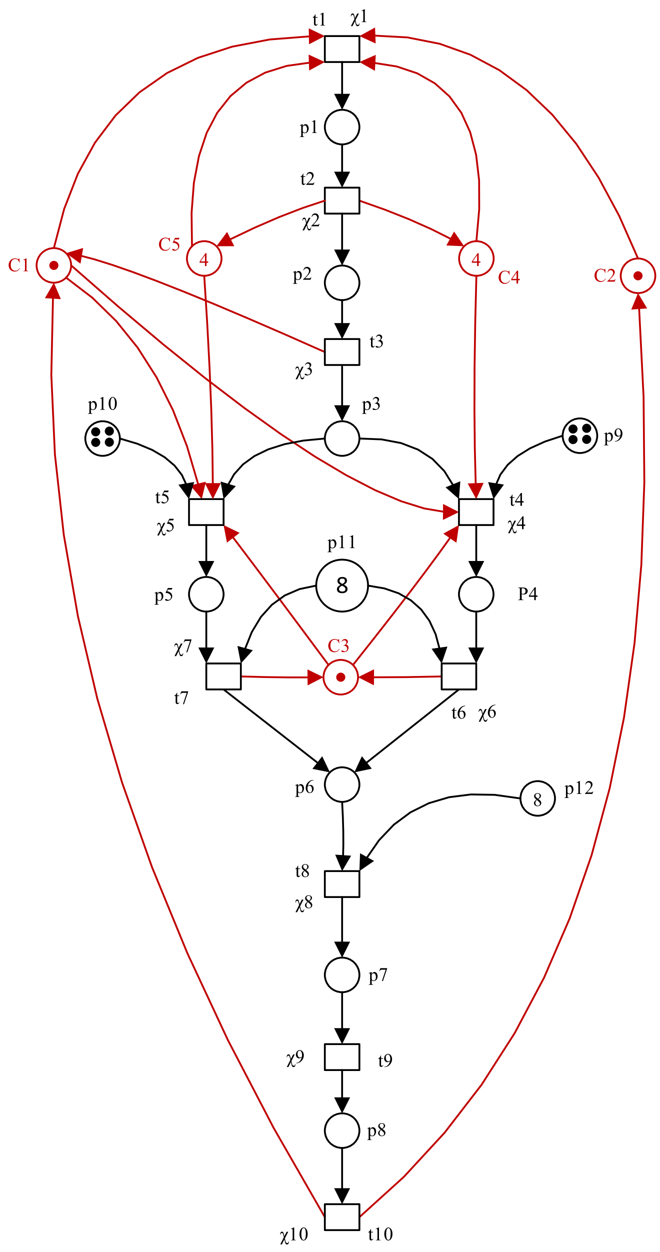


Fig. 6. The controlled model for the task sequence.

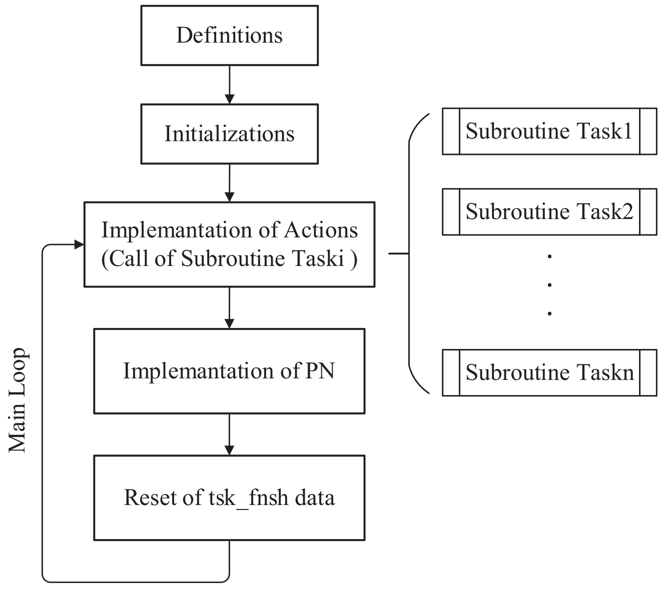


Fig. 7. Program organization for the proposed implementation methodology.

