UNIVERSIDADE FEDERAL DE MINAS GERAIS
School of Engineering
Graduate Program in Mechanical Engineering

Matheus Ungaretti Borges

SIGNAL-INTERPRETED COLOURED PETRI NETS: A MODELLING TOOL FRAMEWORK FOR IMPLEMENTATION OF FEEDBACK-BASED CONTROL OF DISCRETE-EVENT SYSTEMS

Belo Horizonte
2024

Matheus Ungaretti Borges

SIGNAL-INTERPRETED COLOURED PETRI NETS: A MODELLING TOOL FRAMEWORK FOR IMPLEMENTATION OF FEEDBACK-BASED CONTROL OF DISCRETE-EVENT SYSTEMS

Thesis presented to the Graduate Program in Mechanical Engineering from Universidade Federal de Minas Gerais, as a partial requirement for obtaining the Doctor degree in Mechanical Engineering.

Supervisor: Prof. Dr. Eduardo José Lima II

Belo Horizonte
2024

B732s

Borges, Matheus Ungaretti.
Signal-Interpreted Coloured Petri Nets [recurso eletrônico] : a modelling tool framework for implementation of feedback-based control of discrete-event systems / Matheus Ungaretti Borges. – 2024.
1 recurso online (74 f. : il., color.) : pdf.

Orientador: Eduardo José Lima II.

Tese (doutorado) – Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 72-74.

Bibliografia: f. 66-71.

1. Engenharia mecânica – Teses. 2. Modelagem matemática – Teses. 3. Redes de petri – Teses. 4. Microcontroladores – Teses. I. Lima II, Eduardo José. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621(043)

Ficha catalográfica elaborada pelo bibliotecário Marcio A. A. Gomes CRB/6 2812
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

FOLHA DE APROVAÇÃO

"SIGNAL-INTERPRETED COLOURED PETRI NETS: A MODELLING TOOL FRAMEWORK FOR IMPLEMENTATION OF FEEDBACK-BASED CONTROL OF DISCRETE EVENT SYSTEMS"

MATHEUS UNGARETTI BORGES

Tese submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Minas Gerais, constituída pelos Professores: **Dr. Eduardo Jose Lima II (orientador - Departamento de Engenharia Mecânica/UFMG), Dr. Paulo Eigi Miyagi (Escola Politécnica - USP), Dr. Pedro Henrique Ferreira Machado (Instituto Federal de Minas Gerais), Dr. Carlos Andrey Maia (Departamento de Engenharia Elétrica/UFMG), Dr. Alessandro Pilloni (Department of Electrical and Electronic Engineering (DIEE), University of Cagliari – Italia) e Dra. Carla Seatzu (Department of Electrical and Electronic Engineering (DIEE), University of Cagliari – Italia),** como parte dos requisitos necessários à obtenção do título de **"Doutor em Engenharia Mecânica"**, na área de concentração de **"Projetos e Sistemas"**.

Tese aprovada no dia 02 de outubro de 2024.

Por:

Documento assinado eletronicamente por **Eduardo Jose Lima Ii**, **Professor do Magistério Superior**, em 10/10/2024, às 14:42, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Paulo Eigi Miyagi**, **Usuário Externo**, em 10/10/2024, às 18:06, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Carlos Andrey Maia**, **Professor do Magistério Superior**, em 25/11/2024, às 17:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Carla Seatzu**, **Usuária Externa**, em 27/11/2024, às 12:34, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

To my family, whom I can always count on.

Acknowledgements

This thesis is the result of a lot of dedication and study. In addition to my efforts, I, fortunately, could count on the support of many people. From the bottom of my heart, I want to express my most sincere gratitude to everyone who helped me in some way along this path.

Firstly, I thank my family, parents (João and Ruth) and siblings (Luciana and Daniel), as they are a source of affection, understanding, inspiration and support. Their examples throughout my life were fundamental in building who I am today. May we celebrate this achievement and the ones to come. Thank you for being with me in every moment of my life.

To my supervisor Dr. Eduardo José Lima II. His trust in my research, encouragement and all the opportunities provided enriched my academic experience. Thank you for everything!

To Professor Dr. Carla Seatzu and Professor Dr. Alessandro Pilloni. Thank you for hosting me in your laboratory for a year during my exchange in Italy and for supporting this research with meaningful contributions.

To all my doctoral colleagues, especially, Augusto Moura and Bryan Castro, who shared many stories and over a decade of friendship with me. The conversations during the break were fundamental, as rest is necessary for accomplishing the work.

To my friends, who always supported me and cheered for me. Even if it is not always possible to be with you all, you will always be in my heart. May each opportunity that life gives us to encounter be a different joy to share and build good memories.

To all my teachers, from childhood to graduate school, for all the knowledge and experience they imparted to me. They were all important in building who I am today and some of them undoubtedly gave me lifetime memories. I thank you from the bottom of my heart for dedicating your life to this beautiful profession of teaching people!

To the Postgraduate Program in Mechanical Engineering of the School of Engineering (PPGMEC) of the Universidade Federal de Minas Gerais (UFMG), for allowing the development of this work. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001.

To everyone who, directly or indirectly, helped in the development of the work.

My sincerest thanks.

"I'll carry on! Carry on!
This end is my start
We are such stuff as dreams are made on"
— Ashes (Angra)

"Here we go, carrying no longer sorrow
Standing up in the wind
Walk along, marching on for tomorrow
In this neverending way"
— For Tomorrow (Shaman)

RESUMO

Redes de Petri são normalmente utilizadas para design e verificação em vez de implementação de controle direto. No entanto, em linha com o foco do paradigma da Indústria 4.0 em sistemas de controle flexíveis e reconfiguráveis, é proposta uma ferramenta de modelagem para prototipagem rápida de algoritmos de controle de eventos discretos baseados em feedback em controladores programáveis, como CLPs ou placas de microcontroladores. Esta ferramenta de modelagem, chamada Redes de Petri Coloridas Interpretadas, visa combinar a expressividade de modelagem formal de das Redes de Petri Coloridas com as capacidades das Redes de Petri Interpretadas, que são especializadas no processamento de medições de plantas e na determinação de comandos de atuadores. Ao mesclar essas propriedades, a nova estrutura pode resolver problemas que não foram considerados antes. A contribuição da tese envolve: a) A definição formal da estrutura das Redes de Petri Coloridas Interpretadas; b) A modelagem da Estação de Processamento Modular FESTO MPS construindo um Gêmeo Digital em Matlab; c) A implementação da estrutura para controle por feedback de Sistemas de Eventos Discretos em um Arduino compatível, via linguagem C++, para dar suporte à tomada de decisão baseada em feedback dentro das Redes de Petri Coloridas Interpretadas, destacando o chamado *Token-Player*, estrutura projetada para representar o comportamento dinâmico do sistema modelado; d) A validação da eficácia do formalismo proposto no controle de um Gêmeo Digital de uma configuração estendida da FESTO MPS usando um microcontrolador Arduino via comunicações seriais UART bidirecionais. Os testes demonstram que, durante as transições, as condições de interpretação de cor e sinal permitem ao microcontrolador programar com precisão e reconfigurar dinamicamente as ações de controle sem explosão do modelo do controlador baseado em Redes de Petri, dada a complexidade do problema de controle.

Palavras-chave: Controle de sistema a eventos discretos; Redes de Petri; Controle do sistema de fabricação; Controladores Lógicos Programáveis; IEC61131-3; Microcontroladores.

ABSTRACT

Petri nets (PNs) are typically utilised for design and verification rather than direct control implementation. However, in line with the Industry 4.0 paradigm's focus on flexible and reconfigurable control systems, a modelling tool for rapidly prototyping feedback-based discrete-event control algorithms on programmable controllers such as PLCs or microcontroller boards is proposed. 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. By merging these properties, the new framework can solve problems that were not considered before. The thesis' contribution involves: a) The formal definition of the framework SICPN; b) The DT FESTO Modular Processing Station (MPS) testbed modelling in Matlab; c) The implementation of the framework for feedback-control of Discrete-Event Systems (DES) in an Arduino compliant, via C++ language, to support feedback-based decision-making within the SICPN, highlighting the so-called *Token-Player*, structure designed to represent the dynamic behaviour of the modelled system; d) The validation of the effectiveness of the proposed formalism in controlling a Digital Twin (DT) of an extended configuration of the FESTO MPS using an Arduino microcontroller via two-way UART serial communications. The tests demonstrate that, during transitions, the colour and signal interpretation conditions enable the microcontroller to accurately schedule and dynamically reconfigure control actions without explosion of the PN-based model of the controller given the control problem complexity.

Keywords: Discrete-event system control; Petri Nets; Manufacturing system control; Programmable Logic Controllers; IEC61131-3; Microcontrollers.

LIST OF FIGURES

LIST OF TABLES

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

Contents

1 INTRODUCTION

Discrete-event systems (DES) were based on the system theory, whose main objectives were described by Cassandras and Lafortune (2008).

1. Modelling and Analysis:

   This is the first step toward understanding how an existing system actually works. A model should be developed in order to see if it can reproduce the physical system (mathematically, in a laboratory, or through computer simulation). Once the accuracy of the model is verified, the system behaviour should be analysed under different conditions (e.g., different parameter values or input functions);

2. Design and Synthesis:

   Having accurate modelling techniques at our disposal, the next issue to be addressed is "How to build a system that behaves as we desire." Therefore, parameter values of the system components should be selected to result in a "satisfactory" design.

3. Control:

   This is the next logical step to the basic design process. Attempt to select the input functions to ensure the system behaves as expected under a variety of (possibly adverse) operating conditions. Once again, a model capable of testing and validating various control approaches is needed. In addition, to make the process of selecting the right control as efficient as possible some techniques are required;

4. Performance Evaluation:

   After a system has been designed and controlled to ensure proper behaviour, issues such as "How well is the system really performing?" are addressed. Measures of system performance may often be subjective or application-dependent. Several different controls may accomplish basic design objectives. When evaluating the performance of a system, it is usually needed to refine or limit control schemes to the select few that satisfy the performance objectives;

5. Optimisation:

   Considering that a system can be designed and controlled to achieve desirable performances, a natural question is "How can we control it to achieve the best possible performance?" This requires the development of additional analytical or experimental techniques for efficiently determining the optimal system behaviour and the means for attaining it.

For historical context, please refer to "On the history of Discrete Event Systems" (M. Silva, 2018), "Petri nets and Automatic Control: A historical perspective" (Giua & Silva, 2018) and "50 years after the PhD thesis of Carl Adam Petri: A perspective" (M. Silva, 2012).

From the second half of 20^{th} century onwards, several methods were developed to represent DES, such as automata (Ramadge & Wonham, 1989), Mark Flow Graph (MFG) (P. E. Miyagi et al., 1988; P. Miyagi et al., 1995; J. R. Silva & Miyagi, 1995), Grafcet (David, 1995), Sequential Function Chart (SFC) (IEC, 2003), Input-Output Place-Transition (IOPT) (Gomes & Barros, 2018; Gomes et al., 2007), Petri Nets (Murata, 1989).

Petri nets (PNs) have been extensively used across various domains as a declared for modelling discrete events since their introduction by Petri (1962).

The mathematical formalism of Petri allowed the replacement of the temporal relationship by the causal relationship, in addition to defining concepts such as concurrency, parallelism, cooperation, competition and resource allocation, inherent to the behaviour of discrete-event systems (DES) (Ramírez-Treviño et al., 2003; M. Silva, 2012).

