Abstract: Petri nets (PNs) are typically used for design and verification rather than direct control implementation. In this paper, aligning with the Industry 4.0 paradigm's focus on flexible and reconfigurable control systems, we propose a modelling tool for rapidly prototyping feedback-based discrete-event control algorithms on programmable controllers such as PLCs or microcontroller boards. This modelling tool, named Signal Interpreted Coloured Petri Nets (SICPNs), aims to combine the formal modelling expressiveness of Coloured PNs with the capabilities of Signal Interpreted PNs, which are specialised in processing plant measurements and determining actuator commands. This contribution involves: (a) the formal definition of SICPN; (b) the presentation in the IEC61131-3 compliant SCL language of the so-called Token Player, a software entity designed to support feedback-based decision-making within the SICPN; (c) the validation of the effectiveness of the proposed formalism in controlling an extended configuration of the FESTO Modular Processing Station (MPS) using an Arduino microcontroller via two-way UART serial communications; (d) the modelling of a Digital Twin of the FESTO MPS testbed. The tests demonstrate that, during transitions, the colour and signal interpretation conditions enable the microcontroller to accurately schedule and dynamically reconfigure control actions while keeping the size of the PN-based controller small relative to the control problem's complexity.

Keywords: Discrete-event system control; Petri Nets; Manufacturing system control; Programmable Logic Controllers; IEC61131-3; Structured Control Language; Siemens TIA Portal.

Signal-Interpreted Coloured Petri Nets: A Modelling Tool for Rapid Prototyping in Feedback-Based Control of Discrete Event Systems

Matheus Ungaretti Borges[a], Alessandro Pilloni[b], Gustavo Ribeiro Pontes[c], Carla Seatzu[b], Eduardo José Lima II[a]

[a] *Graduate Program in Mechanical Engineering, Universidade Federal de Minas Gerais, Av. Pres. Antônio Carlos, 6627, Belo Horizonte, 31270-901, MG, Brazil*
[b] *Department of Electrical and Electronic Engineering, University of Cagliari, Piazza D'Armi, Cagliari, 09123, CA, Italy*
[c] *Universidade Federal de Minas Gerais, Av. Pres. Antônio Carlos, 6627, Belo Horizonte, 31270-901, MG, Brazil*

Abstract

Petri nets (PNs) are typically used for design and verification rather than direct control implementation. In this paper, aligning with the Industry 4.0 paradigm's focus on flexible and reconfigurable control systems, we propose a modelling tool for rapidly prototyping feedback-based discrete-event control algorithms on programmable controllers such as PLCs or microcontroller boards. This modelling tool, named *Signal Interpreted Coloured Petri Nets* (SICPNs), aims to combine the formal modelling expressiveness of Coloured PNs with the capabilities of Signal Interpreted PNs, which are specialised in processing plant measurements and determining actuator commands.
This contribution involves: a) the formal definition of SICPN; b) the presentation in the IEC61131-3 compliant SCL language of the so-called *Token Player*, a software entity designed to support feedback-based decision-making within the SICPN; c) the validation of the effectiveness of the proposed formalism in controlling an extended configuration of the FESTO Modular Processing Station (MPS) using an Arduino microcontroller via two-way UART serial communications; d) the modelling of a Digital Twin of the FESTO MPS testbed. The tests demonstrate that, during transitions, the colour and signal interpretation conditions enable the microcontroller to accurately schedule and dynamically reconfigure control actions while keeping the size of the PN-based controller small relative to the control problem's complexity.

Keywords: Discrete-event system control, Petri Nets, Manufacturing system control, Programmable Logic Controllers, IEC61131-3, Structured Control Language, Siemens TIA Portal

List of symbols:

$\mathbb{N}$: set of natural numbers;
$\mathbb{Z}$: set of integer numbers;
$\mathbb{N}_0$: set of non-negative integer numbers;
$\mathcal{Z}(A)$: set of all multisets over A;
$\mathcal{N}(A)$: set of all non-negative multisets over A;
$\boldsymbol{\varepsilon}$: empty multiset;
Co : colour function;
C_ℓ : set of possible colours;
P : set of places;
T : set of transitions;
I : set of logical inputs;
O : set of logical outputs;
$\boldsymbol{m}_i$: marking of place p_i;
$\boldsymbol{M}$: net marking vector;
φ : logical firing conditions;
ω : candidate output function;
Ω : output function.

1. Introduction

Petri nets (PNs) are extensively used across various domains as a formalism for modelling discrete event systems, since their introduction by Petri (1962).

A limitation of PNs is the explosion in size when used to model complex systems such as multi-product manufacturing systems and reconfigurable production lines. To tackle this challenge, *high-level Petri net* models have been derived. In some cases, they allow a more compact representation than a standard PN model. In other cases, the modelling power is increased. However, this typically leads to a loss in terms of capability analysis. In this regard, Long et al. (2015) recognise that High-Level Petri Nets (HLPNs), including Coloured Petri Nets (CPNs), revealed superior in modelling Industry 4.0 compliant applications than alternative formalisms. Also, the EU Commission's report on Digital Twins (DTs) (Flamigni et al., 2020) mentions HLPNs

as a candidate formalism for the DT implementation, emphasising the need for advanced PN models. As a result, HLPNs have recently gained standardisation as a *software engineering tool* for a wide range of concurrent Discrete Event Systems (DES) and, in particular, distributed systems, see ISO/IEC 15909-1 (IEC, 2019) for details in this regard.

1.1. Literature Review

A typical way to represent the structure of a PN is through the use of graphical elements such as *places*, *transitions*, *arcs* and *tokens*. The dynamics of the net involve the movement of tokens through the net when a sequence of enabled transitions fire, causing an update in the net marking (Murata, 1989).

The PN formalism is generally used to solve supervisory control problems (Lima II and Dórea, 2004; Awad, 2018; Singh and Singh, 2019; Flochová and Lojan, 2019; Chen and Hu, 2020; Bashir et al., 2021; Basile et al., 2021; Hu et al., 2021; Wang et al., 2021; You et al., 2021), and to model complex and flexible systems such as manufacturing systems (Simon et al., 2018; Zhang et al., 2018; Nabi and Aized, 2019; Gaona et al., 2021; Fernández et al., 2021; Azkarate et al., 2021; Grobelna and Karatkevich, 2021), embedded systems (Xia, 2016; Comlan et al., 2017; Gomes and Barros, 2018; Berger et al., 2019) and communication networks (Machado et al., 2018; Juranić et al., 2019; Farah et al., 2019; Cao et al., 2020; Wu et al., 2021).

To enhance the expressiveness of PNs, various types of HLPNs have been developed, including continuous (Vázquez et al., 2014; Desirena-López et al., 2019; Liu et al., 2021), hybrid (Outafraout et al., 2020; Hüls et al., 2021; Li et al., 2022; Liu et al., 2022), and coloured PNs (Jensen, 1981, 1987; Long et al., 2015; Gehlot, 2019; Sheng and Prescott, 2019), to name a few.

CPNs were first introduced by Jensen (1981) and further developed into a more detailed methodology (Jensen, 1987). Unlike classical PNs, CPNs use tokens of different colours, thus enabling the representation of specific system features or characteristics. This makes CPNs particularly well-suited for modelling multi-product manufacturing systems and priority queue data structures.

Notable recent applications of CPNs include their use to model the process of sending different printing jobs to different printers (Gehlot, 2019), and applications to model all the factors and activities related to aircraft fleet maintenance (Sheng and Prescott, 2019). For additional examples of applications, refer to Jensen and Kristensen (2009, Chapter 14). Finally, it is worth mentioning Farah et al. (2019), where the CPN framework is used for modelling and analysing networked systems in which competition, synchronisation, resource sharing and parallelism are present, as for the Ethernet communication system.

Considering that the input-output response of a dynamical system is often referred to as *low-level behaviour*, some authors (Frey, 2000, 2002; Minas and Frey, 2002; Ramírez-Treviño et al., 2003) use the term low-level PNs to characterise the use of PNs for fine-grained modelling of interacting systems by acquiring and interpreting *low-level signals* from the field. Consequently, to employ them for feedback control purposes, definitions for Signal Interpreted Petri Nets (SIPN) were introduced in the early 2000s (Ramírez-Treviño et al., 2003; Frey, 2000, 2002). Further note the IEC61131-3 Sequential Function Chart (SFC) visual programming language for PLCs is also a primitive version of binary SIPN (Lewis, 1998). In essence, a SIPN can be regarded as a specialised type of PN designed to incorporate input signals into their firing conditions and generate outputs in each place. This feature makes them well-suited for formally representing sequential algorithms, which can then be implemented on Programmable Logic Controllers (PLCs) (Frey, 2000; Klein et al., 2003) or translated into structured text for microcontroller code (Borges and Lima II, 2018).

As an illustration of SIPN applications, Lima II and Dórea (2004) utilised them to control a manufacturing cell system where identical workpieces underwent sequential drilling, testing, and unloading, all managed by a PLC. There it is shown also how translating a SIPN into a controller using the IEC61131-3 compliant Structured Test (ST) language. Among others, notable due to the growing demand in the field is Grobelna et al. (2016), where a SIPN is used to automate a smart home system; Grobelna and Szcześniak (2022), focusing on SIPN applications in battery management system control; Basile et al. (2021), who propose a novel framework based on Timed Petri Nets (TPNs) for supervisory control purposes; and Basile and Ferrara (2022), who utilise TPNs to develop a fault detection algorithm for discrete event systems within a PLC.

1.2. Paper contributions

PNs and CPNs are typically used for hierarchical design and control verification at a high level, rather than for control implementation, due to the lack of low-level signal-interpretation characteristics (Grobelna et al., 2016; Farah et al., 2019). In contrast, when compared to CPNs, SIPNs lack high-level expressivity features, resulting in high-dimensional models. Therefore, the **primary contribution** of this work is to introduce a

new PN formalism named Signal Interpreted Coloured Petri Net (SICPN). This model combines the advantages of CPNs by providing them with low-level signal interpretation functionalities, which are useful for control purposes.

In line with this perspective, the paper's **second contribution** is to introduce what we call a *Token player*, a software entity aimed at supporting input-output feedback-based decision-making within the SICPN controller. Then, we illustrate how to deploy a SICPN-based controller into a PLC. Specifically, among the five IEC 61131-3 PLC languages, we utilise the Structured Control Language (SCL), namely the Siemens' compliant version of the ST language.

It is important to note that, compared to CPN supervisory control solutions and low-level SIPN controllers, in a SICPN, there is no hierarchy. Thus, both the interpretation of low-level signals and the verification of colour transitions operate at the same decision level (Frey, 2002; Grobelna et al., 2016; Farah et al., 2019; Basile et al., 2021).

As **third contribution**, we demonstrate the effectiveness of the proposed formalism for prototyping complex feedback controllers, such as control an extended configuration of the *FESTO Modular Processing Station* (MPS) (Ebel and Pany, 2015). More precisely, as proof-of-concept, we utilised an inexpensive Arduino MEGA board to run an ad-hoc designed SICPN-based controller for controlling operations and the control reconfiguration.

Translating a PN into C-code or running it on Arduino is not new. For example, Comlan and Delfieu (2019) introduced a tool named Petri Nets to Arduino (PN2A), which facilitates the modelling of systems with embedded Time Petri Nets (TPNs). Additionally, Borges and Lima II (2018) discuss conversion methodologies from SIPN to either Ladder diagram (LD) or C-code. On the other hand, Gomes and Barros (2018) provide a tool to translate Input-Output Place Transition (IOPT), which is a class of PNs supported by the IOPT Tools. However, unlike these works, our formalism encompasses both coloured firing transitions and tokens and includes I/O decision-making. Moreover, it is also noteworthy that while previous works simply show how to implement a single transition into a control language, here our approach showcases the implementation of the overall Token-Player.

Finally, as the **fourth and last contribution**, and in light of the absence of a real FESTO MPS system, we provide details for the modelling and implementation of a Digital Twin (DT) of the FESTO MPS testbed. Experimental tests demonstrating the interaction between the controller running on the Arduino and the Digital Twin, which operates on MATLAB through two-way UART serial communications, are also discussed. As a result, the microcontroller operates seamlessly, whether controlling a real plant or a simulated one. Additional studies involving the use of Digital Twins for the FESTO MPS system include Mykoniatis and Harris (2021) and Abdelsattar et al. (2022), which focus on the standard configuration depicted in Figure 1, as well as Batchkova et al. (2013), which explore an extended configuration closely resembling the one under consideration, but simpler.