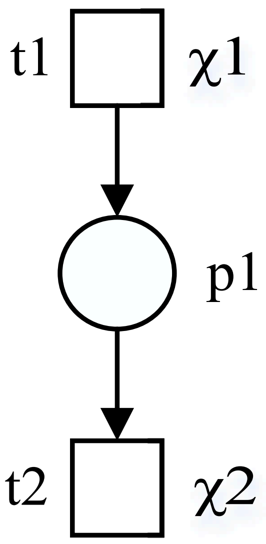


Fig. 8. A sample APN.

the first specification i.e., only one task can be performed at any given time.

To test the second control specification (SPEC2), a new body is placed into the system while the assembly of the previous one has not been completed. As can be seen in Fig. 10, the robot does not pick the new part before completing the required tasks for the existing assembly. Therefore, only one part is assembled at any given time.

According to SPEC3, one piston type (metal or black) should be used in the same assembly operation, i.e., only one of Task 3 or Task 4 should be active during any assembly operation. This means that Task 3 and Task 4 cannot be active simultaneously. Fig. 9 presents a timing diagram for the assembly of red, black, and metallic parts. As seen in the figure, only one of Task 3 or Task 4 is executed during the assembly of each part. Four red-bodied products were assembled in the cell to use up all the black pistons on the piston pallet. As there is no piston to be mounted for the new body added to the system after the fourth body, a new cylinder body should not be put into operation by the robot as stated in SPEC 4. In the timing diagram shown in Fig. 11, TASK1 is not operated for the new body added to the system after the assembly of the 4th part is completed, that is, the robot does not process the new part. A similar scenario has been tested for the black body and metallic-colored piston (Fig. 12) and it has been seen that in case there is no available metallic-colored piston, a new cylinder body should not be put into operation. It is observed that the computed control places and generated code provide the desired behavior. The supplementary materials include animation videos demonstrating considered scenarios created with the CIROS Studio FESTO program.

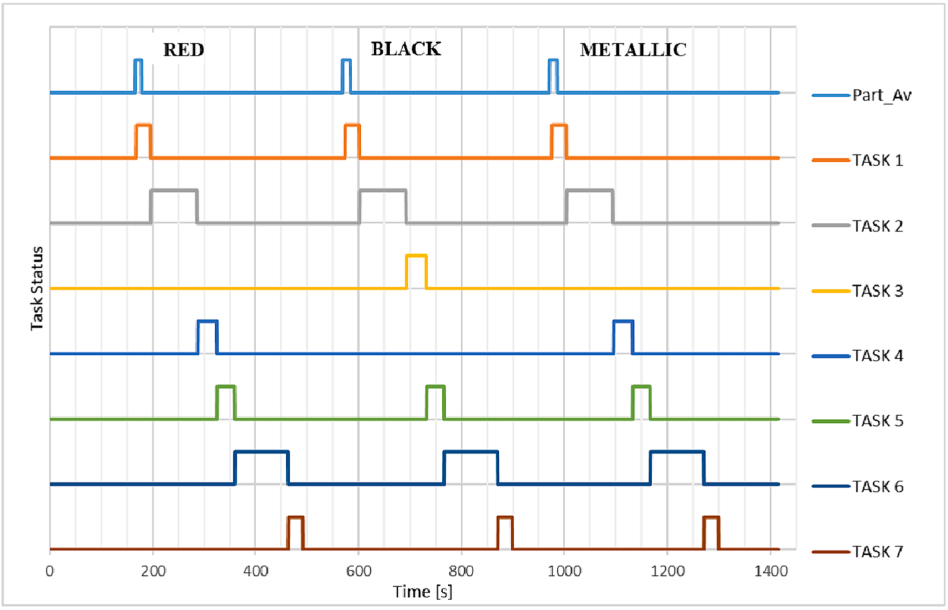


Fig. 9. Timing diagram for the first assembly scenario.