Murata (1989) wrote a paper called "*Petri Nets: Properties, Analysis and Applications*", in which he explains the basic elements of a Petri net, including weighting and initial marking.

There are plenty of examples that can be represented by using PNs that go from a simple production schema and the change of seasons (Petri, 1980) to chemical equations, a state diagram of a vending machine and even dataflow computation of mathematical expressions (Murata, 1989). Some new Petri nets were derived from the original by aggregating new features into their definition. Thus, they can surpass these simple examples and represent more complex systems.

1.1 Motivation

There are some limitations when modelling a system using the original PN. As a natural course of science, these problems appeared and needed to be solved. Hence, researchers have studied for decades and continue to study Petri's thesis to develop new definitions, creating networks capable of aggregating even more information, allowing different systems to be described in the most realistic way possible, thus becoming a very versatile tool to build a framework. As a motivation, two of these limitations are discussed presenting the problem and a possible solution.

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 (Gratie & Petre, 2014; Jensen, 1992). 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), demonstrated superior in modelling Industry 4.0 compliant applications than alternatives declared. Also, the EU Commission's report on Digital Twins (DTs) (Flamigni et al., 2020) mentions HLPNs as a candidate declared 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 and, in particular, distributed systems, see ISO/IEC 15909-1 (IEC, 2019) for details in this regard.

Another limitation of PNs is the incapability of read and process external signals due to the lack of low-level signal interpretation. In this regard, Frey (2000) derived the Signal-Interpreted Petri Nets (SIPN) adding these characteristics allowing the model to recognise the signal inputs and dynamically respond to them determining the system behaviour. As a result, SIPN can be applied to manufacturing lines through PLC (Frey, 2000) or microcontroller (Borges & Lima II, 2018) implementation.

Both limitations were solved by developing new types of PN considering the characteristics of the respective problem. For the first limitation, CPNs were developed, and for the second, SIPNs.

1.2 Proposition

CPNs and SIPNs frameworks were studied separately and solve different questions. For example, 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 (Farah et al., 2019; Grobelna et al., 2016). In contrast, when compared to CPNs, SIPNs lack high-level expressivity features, resulting in high-dimensional models.

Therefore, a new PN formalism, named Signal-Interpreted Coloured Petri Net (SICPN), is proposed in this thesis. This framework merges colour and signal interpretation properties to solve new problems taking advantages of CPNs by providing them with low-level signal interpretation functionalities, which are useful for control purposes.

1.3 Objectives

In line with the perspective of section 1.2, the main objective of this thesis is to devise a SICPN-based control and to implement it on an Arduino microcontroller for validation.

To use a PN as a plant controller requires the modelling of the framework for feedback-control of DES in Arduino to support feedback-based decision-making within the SICPN, highlighting the so-called *Token-Player*, a structure designed within the controller managing input signals and validating transition firing conditions, i.e. verify if a transition is enabled or not given the presented condition of the system. Then, it is illustrated how to deploy a SICPN-based controller into an Arduino microcontroller via C++ 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, e.g. external sensor reading, and the verification of colour transitions, i.e. a transition can fire differently depending on the given conditions, operate at the same decision level (Basile et al., 2021; Farah et al., 2019; Frey, 2002;

Grobelna et al., 2016). As a case study, a Digital Twin of an extended configuration of the *FESTO MPS modular processing station* (MPS) is modelled in closed-loop.

Thus, the specific objectives are:

- Modelling and implementation of a Digital Twin (DT) on Matlab:

  In light of the absence of a real FESTO MPS system, details for the modelling and implementation of a Digital Twin (DT) of the FESTO MPS testbed is provided on Matlab. 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 4.1, as well as Batchkova et al. (2013), which explores an extended configuration closely resembling the one under consideration, but simpler.

- Implementation of the SICPN framework in Arduino:

  Translating a PN into C++ language 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) discusses conversion methodologies from SIPN to either Ladder diagram (LD) or C++. On the other hand, Gomes and Barros (2018) provides a tool to translate Input-Output Place Transition (IOPT), which is a class of PNs supported by the IOPT Tools. However, unlike these works, this thesis's 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 the approach showcases the implementation of the entire system, giving emphasis to the Token-Player.

- Run an SICPN-based controller in an Arduino:

  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 & Pany, 2015) is demonstrated. More precisely, as proof-of-concept, an inexpensive Arduino MEGA board is utilised to run an ad-hoc designed SICPN-based controller for controlling operations and the control reconfiguration.

  Experimental tests demonstrating the interaction between the controller running on the Arduino and the Digital Twin, which operated 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.

Remarking that the SICPN framework is applied specifically to FESTO MPS as a Digital Twin, however it could be applied in the real plant as well. The objective is to have feedback on the controller's response given the reading of input signals from external sensors. This case study is utilised to validate the formalism of the thesis' proposition. Knowing FESTO as an important company of system automation, using one of its simulation plants is relevant to show the applicability and the reproducibility of the case study and also to use in the education of new engineers. Therefore, the representativeness of this case study is proven. Once one get used to the language of this new formalism, it can be extended to other cases and the development time to model the new system is reduced.

1.4 Thesis structure

The chapters if this thesis are organised as follows:

- **Chapter 1**. The introduction gives context to the worked theme and presents the motivation, proposition, objectives and structure organisation of the thesis.
- **Chapter 2**. The literature review discusses themes such as Discrete-Event Systems (DES), Petri Nets - their derivations and applications - and the state-of-art of the main themes studied in this Ph.D thesis.
- **Chapter 3**. The SICPN model definition is introduced, explaining the dynamic behaviour of the net, with particular emphasis on its formal definition and an in-depth exploration of its sequential behaviour, including dynamic and I/O aspects. Thus, a simple example of its modelling is given. Furthermore, it is presented the output function implementation and the IEC61131-3 compliant SICPN compiler to implement the example in a PLC.
- **Chapter 4**. A discussion is done by presenting a case study and formulating the control problem (section 4.1). Afterwards, the case is modelled using the proposed formalism (section 4.2). Subsequently, a compiler designed to translate a SICPN into C++ to support the prototyping of feedback-based controllers within a Arduino microcontroller architecture is presented (section 4.3). Then, details regarding the implementation of the plant Digital Twin in MATLAB, and how the experimental tests are conducted, are explained (section 4.4).
- **Chapter 5**. The conclusion highlights the most relevant topics in this thesis and also presents the publication derived from this work and suggests ideas and directions to carry out future research.

2 LITERATURE REVIEW

2.1 Descrete Event Systems and Petri Nets

Discrete-event systems (DES) are dynamical systems whose states take value from a set that may either be finite or infinite. The state changes because of the occurrence of events that are typically asynchronous. Appropriate conditions should be satisfied in order to enable the occurrence of an event. An evolution of a DES is thus described by a sequence of events interleaved by the sequence of states that are visited (Sköldstam et al., 2007). For example, an event may represent the arrival or departure of a customer in a queue, the completion of a task or the failure of a machine in a production system, among other examples that are in different domains of knowledge such as production, robotics, logistics, computing, communication networks (Ramadge & Wonham, 1989).

As previously mentioned, one form of representation is through the finite state automaton, as exemplified by the authors Sköldstam et al. (2007), and which may be unfeasible to represent more complex cases because of the explosion of states (M. Silva, 2018). To deal with this problem, researchers began to model DESs using the Petri net formalism to represent, for example: manufacturing systems, communication networks, supervisory control systems (such as PLC or SCADA) and embedded systems.

There are other modelling approaches such as continuous and hybrid systems. For a brief discussion and examples, please refer to Vázquez et al. (2014), Desirena-López et al. (2019), F. Liu et al. (2021) for continuous systems and Dotoli et al. (2008), Outafraout, Nait-Sidi-Moh, et al. (2020), Hüls et al. (2021), Li et al. (2022), Z. Liu et al. (2022) for hybrid systems. However, the chosen modelling is DES because it can be modelled by using PN. 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 declared is generally used to solve supervisory control problems (Awad, 2018; Bashir et al., 2021; Basile et al., 2021; Chen & Hu, 2020; Flochová & Lojan, 2019; Hu et al., 2021; Lima II & Dórea, 2004; Singh & Singh, 2019; Wang et al., 2021; You et al., 2021), and to model complex and flexible systems such as manufacturing systems (Azkarate et al., 2021; Fernández et al., 2021; Gaona et al., 2021; Grobelna & Karatkevich, 2021; Nabi & Aized, 2019; Simon et al., 2018; Zhang et al., 2018), embedded systems (Berger et al., 2019; Comlan et al., 2017; Gomes & Barros, 2018; Xia, 2016) and communication networks (Cao et al., 2020; Farah et al., 2019; Juranić et al., 2019; Machado et al., 2018; Wu et al., 2021).

A discussion of several topics within supervisory control problems (subsection 2.1.1), manufacturing systems (subsection 2.1.2), embedded systems (subsection 2.1.3) and communication networks (subsection 2.1.4) will be made to show that Petri nets have several applications. These topics, directly or indirectly, helped to idealise the SICPN framework (Chapter 3).

2.1.1 Supervisory control problems

This subsection only discuss proposals based on the use of PN. From a practical point of view, supervisory control systems are, in general, designed based on experiences derived from previous similar applications and professionals specialised in the processes involved.

Lima II and Dórea (2004) developed a controller supervisor based on the concept of place invariants of PNs, which are capable to describe discrete-event systems with a set of operation constraints that must be respected by the controlled system. In their case study, they present a practical implementation of supervisory control in a manufacturing cell. The closed loop system (plant + supervisor), modelled by a PN, is implemented through a program written in STL in a Programmable Logic Controller (PLC), which drives the plant operation.

Awad (2018) presented a step-by-step on how to construct a supervisory control scheme in the field of DES modelling and control and also shows how the continuous activities, such as temperature control, pressure control, etc. are represented by few places resided in the embedded PN models.

Singh and Singh (2019) introduced the concept of performance-based risk in case of safety-critical systems. The performance analysis can be done by using deterministic or stochastic models. The authors illustrated the technique on a case study using PNs to perform the performance analysis to validate the design of safety-critical systems of a Nuclear Power Plant.

Flochová and Lojan (2019) described principles and methods of supervisory control of discrete-event systems initiated by Ramadge and Wonham and introduced three supervisory control methods based on the PN models highlighting the key features of the Petri tool software application for the supervisory control of discrete-event systems.

Chen and Hu (2020) proposed the extended P-invariant control principle in order to extend the application of P-invariants and provide a general methodology for the control of siphons, which can bring a concern regarding supervisors' structural complexity problem. The numbers of siphons and states are exponential with regard to the size of PNs. Thus, supervisors often suffer from huge size and complex structure. Therefore, supervisor simplification is indispensable before their implementation. This principle was applied to solve the supervisor simplification problem, which relies mainly on structural, reachability and algebra analysis.

Bashir et al. (2021) proposed a new method to construct a supervisory structure using combine control places and control transitions to ensure flexible manufacturing systems' smooth operation. This new method is used to design supervisor for FMSs using a Place-Transition Controller (PTC) and a Transition-Place Controller (TPC). A loop marking is computed for each concurrent processes in the FMSs to specify where the deadlock markings return in the live marking without affecting the proper operation of the system.

Basile et al. (2021) proposed a novel framework for the supervisory control of timed discrete-event systems based on Time Petri nets (TPNs). This method addresses logical specifications, including target markings to be reached in succession (reachability) and markings to avoid (safety); and also temporal specifications, including arrival and departure times in the target markings are required to be in prescribed time intervals (performance).