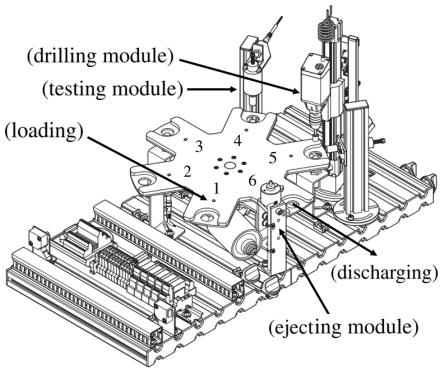


Figure 1: Standard Configuration of the FESTO S-BE-M Processing Station (Ebel and Pany, 2015). Note the extended configuration we are going to consider encompasses supplementary testing and drilling modules, situated at position indexes 2 and 3, respectively.

1.3. Paper organisation

In Section 2, we introduce the SICPN model, with particular emphasis on its formal definition and an in-depth exploration of its sequential behaviour, including dynamic and I/O aspects. Subsequently, we present a compiler designed to translate a SICPN into IEC61131-3 SCL code to support the prototyping of feedback-based controllers within a PLC architecture.

In Section 3, we illustrate our manufacturing domain case study. Section 4 details the modelling of the FESTO MPS controller using the proposed formalism. Then, details regarding the implementation of the plant Digital Twin, and how the experimental tests are conducted, are explained in Section 5. Finally, Section 6 offers a summary of our findings and outlines future directions.

2. Signal Interpreted Coloured Petri Nets

2.1. Preliminary notions on multisets

Let A be a *set*, its *cardinality* is $|A|$. A *multiset* (resp. *non-negative multiset*) $\boldsymbol{\alpha} = \sum_{a \in A} \alpha(a) \otimes a$ is defined by

a mapping $\alpha : A \to \mathbb{Z}$ (resp. $\alpha : A \to \mathbb{N}_0$), where $\otimes$ denotes the *multiset constructor operator*.

Then, $\mathcal{Z}(A)$ (resp. $\mathcal{N}(A)$) denotes the set of *all multisets* (resp. *all non-negative multisets*) over A, whereas $\boldsymbol{\varepsilon} = \sum_{a \in A} \varepsilon(a) \otimes a$ is the so-called *empty multiset* where $\forall\ a \in A$ it results that $\varepsilon(a) = 0$. Finally, $\wedge$, $\vee$ and $\neg$ denote, resp., the *and*, *or*, and *not* logical operators.

2.2. SICPN Definition

A SICPN N is defined by the following 11-tuple

$$N = (P,\ T,\ Co,\ \boldsymbol{Pre},\ \boldsymbol{Post}, \boldsymbol{M}_0,\ I,\ O,\ \varphi,\ \omega,\ \Omega), \quad (1)$$

where $P = \{p_1, p_2, \ldots, p_m\}$, with cardinality $|P| = m$, and $T = \{t_1, t_2, \ldots, t_n\}$, with cardinality $|T| = n$, are the finite sets of *places* and *transitions*, respectively, and $Co : P \cup T \to C_\ell$ is a *colour function* that associates to each element in $P \cup T$ a non-empty ordered set of colours in the set of possible colours C_ℓ. Therefore, $\forall\ p_i \in P$, $Co(p_i) = \{a_{i,1}, a_{i,2}, \ldots, a_{i,u_i}\} \subseteq C_\ell$ is the *ordered set of possible colours of tokens in* p_i, and $u_i = |Co(p_i)|$ is the *number of possible colours of tokens in* p_i. Analogously, $\forall\ t_j \in T$, $Co(t_j) = \{b_{j,1}, b_{j,2}, \ldots, b_{j,v_j}\} \subseteq C_\ell$ is the *ordered set of possible occurrence colours of* t_j, and $v_j = |Co(t_j)|$ is the *number of possible occurrence colours in* t_j.

Then, with $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ we represent the *pre-* and *post-incidence* matrices with dimensions $m \times n$. Here, each element (i, j) of either $\boldsymbol{Pre}$ or $\boldsymbol{Post}$ is denoted by $\boldsymbol{Pre}(p_i, t_j) : Co(t_j) \to \mathcal{N}(Co(p_i))$, or $\boldsymbol{Post}(p_i, t_j) : Co(t_j) \to \mathcal{N}(Co(p_i))$, which is a mapping from the set of occurrence colours of t_j (with cardinality v_j) to a *non-negative multiset* over the set of colours of place p_i (with cardinality u_i), $\forall\ i = 1, 2, \ldots, m$, and $j = 1, 2, \ldots, n$.

In simple words, $\boldsymbol{Pre}(p_i, t_j)$ ($\boldsymbol{Post}(p_i, t_j)$) specifies how many tokens of each colour in p_i should be removed from p_i (put in p_i, respectively) when t_j fires with respect to any possible colour associated with it. To maintain a concise notation, as depicted in Figure 2, we compactly represent each entry of $\boldsymbol{Pre}$ and $\boldsymbol{Post}$, resp. $\boldsymbol{Pre}(p_i, t_j)$ and $\boldsymbol{Post}(p_i, t_j)$, as $|Co(p_i)| \times |Co(t_j)|$ matrices. Thus, the generic entry $\boldsymbol{Pre}(p_i, t_j)(h, k)$ is equal to the number of tokens of colour $a_{i,h}$ removed from p_i when transition t_j fires w.r.t. colour $b_{j,k}$. Similarly, $\boldsymbol{Post}(p_i, t_j)(h, k)$ is equal to the number of tokens of colour $a_{i,h}$ put in p_i when transition t_j fires w.r.t. colour $b_{j,k}$. This holds for $h = 1, 2, \ldots, u_i$, and $k = 1, 2, \ldots, v_j$.

Let $m_i : Co(p_i) \to \mathbb{N}_0$ be the mapping associating with each possible token colour $a_{i,h} \in Co(p_i)$ a non-negative integer representing the number of tokens of that colour in the given place. Then the *marking of place* p_i is

$$\boldsymbol{m}_i = \sum_{a_{i,h} \in Co(p_i)} m_i(a_{i,h}) \otimes a_{i,h}. \quad (2)$$

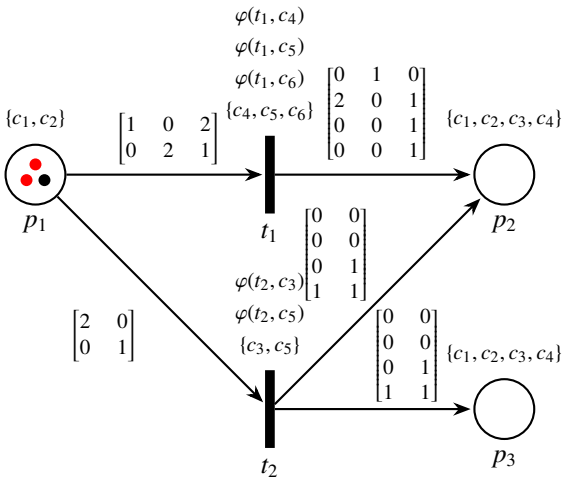


Figure 2: A SICPN $\langle N, \boldsymbol{M}_0 \rangle$ with $\boldsymbol{M}_0 = [2 \otimes c_1 + 1 \otimes c_2,\ \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon}]$.

The *net's marking* is an m-dimensional column vector of non-negative multisets

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{m}_1, & \boldsymbol{m}_2, & \cdots, & \boldsymbol{m}_m \end{bmatrix}, \quad (3)$$

and $\boldsymbol{M}_0$ denotes the *net's initial marking*. Thus, $\langle N, \boldsymbol{M}_0 \rangle$ indicates that N in (1) is initialised at $\boldsymbol{M}_0$.

Taking inspiration from Frey (2000), and to equip a generic CPN $N' = (P, T, Co, \boldsymbol{Pre}, \boldsymbol{Post}, \boldsymbol{M}_0)$ with input signal interpretation and output decision-making, we introduce the following additional fields in (1).

Set $I = \{i_1, i_2, \ldots, i_d\}$ denotes the *set of logical signals* received as *input* to the net from the environment. Set $O = \{o_1, o_2, \ldots, o_z\}$ indicates the *set of logical outputs* (or *action* to be done or not). It is $I \cap O = \varnothing$, and $d = |I|$, $z = |O|$. Notice that I can be seen as a set of *measurements* from a plant to be controlled, while O can be seen as the set of *commands* issued to the plant's actuators.

Then, with $\varphi(t_j, b_{j,k}) : t_j \times Co(t_j) \to f_{j,k}(i_1, i_2 \ldots, i_d)$ is denoted a mapping that associates with each transition $t_j \in T$, and each occurrence colour $b_{j,k}$, a *logical firing condition* w.r.t. I, namely $f_{j,k} : \{i_1, i_2, \ldots, i_d\} \to \{0, 1\}$. Here, $f_{j,k} = 1$ means that t_j is enabled to fire w.r.t. $b_{j,k}$ provided that it is marking enabled, i.e., a sufficiently large number of tokens of appropriate colours are contained in it. On the contrary, $f_{j,k} = 0$ means that t_j is not enabled to fire w.r.t. $b_{j,k}$ even if it is marking enabled.

Furthermore, with $\omega(p_i, a_{i,h}) : p_i \times Co(p_i) \to \{0, 1, -\}^{|O|}$ is denoted a *candidate output configuration*

for each $p_i \in P$ marked with a token of colour $a_{i,h}$. Here, $\omega(p_i, a_{i,h})(r) = 0$ indicates the candidate status for the action associated with o_r is *not requested* when place p_i is marked w.r.t. colour $a_{i,h}$; conversely $\omega(p_i, a_{i,h})(r) = 1$ signifies that such an action is *requested*. Finally, $\omega(p_i, a_{i,h})(r) = -$ implies that performing or not such an action is *irrelevant*, with '$-$' signifying '*don't care*'.

We define the *output function*

$$\Omega : \boldsymbol{M} \rightarrow \{\ 0, 1, -, q, r_0, r_1, q_0, q_1, q_{01}\ \}^{|O|} \qquad (4)$$

that computes the output configuration at marking $\boldsymbol{M}$, for each output o_r, by verifying the congruence of all the candidate output configurations. To formalise this, we preliminarily introduce the set

$$\Gamma = \{\ (p_i, a_{i,h}) \in P \times C_\ell\ :\ \boldsymbol{m}_i \neq \boldsymbol{\varepsilon},\ m_i(a_{i,h}) \neq 0\ \} \qquad (5)$$

associated with marking $\boldsymbol{M} = [\ \boldsymbol{m}_1, \cdots, \boldsymbol{m}_m\ ]$. The generic r-th component of Ω is equal to:

- *zero* (0) (resp. *one* (1)) if and only if there is only one pair $(p_i, a_{i,h}) \in \Gamma$ such that $\omega(p_i, a_{i,h})(r) = 0$ (resp. 1), while for all the other pairs in Γ, namely for all $(p, a) \in \Gamma \setminus \{(p_i, a_{i,h})\}$, it is $\omega(p, a)(r) = -$;
- *don't care* ($-$), if $\omega(p, a)(r) = -$ for all $(p, a) \in \Gamma$;
- *contradictory* (q) when there are 2 pairs $(p_i, a_{i,h}), (p_j, a_{j,k}) \in \Gamma$, (with p_i not necessarily different from p_j) such that $\omega(p_i, a_{i,h})(r) = 0$ and $\omega(p_j, a_{j,k})(r) = 1$, while for all the other pairs in Γ, it is $\omega(p, a)(r) = -$;
- *redundant zero* (r_0) (resp. *redundant one* (r_1) if there are at least 2 pairs $(p_i, a_{i,h}), (p_j, a_{j,k}) \in \Gamma$ (with p_i not necessarily different from p_j) such that $\omega(p_i, a_{i,h})(r) = \omega(p_j, a_{j,k})(r) = 0$ (resp. 1), while for all the other pairs in Γ, it is $\omega(p, a)(r) = -$;
- *contradiction and redundant zero* $\{q_0\}$ means there are at least 2 pairs $(p_i, a_{i,h})$, $(p_j, a_{j,k}) \in \Gamma$ (with p_i not necessarily different from p_j) such that $\omega(p_i, a_{i,h})(r) = \omega(p_j, a_{j,k})(r) = 0$ and exactly one pair is $\omega(p_x, a_{x,x})(r) = 1$, while for all the other pairs in Γ, it is $\omega(\cdot, \cdot)(r) = -$;
- *contradiction and redundant one* $\{q_1\}$ means there are at least 2 pairs $(p_i, a_{i,h})$, $(p_j, a_{j,k})$ (with p_i not necessarily different from p_j) such that $\omega(p_i, a_{i,h})(r) = \omega(p_j, a_{j,k})(r) = 1$ and exactly one pair is $\omega(p_x, a_{x,h})(r) = 0$, while for the other pairs in Γ, it is $\omega(\cdot, \cdot)(r) = -$;
- *contradiction and redundant zero and redundant one* $\{q_{01}\}$, means that there are at least 2 pairs $(p_i, a_{i,h}), (p_j, a_{j,k}) \in \Gamma$ such that $\omega(p_i, a_{i,h})(r) = \omega(p_j, a_{j,k})(r) = 0$ and at least 2 more different pairs $(p_x, a_{x,h})$, $(p_y, a_{y,k})$ such that $\omega(p_x, a_{x,h})(r) = \omega(p_y, a_{y,k})(r) = 1$, while for all the other pairs in Γ, it is $\omega(\cdot, \cdot)(r) = -$.