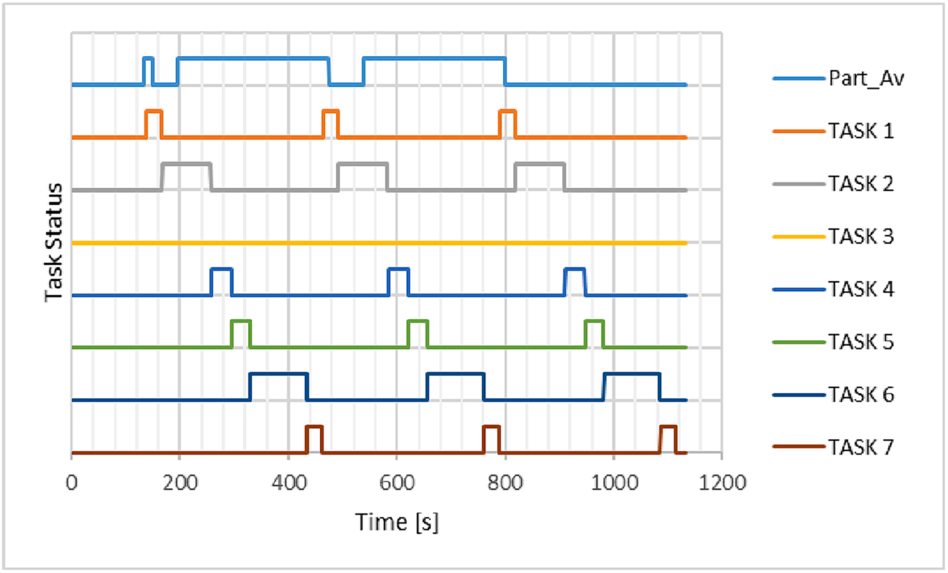


Fig. 10. Timing diagram for the second assembly scenario.

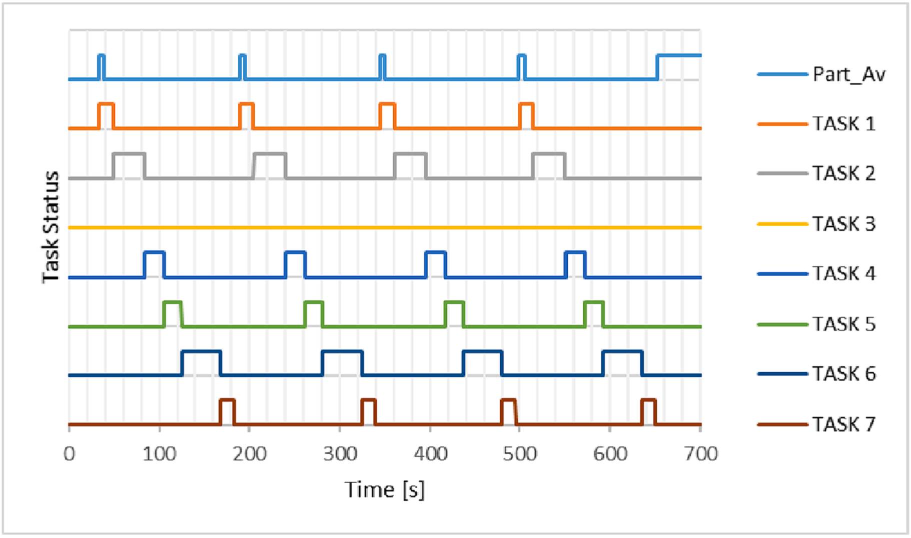


Fig. 11. Timing diagram for the third assembly scenario.

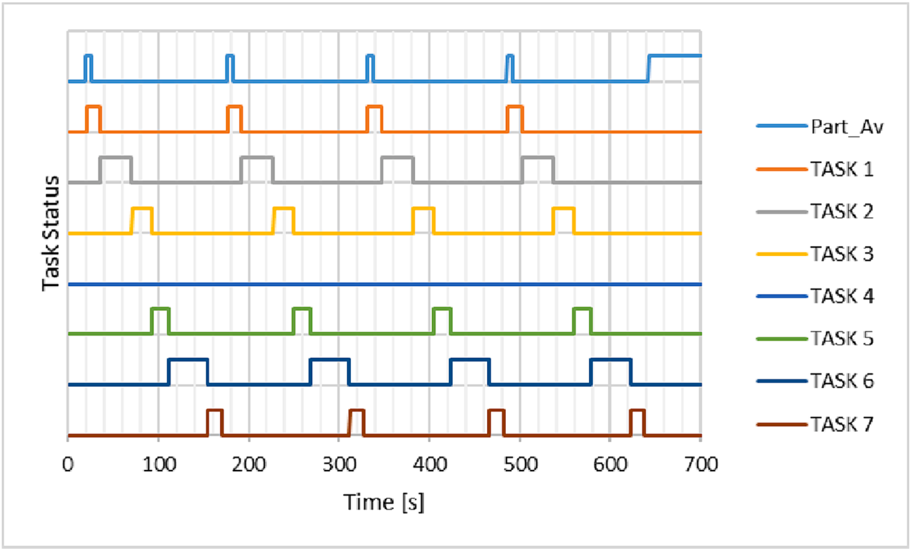


Fig. 12. Timing diagram for the fourth assembly scenario.