Hu et al. (2021), based on the active diagnosis problem in labelled Petri nets, developed a supervisor for a plant such that the closed-loop system is diagnosable. Considering that control actions may introduce deadlocks even if an original plant is deadlock-free, the classical notion of diagnosability in labeled Petri nets is generalised to the nets that may contain potential deadlocks. The authors also developed a structure called quiescent basis reachability graph to avoid enumerating all reachable markings of a plant and accordingly proposed a structure named Q-diagnoser to verify the diagnosability of a net.

Wang et al. (2021) researched the supervisory control problem of a DES modelled with labeled Petri nets under malicious attacks, which can be divided into two groups: (i) actuator attacks, that may cause a failure of an actuator for executing the commands issued from a supervisor that enforces a specification; and (ii) sensor attack, that may corrupt an observation (i.e., a sequence of observable transition labels) from a sensor by different types of attacks such as insertion, removal, and replacement of transition labels. The authors proposed a novel structure, namely a product observation reachability graph constructed from a plant and its specification, to decide the existence of such a supervisor by checking whether each state in the graph satisfies a particular condition.

You et al. (2021) used Petri nets as the reference declared to model the plant and assume a control specification in terms of a generalised mutual exclusion constraint. The authors also presented three different methods to derive a control policy that is robust to the possible replacement attacks. The first method provides an optimal (namely, maximally permissive) control policy based on the enumeration of markings consistent with the current observation. The second method derives an optimal policy based on constructing a monitor-controlled PN system, which still requires marking enumeration. The third method computes a control policy with timely response even for large-size PN systems at the expense of optimality.

2.1.2 Manufacturing systems

Simon et al. (2018) investigated the suitability of a DES environment for the modelling of Petri nets in the context of manufacturing systems. Their study tackled the issue of simulation for the modelling of manufacturing systems and the advantages of using Petri net models over classic DES element-based models. Petri nets were chosen because it has a solid mathematical ground with a simple language that enables the development of transparent models, which allow increased flexibility and control for designers.

Zhang et al. (2018) declared that the wide use of Internet of Things (IoT) technologies in manufacturing shop-floor creates an opportunity to turn the traditional manufacturing factories into smart ones. Zhang et al. (2018) tackled problems such as the large amount of raw data to process and the quantity of connected devices. Based on these concerns, the authors proposed a systematic graphics-based modelling approach for manufacturing information sensing, namely CPN-based active sensing system of real-time and multi-source manufacturing information (CPN-MIASS), which can assist the general operators in monitoring and controlling the real-time manufacturing process easily and dynamically.

Nabi and Aized (2019) proposed a hierarchical CPN model which is developed to analyse the performance of flexible manufacturing systems. The system performance has been investigated in relation to material supply and handling system, process execution, and production resources reliability variables. Different input factors are considered for simulation modelling such as mean machining time, mean loading/unloading time, mean assembly time, buffer capacity, material supply inter-arrival time, number of operations between failures, and mean time to repair for production resources; a variation in input factors has shown a significant impact on system performance measures. The CPN–based modelling, simulation, and analysis approach has been demonstrated as an efficient method for carousel-based mixed-model configured flexible manufacturing system.

Gaona et al. (2021) explored a different control framework, named Regulation control, in which the system to be controlled, named Plant, is represented by an Interpreted Petri net (IPN), which is an extension to PN in which labels are associated to places (representing sensors) and transitions (representing actuators). In this framework, the goal is to control the Plant in such a way that its output becomes equal to the output of another IPN, named Specification, representing the required behaviour of the system. The authors introduced a Matlab app named RCPetri, which allows to automatically compute regulation controllers. In particular, the app allows to build, compose and simulate IPN models in an efficient and graphical way. Moreover, the app allows to automatically generate the IPN Specification from an Excel table describing the required tasks in a high-level fashion. The synthesised IPN control program can be automatically translated to PLC code, for its final implementation.

Fernández et al. (2021) raised the possibility of reducing that probability of error when programming discrete-event dynamic systems by implementing a PN managing algorithm. A framework is presented which combines the use of this algorithm, by means of pre-incidence and post-incidence matrices and initial marking vector of a net, with code validation through emulation. A use case is brought forward in which the control program of a sequential process with parallel operations is implemented, with both virtual (VC) and real commissioning. Focusing on PNs and using their matrix representation and evolution rules, allowing tokens moving among net places as transitions firing consequence, opens a new perspective to the programming of this type of processes. The authors proposed a methodology for the development and commissioning of sequential systems controlled by PLC, based on a semi-compiled approach, which manages the evolution of any PN, by means of their pre-incidence and post-incidence matrices and initial marking vector.

Azkarate et al. (2021) declared that industrial discrete-event dynamic systems are commonly modelled by means of PNs, which have the capability to model behaviours such as concurrency, synchronization and resource sharing. The authors stated that there is not an effective systematic way to implement a PN in a PLC, and so the implementation of such a controller outside a PLC in some external software that will communicate with the PLC is very common. There have been some attempts to implement PNs within a PLC, but they are dependent on how the logic of places and transitions is programmed for each application. They proposed a novel application-independent and platform-independent PN implementation methodology, which is a systematic way to implement a PN controller within industrial PLCs. A great portion of the code will be validated automatically prior to PLC implementation. Net structure and marking evolution will be checked on the basis of PN model structural analysis, and only net interpretation will be manually coded and error-prone. Thus, this methodology represents a systematic and semi-compiled PN implementation method. A use case supported by a digital twin (DT) is shown where the automated solution required by a manufacturing system is carried out and executed in two different devices for portability testing, and the scan cycle periods are compared for both approaches.

Grobelna and Karatkevich (2021) affirmed that PNs are a useful mathematical declared for specification of manufacturing systems, supported by various analysis and verification methods. The progress made in automating control systems and the widespread use of Industry 4.0 pose a number of challenges to their application, starting from the education at university level and ending with modelling of real case studies. The authors presented and analysed the most relevant challenges and opportunities related to the use of Petri nets as a modelling technique of manufacturing systems.

2.1.3 Embedded systems

Xia (2016) stated that Petri net based Representation for Embedded Systems (PRES+) is a promising methodology for modelling, verification, analysis and control of embedded systems. But the state space explosion problem is somewhat tedious for PRES+ to specify and analyse large complex embedded systems. To solve state space explosion problem of PRES+, the authors proposed a method for expanding PRES+ model to the desired level of detail using a refinement approach. The authors presented definitions of two PRES+ models have the same dynamic properties, such as reachability, timing and functionality. Definitions of liveness and boundedness of PRES+ are also presented. A temperature measure and control example illustrates the efficiency of our refinement approach on practical applications.

Comlan et al. (2017) presented a tool, named Petri Net to Arduino (PN2A), which embeds Time Petri Nets (TPN) to Arduino micro-controller architecture. PN2A imports TPN and generates Arduino sketches, which can be then compiled and uploaded to a micro-controller architecture. Some transitions (resp. places) of the transition set (resp. place set) can be assigned to pins of the micro-controller. Embedded, the TPN becomes partially non-autonomous and can be defined as a microcontroller Synchronised Time Petri net (mSTPN). The structure of the TPN is translated in an incident matrix and enabling and firing functions are defined. In addition, one originality of the approach is the possibility of a “partial association”: a subset of places and transitions can be associated to the pins of a microcontrollers. The environment is external and physical: actuators, sensors, motors, LEDs, human interactivity. The modelling can be simulated in a real context.

Borges and Lima II (2018) equipped a laboratory bench with a two-cylinder pneumatic system whose function is to separate metallic blocks from other materials by using two inductive sensors (one for each cylinder). Other components of the bench are four end-course sensors and a start button. When the blocks are put in front of the cylinders, inductive sensors will respond a Boolean signal (TRUE or FALSE) to the system, informing if both blocks are metallic or not. If the first condition is true and the start button is pressed, both cylinders will advance simultaneously to collect the blocks together and return one at a time, whereas, in any other situation, they will advance separately and return simultaneously. At the end of this process, it will only restart when new blocks are put in their initial positions for new analysis and the firing conditions are satisfied again.

Gomes and Barros (2018) presented the Input-Output Place Transition (IOPT), which is a class of Petri nets supported by the IOPT Tools, with some additions. The corresponding formal syntax and semantics rules are also presented, followed by an illustrative example.

Berger et al. (2019) stated that concepts like the Internet of Things (IoT) or Cyber-physical Systems (CPS) accelerate the development from traditional production facilities towards smart factories. Due to their reliance on information flows and the high degree of cross-linking, these networks are, in particular, vulnerable to availability risks caused by attacks and errors. To address this problem, the authors aim to identify and analyse availability threats by developing a modelling approach that depicts specific characteristics of smart factory networks and, then, to propose a modular Petri net approach. To demonstrate the usefulness and applicability of this model, one real-world use case and two planned extensions of a mechanical engineering company were simulated. The model depicts information-based dependencies within smart factory networks and allows for the simulation and analysis of threat propagation. Thereby, it enables both researchers and practitioners to identify critical network connections and components, serving as a basis for layout decisions and IT security mitigation measures.

2.1.4 Communication networks

Machado et al. (2018) stated that the increasing penetration of distributed energy resources (DERs) and local/small-scale power systems, named microgrids (MG), becomes a growing challenge for distribution system operators (DSOs). In general, the DER units and MG operation are based on a decentralized and time-critical decisions, which performs a complex behaviour. To deal with this complex distributed system, the information and communication schemes are essential. Considering this, the IEC 61580 standard shows up as an appropriate solution, as it fulfills all related specifications of distributed systems. In this context, the authors' goal is to present and demonstrate a formal approach to deal with uncertainties and event-driven behaviours of an IEC 61850-based automated microgrid, showing their impacts in power system protection/control scheme. In addition, this work attempts to aid network designers suggesting a systematic approach to network modelling, especially if specific components need to be developed. To achieve this objective, it is suggested a methodology using the CPN declared as a tool to model an IEC 61850-based microgrid.

Juranić et al. (2019) affirmed that the increasing complexity of engineering activities emphasises the need to enhance software support for design team collaboration and improve the management of design process dynamics, especially in critical situations. Thus, the authors proposed a new approach in which a set of CPNs enable a practical implementation of design activities ontology by modelling rules and relationships simultaneously with the instantiation of taxonomy elements. The development of the proposed model is based on an in-depth analysis of several long-lasting development projects in a large industrial company and three case studies on designers' collaboration. To facilitate the implementation of ontology models into industrial practice, this work focuses on the issues of interoperability between design support systems and ontology models. The proposed method has significant potential to provide real-time updating and propagation of design information in teamwork.

Farah et al. (2019) declared that the Networked control system (NCS) is a set of entities communicating and exchanging data via a network, which makes its analysis and synthesis complex. Coloured Petri nets (CPN) provide a very compact way and a well-adapted and progressive framework for modelling and analysis complex systems in which competition, synchronisation, resource sharing and parallelism are present. The authors used a CPN for designing of NCS. For that goal, a software named CPNtools simulator is used to present a graphical model with hierarchical coloured Petri nets (HCPN) for the Ethernet network in the first step, starting with the global model, then going through its internal modules. Then, the authors proposed a model for different entities that constitute the control system.

Cao et al. (2020) defined that a cyber-physical system (CPS) typically consists of the plant, sensors, actuators, the controller and a communication network. The communication network connects the individual components to achieve the computing and communication in the CPS. It also makes the CPS vulnerable to network attacks. How to deal with the network attacks in CPSs has become a research hotspot. The authors classified network attacks in CPSs and review the work on intrusion detection and defence strategies.