2.3. Net dynamics

Consider a SICPN $\langle N, \boldsymbol{M}_0 \rangle$. A transition $t_j \in T$ is said to be *enabled* w.r.t. colour $b_{j,k}$ at a marking $\boldsymbol{M}$ if and only if for each $p_i \in P$ and for all $h = 1, 2, \ldots, u_i$, it is $m_i(a_{i,h}) \geq \boldsymbol{Pre}(p_i, t_j)(h, k)$ and simultaneously the firing condition $\varphi(t_j, b_{j,k}) = f_{j,k}$ holds *true*. If an *enabled transition* t_j fires at $\boldsymbol{M}$ w.r.t. $b_{j,k}$, then we get a new marking $\boldsymbol{M}'$ where, for all $p_i \in P$ and for all $h = 1, 2, \ldots, u_i$, $m'_i(a_{i,h}) = m_i(a_{i,h}) + \boldsymbol{Post}(p_i, t_j)(h, k) - \boldsymbol{Pre}(p_i, t_j)(h, k)$. In the following, $\boldsymbol{M}[t_j(k)\rangle \boldsymbol{M}'$ denotes that t_j fires at $\boldsymbol{M}$ w.r.t. colour $b_{j,k}$ yielding $\boldsymbol{M}'$. A *firing sequence* from $\boldsymbol{M}_0$ is a (possibly empty) sequence of transitions

$$\sigma = t_{j_1}(k_{j_1})\ t_{j_2}(k_{j_2})\ \cdots\ t_{j_\tau}(k_{j_\tau}), \qquad (6)$$

each one firing w.r.t. a given colour, such that $\boldsymbol{M}_0[t_{j_1}(k_{j_1})\rangle \boldsymbol{M}_1[t_{j_2}(k_{j_2})\rangle \cdots \boldsymbol{M}_{\tau-1}[t_{j_\tau}(k_{j_\tau})\rangle \boldsymbol{M}$. Finally, $\boldsymbol{M}$ is said to be *reachable* from M_0 if and only if there exists a sequence σ such that $\boldsymbol{M}_0[\sigma\rangle \boldsymbol{M}$.

Remark 1. *A SICPN model assumes the presence of a controller who reads from the environment sensors the input signals, processes them along with the current net marking, and generates the outputs. Then, the outputs will set the plant actuator's commands, thus closing a feedback control loop. However, it is important to remark that the outputs do not directly affect the SICPN's dynamics.* ■

Remark 2. *Unlike standard PNs and CPNs, in SICPNs, as well as in SIPNs, the input signals I affect the net dynamics. As a consequence, well-known properties of PNs should be completely revisited in their definition and even more, in their analysis. Examples in this respect are: (a) liveness, indicating whether or not the model gets stuck in a deadlock state; (b) reachability, which refers to the ability to reach a particular marking or state; (c) reversibility, assessing whether it is possible to return to the initial state; (d) safety, determining whether all places contain at most one token. In addition, when SICPNs are used to derive a feedback controller as in this paper, what would be most interesting and challenging is the definition of the above properties w.r.t. the closed-loop system. Formalising such properties, establishing if they are decidable, and eventually*

proving an analysis approach, clearly goes beyond the scope of this paper and is left as a future work as mentioned in Section 6. ■

2.3.1. Example of SICPN dynamics

Consider the SICPN in Figure 2 with $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2\}$ and $C_\ell = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. The set of logical inputs and outputs are equal to $I = \{i_1, i_2, i_3, i_4\}$ and $O = \{o_1, o_2, o_3, o_4, o_5\}$, respectively. In this example, each input and output is assumed to be boolean. Furthermore, inputs are initialised at

$$i_1 = 1,\ i_2 = 1,\ i_3 = 0,\ i_4 = 0 \tag{7}$$

Place p_1 (resp. p_2, p_3) may only contain tokens of colours $Co(p_1) = \{c_1, c_2\}$ (resp. $Co(p_2) = \{c_1, c_2, c_3, c_4\}$, $Co(p_3) = \{c_1, c_2, c_3, c_4\}$) and $Co(t_1) = \{c_4, c_5, c_6\}$ (resp. $Co(t_2) = \{c_3, c_5\}$) means that t_1 (resp. t_2) may fire only w.r.t. to either c_4, or c_5, or c_6 (resp. c_3 or c_5) provided that corresponding firing conditions, which are defined as follows, hold true:

$$\varphi(t_1, c_4) = f_{1,4}(i_1, i_2, i_3, i_4) = i_1 \wedge i_2, \tag{8}$$
$$\varphi(t_1, c_5) = f_{1,5}(i_1, i_2, i_3, i_4) = i_1 \wedge \neg i_3, \tag{9}$$
$$\varphi(t_1, c_6) = f_{1,6}(i_1, i_2, i_3, i_4) = i_2 \wedge \neg i_4, \tag{10}$$
$$\varphi(t_2, c_3) = f_{2,3}(i_1, i_2, i_3, i_4) = i_1 \wedge \neg i_3, \tag{11}$$
$$\varphi(t_2, c_5) = f_{2,5}(i_1, i_2, i_3, i_4) = i_1 \wedge i_2. \tag{12}$$

Given the net's topology, each entry (i, j) of matrices $\boldsymbol{Pre}$ (resp. $\boldsymbol{Post}$), namely $\boldsymbol{Pre}(p_i, t_j)$ (resp. $\boldsymbol{Post}(p_i, t_j)$) is a non-empty non-negative multiset if and only if the pre-arc (resp. post-arc) (p_i, t_j) exists, otherwise $\boldsymbol{Pre}(i, j) = \boldsymbol{\varepsilon}$. In this example, the only non-empty $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ entries are

$$\boldsymbol{Pre}(p_1, t_1)(c_4) = 1 \otimes c_1, \tag{13}$$
$$\boldsymbol{Pre}(p_1, t_1)(c_5) = 2 \otimes c_2, \tag{14}$$
$$\boldsymbol{Pre}(p_1, t_1)(c_6) = 2 \otimes c_1 + 1 \otimes c_2, \tag{15}$$
$$\boldsymbol{Pre}(p_1, t_2)(c_3) = 2 \otimes c_1, \tag{16}$$
$$\boldsymbol{Pre}(p_1, t_2)(c_5) = 1 \otimes c_2, \tag{17}$$
$$\boldsymbol{Post}(p_2, t_1)(c_4) = 2 \otimes c_2, \tag{18}$$
$$\boldsymbol{Post}(p_2, t_1)(c_5) = 1 \otimes c_1, \tag{19}$$
$$\boldsymbol{Post}(p_2, t_1)(c_6) = 1 \otimes c_2 + 1 \otimes c_3 + 1 \otimes c_4, \tag{20}$$
$$\boldsymbol{Post}(p_2, t_2)(c_3) = 1 \otimes c_4, \tag{21}$$
$$\boldsymbol{Post}(p_2, t_2)(c_5) = 1 \otimes c_3 + 1 \otimes c_4, \tag{22}$$
$$\boldsymbol{Post}(p_3, t_2)(c_3) = 1 \otimes c_4, \tag{23}$$
$$\boldsymbol{Post}(p_3, t_2)(c_5) = 1 \otimes c_3 + 1 \otimes c_4. \tag{24}$$

A simpler and smart way to define the $\boldsymbol{Pre}$ (resp. $\boldsymbol{Post}$) matrix is depicted in Figure 2. Here, each non-empty multiset $\boldsymbol{Pre}(p_i, t_j)$ (resp. $\boldsymbol{Post}(p_i, t_j)$) is a $|Co(p_i)| \times |Co(t_j)|$ matrix of non-negative integers. Let the net's initial marking be

$$\boldsymbol{M}_0 = \left[\ 2 \otimes c_1 + 1 \otimes c_2,\quad \boldsymbol{\varepsilon},\quad \boldsymbol{\varepsilon}\ \right], \tag{25}$$

upon verification of (8)-(12), it is observed that both t_1 and t_2 may fire. Specifically, t_1 could fire w.r.t. either c_4 or c_6, while t_2 could fire w.r.t. either c_3 or c_5. Notice that we assume a policy that first gives priority to transitions with lower index and then gives priority to colours with lower index. Therefore, in this case, transition t_1 fires w.r.t. c_4. After that, the net's marking becomes

$$\boldsymbol{M}_1 = \left[\ 1 \otimes c_1 + 1 \otimes c_2,\quad 2 \otimes c_2,\quad \boldsymbol{\varepsilon}\ \right]. \tag{26}$$

Note that since in a SICPN several signals can change simultaneously, it is fundamental to update the net's marking as soon as a transition fires. Following this policy, from $\langle N, \boldsymbol{M}_1 \rangle$ and recalling (7), then only t_2 w.r.t. c_5 can fire, thus yielding the next marking

$$\boldsymbol{M}_2 = [1 \otimes c_1,\ 2 \otimes c_2 + 1 \otimes c_3 + 1 \otimes c_4, 1 \otimes c_3 + 1 \otimes c_4]\,.$$

Now, assume the candidate output functions ω are defined as follows:

$$\omega(p_1, c_1) = \left[1,\ -,\ -,\ -,\ -\right], \tag{27}$$
$$\omega(p_1, c_2) = \left[-,\ -,\ -,\ -,\ -\right], \tag{28}$$
$$\omega(p_2, c_1) = \left[1,\ -,\ 1,\ -,\ -\right], \tag{29}$$
$$\omega(p_2, c_2) = \left[-,\ 1,\ 0,\ -,\ 0\right], \tag{30}$$
$$\omega(p_2, c_3) = \left[-,\ 0,\ 0,\ 1,\ -\right], \tag{31}$$
$$\omega(p_2, c_4) = \left[1,\ -,\ 0,\ 0,\ 1\right], \tag{32}$$
$$\omega(p_3, c_1) = \left[1,\ -,\ -,\ -,\ -\right], \tag{33}$$
$$\omega(p_3, c_2) = \left[-,\ 1,\ 0,\ -,\ -\right], \tag{34}$$
$$\omega(p_3, c_3) = \left[-,\ -,\ 0,\ 1,\ 0\right], \tag{35}$$
$$\omega(p_3, c_4) = \left[0,\ -,\ 0,\ -,\ 1\right]. \tag{36}$$

We will now explain how the output function Ω in (4) relates to the current marking. Let us focus on $\boldsymbol{M}_0$. Since only p_1 is marked (with colours c_1 and c_2), based on eq. (27)-(28), it is $\Omega(\boldsymbol{M}_0) = [\ 1,\ -,\ -,\ -,\ -\ ]$.
If instead we consider $\boldsymbol{M}_1$, where p_1 is marked with colours c_1 and c_2, while p_2 is marked with colour c_2. Thus, based on eq. (27), (28) and (30), it is $\Omega(\boldsymbol{M}_1) = [\ 1,\ 1,\ 0,\ -,\ 0\ ]$. Finally, if we consider $\boldsymbol{M}_2$, from (30)-(32), it is $\Omega(\boldsymbol{M}_1) = [\ r_1,\ q,\ r_0,\ q_1,\ q_{01}\ ]$. The first entry follows from eq. (27) and (32); the second entry follows from eq. (30) and (31); the third entry follows

from (30)-(32), (35) and (36); while the fourth entry is because there is simultaneously a contradiction and a redundancy 1 due to (31), (32) and (35); finally, the fifth entry follows from eq. (30), (35) that propose 0 while (32), (36) that propose 1.