7. Conclusions

This paper introduces a novel approach for obtaining control codes for industrial robots by employing supervisory control theory and Petri nets. The proposed methodology offers a general framework, allowing the computation of control places (supervisors) based on the task sequence model and control constraints. To demonstrate the effectiveness of this implementation approach, an experimental assembly cell featuring an industrial robot has been used. The results showed that the robotic assembly cell achieved excellent performance under the proposed control strategy. In practical terms, incorporating a Petri net supervisor into a robot controller allows for the utilization of formal methods in robot programming. By employing this approach, the programming process becomes more systematic and rigorous, following established principles of formal methods. This brings several benefits to the development and maintenance of robot systems. Additionally, the code becomes more comprehensible and modifiable if necessary. This paper primarily focuses on high-level control, where the supervisor's role involves coordinating robotic tasks. Future research will concentrate on investigating low-level control, in which the supervisor orchestrates interactions between control devices and tasks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A:. MELFA BASIC code for simple APN model presented in Fig. 10

'The MELFA BASIC code implementation of simple APN model presented in Fig. 10

```
'--- Variable definitions ------------------
Def Inte P1
Def Io sensor  = Bit,1
Def Inte T1_FNS
'----- Initialization -------------------------
P1 = 0
T1_FNS = 0
*Start
'----- Actions (call of subroutines) -----
  If P1 = 1 Then GoSub *TASK1
'----- Transitions Mechanism -------
'----- t1 -------
   If (P1 = 0 AND sensor = 1) Then
        P1 = 1
   ENDIF
'----- t2 -------
   If (P1 = 1 AND T1_FNS = 1) Then
        P1 = 0
   ENDIF
'----- Reset of Task-finished data -----
T1_FNS = 0
'----- End of Main Program -------------
GoTo *Start
END
'----- Task Subroutines ------------------
*TASK1
T1_FNS = 0
----------
----------
----------
T1_FNS = 1
Return
```

.

Appendix B. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.asej.2024.102804.

References

[1] Alatartsev S, Stellmacher S, Ortmeier F. Robotic task sequencing problem: a survey. J Intell Robot Syst 2015;80:279–98. https://doi.org/10.1007/s10846-015-0190-6.

[2] Rosell J. Assembly and task planning using petri nets: a survey. Proc Inst Mech Eng B J Eng Manuf 2004;218:987–94. https://doi.org/10.1243/0954405041486019.

[3] Costelha H, Lima P. Robot task plan representation by petri nets: modelling, identification, analysis and execution. Auton Robot 2012;33:337–60. https://doi.org/10.1007/s10514-012-9288-x.

[4] Cassandras CG, Lafortune S. Introduction to discrete event systems. Cham: Springer International Publishing; 2021. DOI: 10.1007/978-3-030-72274-6.

[5] Deplano D, Franceschelli M, Ware S, Rong S, Giua A. A discrete event formulation for multi-robot collision avoidance on pre-planned trajectories. IEEE Access 2020;8:92637–46. https://doi.org/10.1109/ACCESS.2020.2994472.

[6] Hussain R, Zielinska T, Hexel R. Finite state automaton based control system for walking machines. Int J Adv Rob Syst 2019;16. https://doi.org/10.1177/1729881419853182. 1729881419853182.

[7] Tatsumoto Y, Shiraishi M, Cai K, Lin Z. Application of online supervisory control of discrete-event systems to multi-robot warehouse automation. Control Eng Pract 2018;81:97–104. https://doi.org/10.1016/j.conengprac.2018.09.003.

[8] Da Mota FAX, Rocha MX, Rodrigues JJPC, De Albuquerque VHC, De Alexandria AR. Localization and navigation for autonomous mobile robots using petri nets in indoor environments. IEEE Access 2018;6:31665–76. https://doi.org/10.1109/ACCESS.2018.2846554.

[9] Figat M, Zieliński C. Methodology of designing multi-agent robot control systems utilising hierarchical petri nets. International Conference on Robotics and Automation (ICRA) 2019;2019:3363–9. https://doi.org/10.1109/ICRA.2019.8794201.

[10] Figat M, Zieliński C. Robotic system specification methodology based on hierarchical petri nets. IEEE Access 2020;8:71617–27. https://doi.org/10.1109/ACCESS.2020.2987099.