Wu et al. (2021) proposed a Petri net of network attack and defence stochastic evolutionary game based on the effectiveness constraints on both sides with network attack and defence with the help of stochastic Petri net and evolutionary game theory. Using this model, a quantitative analysis of network attack events is carried out to solve a series of indicators related to system security, namely, attack success rate, average attack time, and average system repair time. Finally, the proposed model and analysis method are applied to a classic network attack and defence process for experimental analysis, and the results verify the rationality and accuracy of the model and analysis method.

2.2 Coloured Petri Nets

To enhance the expressiveness of PNs, various types of HLPNs have been developed, including continuous (Desirena-López et al., 2019; F. Liu et al., 2021; Vázquez et al., 2014), hybrid (Dotoli et al., 2008; Hüls et al., 2021; Li et al., 2022; Z. Liu et al., 2022; Outafraout, Nait-Sidi-Moh, et al., 2020), and coloured PNs (Gehlot, 2019; Jensen, 1981, 1987; Long et al., 2015; Sheng & 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, since information is attached to each token. The information can be inspected and modified when a transition fires. This makes CPNs particularly well-suited for modelling multi-product manufacturing systems and priority queue data structures and yields to a much smaller net.

Notable recent applications of CPNs include their use to model the process of sending different printing jobs to different printers (Gehlot, 2019), applications to model all the factors and activities related to aircraft fleet maintenance (Sheng & Prescott, 2019) and applications to model railway networks systems (Fanti et al., 2006) and also traffic lights, readers and writers, data base and telephone systems (Jensen, 1987). It is worth mentioning Farah et al. (2019), where the CPN framework is used for modelling and analysis networked systems in which competition, synchronisation, resource sharing and parallelism are present, as for the Ethernet communication system. For additional examples of applications, refer to Jensen and Kristensen, Chapter 14 (2009).

2.3 Signal-Interpreted Petri Nets

Considering that the input-output response of a dynamical system is often referred to as *low-level behaviour*, some authors (Frey, 2000, 2002; Minas & Frey, 2002; Ramírez-Treviño et al., 2003) use the term Low-level PN to characterise the use of PN for fine-grained modelling of interacting systems by acquiring and interpreting *low-level signals* from a system (OR external *low-level signals*). Consequently, to employ them for feedback control purposes, definitions for Signal-Interpreted Petri Nets (SIPN) were introduced in the early 2000s (Frey, 2000, 2002; Ramírez-Treviño et al., 2003). 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 & Lima II, 2018; Comlan & Delfieu, 2019).

Real-life and industrial processes modelled with application of Petri nets can be easily implemented in various hardware platforms with the guarantee that behavioural user-defined requirements are satisfied. The system can be decomposed into smaller parts, called modules, and implemented over the distributed devices working in different time-domains with the proper synchronization of the whole system (Grobelna et al., 2016).

As an illustration of SIPNs application, 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; Among others, notable recent applications also include Grobelna et al. (2016), where a SIPN is used to automate smart home systems, catering to the growing demand in this field; Grobelna and Szcześniak (2022), focusing on SIPN application in battery energy storage system management control; and , Basile et al. (2021), which proposes a novel framework based on Timed Petri nets (TPNs) for supervisory control purposes; and Basile and Ferrara, 2022, which utilises TPNs to develop a fault detection algorithm for discrete-event systems within a PLC.

2.3.1 Sequential Function Chart as Signal-Interpreted Petri Nets

The standard IEC 848 defined Sequential Function Chart (SFC) as a graphical programming language based on Grafcet (M. Silva, 2012). Later, the standard IEC 61131-3 specifies syntax and semantics of programming languages for programmable controllers (IEC, 2003).

SFC elements provide a means of partitioning and organizing controller programs with a set of steps and transitions connected by directed connector lines. A set of actions is associated with each place and a set of conditions is associated with each transition. An action can be represented in several ways: as a Boolean variable, as a collection of statements in the IL language, as a collection of statements in the ST language, as a collection of lines in the LD language, as a collection of lines/blocks in the FBD language, or even as a sequential function diagram in the SFC language (IEC, 2003). Refer to IEC (2003) standard for more information on rules for steps, transitions and evolution of a SFC.

According to de Mello et al. (2012), there is an intrinsic relationship between the elements of SIPN and the SFC language, as shown in Table 2.1.

Table 2.1 – Mapping between Signal-Interpreted Petri Net and Sequential Function Chart. Adapted from (de Mello et al., 2012).

Signal-Interpreted Petri Net	**Sequential Function Chart**
Place	Step
Transition	Transition
Arc	Connection
Input variables	Receptivity of each transition
Output variables	Action of each step

The SFC language has similarities with the SIPN. However, David (1995) highlights two differences: (i) marking a step in Grafcet is Boolean, that is, each step is associated with the value 0 (*False*) or 1 (*True*), while marking a place in SIPN is numeric and can be represented by a non-negative integer of tokens (resources). In the definition of Frey (2000), which is a safe network, the maximum value that the weight function can assume is unity; (ii) in Grafcet, all simultaneously enabled transitions fire simultaneously, while in SIPN, if there are conflicting enabled transitions, only one of them will be fired and its behaviour will be non-deterministic unless there is a condition that selects the transition to be fired. Despite these differences, if a SIPN is safe (its marks behave like logical variables) and deterministic (if there are no conflicting transitions that present true firing conditions simultaneously), it can be represented in the SFC language.

In addition to these differences mentioned above, Frey (2000) highlights a difference in the dynamic behaviour of states. The flowchart on Figure 2.1 shows that the states of a SIPN can be unstable (transient), while the flowchart on Figure 2.2 shows that in the SFC language these states are kept active for at least one PLC cycle, also known as the scan cycle. However, given the short duration of the PLC cycle, these states can be viewed as almost transient.

Figure 2.1 – SIPN flowchart. (Frey, 2002)

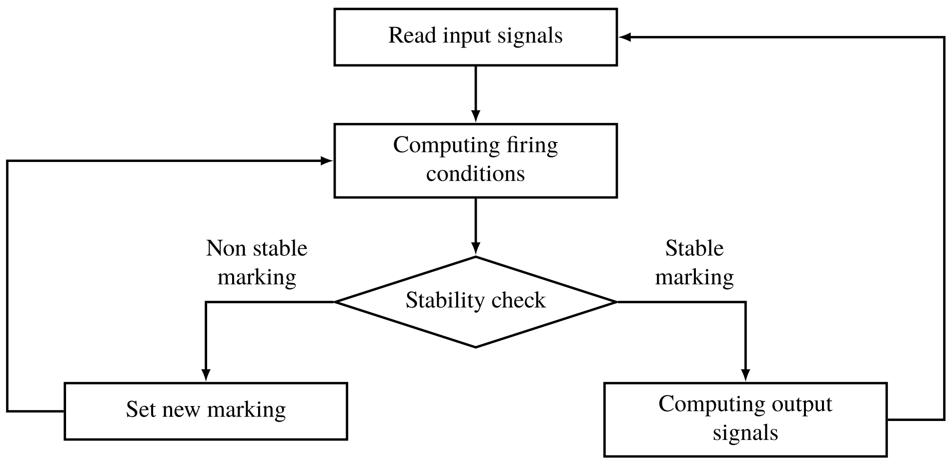
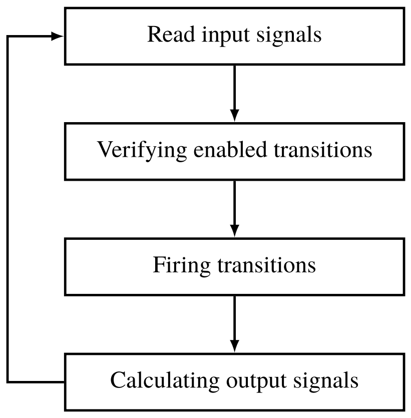


Figure 2.2 – SFC scan cycle flowchart.



Therefore, this subsection is important to contextualise because the modelling of the operational plant was done in SFC and implemented in Matlab, even if the control is applied in Arduino later.

3 SIGNAL-INTERPRETED COLOURED PETRI NETS

3.1 Preliminary notions on multisets

Some useful notation is introduced to formally define the SICPN model.

The sets of integers, naturals, and non-negative integers are denoted as $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$. 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.

For example, consider $\{a, b, d, b, a, b\}$ a finite multi-set over the set $\{a, b, c, d\}$, and it is represented by the formal sum $2 \otimes a +\ 3 \otimes b\ +\ 1 \otimes d$ (Jensen, 1987).

For more examples on multisets, I refer to Fanti et al. (2006) in the thesis' Annex A.

3.2 SICPN Definition

A SICPN N is defined by the following 11-tuple

$$N = (P,\ T,\ Co,\ \boldsymbol{Pre},\ \boldsymbol{Post}, \boldsymbol{M}_0,\ I,\ O,\ \varphi,\ \omega,\ \Omega), \tag{3.1}$$

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. Figure 3.1 highlights the elements of the structure of the net.

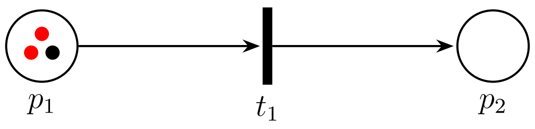


Figure 3.1 – PN structure elements

$Co : P \cup T \to C_\ell$ 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 $\mathcal{C}_\ell$. Therefore, $\forall\ p_i \in P$, $Co(p_i) = \{a_{i,1}, a_{i,2}, \ldots, a_{i,u_i}\} \subseteq \mathcal{C}_\ell$ 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, $\forall\ t_j \in T$, $Co(t_j) = \{b_{j,1}, b_{j,2}, \ldots, b_{j,v_j}\} \subseteq \mathcal{C}_\ell$ 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, $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ represent the *pre-* and *post-incidence* matrices with dimensions $n \times m$. Here, each element of either $\boldsymbol{Pre}(p_i, t_j) : Co(t_j) \to \mathcal{N}(Co(p_i))$, or $\boldsymbol{Post}(p_i, t_j)$:

$Co(t_j) \to \mathcal{N}(Co(p_i))$, 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, $\boldsymbol{Pre}(p_i, t_j)$ (resp. $\boldsymbol{Post}(p_i, t_j)$) 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. Note SICPN definitions are more complex than in uncoloured PNs because the weights of the arcs going from a place to a transition and from a transition to a place, depend on the colour with respect to which the transition is fired.

To maintain a concise notation, 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 $k = 1, 2, \ldots, v_j$.

Based on the $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ definitions it is further defined the $n \times m$ *incidence matrix* as $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$, where each element $\boldsymbol{C}(p_i, t_j) : Co(t_j) \to \mathcal{Z}(Co(p_i))$.

Let $m_i : Co(p_i) \to \mathbb{N}_0$ be the mapping associating to 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

$$\boldsymbol{m}_i = \sum_{a_{i,h} \in Co(p_i)} m_i(a_{i,h}) \otimes a_{i,h}, \tag{3.2}$$

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

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{m}_1, & \boldsymbol{m}_2, & \cdots, & \boldsymbol{m}_m \end{bmatrix}, \tag{3.3}$$

whereas with $\boldsymbol{M}_0$ is denoted the *net's initial marking*. The notation $\langle N, \boldsymbol{M}_0 \rangle$ indicates that N in (3.1) is initialised $\boldsymbol{M}_0$.