Remark 3. *In real discrete-event control systems, outputs are typically Booleans. Moreover, at each time cycle, they must be uniquely assigned to avoid unexpected behaviours. In SICPNs the output function* Ω *generates more than simply "*1*" or "*0*" symbols. This allows* Ω *to function as an "Always On WatchDog", aimed at detecting bugs in the code during the controller commissioning or generating alarms whenever a contradiction is detected due to faults or other issues. Also having undefined outputs "–" should also be avoided by design, while any redundant zeros or redundant ones will be translated to* 0 *or* 1*, respectively.* ■

2.4. IEC61131-3 compliant SICPN compiler

A SICPN can be viewed as an input-output function that determines the configuration for O based on an input configuration I and the current net marking $\boldsymbol{M}$. This makes it suitable for designing feedback controllers for discrete-event systems. Moreover, compared with SIPN (Frey, 2002; Borges and Lima II, 2018) or more primitive formalisms such as the SFC (Sequential Function Chart) (which is based on binary PNs (Lewis, 1998)), the presence of coloured tokens/transitions may lead to lower-dimensional models.

To run a SICPN within a PLC or any other control board, two key aspects are crucial:

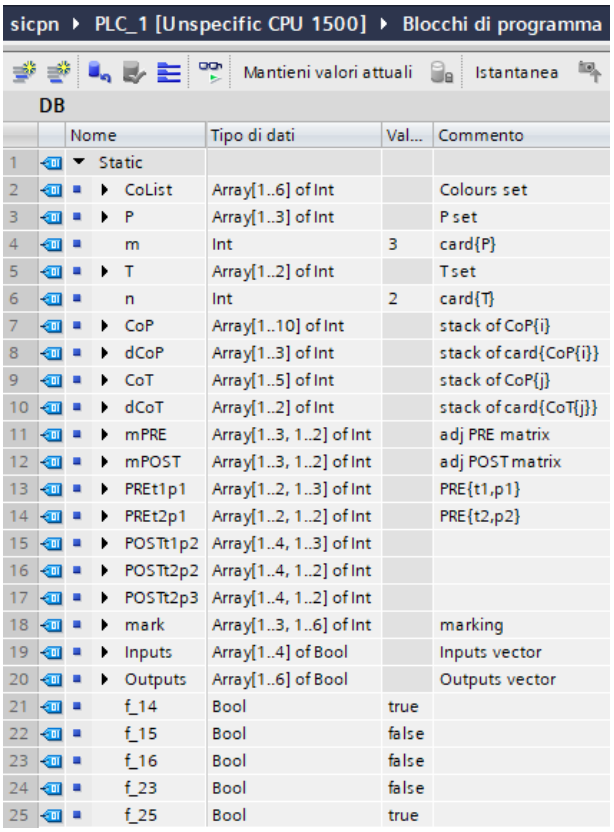
a) properly defining the net's primitives within the controller's memory. This involves aligning them with available data-type structures;

b) developing a software entity, referred to as the Token-Player, tasked to correctly implement the SICPN dynamics, see e.g. Subsection 2.3.1.

Here we will focus on explaining how to implement it within a PLC. Specifically, among the five IEC 61131-3 languages (LD, ST, SFC, FBD, IL), we utilise the Siemens' compliant version of ST, known as SCL (Structured Control Language). Note that since the SCL is based on *Pascal*, the migration to other textual languages, e.g. C or C++, becomes quite straightforward.

2.4.1. Memory allocations and primitive definitions

To formalise this task, we initially worked in MATLAB and then transitioned to SCL. In this regard, and following point a), the major difficulty we encountered was the lack of advanced data-type structures such as *cell-arrays* (Mathworks, 2024), which would have been ideal for defining each multiset entry (i, j) of $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ as a $|Co(p_i)| \times |Co(t_j)|$ indexed matrix if the arc (p_i, t_j) exists, otherwise, such (i, j) entry would be an empty cell. Unfortunately, PLCs do not support such advanced containers but only traditional multi-dimensional arrays, see Figure 3. To overcome this issue, we thus introduced some additional support variables in the IEC61131-3 compliant `GLOBAL DB` used to store the Net's primitives, the current marking (2) and values of the transition functions $f_{j,k}$, see Figure 3 for details. More precisely, concerning the example in Figure 2, and let $C_\ell = \{c_1, c_2, \cdots, c_6\}$ for simplicity be represented by the ordered set of natural numbers from 1 to 6, these additional support variables are



	Nome	Tipo di dati	Val...	Commento
1	Static			
2	CoList	Array[1..6] of Int		Colours set
3	P	Array[1..3] of Int		P set
4	m	Int	3	card{P}
5	T	Array[1..2] of Int		T set
6	n	Int	2	card{T}
7	CoP	Array[1..10] of Int		stack of CoP{i}
8	dCoP	Array[1..3] of Int		stack of card{CoP{i}}
9	CoT	Array[1..5] of Int		stack of CoP{j}
10	dCoT	Array[1..2] of Int		stack of card{CoT{j}}
11	mPRE	Array[1..3, 1..2] of Int		adj PRE matrix
12	mPOST	Array[1..3, 1..2] of Int		adj POST matrix
13	PREt1p1	Array[1..2, 1..3] of Int		PRE{t1,p1}
14	PREt2p1	Array[1..2, 1..2] of Int		PRE{t2,p2}
15	POSTt1p2	Array[1..4, 1..3] of Int		
16	POSTt2p2	Array[1..4, 1..2] of Int		
17	POSTt2p3	Array[1..4, 1..2] of Int		
18	mark	Array[1..3, 1..6] of Int		marking
19	Inputs	Array[1..4] of Bool		Inputs vector
20	Outputs	Array[1..6] of Bool		Outputs vector
21	f_14	Bool	true	
22	f_15	Bool	false	
23	f_16	Bool	false	
24	f_23	Bool	false	
25	f_25	Bool	true	

Figure 3: Screenshot of the Siemens TIA Portal V16 IDE, displaying the `GLOBAL DB` used to store the primitives of the SICPN as shown in Figure 2, and (37)-(39).

$$\texttt{CoP} = \left[\begin{array}{cc|cccc|cccc} 1 & 2 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \end{array}\right]^{\mathsf{T}} \quad (37)$$

$$\texttt{CoT} = \left[\begin{array}{ccc|cc} 4 & 5 & 6 & 3 & 5 \end{array}\right]^{\mathsf{T}} \quad (38)$$

aimed to stack each $Co(p_i)$ (resp. $Co(t_j)$), and

$$\texttt{dCoP} = \left[\begin{array}{ccc} 2 & 4 & 4 \end{array}\right]^{\mathsf{T}} \quad , \quad \texttt{dCoT} = \left[\begin{array}{cc} 3 & 2 \end{array}\right]^{\mathsf{T}} \quad (39)$$

aimed at storing the cardinality of each $Co(p_i)$ (resp. $Co(t_j)$); see lines 7-10 of Figure 3. Moreover, for memory optimisation, only the non-empty entries of ***Pre*** (resp. ***Post***) are stored, as shown in lines 13-17 of Figure 3. Then, we introduced in lines 11-12 of Figure 3 the following support matrices

$$\texttt{mPRE} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \texttt{mPOST} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}, \tag{40}$$

to keep track to which pairs (p_i, t_j) there exists (i.e. 1) or not (i.e. 0) a non-empty $\boldsymbol{Pre}(p_i, t_j)$ (resp. $\boldsymbol{Post}(p_i, t_j)$) matrix, cf. Figure 2 with (40) and with line 3 of Figure 4. The meaning of all the other variables in Figure 3 can instead be easily inferred from the comments provided on the right of the `GLOBAL DB` interface.

2.4.2. *Token-Player implementation*

In Figure 4 is provided a screenshot of the Siemens TIA Portal V16 IDE displaying the list of the SCL instructions we included in the IEC61131-3 compliant `FC` (function) tasked to implement the dynamics of the SICPN displayed in Figure 2, and with primitives in the `GLOBAL DB` depicted as in Figure 3.

Further note that the definition of the support variables in (37)-(39) was a strategy to allow the verification of the firing condition with respect to:

a) all transitions t_j (see index `#j` in line 1);

b) *all places* p_i (see index `#i` in line 2);

c) *all colours transitions* $b_{j,k} \in Co(t_j)$ (see `#b_jk` in line 6, or `k=DB.CoT[#b_jk]` in line 25);

d) *all colours places* $a_{i,h} \in Co(p_i)$ (see line 8 where `#h=1, 2,..., DB.CoP[#i]`);

while giving priority to lower indices. Further note that since the Token-Player in Figure 4 is thoroughly commented within the same figure, we will not delve deeply into its details hereafter.

However, it is essential to highlight that the red-highlighted blocks of code, i.e., lines 9-20, 28-39, and 45-59, are specialised for the SICPN shown in Figure 2.

In general, those code portions should be replaced with function calls specifically designed for the given SICPN (see the green comment in lines 9-11). However, we chose to make these instructions explicit here for clarity. With that said, please note that lines 12-20 are dedicated to checking whether the generic transition t_j is enabled to fire w.r.t. a pair of colours $b_{j,k}$, given the actual marking $\boldsymbol{m}_i$ and the $\boldsymbol{Pre}(p_i, t_j)(\cdot, k)$ conditions.

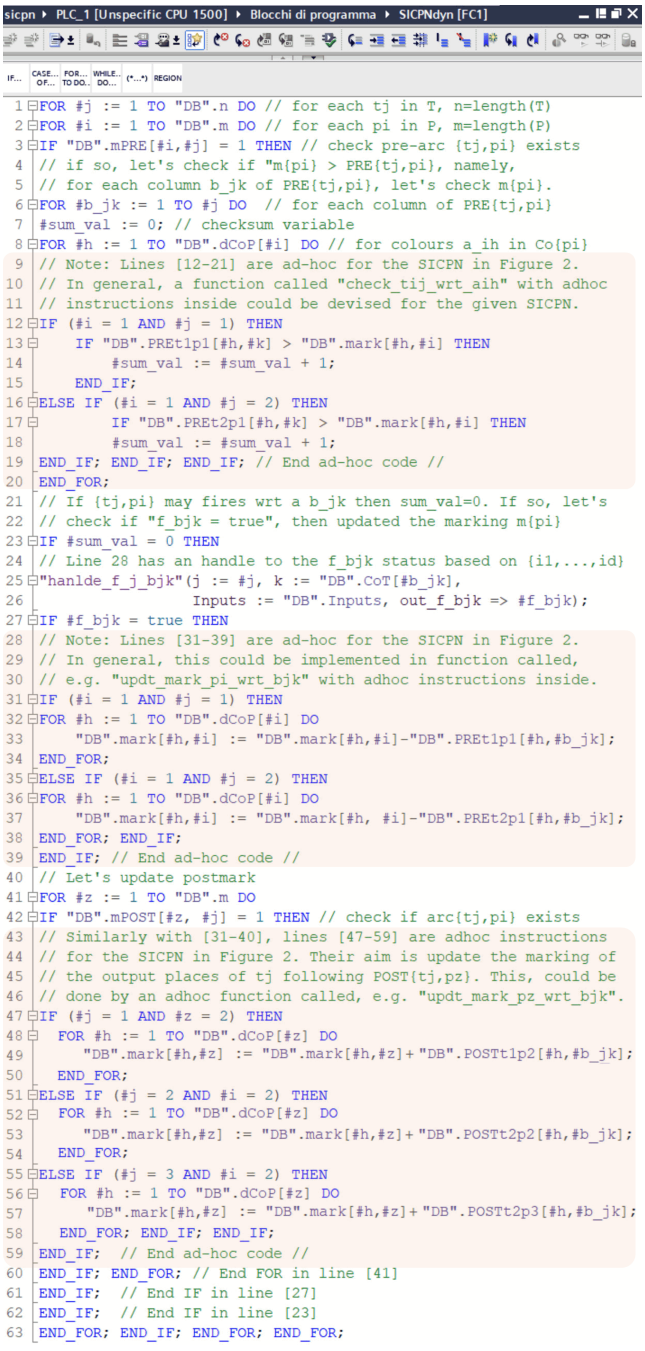


Figure 4: Screenshot of the Siemens TIA Portal V16 IDE displaying the list of SCL instructions included in the `FC` implementing the dynamics of the SICPN displayed in Figure 2, and with primitives defined as in Figure 3.

On the other hand, lines 31-39 deal with the net marking update based on the transition costs associated with the $\boldsymbol{Pre}(p_i, t_j)(\cdot, k)$ arc-weights, where there $\varphi_{j,k}(I)$ holds true (as indicated by `#f_jk=true` in line 27).