[11] Lacerda B, Lima PU. Petri net based multi-robot task coordination from temporal logic specifications. Rob Auton Syst 2019;122:103289. https://doi.org/10.1016/j.robot.2019.103289.

[12] Moody JO, Antsaklis PJ. Petri net supervisors for DES with uncontrollable and unobservable transitions. IEEE Trans Autom Control 2000;45:462–76. https://doi.org/10.1109/9.847725.

[13] Filipescu A, Petrea G, Filipescu A, Filipescu S. Modeling and control of a mechatronics system served by a mobile platform equipped with manipulator. Proceedings of the 33rd Chinese Control Conference, Nanjing, China: IEEE; 2014, p. 6577–82. DOI: 10.1109/ChiCC.2014.6896078.

[14] Filipescu A, Mincă E, Filipescu A, Coandă H-G. Manufacturing technology on a mechatronics line assisted by autonomous robotic systems, robotic manipulators and visual servoing systems. Actuators 2020;9:127. https://doi.org/10.3390/act9040127.

[15] Caloini A, Magnani G, Pezze M. A technique for designing robotic control systems based on petri nets. IEEE Trans Contr Syst Technol 1998;6:72–87. https://doi.org/10.1109/87.654878.

[16] Yasuda G. Implementation of Distributed Control Architecture for Multiple Robot Systems Using Petri Nets. In: Pawlewski P, editor. Petri Nets - Manufacturing and Computer Science, InTech; 2012. DOI: 10.5772/50577.

[17] Ziparo VA, Iocchi L, Lima PU, Nardi D, Palamara PF. Petri net plans: a framework for collaboration and coordination in multi-robot systems. Auton Agent Multi-Agent Syst 2011;23:344–83. https://doi.org/10.1007/s10458-010-9146-1.

[18] Azkarate I, Ayani M, Mugarza JC, Eciolaza L. Petri net-based semi-compiled code generation for programmable logic controllers. Appl Sci 2021;11:7161. https://doi.org/10.3390/app11157161.

[19] Gelen G, Uzam M. The synthesis and PLC implementation of hybrid modular supervisors for real time control of an experimental manufacturing system. J Manuf Syst 2014;33:535–50. https://doi.org/10.1016/j.jmsy.2014.04.008.

[20] Hellgren A, Fabian M, Lennartson B. On the execution of sequential function charts. Control Eng Pract 2005;13:1283–93. https://doi.org/10.1016/j.conengprac.2004.11.011.

[21] Moreira MV, Basilio JC. Bridging the gap between design and implementation of discrete-event controllers. IEEE Trans Automat Sci Eng 2014;11:48–65. https://doi.org/10.1109/TASE.2013.2281733.

[22] Uzam M, Gelen G. The real-time supervisory control of an experimental manufacturing system based on a hybrid method. Control Eng Pract 2009;17:1174–89. https://doi.org/10.1016/j.conengprac.2009.05.004.

[23] Uzam M, Jones AH. Discrete event control system design using automation petri nets and their ladder diagram implementation. Int J Adv Manuf Technol 1998;14: 716–28. https://doi.org/10.1007/BF01438224.

[24] Hrúz B, Zhou M. Modeling and control of discrete-event dynamical systems: with petri nets and other tool. London: Springer; 2007.

[25] Ebel F, Ersoy M, Pensky D. Robot station with MPS modules. 73770. Denkendorf, Germany: Festo Didactic GmbH & Co. KG,; 2015.

Gökhan GELEN received the B.Sc. degree in electrical and electronics engineering from İnönü University, Malatya, Turkey, 2003. He received the M.Sc. and Ph. D. degrees from Nigde University, Nigde, Turkey, 2006 and 2010, respectively. He was with Nigde University, from 2004 to 2011. He was with Gaziosmanpaşa University, Turkey, from 2011 to 2015. He is currently an Assocciated Professor in the Department of Mechatronics Engineering, at Bursa Technical University, Bursa, Turkey, since 2015. His research interests include design and implementation of Supervisory controller for Discrete Event Control Systems.

Yasemin İçmez received the B.Sc. degree in electrical and electronics engineering from Karadeniz Technical University, Trabzon, Turkey, 2012. She received the M.Sc. degree from Tokat Gaziosmanpaşa University, Tokat, Turkey, 2015. She is currently pursuing a Ph.D. degree in Electrical and Electronics Engineering at Tokat Gaziosmanpaşa University. Her research interests include control an automation systems.