Figure 3.2 highlights the colour elements of the net, including the initial marking $\boldsymbol{M}_0$.

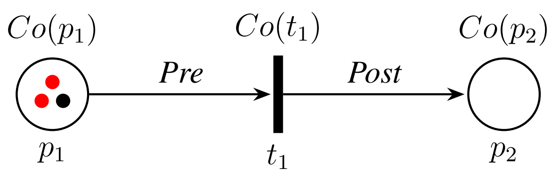


Figure 3.2 – PN coloured elements

By drawing 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, it is introduced the next additional fields in (3.1).

The set $I = \{i_1, i_2, \dots, i_d\}$ denotes the *set of logical signals* received as *input* to the net from the environment. The set $O = \{o_1, o_2, \dots, o_z\}$ indicates the *set of logical outputs* (or *action* to be done or not) computed by the net after processing the current marking $\boldsymbol{M}$ and inputs I, with $I \cap O = \emptyset$, and $d = |I|$, $z = |O|$. Notice that, I can be understood as a set of *measurements* from a plant to be controlled, while O as the set of *commands* issued to the plant's actuators.

Then, with $\varphi(t_j, b_{j,k}) : t_j \times Co(t_j) \rightarrow f_{j,k}(i_1, i_2 \dots, i_d)$ 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 $f_{j,k} : \{i_1, i_2, \dots, i_d\} \rightarrow \{0, 1\}$. Here, $f_{j,k} = \{1\}$ means such condition holds *true*, thus the I configuration is such that t_j is enabled to fire w.r.t. $b_{j,k}$. On the contrary, $f_{j,k} = \{0\}$ means t_j is not enabled to fire w.r.t. $b_{j,k}$.

Figure 3.3 highlights the signal interpretation elements represented on the net. It is noteworthy that only $\varphi(t_1, b_{1,k})$ appears explicitly on the net representation. Input I is an argument of φ function, whereas output O and functions ω and Ω are the system's response.

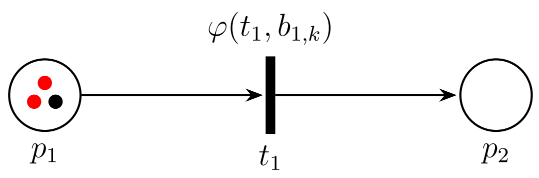


Figure 3.3 – PN signal interpretation elements

Then, with $\omega(p_i, a_{i,h}) : P \times Co(p_i) \rightarrow \{0, 1, -\}^z$ is denoted a *candidate output configuration* for each $p_i \in P$ marked w.r.t. 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 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*'.

The *output function* is defined as follows:

$$\Omega : \boldsymbol{M} \rightarrow \left\{ \; 0, 1, -, q, r_0, r_1, q_0, q_1, q_{01} \; \right\}^{|O|} \tag{3.4}$$

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, the following set is preliminarily introduced as follows:

$$\Gamma = \left\{ \; (p_i, a_{i,h}) \in P \times C_\ell \; : \; \boldsymbol{m}_i \neq \boldsymbol{\varepsilon}, \; m_i(a_{i,h}) \neq 0 \; \right\} \tag{3.5}$$

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 $\omega(\cdot,\cdot)(r) = -$.

It is remarkable to note that a SICPN behaves as a standard Petri net with four main differences:

1. Places may be marked by tokens with different colours;
2. Transitions may fire w.r.t. different colours;
3. The enabling of a transition w.r.t. a certain colour $b_{j,k} \in Co(t_j)$ does not depend only on the content of the input places, but also on some input conditions, encoded by the transition functions $\varphi(t_j, b_{j,k})$;
4. When a transition fires w.r.t. a certain colour, not only the PN marking is updated, but also some output signals are produced, as clarified by functions ω and Ω.

3.3 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 it is reached 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

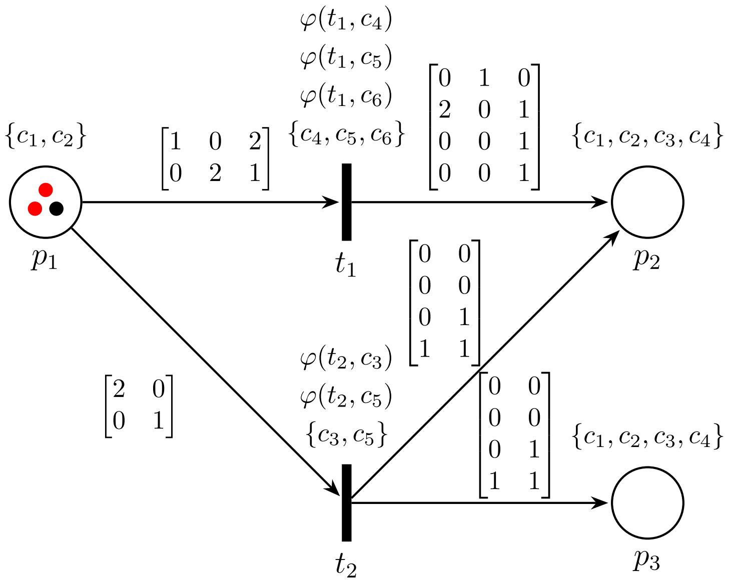
$$\sigma = t_{j_1}(k_{j_1})\, t_{j_2}(k_{j_2})\, \cdots\, t_{j_r}(k_{j_\tau}), \tag{3.6}$$

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

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

Unlike standard PNs and CPNs, in a SICPN, as well as in its primitive form, the SIPN, 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.

3.3.1 Example of SICPN dynamics



$\varphi(t_1, c_4)$
$\varphi(t_1, c_5)$
$\varphi(t_1, c_6)$
$\{c_4, c_5, c_6\}$
$\{c_1, c_2\}$
$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \end{bmatrix}$
$\begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
$\{c_1, c_2, c_3, c_4\}$
p_1
t_1
p_2
$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$
$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$
$\varphi(t_2, c_3)$
$\varphi(t_2, c_5)$
$\{c_3, c_5\}$
$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$
$\{c_1, c_2, c_3, c_4\}$
t_2
p_3

Figure 3.4 – A SICPN $\langle N, \boldsymbol{M}_0\rangle$ with $\boldsymbol{M}_0 = [2 \otimes c_1 + 1 \otimes c_2,\ \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon}]$.

Consider the SICPN in Figure 3.4 with $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2\}$ and $\mathcal{C}_\ell = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. 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

$$i_1 = 1,\ i_2 = 1,\ i_3 = 0,\ i_4 = 0 \tag{3.7}$$

Here, 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:

$$\varphi(t_1, c_4) = f_{1,4}(i_1, i_2, i_3, i_4) = i_1 \wedge i_2 \tag{3.8}$$
$$\varphi(t_1, c_5) = f_{1,5}(i_1, i_2, i_3, i_4) = i_1 \wedge \neg i_3 \tag{3.9}$$
$$\varphi(t_1, c_6) = f_{1,6}(i_1, i_2, i_3, i_4) = i_2 \wedge \neg i_4 \tag{3.10}$$
$$\varphi(t_2, c_3) = f_{2,3}(i_1, i_2, i_3, i_4) = i_1 \wedge \neg i_3 \tag{3.11}$$
$$\varphi(t_2, c_5) = f_{2,5}(i_1, i_2, i_3, i_4) = i_1 \wedge i_2 \tag{3.12}$$

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 $\boldsymbol{Pre}(i, j) = \boldsymbol{\varepsilon}$. Following Figure 3.4, the only non-empty $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ entries are

$$\boldsymbol{Pre}(p_1, t_1)(c_4) = 1 \otimes c_1 \tag{3.13}$$
$$\boldsymbol{Pre}(p_1, t_1)(c_5) = 2 \otimes c_2 \tag{3.14}$$
$$\boldsymbol{Pre}(p_1, t_1)(c_6) = 2 \otimes c_1 + 1 \otimes c_2 \tag{3.15}$$
$$\boldsymbol{Pre}(p_1, t_2)(c_3) = 2 \otimes c_1 \tag{3.16}$$
$$\boldsymbol{Pre}(p_1, t_2)(c_5) = 1 \otimes c_2 \tag{3.17}$$
$$\boldsymbol{Post}(p_2, t_1)(c_4) = 2 \otimes c_2 \tag{3.18}$$
$$\boldsymbol{Post}(p_2, t_1)(c_5) = 1 \otimes c_1 \tag{3.19}$$
$$\boldsymbol{Post}(p_2, t_1)(c_6) = 1 \otimes c_2 + 1 \otimes c_3 + 1 \otimes c_4 \tag{3.20}$$
$$\boldsymbol{Post}(p_2, t_2)(c_3) = 1 \otimes c_4 \tag{3.21}$$
$$\boldsymbol{Post}(p_2, t_2)(c_5) = 1 \otimes c_3 + 1 \otimes c_4 \tag{3.22}$$
$$\boldsymbol{Post}(p_3, t_2)(c_3) = 1 \otimes c_4 \tag{3.23}$$
$$\boldsymbol{Post}(p_3, t_2)(c_5) = 1 \otimes c_3 + 1 \otimes c_4 \tag{3.24}$$

A simpler and smarter way to define the $\boldsymbol{Pre}$ (resp. $\boldsymbol{Post}$) matrix is depicted in Figure 3.4. 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

$$\boldsymbol{M}_0 = \left[\ 2 \otimes c_1 + 1 \otimes c_2,\ \ \boldsymbol{\varepsilon},\ \ \boldsymbol{\varepsilon}\ \right], \tag{3.25}$$

upon verification of (3.8)-(3.12), 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. It is crucial to note that this example is designed to highlight such ambiguity and emphasise that the order in which a transition will fire is not unique and depends on the user's policy and here it is assumed a policy that first gives priority to transitions with lower index and then gives priority to colours with lower index. Therefore, given that P, T, and C_ℓ are ordered sets, transition t_1 fires w.r.t. c_4 because when iterating through all the transitions and places it will be detected first. After that, then the net's marking becomes

$$\boldsymbol{M}_1 = \left[\ 1 \otimes c_1 + 1 \otimes c_2,\ \ 2 \otimes c_2,\ \ \boldsymbol{\varepsilon}\ \right]. \tag{3.26}$$

Notice that since in a SICPN several signals can change simultaneously, it is essential to update the net's marking as soon as a transition fires to avoid critical situations. Thus, following this policy, from $\langle N, \boldsymbol{M}_1 \rangle$ and recalling (3.7) 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:

$$\begin{aligned}
\omega(p_1, c_1) &= \begin{bmatrix}1, & -, & -, & -, & -\end{bmatrix} && (3.27)\\
\omega(p_1, c_2) &= \begin{bmatrix}-, & -, & -, & -, & -\end{bmatrix} && (3.28)\\
\omega(p_2, c_1) &= \begin{bmatrix}1, & -, & 1, & -, & -\end{bmatrix} && (3.29)\\
\omega(p_2, c_2) &= \begin{bmatrix}-, & 1, & 0, & -, & 0\end{bmatrix} && (3.30)\\
\omega(p_2, c_3) &= \begin{bmatrix}-, & 0, & 0, & 1, & -\end{bmatrix} && (3.31)\\
\omega(p_2, c_4) &= \begin{bmatrix}1, & -, & 0, & 0, & 1\end{bmatrix} && (3.32)\\
\omega(p_3, c_1) &= \begin{bmatrix}1, & -, & -, & -, & -\end{bmatrix} && (3.33)\\
\omega(p_3, c_2) &= \begin{bmatrix}-, & 1, & 0, & -, & -\end{bmatrix} && (3.34)\\
\omega(p_3, c_3) &= \begin{bmatrix}-, & -, & 0, & 1, & 0\end{bmatrix} && (3.35)\\
\omega(p_3, c_4) &= \begin{bmatrix}0, & -, & 0, & -, & 1\end{bmatrix} && (3.36)
\end{aligned}$$

Now it is explained how the output function Ω in (3.4) relates to the current marking.