Table 1: Logical Statements of Sensors for the PI-1 Identification Task

Logical statement	Conclusion
IND	metal (M)
¬IND ∧ OPT	red plastic (RP)
¬IND ∧ ¬OPT ∧ CAP	black plastic (BP)
otherwise	no workpiece

Finally, lines 47-59 deal with the net marking update based on the $\boldsymbol{Post}(\cdot, t_j)(\cdot, k)$ arc-weights.

Notice that the given Token-Player `FC` in Figure 4 is expected to be executed within the main cyclic `OB1` organization block of the PLC. This ensures that during each scan cycle, the entire SICPN marking is continuously updated based on the current values of the input set I and marking $\boldsymbol{M}$. Further note immediately after the net marking is updated, a second `FC` aimed at implementing the output function Ω should be executed. This `FC` should implement the output function Ω, but it is omitted here due to space limitations. However, it can be easily derived based on equations (27)-(36).

3. Case study description

To validate the effectiveness of our formalism for directly implementing control algorithms, we will model and test a SICPN-based feedback control on an Arduino microcontroller equipped with a 16 MHz ATmega2560 CPU, 4kB of SRAM, and 256kB of Flash memory. The control is devised to coordinate actions on a Digital Twin representing an *extended configuration* of the FESTO MPS (Ebel and Pany, 2015). The control loop between the Arduino and the DT, which runs in MATLAB, is facilitated through two-way UART serial communications. Consequently, we will show that the proposed controller effectively and seamlessly coordinates the plant's module operations.

The FESTO MPS has long served as a benchmark in the field of manufacturing automation (Lima II and Dórea, 2004). Recently, it has also emerged as a common reference for DT applications and PLC virtual commissioning (Batchkova et al., 2013; Fernández et al., 2021; Mykoniatis and Harris, 2021; Abdelsattar et al., 2022). In contrast to previous studies, which typically focus on the standard FESTO MPS configuration as depicted in Figure 1, our research, akin to Batchkova et al. (2013), delves into an extended configuration. This expanded setup includes additional and different testing and drilling modules. Moreover, we consider workpieces of various materials, and requiring different operations.

3.1. Control problem formulation

The process to be automated is the FESTO MPS of Festo Didactic, global leader in technical training solutions for industrial and process automation. The testbed core is a DC-motor driven rotating table as shown in Figure 1. As the table rotates clockwise, the workpieces (which are cylindrical blocks of different materials) undergo different stations identified by a position-index (PI) until they are withdrawn, see Figure 1. Each PI identifies a specific station and a function, resp.:

- PI-1 Identification (ID) → A workpiece is here loaded from a buffer, and its material is identified;
- PI-2 Testing A (TA) → A testing module checks for a hole with a diameter $d_A > 0$ or larger in the piece;
- PI-3 Processing A (PA) → A drilling module pierces the workpiece using a tip with a diameter $d_A > 0$;
- PI-4 Testing B (TB) → A testing module checks for a hole with a diameter $d_B > d_A$ or larger in the piece;
- PI-5 Processing B (PB) → A drilling module pierces the workpiece using a tip with a diameter $d_B > d_A$;
- PI-6 Ejection (EJ) → The worked piece is pushed out using an ejector module.

Three types of block materials are considered: *metal* (M), *black plastic* (BP), and *red plastic* (RP). Additionally, we assume that metal blocks require two drilling operations, one at PA and another at PB, where the tip diameters satisfy $d_B > d_A > 0$. In contrast, plastic blocks require a single drilling operation at PB.

Remark 4. *Unlike the previously mentioned literature, the control will be formulated to handle* ***six pieces*** *being worked on* ***simultaneously*** *instead of just one. Additionally, in the workpiece arrival process, we assume that some blocks may not require any operations, while the control system must be capable of identifying and dealing with them. We emphasise this to underscore our problem's complexity and the fundamental role of using SICPN to keep the controller dimension small.* ■

To identify the materials, PI-1 is equipped with digital sensors. Specifically, a capacitive sensor (CAP) to detect the presence of a block, an optical sensor (OPT) identifying the material's reflectiveness, and an inductive sensor (IND) sensitive to metallic objects. Table 1 summarise the material identification logic.

Additional digital measurements from the table can serve purposes beyond the PI-1 task. When the table is aligned correctly with a PI (and consequently with others, given their equispaced), the signal IDX is high.

Sensors TA0, PA0, and TA1, PA1 (and their counterparts TB0, PB0, and TB1, PB1) correspond to the lower

Table 2: Meaning of the SICPN places in the case study.

Modules	Place	Meaning
Table	p_1	Number of Idle stations
	p_2	DC-Motor starts
	p_3	DC-Motor rotating
Resources	p_4	Table rotation request
	p_5	1 piece limiter on PI-1
PI-1 (Identification)	p_6	Workpiece arrived
	p_7	Waiting synchronisation
Rotation	p_8	Table is turning
PI-2 (Testing A)	p_9	Block on PI-2
	p_{10}	Testing block
	p_{11}	Testing block
	p_{12}	Block on PI-2
	p_{13}	Waiting synchronisation
Rotation	p_{14}	Table is turning
PI-3 (Processing A)	p_{15}	Block on PI-3
	p_{16}	Processing block
	p_{17}	Processing block
	p_{18}	Block on PI-3
	p_{19}	Waiting synchronisation
Rotation	p_{20}	Table is turning
PI-4 (Testing B)	p_{21}	Block on PI-4
	p_{22}	Testing block
	p_{23}	Testing block
	p_{24}	Block on PI-4
	p_{25}	Waiting synchronisation
Rotation	p_{26}	Table is turning
PI-5 (Processing B)	p_{27}	Block on PI-5
	p_{28}	Processing block
	p_{29}	Processing block
	p_{30}	Block on PI-5
	p_{31}	Waiting synchronisation
Rotation	p_{32}	Table is turning
PI-6 (Ejection)	p_{33}	Block on PI-6

and upper limit switches associated with the testing piston and processing tip at stations A and B, respectively. These signals indicate whether the actuator is retracted or extended. Finally, TAOK represents the output of the test at TA. Here, we assumed that this test is only required for "M" parts while "RP" or "BP" blocks can skip the test at PI-2. In contrast, all parts must undergo TB in PI-3, and TBOK denotes the test's output. Finally, CAPF is the capacitive sensor that identifies whether there is a block at the final PI-6 station, indicating that the block is ready to be removed from the table.

4. Feedback controller modelling via SICPN

Here is presented the modelling of the proposed SICPN feedback controller for the given case study, and each entry of 11-tuple in (1) will now be characterised.

Table 3: Token colours list and corresponding 4-bit lookup table.

Colour index	Material	$drilled_A$	$drilled_B$
c_0	00 Generic	-	-
c_1	01 Metal block	0	0
c_2		1	0
c_3		1	1
c_4	10 Red plastic block	0	0
c_5		1	1
c_6	11 Black plastic block	0	0
c_7		1	1

4.1. Modelling structure

Figure 5 illustrates the bipartite graph associated with the proposed SICPN controller. This is custom-designed to address the specifications outlined in Subsection 3.1. The meanings of each of the 33 places can be found in Table 2.

Remark 5. *The proposed 33 places SICPN can be considered compact, considering the control problem complexity and specifications. This involves managing 3 types of workpieces, coordinating up to 6 different operations simultaneously (one for each PI), addressing scenarios where no operation is necessary (e.g., when TA or TB tests fail or when there is no block in PI-1), and controlling the table rotation's DC motor.* ■

In Figure 5, the dashed boxes labelled PI-X with $X = 1, 2, \ldots, 6$, highlight the portion of our SICPN responsible for coordinating each PI task, while those labelled Table and Rotation-Request manage the rotation table's operations. Notably, places p_{21} to p_{32}, responsible for controlling operations in PI-4 and PI-5, have been omitted from Figure 5. This is because TB and PB operate on all the material types, while TA and PA operate only on metal parts, resulting in similar but simpler functionalities to implement.

In Figure 5, places p_4 and p_5 denote two resources:
- p_4 controls the connection and synchronisation of processes and table rotation by functioning as a *rotation request*, indicating that there must be at least one block on the table to request its rotation;
- p_5 limits the sum of tokens in p_6, p_7, and p_8 to one.

The initial marking of the net $\boldsymbol{M}_0$ comprises 6 generic c_0 tokens in p_1, indicating that all table stations are idle or have finished processing a block, if present. Moreover, there is 1 generic c_0 token in p_5, indicating that PI-1 is available to receive a block from the buffer queue (waiting area). It is also important to note that having $\boldsymbol{m}_1 = 6 \otimes c_0$ is a requirement for t_1 to fire.

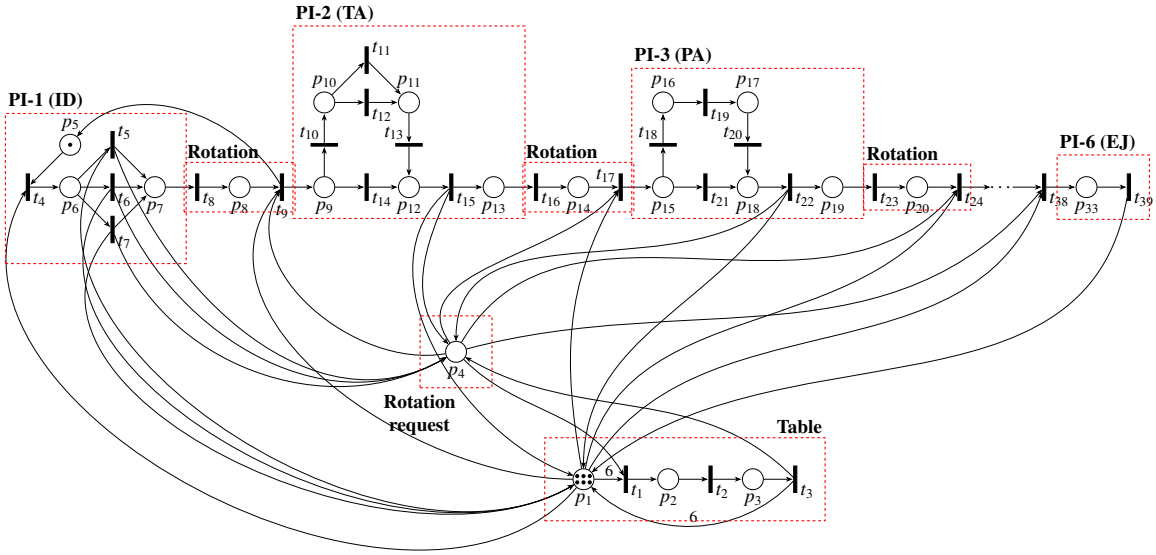


Figure 5: Proposed SICPN-based FESTO MPS control algorithm. Refer to Section 4, Table 2 and Figure 6 for details on the controller primitives.

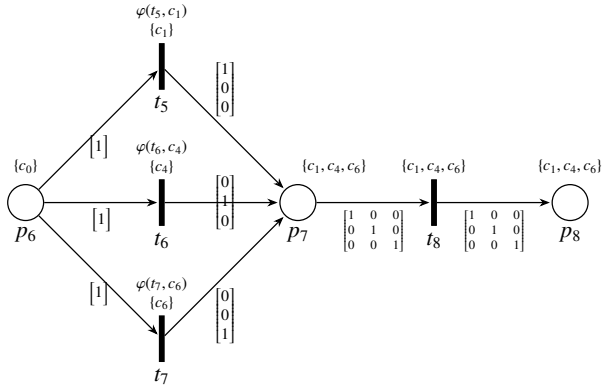


Figure 6: Zoom of a portion of the SICPN controller of Figure 5.

4.2. Colouring and Signal Interpretation

The list of token colours used to describe the system status is given in Table 3 along with the corresponding 4-bit lookup table. The set of input is as next

$$I = \{\ \text{IDX}, \text{IND}, \text{OPT}, \text{CAP}, \text{TA0}, \text{TA1}, \text{TAOK}, \quad (41)$$
$$\text{PA0}, \text{PA1}, \text{TB0}, \text{TB1}, \text{TBOK}, \text{PB0}, \text{PB1}, \text{CAPF}\ \}$$

Please refer to Section 3 for details on their meaning. Finally, the net's controller output, namely the set of commands to be delivered to the plant's actuators, is

$$O = \{\ MT,\ TA_{adv},\ TA_{ret},\ PA_{adv},\ PA_{ret},$$
$$TB_{adv},\ TB_{ret},\ PB_{adv},\ PB_{ret}\ \}, \quad (42)$$

where MT is an output enabling the motor to rotate the table, while TA_{adv}, TA_{ret}, PA_{adv}, PA_{ret}, TB_{adv}, TB_{ret}, PB_{adv}, and PB_{ret} are outputs that provide advance and retract commands for the pneumatic actuators associated with the station TA, PA, and TB, PB, respectively.

4.3. A detailed description of PI-1

To provide an understanding of how the controller in Figure 5 operates, we delve into more detail regarding the functions associated with the portions of the net labelled *PI-1* and *Rotation* in Figure 5. A zoomed-in view of this sub-net is provided in Figure 6, where the remaining net's primitives such as $\boldsymbol{Pre}(p_i, t_j)$, $\boldsymbol{Post}(p_i, t_j)$, $Co(p_i)$, $Co(t_j)$, and $\varphi(t_j, b_{j,k})$ are explicitly shown.

Here, when an unidentified (generic) workpiece arrives at PI-1, it triggers the firing of transition t_4 in Figure 5. Subsequently, a token c_0 enters place p_6 in Figure 6. Then, sensors IND, OPT, and CAP located in PI-1 perform the identification. More precisely, following Figure 6 and Table 3, the following transition rules are devised to enable the workpiece material identification:

$$\varphi(t_5, c_1) = \text{IND} \quad (43)$$
$$\varphi(t_6, c_4) = \neg\text{IND} \wedge \text{OPT} \quad (44)$$
$$\varphi(t_7, c_6) = \neg\text{IND} \wedge \neg\text{OPT} \wedge \text{CAP} \quad (45)$$

Then, $\boldsymbol{Pre}(p_i, t_j)$ and $\boldsymbol{Post}(p_i, t_j)$ in Figure 6 define which tokens are consumed from each pre-place and which tokens enter a post-place. Notice that the firing of either t_5, or t_6, or t_7 models the material identification task. Following Table 3, the corresponding post-place

will be marked by either a c_1, or c_4, or c_6 token, while c_0 in p_6 is consumed, namely

$$\boldsymbol{Pre}(p_6, t_5)(c_1) = 1 \otimes c_0 \quad (46)$$
$$\boldsymbol{Pre}(p_6, t_6)(c_4) = 1 \otimes c_0 \quad (47)$$
$$\boldsymbol{Pre}(p_6, t_7)(c_6) = 1 \otimes c_0 \quad (48)$$
$$\boldsymbol{Post}(p_7, t_5)(c_1) = 1 \otimes c_1 \quad (49)$$
$$\boldsymbol{Post}(p_7, t_6)(c_4) = 1 \otimes c_4 \quad (50)$$
$$\boldsymbol{Post}(p_7, t_7)(c_6) = 1 \otimes c_6 \quad (51)$$

Moreover, as soon as t_5, or t_6, or t_7 fires, in accordance with Figure 5, we also have

$$\boldsymbol{Post}(p_4, t_5)(c_1) = 1 \otimes c_0 \quad (52)$$
$$\boldsymbol{Post}(p_4, t_6)(c_4) = 1 \otimes c_0 \quad (53)$$
$$\boldsymbol{Post}(p_4, t_7)(c_6) = 1 \otimes c_0 \quad (54)$$
$$\boldsymbol{Post}(p_1, t_5)(c_1) = 1 \otimes c_0 \quad (55)$$
$$\boldsymbol{Post}(p_1, t_6)(c_4) = 1 \otimes c_0 \quad (56)$$
$$\boldsymbol{Post}(p_1, t_7)(c_6) = 1 \otimes c_0 \quad (57)$$

which means that in p_4 and p_1 a c_0 token will enter, meaning that the identification has ended, and the table can rotate from the PI-1 station's perspective.

However, notice that two conditions must hold to start the table rotation:

(i) there is at least 1 rotation request, namely p_4 contains at least 1 generic c_0 token (see Figure 5). To ensure this condition, a c_0 token is added to place p_4 when any process is finished, namely once one among t_5, t_6, t_7, t_{15}, or t_{22} is fired, cf. e.g. (52)-(54) w.r.t. PI-1;

(ii) all the workpieces on the table have finished processing if necessary, which is satisfied when p_1 contains 6 generic tokens (see Figure 5). To ensure this condition, a token is removed from p_1 whenever a block is being processed, and it is added to p_1 again when the process is finished, as indicated by (55)-(57).

If (i) and (ii) hold, then t_1 in Figure 5 fires, resulting with the following output function configurations

$$\omega(p_1, c_0) = \left[0, \; -, \; -, \; -, \; \cdots\right] \quad (58)$$
$$\omega(p_2, c_0) = \left[1, \; -, \; -, \; -, \; \cdots\right] \quad (59)$$
$$\omega(p_3, c_0) = \left[1, \; -, \; -, \; -, \; \cdots\right] \quad (60)$$

Here, the electric motor will be turned on (i.e. $MT = 1$) when either p_2 or p_3 is marked by a generic c_0 token. Conversely, if p_1 is marked $MT = 0$.

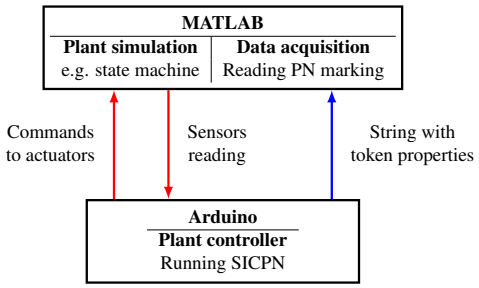


Figure 7: Close-loop interaction between the SICPN controller running on Arduino and the plant DT running in MATLAB.

Finally, with reference to Figure 6, the transition functions of t_8 are as follows:

$$\varphi(t_8, c_1) = \varphi(t_8, c_4) = \varphi(t_8, c_6) = \neg\text{IDX} \quad (61)$$

indicating that the table is rotating independently of the block type until IDX = 1. Nevertheless, it results

$$\boldsymbol{Pre}(p_7, t_8)(c_1) = 1 \otimes c_1 \quad (62)$$
$$\boldsymbol{Pre}(p_7, t_8)(c_4) = 1 \otimes c_4 \quad (63)$$
$$\boldsymbol{Pre}(p_7, t_8)(c_6) = 1 \otimes c_6 \quad (64)$$
$$\boldsymbol{Post}(p_8, t_8)(c_1) = 1 \otimes c_1 \quad (65)$$
$$\boldsymbol{Post}(p_8, t_8)(c_4) = 1 \otimes c_4 \quad (66)$$
$$\boldsymbol{Post}(p_8, t_8)(c_6) = 1 \otimes c_6 \quad (67)$$

which means that we are keeping track of the block material currently on PI-1 and the material approaching the PI-2 station in the controller. This will occur when p_9 will be marked, cf. with Table 2.

Finally, note that places p_8, p_{14}, p_{20}, p_{26}, and p_{32} are not responsible for defining the output signal MT, namely $\omega(p_8, \cdot)(1) = \{-\}$, but instead they represent movement of the workpiece from one station to the next. On the other hand, since all blocks will rotate simultaneously, MT is set *high* only by places p_2 and p_3.

5. Closed-loop test design and Results

In the absence of the real FESTO MPS testbed, the proposed SICPN controller underwent testing using a Digital Twin of the plant. We employed MATLAB to simulate the plant dynamics, while implementing the controller depicted in Figure 5 with an Arduino. The control loop was closed using two-way UART serial communications between the Arduino and MATLAB, as illustrated in Figure 7.

To facilitate real-time communication between Arduino and MATLAB, both the input and output of the controller and of the plant DT were encoded as `string`,

as this is the only data type supported by the serial protocol. Synchronisation between Arduino and MATLAB was achieved by adjusting the scan cycle in both codes. Specifically, we set the Arduino scan cycle to 1 second, enabling safe bidirectional serial communication.

5.1. Modelling the FESTO MPS Testbed Using SFC

To test our controller, it was necessary to have a model of the FESTO MPS system capable of emulating the input-output behaviour of the plant, thereby generating the correct input signals to the controller (denoted as I) based on the provided commands by the controller (denoted as O).

To tackle this task, we utilised the Sequential Function Chart (SFC) language to model the plant dynamics, which were subsequently translated into a timed MATLAB script. The plant's dynamics were represented by five SFCs running in parallel, one for each module of the FESTO MPS system:

• Figure 8 represents the input-output response of the rotating table. Here, the table is modelled as a circular buffer with 6 positions (cf. with Figure 1) and the table rotates clockwise until the SICPN generated command MT is *high*. More precisely, suppose the table is initially correctly aligned and at rest (Step {1} in Figure 8). Here, the optical sensor IDX is *high*, indicating that the table positions are correctly aligned with the processing stations. Then, once a workpiece arrives, and after the identification process described in subsection 4.3 is finished, a rotation request is called, thus p_2 is marked, and MT is set to *high*. The SFC then moves to Step {2}, indicating that the motor begins rotating. During the first second, IDX stays *high* due to some motor and sensor inertia. Then, in Step {3}, IDX becomes *low*, and 4 seconds later, in Step {4}, IDX returns to *high*, indicating that the table has finished the rotation and reached the correct alignment again.
Simultaneously, the workpieces are shifted by one position, i.e., "partpos(2:6) ← partpos(1:5)", so that they are ready for the new type of service, depending on their location and the current marking of the SICPN-based controller. For example, if place p_{33} in Figure 5 is marked, it means there is a block in position PI-6 which will be ejected by the ejection module.

• Figure 9 illustrates the input-output response of the Testing A station (resp. Testing B station). When a material is already identified, the sequence of operations unfolds as follows: upon receiving a metal workpiece (resp. any workpiece), the testing piston is triggered by the controller to advance, using the command TA_{adv} (resp. TB_{adv}), and the SFC progresses to Step {2} to initiate the test. The test lasts 3 seconds, leading to Step {3}. Depending on the test output, the SFC moves to either Step {4} if the workpiece is drilled or to Step {5} if it is not drilled. Subsequently, the testing piston receives the command TA_{ret} (resp. TB_{ret}) from the SICPN controller to retract, and the SFC proceeds to Step {6}, where it waits for 2 seconds before returning to {1}. If instead the material is not metallic, both the commands TA_{adv} and TA_{ret} will remain *low*.

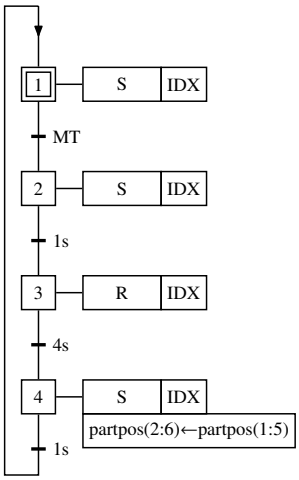


Figure 8: SFC modelisation of the plant's rotating table.

• Figure 10 illustrates the input-output response of the Processing A station (resp. Processing B station). Specifically: When a workpiece arrives at the PA (resp. PB) station, it encounters two possible outcomes based on the results from Testing A (resp. Testing B). If the workpiece is already drilled, it will wait until the next table rotation, thus both PA_{adv} and PA_{ret} remain low. However, if the workpiece is not drilled, it requires processing. In the latter case, the SFC is at Step {1}, and the drilling piston will be commanded through PA_{adv} (resp. PB_{adv}) by the SICPN controller to advance. Subsequently, the SFC progresses to Step {2} to initiate drilling, which lasts 3 seconds, culminating in Step {3}. Following drilling, the piston will receive the command PA_{ret} (resp. PB_{ret}) to retract, while the SFC advances to Step {4}. After a 2-second wait, the SFC returns to {1}.

5.2. Closed-loop tests

As the plant controller, Arduino manages the Digital Twin (DT) dynamics by executing the Token-Player associated with the SICPN depicted in Figure 5. For this purpose, the compiler discussed in subsection 2.4 has been translated into C.