Let us consider $\boldsymbol{M}_0$. Since only p_1 is marked (with colours c_1 and c_2), based on eq. (3.27)-(3.28), 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. (3.27), (3.28) and (3.30), it is $\Omega(\boldsymbol{M}_1) = [\ 1,\ 1,\ 0,\ -,\ 0\ ]$.

Finally, if we consider $\boldsymbol{M}_2$, from (3.30) - (3.32), it is $\Omega(\boldsymbol{M}_2) = [\ r_1,\ q,\ r_0,\ q_1,\ q_{01}\ ]$. The first entry follows from eq. (3.27) and (3.32); the second entry follows from eq. (3.30) and (3.31); the third entry follows from (3.30) - (3.32), (3.35) and (3.36); while the fourth entry is because there is simultaneously a contradiction and a redundancy 1 due to (3.31), (3.32) and

(3.35); finally, the fifth entry follows from eq. (3.30), (3.35) that propose 0 while (3.32), (3.36) that propose 1.

3.4 Output Function implementation

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, as shown in (3.4). 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 to prevent unexpected behaviours, while any redundant zeros or redundant ones will be translated to 0 or 1, respectively.

3.5 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 (Borges & Lima II, 2018; Frey, 2002) 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:

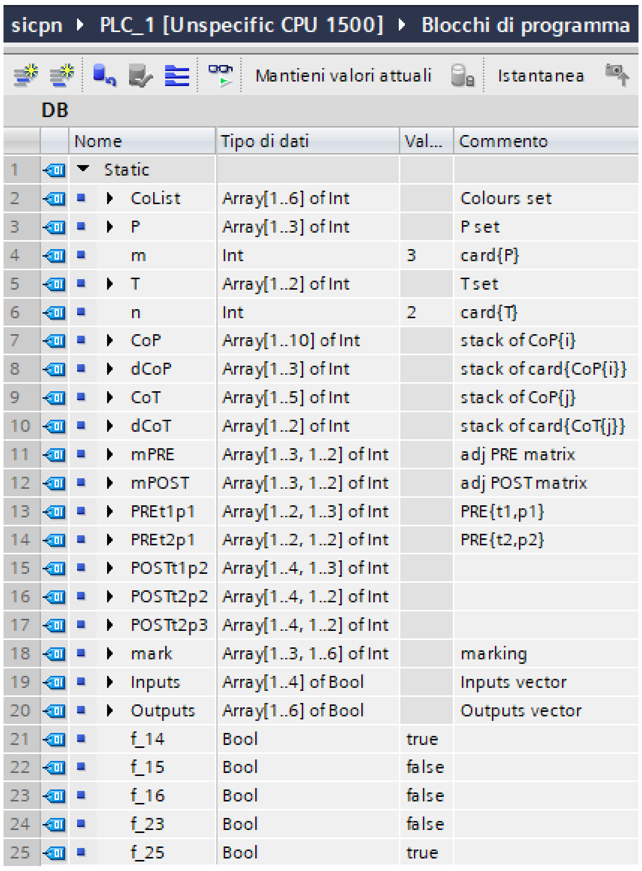
a) properly defining the net's primitives within the controller's memory. This involves aligning them with available data-type structures;

b) developing a software entity, referred to as the Token-Player, tasked to correctly implement the SICPN dynamics, see e.g. subsection 3.3.1.

Here the focus is explaining how to implement it within a PLC. Specifically, among the five IEC 61131-3 languages (LD, ST, SFC, FBD, IL), it is utilised 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.

3.5.1 Memory allocations and function definitions

To achieve this task, the Token-Player was initially implemented in MATLAB and then transitioned to SCL. In this regard, and following point a), the major difficulty was the lack of advanced data-type structures such as cell-arrays (Mathworks, 2024), which would have been ideal for defining each entry (i, j) of $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ as a $|Co(p_i)| \times |Co(t_j)|$ indexed matrix if



	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 3.5 – Screenshot of the Siemens TIA Portal V16 IDE, displaying the `GLOBAL DB` used to store the primitives of the SICPN as shown in Figure 3.4, and (3.37)-(3.39).

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 3.5. To overcome this issue, some additional support variables were introduced in the IEC61131-3 compliant `GLOBAL DB` used to store the Net's primitives, the current marking (3.2) and values of the transition functions $f_{j,k}$, see Figure 3.5 for details. More precisely, and concerning the example in Figure 3.4, these additional variables are

$$\mathtt{CoP} = \left[\begin{array}{cc|cccc|cccc} 1 & 2 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \end{array}\right]^{\mathsf{T}} \tag{3.37}$$

$$\mathtt{CoT} = \left[\begin{array}{ccc|cc} 4 & 5 & 6 & 3 & 5 \end{array}\right]^{\mathsf{T}} \tag{3.38}$$

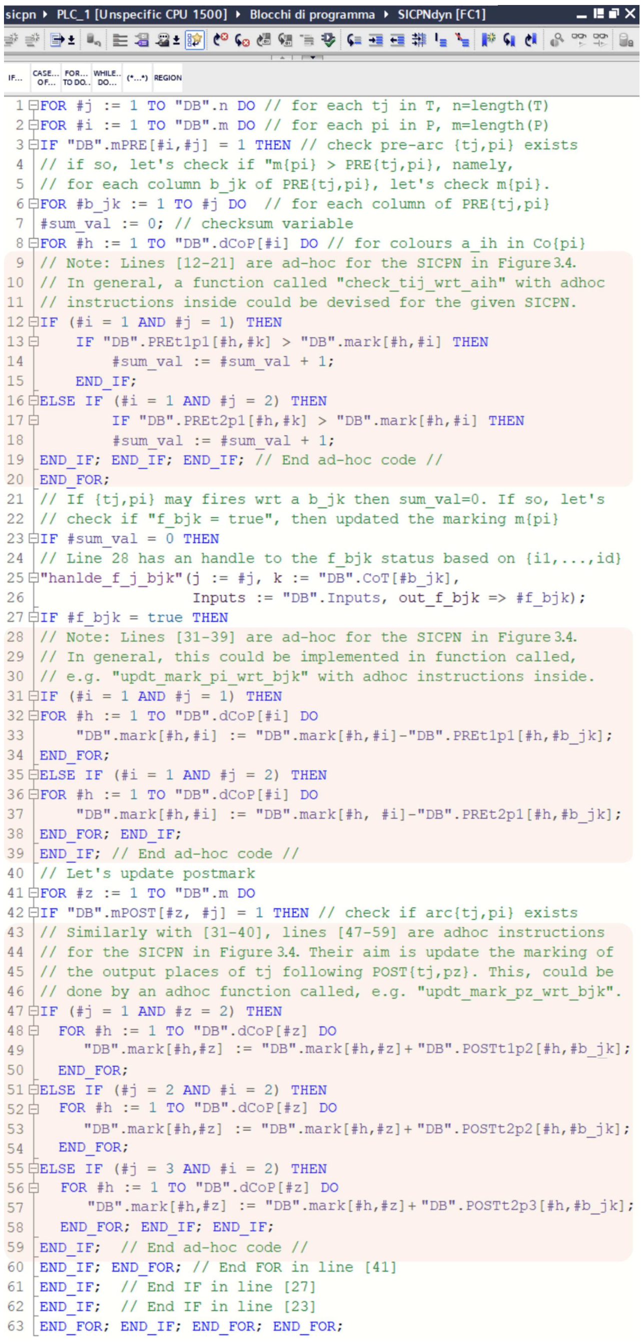
aimed to stack each $Co(p_i)$ (resp. $Co(t_j)$), and

$$\mathtt{dCoP} = \begin{bmatrix} 2 & 4 & 4 \end{bmatrix}^{\mathsf{T}} \quad , \quad \mathtt{dCoT} = \begin{bmatrix} 3 & 2 \end{bmatrix}^{\mathsf{T}} \tag{3.39}$$

aimed at storing the cardinality of each $Co(p_i)$ (resp. $Co(t_j)$); see lines 7-10 of Figure 3.5. Moreover, for memory optimisation, only the non-empty entries of $\boldsymbol{Pre}$ (resp. $\boldsymbol{Post}$) are memorised, as shown in lines 13-17 of Figure 3.5. Then, the matrices

$$\mathtt{mPRE} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad , \quad \mathtt{mPOST} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{3.40}$$

were used to keep track to which pairs (p_i, t_j) there exists (i.e. 1) or not (i.e. 0) an non-empty $\boldsymbol{Pre}(p_i, t_j)$ (resp. $\boldsymbol{Post}(p_i, t_j)$) matrix, cf. Figure 3.4 with (3.40) and with line 3 of Figure 3.6. The meaning of all the other variables in Figure 3.5 can instead be easily inferred from the comments provided on the right of the `GLOBAL DB` interface.