Based on the input signals I (i.e., the plant measurements) received through MATLAB in each scan cycle,

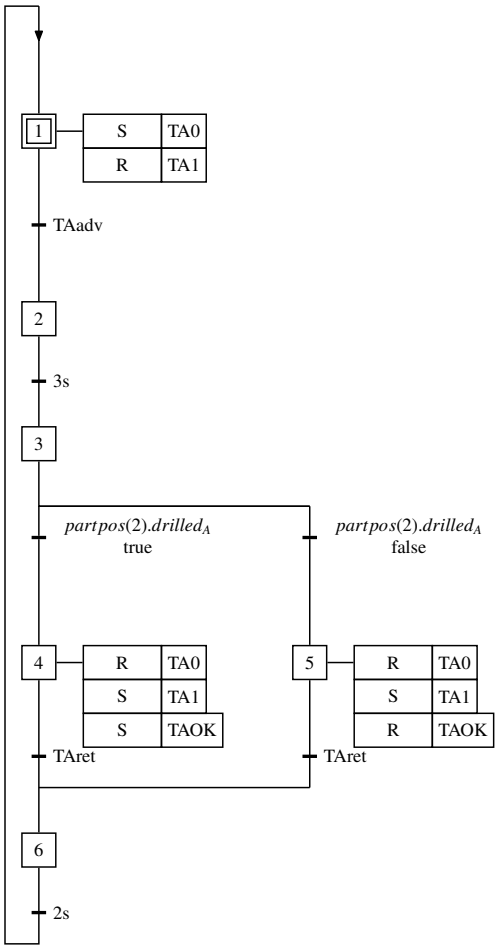


Figure 9: SFC modelisation of the Testing A station (resp. Testing B station).

the SICPN controller drives the plant by sending back to MATLAB its output signals O as commands. The main steps of the closed-loop implementation are listed in Algorithm 1.

Two types of closed-loop tests were conducted to verify the correctness of the proposed SICPN controller:

- Test 1: This test aims to track the operations of a not-drilled metal part among all stations, while verifying the correctness of the operations it undergoes.

- Test 2: This test aims to track the operations of four a-priory known blocks with exponentially distributed arrival times among all the service stations, while verifying the correctness of the operations they undergo. Clearly, Test 2 takes into account parallel simultaneous operations.

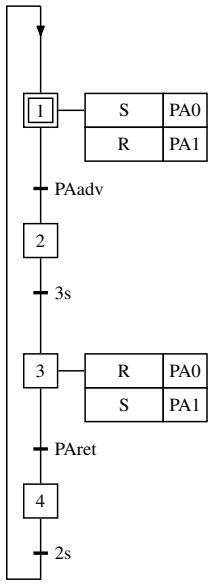


Figure 10: SFC modelisation of the Processing A station (resp. Processing B station).

Algorithm 1 Closed-loop tests' main steps.

```
System configuration                    ▷ Arduino's setup
    Define I and O signals
    Begin serial port connection
    Define the initial net's marking M_0
while true do                           ▷ Arduino's loop
    SICPN Token Player (net dynamics)
    Arduino write outputs O
    Communication (serial port)
        MATLAB reads O from the Arduino
        MATLAB calculates I
        MATLAB sends back I to Arduino
    Arduino reads inputs I
end while
```

Table 4 shows the properties of the workpieces and the order of entrance of the blocks at the rotating table in the second test. Remember that the materials of the blocks are initially unknown to the controller, as they have not been identified yet.

Table 4: Ordered list of workpieces and corresponding expected operations in the second test.

Arrivals	Initial status	Requests operations
Q1	M, $\neg d_A$, $\neg d_B$	1, 2, 3, 4, 5 (ID, TA, PA, TB, PB)
Q2	RP, d_A, $\neg d_B$	1, 4, 5 (ID, TB, PB)
Q3	BP, d_A, d_B	1, 4 (ID, TB)
Q4	M, d_A, d_B	1, 2, 4 (ID, TA, TB)

Table 5 further illustrates the list of work processes

each workpiece is expected to be subjected to at a time, along with the number of simultaneous operations the modular processing station is performing at a time.

Table 5: Dynamic processes for the controlled queue with four blocks. Note that the **bold numbers** represents in which station the given block will be processed at the table.

Expected processes	**Temporal process execution**								
Q1 (1, 2, 3, 4, 5)	**1**	**2**	**3**	**4**	**5**	6			
Q2 (1, 4, 5)		**1**	2	3	**4**	**5**	6		
Q3 (1, 4)			**1**	2	3	**4**	5	6	
Q4 (1, 2, 4)				**1**	**2**	3	**4**	5	6
# Processes	1	2	2	2	3	2	1	0	0

As an example, at the time **Q3** enters PI-1, simultaneously the FESTO MPS is performing two operations: **Q3** is under identification (ID), while **Q1** is being drilled at PI-3 station (PA). Meanwhile, **Q2** is not undergoing any operation because red plastic blocks (RP) do not require TA processing at the PI-2 station.

5.2.1. *Acquiring and plotting data*

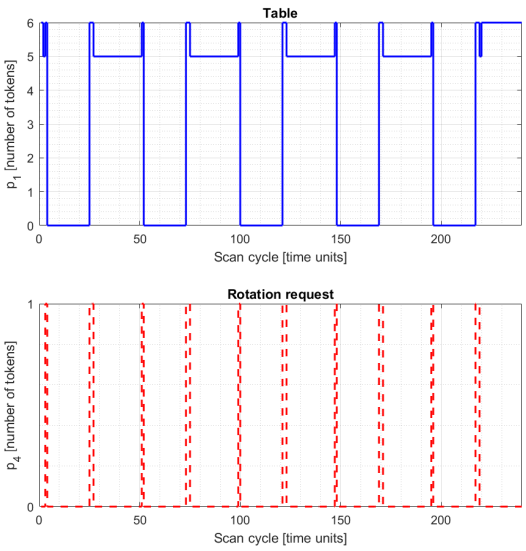
Among the many verification tests we conducted to ensure the correct design and implementation of the proposed SICPN controller, two plots are of particular interest for this analysis.

The first plot shows the marking of places p_1 and p_4 at each scan cycle. This plot relates to the condition of a processing operation being completed for a given block, represented as $\boldsymbol{m}_1 = 6 \otimes c_0$, and the condition of a *rotation request*, corresponding to $m_4 \geq 1 \otimes c_0$.

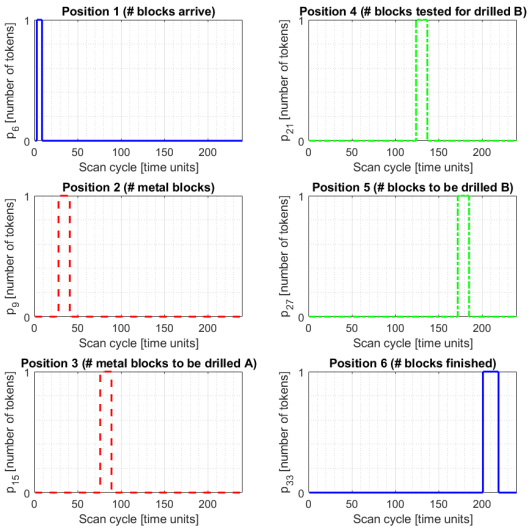
The second plot tracks the markings of places p_6, p_9, p_{15}, p_{21}, p_{27}, and p_{33}, indicating when each block enters a new process station.

5.2.2. *Test 1: One metal block on the table*

One can observe the results of Test 1 in Figure 11. More precisely, in Figure 11a, it shows that the rotation request is synchronised with the table rotation. Furthermore, when the metal part leaves the table, it stands still waiting for the next workpiece to arrive, i.e., $m_1(t) = 6 \otimes c_0$ for all $t > 220$ time units. Figure 11b shows that the metal block passes through all positions, as expected.



(a) Table and request rotation

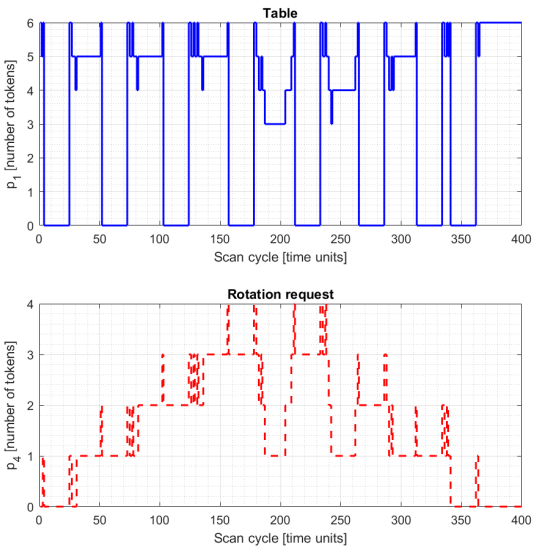


(b) Positions of the block

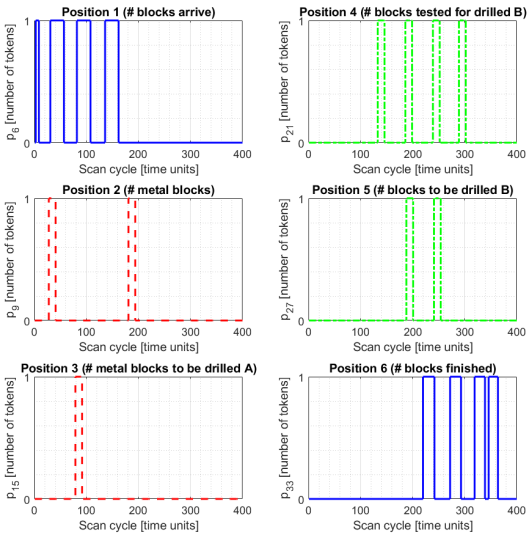
Figure 11: Results of Test 1: One metal block on the table.

5.2.3. *Test 2: Four different blocks on the table*

One can observe the results of Test 2 in Figure 12. More precisely, Figure 12a shows that place p_4 is marked at each time by a number of tokens equal to the number of blocks on the table that generate a *rotation request*. Moreover, m_4 is also always upper-bounded by the total number of blocks on the table, namely $\boldsymbol{m}_4 \leq 4 \otimes c_0$. Furthermore, Figure 12b shows which blocks, according to their properties (see Table 4), are processed in each position as time increases. By inspection of the expected results illustrated in Table 5, it is verified that the plant is behaving correctly under the proposed SICPN-based feedback controller.



(a) Table and request rotation



(b) Positions of the blocks

Figure 12: Results of Test 2: Four heterogeneous blocks on the table.

6. Conclusions

This study presents the formal definition of SICPN, accompanied by an algorithm tailored for implementing a SICPN controller using an IEC61131-3 compliant SCL language, thereby facilitating feedback-based decision-making prototyping in discrete event systems. The efficacy of this formalism is showcased in its application to control an extended configuration of a FESTO Modular Processing Station, where the SICPN controller is implemented on an Arduino board.

Building on these findings, several avenues for future research emerge. Exploring the versatility of SICPNs across diverse domains beyond manufacturing, such as transportation, healthcare, or smart cities, offers the potential for addressing intricate control challenges and driving innovation across industries. This endeavour could entail the development of standardised SICPN-based modules aimed at working towards standardisation and interoperability of SICPN-based control systems, facilitating seamless integration with existing industrial automation platforms and devices. By pursuing these lines of inquiry, we can advance the capabilities and applicability of SICPN-based control systems across various domains, thereby propelling the evolution of automation and decision-making processes.

Moreover, regarding more theoretical aspects, efforts to formalise well-known concepts in PNs such as reachability, liveness, and determinism for either SIPNs (which have not been studied yet), or SICPNs would also need to be undertaken. This task presents significant challenges since, in either SIPNs or SICPNs, transitions depend on exogenous inputs which can change simultaneously, rendering the standard definitions less applicable. It would be also of interest to explore the extension of these concepts from a closed-loop perspective. This involves modeling both the controller and the plant as coupled SICPNs, where the inputs of one are the outputs of the other and vice versa. In this sense, and by developing tailored composition rules between the two models, we could strive to formalise some closed-loop counterparts of these properties.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. Alessandro Pilloni thanks Fondazione di Sardegna for the grant supporting “IQSS-Information Quality aware and Secure Sensor networks for smart cities” (CUP: F75F21001400007). Carla Seatzu appreciates the funding received from MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN), call 2022, ERC Sector: PE6, for the project “Motown: Smart Production Planning and Control for Manufacturing of Electric Vehicle Powertrain in Industry 4.0 Environment” (CUP:20228XEHRE).

References

Abdelsattar, A., Park, E.J., Marzouk, A., 2022. An OPC UA client/gateway-based digital twin architecture of a SCADA system with embedded system connections, in: IEEE/ASME Int. Conf. Adv. Intell. Mechatron., IEEE. pp. 798–803. doi:10.1109/AIM52237.2022.9863367.

Awad, H., 2018. Supervisory control systems: Theory and industrial applications, in: Petri Nets in Science and Engineering. IntechOpen, pp. 93–109. doi:10.5772/intechopen.75166.