sicpn ▸ PLC_1 [Unspecific CPU 1500] ▸ Blocchi di programma ▸ SICPNdyn [FC1]
IF...
CASE... OF...
FOR... TO DO..
WHILE.. DO...
(*...*)
REGION

```
 1 FOR #j := 1 TO "DB".n DO // for each tj in T, n=length(T)
 2 FOR #i := 1 TO "DB".m DO // for each pi in P, m=length(P)
 3 IF "DB".mPRE[#i,#j] = 1 THEN // check pre-arc {tj,pi} exists
 4 // if so, let's check if "m{pi} > PRE{tj,pi}, namely,
 5 // for each column b_jk of PRE{tj,pi}, let's check m{pi}.
 6 FOR #b_jk := 1 TO #j DO  // for each column of PRE{tj,pi}
 7 #sum_val := 0; // checksum variable
 8 FOR #h := 1 TO "DB".dCoP[#i] DO // for colours a_ih in Co{pi}
 9 // Note: Lines [12-21] are ad-hoc for the SICPN in Figure 3.4.
10 // In general, a function called "check_tij_wrt_aih" with adhoc
11 // instructions inside could be devised for the given SICPN.
12 IF (#i = 1 AND #j = 1) THEN
13     IF "DB".PREt1p1[#h,#k] > "DB".mark[#h,#i] THEN
14         #sum_val := #sum_val + 1;
15     END_IF;
16 ELSE IF (#i = 1 AND #j = 2) THEN
17         IF "DB".PREt2p1[#h,#k] > "DB".mark[#h,#i] THEN
18         #sum_val := #sum_val + 1;
19 END_IF; END_IF; END_IF; // End ad-hoc code //
20 END_FOR;
21 // If {tj,pi} may fires wrt a b_jk then sum_val=0. If so, let's
22 // check if "f_bjk = true", then updated the marking m{pi}
23 IF #sum_val = 0 THEN
24 // Line 28 has an handle to the f_bjk status based on {i1,...,id}
25 "hanlde_f_j_bjk"(j := #j, k := "DB".CoT[#b_jk],
26                  Inputs := "DB".Inputs, out_f_bjk => #f_bjk);
27 IF #f_bjk = true THEN
28 // Note: Lines [31-39] are ad-hoc for the SICPN in Figure 3.4.
29 // In general, this could be implemented in function called,
30 // e.g. "updt_mark_pi_wrt_bjk" with adhoc instructions inside.
31 IF (#i = 1 AND #j = 1) THEN
32 FOR #h := 1 TO "DB".dCoP[#i] DO
33     "DB".mark[#h,#i] := "DB".mark[#h,#i]-"DB".PREt1p1[#h,#b_jk];
34 END_FOR;
35 ELSE IF (#i = 1 AND #j = 2) THEN
36 FOR #h := 1 TO "DB".dCoP[#i] DO
37     "DB".mark[#h,#i] := "DB".mark[#h, #i]-"DB".PREt2p1[#h,#b_jk];
38 END_FOR; END_IF;
39 END_IF; // End ad-hoc code //
40 // Let's update postmark
41 FOR #z := 1 TO "DB".m DO
42 IF "DB".mPOST[#z, #j] = 1 THEN // check if arc{tj,pi} exists
43 // Similarly with [31-40], lines [47-59] are adhoc instructions
44 // for the SICPN in Figure 3.4. Their aim is update the marking of
45 // the output places of tj following POST{tj,pz}. This, could be
46 // done by an adhoc function called, e.g. "updt_mark_pz_wrt_bjk".
47 IF (#j = 1 AND #z = 2) THEN
48   FOR #h := 1 TO "DB".dCoP[#z] DO
49      "DB".mark[#h,#z] := "DB".mark[#h,#z]+"DB".POSTt1p2[#h,#b_jk];
50   END_FOR;
51 ELSE IF (#j = 2 AND #i = 2) THEN
52   FOR #h := 1 TO "DB".dCoP[#z] DO
53      "DB".mark[#h,#z] := "DB".mark[#h,#z]+"DB".POSTt2p2[#h,#b_jk];
54   END_FOR;
55 ELSE IF (#j = 3 AND #i = 2) THEN
56   FOR #h := 1 TO "DB".dCoP[#z] DO
57      "DB".mark[#h,#z] := "DB".mark[#h,#z]+"DB".POSTt2p3[#h,#b_jk];
58   END_FOR; END_IF; END_IF;
59 END_IF;  // End ad-hoc code //
60 END_IF; END_FOR; // End FOR in line [41]
61 END_IF;  // End IF in line [27]
62 END_IF;  // End IF in line [23]
63 END_FOR; END_IF; END_FOR; END_FOR;
```

Figure 3.6 – 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 3.4, and with primitives defined as in Figure 3.5.

3.5.2 Token-Player implementation

In Figure 3.6 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 3.4, and with primitives in the `GLOBAL DB` depicted in Figure 3.5.

Note that the definition of the support variables in (3.37)-(3.39) was a strategy to allow the verification of the firing condition for all the net transitions t_j (see index `#j` in line 1) w.r.t.: *all places* p_i (see index `#i` in line 2); *all colours transitions* $b_{j,k} \in Co(t_j)$ (see index `#b_jk` in line 6, and e.g. `k=DB.CoT[#b_jk]` in line 25); *all colours places* $a_{i,h} \in Co(p_i)$ (see line 8 where `#h=1, 2,..., DB.CoP[#i]`).

Further note that since the Token-Player in Figure 3.6 is thoroughly commented within the figure. Thus, 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 3.4.

In general, those code portions should be replaced with function calls specifically designed for the given SICPN (see the comment in line 9-11). However, these instructions were made 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 $b_{j,k}$, considering the current marking $\boldsymbol{m}_i$ and the condition $\boldsymbol{Pre}(p_i, t_j)(\cdot, k)$.

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

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

Moreover, it is important to note that the provided Token-Player `FC` in Figure 3.6 is expected to be executed within the `OB1` main cyclic organisation block of the PLC. This ensures that during each scan cycle, the entire SICPN marking is continuously updated based on the current input status I and marking $\boldsymbol{M}$. Further note immediately after the net marking is updated, a second `FC` aimed at implementing the output function Ω is requested to be executed in subsection 3.3.1. Here, the `FC` corresponding to the output function Ω is omitted due to space limitations. However, it can be easily derived based on equations (3.27)-(3.36).

4 DISCUSSION BASED ON A CASE STUDY

4.1 Presenting the case study

To validate the effectiveness of this formalism for directly implementing control algorithms, a SICPN-based feedback control is modelled and tested 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 & Pany, 2015). The control loop between the Arduino and the DT, which runs in MATLAB, is facilitated through two-way UART serial communications. Consequently, it is shown the SICPN-based controller's effective and seamless operation with the proposed DT, despite the plant being emulated.

It is noteworthy that this testbench has long served as a benchmark in the field of manufacturing automation (Lima II & Dórea, 2004). Recently, it has also emerged as a common reference for DT applications and PLC virtual commissioning (Abdelsattar et al., 2022; Batchkova et al., 2013; Fernández et al., 2021; Mykoniatis & Harris, 2021). In contrast to previous studies, which typically focus on the standard FESTO configuration as depicted in Figure 4.1, this thesis research, akin to Batchkova et al. (2013), delves into an extended configuration. This expanded setup includes additional and different testing and drilling modules. Moreover, it is considered workpieces of various materials, random arrival times, and potentially requiring different operations.

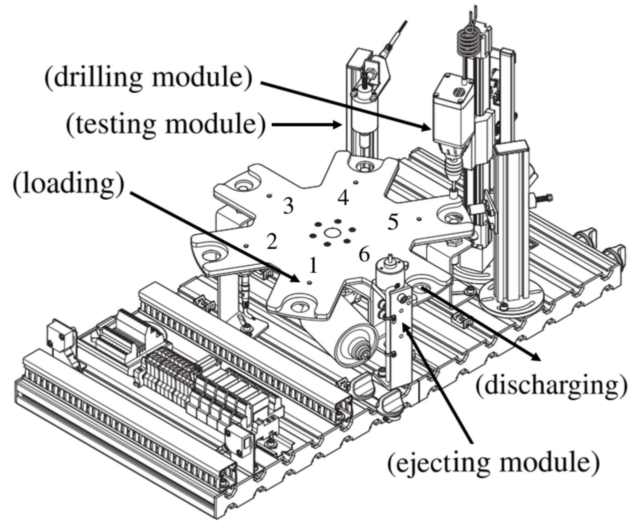


Figure 4.1 – Standard Configuration of the FESTO S-BE-M Processing Station (Ebel & Pany, 2015). Note the extended configuration considered encompasses supplementary testing and drilling modules, situated at position indexes 2 and 3, respectively.

4.1.1 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 core of the station is a DC-motor driven rotating table as shown in Figure 4.1. 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 4.1. Each PI identifies a specific station and a function, resp.:

PI-1 Identification (ID)$\rightarrow$ A workpiece is here loaded from a buffer, and its material is identified.

PI-2 Testing A (TA) $\rightarrow$ A testing module checks for a hole with a diameter $d_A > 0$ or larger in the piece.

PI-3 Processing A (PA) $\rightarrow$ A drilling module pierces the workpiece using a tip with a diameter $d_A > 0$.

PI-4 Testing B (TB) $\rightarrow$ A testing module checks for a hole with a diameter $d_B > d_A$ or larger in the piece.

PI-5 Processing B (PB) $\rightarrow$ A drilling module pierces the workpiece using a tip with a diameter $d_B > d_A$.

PI-6 Ejection (EJ)$\rightarrow$ 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, it is assumed 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 undergo a single drilling operation at PB. The control problem is formulated in a way that allows a maximum of six pieces to be worked on simultaneously. Furthermore, in the workpiece arrival process, it is assumed that some blocks may not require any operation, and the control system must be capable of recognising them.

Here, unlike in the previously mentioned literature, the control will be formulated to handle a maximum of **six pieces** being worked on **simultaneously** instead of just one.

Additionally, in the workpiece arrival process, it is assumed that some blocks may not require any operations, while the control system must be capable of identifying and dealing with them. An important aspect to emphasise and underscore is the complexity of the problem and the fundamental role of using coloured PN 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 4.1 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.

Table 4.1 – Sensor combinations for the PI-1's identification task.

Logical statement	**Conclusion**
IND	metal (M)
$\neg$IND $\wedge$ OPT	red plastic (RP)
$\neg$IND $\wedge$ $\neg$OPT $\wedge$ CAP	black plastic (BP)
otherwise	no workpiece

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

TAOK represents the output of the TA station. However, based on the given specifications, this test is only required for M parts, while RP or BP blocks can skip the test at PI-2. In contrast, all types of blocks must undergo testing in the TB station in PI-4, and TBOK denotes the test's output.

When TAOK is *false*, meaning that M part is not yet drilled, thus PA station PI-3 will drill it. Whereas, when TBOK is *false*, meaning that the block of any material is not yet drilled, thus PB station PI-5 will drill it.

Finally, CAPF is the capacitive sensor that identifies whether there is a block at PI-6 of the table, indicating that the block is ready to be removed from the station as its manufacturing process is completed.

4.2 Feedback controller modelling via SICPN

Before introducing the feedback controller modelling, there are two issues to highlight:

- There is no "uncontrolled" model because it is built considering its controlled behaviour, i.e. the synthesis is done when design the model in SICPN.
- In a decision-making process there is a feedback based on the given information based on the input signals to calculate possible outputs that will generated different control actions to be executed.

Here is presented the modelling of the proposed SICPN feedback controller for the case study under consideration, and each entry of 11-tuple in (3.1) will be now characterised for the control problem at hand.

4.2.1 Modelling structure

Figure 4.2 illustrates the bipartite graph associated with the SICPN-based controller, custom-designed for automating the extended FESTO MPS station. Considering each case study is different, the built model is ad hoc and the researcher must know what behaviour is expected for the system. Despite the consistency of the SICPN notation, the researcher must pay attention if the outputs are contradictory when applying the notation to a model.

Definitions for each of the 33 net's places can be found in Table 4.2.

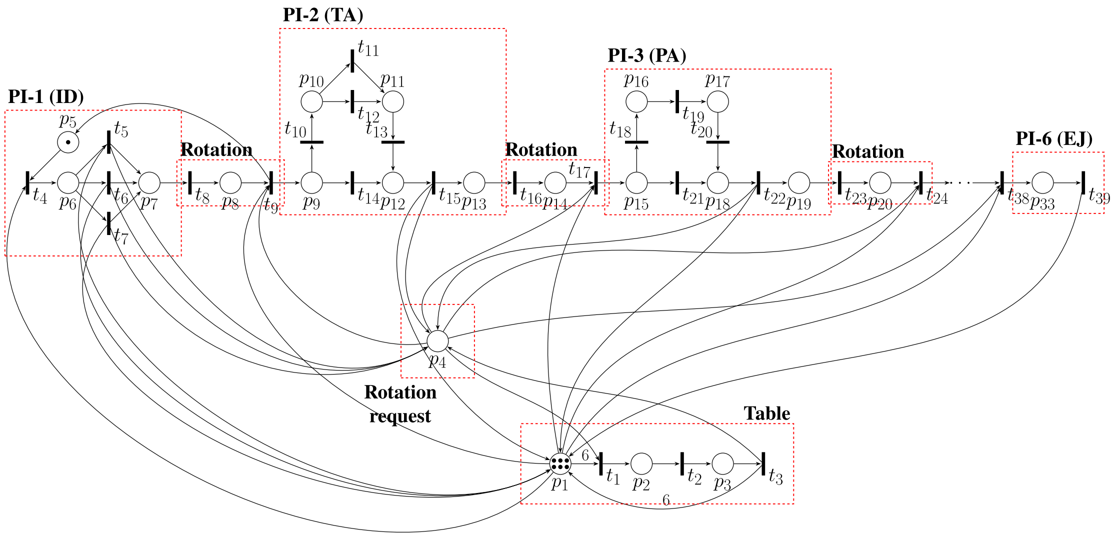

PI-1 (ID)
PI-2 (TA)
PI-3 (PA)
PI-6 (EJ)
Rotation
Rotation
Rotation
Rotation
Rotation request
Table
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12} p_{13} p_{14} p_{15} p_{16} p_{17} p_{18} p_{19} p_{20} p_{33}
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17} t_{18} t_{19} t_{20} t_{21} t_{22} t_{23} t_{24} t_{38} t_{39}
6
6

Figure 4.2 – SICPN-based modelling of FESTO MPS control algorithm.