Azkarate, I., Ayani, M., Mugarza, J.C., Eciolaza, L., 2021. Petri net-based semi-compiled code generation for programmable logic controllers. Applied Sciences 11, 7161. doi:10.3390/app11157161.

Bashir, M., Zhou, J., Muhammad, B.B., 2021. Optimal supervisory control for flexible manufacturing systems model with petri nets: A place-transition control. IEEE Access 9, 58566–58578. doi:10.1109/ACCESS.2021.3072892.

Basile, F., Cordone, R., Piroddi, L., 2021. Supervisory control of timed discrete event systems with logical and timed specifications. IEEE Transactions on Automatic Control doi:10.1109/TAC.2021.3093618.

Basile, F., Ferrara, L., 2022. Residuals-based fault diagnosis of industrial automation systems using timed and untimed interpreted petri nets. Control Engineering Practice 129, 105361. doi:10.1016/j.conengprac.2022.105361.

Batchkova, I., Popov, G., Karamishev, H., Stambolov, G., 2013. Dynamic reconfigurability of control systems using iec 61499 standard. IFAC Proceedings Volumes 46, 256–261. doi:10.3182/20130606-3-XK-4037.00050.

Berger, S., Bogenreuther, M., Häckel, B., Niesel, O., 2019. Modelling availability risks of it threats in smart factory networks–a modular petri net approach. ECIS 2019 - 27th European Conference on Information Systems .

Borges, M.U., Lima II, E.J., 2018. Conversion methodologies from signal interpreted petri nets to ladder diagram and c language in arduino. International Journal of Mechanical Engineering Education Vol. 46, 302–314. doi:10.1177/0306419018759921.

Cao, L., Jiang, X., Zhao, Y., Wang, S., You, D., Xu, X., 2020. A survey of network attacks on cyber-physical systems. IEEE Access 8, 44219–44227. doi:10.1109/ACCESS.2020.2977423.

Chen, C., Hu, H., 2020. Extended place-invariant control in automated manufacturing systems using petri nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems doi:10.1109/TSMC.2020.3035668.

Comlan, M., Delfieu, D., 2019. Petri nets to arduino (PN2A) embedding time petri nets into a microcontroller architecture. Soft Computing and Electrical Engineering (SCEE) 1, 12–25.

Comlan, M., Delfieu, D., Sogbohossou, M., Vianou, A., 2017. Embedding time petri nets, in: 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE. pp. 0404–0409. doi:10.1109/CoDIT.2017.8102625.

Desirena-López, G., Ramírez-Treviño, A., Briz, J.L., Vázquez, C.R., Gómez-Gutiérrez, D., 2019. Thermal-aware real-time scheduling using timed continuous petri nets. ACM Transactions on Embedded Computing Systems (TECS) 18, 1–24. doi:10.1145/3322643.

Ebel, F., Pany, M., 2015. Festo processing station manual 648813 de/en .

Farah, K., Chabir, K., Abdelkrim, M.N., 2019. Colored petri nets for modeling of networked control systems, in: 2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), IEEE. pp. 226–230. doi:10.1109/STA.2019.8717215.

Fernández, I.A., Cortabarría, J.C.M., Echeverría, L.E., 2021. Petri net implementation in programmable logic controllers: methodology for development and validation, in: 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE. pp. 15–20. doi:10.1109/SAMI50585.2021.9378673.

Flamigni, F., Pileggi, P., Barrowclough, O., Haenisch, J., 2020. First report on standards relevant for digital twins. The Change2Twin Consortium .

Flochová, J., Lojan, T., 2019. Supervisors of petri nets. Vedecké Práce Materiálovotechnologickej Fakulty Slovenskej Technickej Univerzity v Bratislave so Sídlom v Trnave 27, 33–41. doi:10.2478/rput-2019-0023.

Frey, G., 2000. Automatic implementation of Petri net based control algorithms on PLC. Proceedings of the IEEE American Control Conference (ACC) 4, 2819–2823. doi:10.1109/ACC.2000.878725.

Frey, G., 2002. Design and formal Analysis of Petri Net based Logic Controllers. Ph.D. thesis. Dissertation, Shaker Verlag, Aachen, Germany.

Gaona, A.C., Chávez, J.M., Vázquez, C.R., 2021. Rcpetri: a matlab app for the synthesis of petri net regulation controllers for industrial automation, in: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE. pp. 01–07. doi:10.1109/ETFA45728.2021.9613441.

Gehlot, V., 2019. From petri nets to colored petri nets: A tutorial introduction to nets based formalism for modeling and simulation, in: 2019 Winter Simulation Conference (WSC), IEEE. pp. 1519–1533. doi:10.1109/WSC40007.2019.9004691.

Gomes, L., Barros, J.P., 2018. Refining IOPT petri nets class for embedded system controller modeling, in: IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, IEEE. pp. 4720–4725. doi:10.1109/IECON.2018.8592921.

Grobelna, I., Karatkevich, A., 2021. Challenges in application of petri nets in manufacturing systems. Electronics 10, 2305. doi:10.3390/electronics10182305.

Grobelna, I., Szcześniak, P., 2022. Interpreted petri nets applied to autonomous components within electric power systems. Applied Sciences 12, 4772. doi:10.3390/app12094772.

Grobelna, I., Wiśniewski, R., Grobelny, M., Wiśniewska, M., 2016. Design and verification of real-life processes with application of petri nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47, 2856–2869. doi:10.1109/TSMC.2016.2531673.

Hu, Y., Ma, Z., Li, Z., Giua, A., 2021. Diagnosability enforcement in labeled petri nets using supervisory control. Automatica 131, 109776. doi:10.1016/j.automatica.2021.109776.

Hüls, J., Pilch, C., Schinke, P., Niehaus, H., Delicaris, J., Remke, A., 2021. State-space construction of hybrid petri nets with multiple stochastic firings. ACM Transactions on Modeling and Computer Simulation (TOMACS) 31, 1–37. doi:10.1145/3449353.

IEC, 2019. Systems and Software Engineering - High-Level Petri Nets - Part 1: Concepts, Definitions and Graphical Notation. Standard. International Organization for Standardization and International Electrotechnical Commission. Geneva, CH.

Jensen, K., 1981. Coloured petri nets and the invariant-method. Theoretical computer science 14, 317–336. doi:10.1016/0304-3975(81)90049-9.

Jensen, K., 1987. Coloured petri nets, in: Petri nets: central models and their properties. Springer, pp. 248–299. doi:10.1007/978-3-540-47919-2_10.

Jensen, K., Kristensen, L.M., 2009. Coloured Petri nets: modelling and validation of concurrent systems. Springer Science & Business Media. doi:10.1007/b95112.

Juranić, J., Pavković, N., Naumann, T., Toepfer, F., 2019. Patterns of engineering design collaboration and reasoning activities modelled with coloured petri nets. Journal of engineering design doi:10.1080/09544828.2019.1630803.

Klein, S., Frey, G., Minas, M., 2003. PLC programming with signal

interpreted petri nets, in: International Conference on Application and Theory of Petri Nets, 2003, Springer. pp. 440–449. doi:10.1007/3-540-44919-1_27.

Lewis, R.W., 1998. Programming industrial control systems using IEC 1131-3. 50, Iet.

Li, H., Yang, D., Cao, H., Ge, W., Chen, E., Wen, X., Li, C., 2022. Data-driven hybrid petri-net based energy consumption behaviour modelling for digital twin of energy-efficient manufacturing system. Energy 239. doi:10.1016/j.energy.2021.122178.

Lima II, E.J., Dórea, C.E., 2004. Synthesis and plc implementation of supervisory control via place invariants for a manufacturing cell. IFAC Proceedings Volumes 37, 259–264. doi:10.1016/S1474-6670(17)36128-1.

Liu, F., Sun, W., Heiner, M., Gilbert, D., 2021. Hybrid modelling of biological systems using fuzzy continuous petri nets. Briefings in Bioinformatics 22, 438–450. doi:10.1093/bib/bbz114.

Liu, Z., Hu, L., Hu, W., Tan, J., 2022. Petri nets-based modeling solution for cyber–physical product control considering scheduling, deployment, and data-driven monitoring. IEEE Transactions on Systems, Man, and Cybernetics: Systems 53, 990–1002. doi:10.1109/TSMC.2022.3170489.

Long, F., Zeiler, P., Bertsche, B., 2015. Potentials of coloured petri nets for realistic availability modelling of production systems in industry 4.0, in: Proceedings of the ESREL 2015 Conference, pp. 4455–4463.

Machado, P., Silva, M.R., de Souza, L.E., de Souza, C.W., Netto, R.S., 2018. Modeling using colored petri net of communication networks based on iec 61850 in a microgrid context. Journal of control, automation and electrical systems 29, 703–717. doi:10.1007/s40313-018-0411-x.

Mathworks, 2024. Cell arrays. MATLAB help URL: www.mathworks.com/help/matlab/cell-arrays.html.

Minas, M., Frey, G., 2002. Visual PLC-programming using signal interpreted petri nets, in: Proc. of IEEE American Control Conf., pp. 5019–5024. doi:10.1109/ACC.2002.1025461.

Murata, T., 1989. Petri nets : Properties , analysis and applications. Proceedings of the IEEE Vol. 77, 541–580. doi:10.1109/5.24143.

Mykoniatis, K., Harris, G.A., 2021. A digital twin emulator of a modular production system using a data-driven hybrid modeling and simulation approach. Journal of Intelligent Manufacturing , 1–13.doi:10.1007/s10845-020-01724-5.

Nabi, H.Z., Aized, T., 2019. Modeling and analysis of carousel-based mixed-model flexible manufacturing system using colored petri net. Advances in Mechanical Engineering 11. doi:10.1177/1687814019889740.

Outafraout, K., Nait-Sidi-Moh, A., et al., 2020. A control approach based on colored hybrid petri nets and (max,+) algebra: Application to multimodal transportation systems. IEEE Transactions on Automation Science and Engineering 17, 1208–1220. doi:10.1109/TASE.2020.2973996.

Petri, C.A., 1962. Kommunikation mit Automaten. Ph.D. thesis. Technische Universität Darmstadt.

Ramírez-Treviño, A., Rivera-Rangel, I., López-Mellado, E., 2003. Observability of discrete event systems modeled by interpreted Petri nets. IEEE Transactions on Robotics and Automation Vol. 19, 557–565. doi:10.1109/TRA.2003.814503.

Sheng, J., Prescott, D., 2019. A coloured petri net framework for modelling aircraft fleet maintenance. Reliability Engineering & System Safety 189, 67–88. doi:10.1016/j.ress.2019.04.004.

Simon, E., Oyekan, J., Hutabarat, W., Tiwari, A., Turner, C., 2018. Adapting petri nets to discrete event simulation for the stochastic modelling of manufacturing systems. International Journal of Simulation Modelling 17, 5–17. doi:10.2507/IJSIMM17(1)403.

Singh, P., Singh, L.K., 2019. Design of safety critical and control systems of nuclear power plants using petri nets. Nuclear engineering and technology 51, 1289–1296. doi:10.1016/j.net.2019.02.014.

Vázquez, C.R., Ramírez-Treviño, A., Silva, M., 2014. Controllability of timed continuous petri nets with uncontrollable transitions. International Journal of control 87, 537–552. doi:10.1080/00207179.2013.846480.

Wang, Y., Li, Y., Yu, Z., Wu, N., Li, Z., 2021. Supervisory control of discrete-event systems under external attacks. Information Sciences 562, 398–413. doi:10.1016/j.ins.2021.03.033.

Wu, Z., Tian, L., Zhang, Y., Wang, Y., Du, Y., 2021. Network attack and defense modeling and system security analysis: A novel approach using stochastic evolutionary game petri net. Security and Communication Networks 2021. doi:10.1155/2021/4005877.

Xia, C., 2016. Property preservation of refinement for petri net based representation for embedded systems. Cluster Computing 19, 1373–1384. doi:10.1007/s10586-016-0597-2.

You, D., Wang, S., Zhou, M., Seatzu, C., 2021. Supervisory control of petri nets in the presence of replacement attacks. IEEE Transactions on Automatic Control 67, 1466–1473. doi:10.1109/TAC.2021.3063699.

Zhang, Y., Wang, W., Du, W., Qian, C., Yang, H., 2018. Coloured petri net-based active sensing system of real-time and multi-source manufacturing information for smart factory. The International Journal of Advanced Manufacturing Technology 94, 3427–3439. doi:10.1007/s00170-017-0800-5.