Remarkably, the SICPN with 33 places can be considered compact, considering the control problem complexity. This problem entails managing 3 types of workpieces, coordinating up to 6 different operations simultaneously (one operation for each PI), addressing scenarios where no operation is required (e.g., when TA or TB tests fail or when there is no block in PI-1), and controlling the DC motor for table rotation.

In Figure 4.2, the dashed boxes, labelled PI-X with $X = 1, 2, \ldots, 6$, highlight the portion of the SICPN net 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 4.2. This is because TB and PB can operate on all the material types, while TA and PA only on metal blocks, resulting in similar but simpler functionalities to implement.

Places p_4 and p_5 denote two resources of the system (see Figure 4.2 and Table 4.2):

- 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, as a place invariant.

It is possible to ensure the system is not blocking by modelling construction. Resource p_5 guarantees there is only one block at a time in station PI-1. For the other stations, this is limited because the rotation is synchronised for all the positions PI-1 to PI-6, controlled by resource p_4.

Table 4.2 – 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

Afterwards, the model was tested by extensive simulation with no errors, i.e. the system behaved as expected.

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.

4.2.2 Colour

The modelled system considers three different materials, as presented at Table 4.1, and this is one of the features incorporated and stored within each token. Here, in respect to colours, the possibilities are: 00 (*Unknown*), before material identification; 01 (*Metal*), 10 (*Red plastic*) and 11 (*Black plastic*), after material identification. Beyond the type of material, there are two features regarding the status: $drilled_A$, for a hole with a diameter $d_A > 0$; and $drilled_B$, for a hole with a diameter $d_B > d_A$.

The list of token colours used to describe the system status is given in Table 4.3 along with the corresponding 4-bit lookup table.

Table 4.3 – Possible colours based on system properties.

Colour index	**Material**	$\boldsymbol{drilled_A}$	$\boldsymbol{drilled_B}$
c_0	00 Unknown	-	-
c_1	01 Metal	0	0
c_2		1	0
c_3		1	1
c_4	10 Red plastic	0	0
c_5		1	1
c_6	11 Black plastic	0	0
c_7		1	1

4.2.3 Signal interpretation

In signal interpretation, the set of input signals is as next:

$$I = \Big\{\text{IDX, IND, OPT, CAP, TA0, TA1, TAOK, PA0, PA1, TB0, TB1, TBOK, PB0, PB1, CAPF}\Big\}. \tag{4.1}$$

Please refer to subsection 4.1.1 for details on their meaning. Finally, the net's controller output, namely the set commands to be delivered to the plant's actuators, is

$$O = \Big\{\text{MT}, TA_{adv}, TA_{ret}, PA_{adv}, PA_{ret}, TB_{adv}, TB_{ret}, PB_{adv}, PB_{ret}\Big\}. \tag{4.2}$$

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.

4.2.4 A detailed description of PI-1

To provide an understanding of how the controller in Figure 4.2 operates, it is given more detail regarding the functions associated with the portions of the net labelled *PI-1* and *Rotation* in Figure 4.3. A zoomed-in view of this sub-net is provided in Figure 4.4, 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 4.3. Subsequently, a token c_0 enters place p_6 in Figure 4.4. Then,

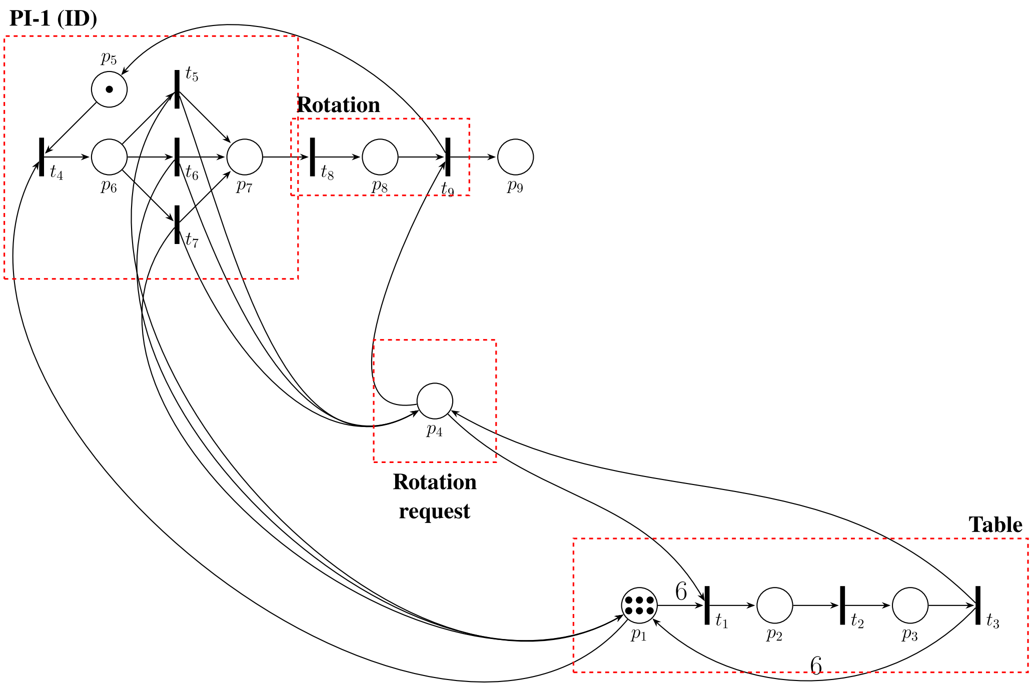


Figure 4.3 – SICPN-based modelling of FESTO MPS control algorithm - Position Index PI-1.

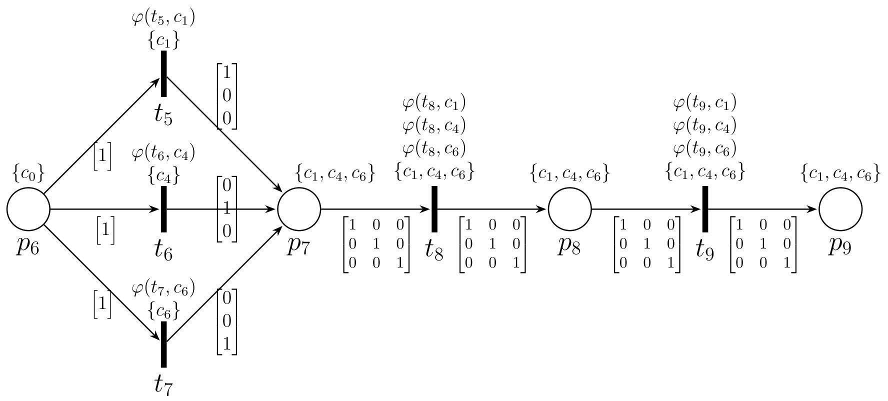


Figure 4.4 – Excerpt from the SICPN in Figure 4.2.

sensors IND, OPT, and CAP located in PI-1 perform the identification. More precisely, following Figure 4.4 and Table 4.3, the following transition rules are devised to enable the workpiece material identification:

$$\varphi(t_5, c_1) = \text{IND} \tag{4.3}$$

$$\varphi(t_6, c_4) = \neg\text{IND} \wedge \text{OPT} \tag{4.4}$$

$$\varphi(t_7, c_6) = \neg\text{IND} \wedge \neg\text{OPT} \wedge \text{CAP} \tag{4.5}$$

Then, $\boldsymbol{Pre}(p_i, t_j)$ and $\boldsymbol{Post}(p_i, t_j)$ in Figure 4.4 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 4.3, 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

$$\boldsymbol{Pre}(p_6, t_5)(c_1) = 1 \otimes c_0 \tag{4.6}$$
$$\boldsymbol{Pre}(p_6, t_6)(c_4) = 1 \otimes c_0 \tag{4.7}$$
$$\boldsymbol{Pre}(p_6, t_7)(c_6) = 1 \otimes c_0 \tag{4.8}$$
$$\boldsymbol{Post}(p_7, t_5)(c_1) = 1 \otimes c_1 \tag{4.9}$$
$$\boldsymbol{Post}(p_7, t_6)(c_4) = 1 \otimes c_4 \tag{4.10}$$
$$\boldsymbol{Post}(p_7, t_7)(c_6) = 1 \otimes c_6 \tag{4.11}$$

Moreover, as soon as t_5, or t_6, or t_7 fires, in accordance with Figure 4.3, we also have

$$\boldsymbol{Post}(p_4, t_5)(c_1) = 1 \otimes c_0 \tag{4.12}$$
$$\boldsymbol{Post}(p_4, t_6)(c_4) = 1 \otimes c_0 \tag{4.13}$$
$$\boldsymbol{Post}(p_4, t_7)(c_6) = 1 \otimes c_0 \tag{4.14}$$
$$\boldsymbol{Post}(p_1, t_5)(c_1) = 1 \otimes c_0 \tag{4.15}$$
$$\boldsymbol{Post}(p_1, t_6)(c_4) = 1 \otimes c_0 \tag{4.16}$$
$$\boldsymbol{Post}(p_1, t_7)(c_6) = 1 \otimes c_0 \tag{4.17}$$

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 4.2). 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. (4.12)-(4.14) 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 c_0 tokens (see Figure 4.2). 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 (4.15)-(4.17).

If (i) and (ii) hold, then t_1 in Figure 4.3 fires, resulting with the following output function configurations

$$\omega(p_1, c_0) = \begin{bmatrix} 0, & -, & -, & -, & \cdots \end{bmatrix} \tag{4.18}$$
$$\omega(p_2, c_0) = \begin{bmatrix} 1, & -, & -, & -, & \cdots \end{bmatrix} \tag{4.19}$$
$$\omega(p_3, c_0) = \begin{bmatrix} 1, & -, & -, & -, & \cdots \end{bmatrix} \tag{4.20}$$

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 $MT = 0$.

Finally, with reference to Figure 4.4, the transition functions of t_8 are as follows:

$$\varphi(t_8, c_1) = \varphi(t_8, c_4) = \varphi(t_8, c_6) = \neg\text{IDX} \tag{4.21}$$

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

$$\begin{aligned}
\boldsymbol{Pre}(p_7, t_8)(c_1) &= 1 \otimes c_1 && (4.22)\\
\boldsymbol{Pre}(p_7, t_8)(c_4) &= 1 \otimes c_4 && (4.23)\\
\boldsymbol{Pre}(p_7, t_8)(c_6) &= 1 \otimes c_6 && (4.24)\\
\boldsymbol{Post}(p_8, t_8)(c_1) &= 1 \otimes c_1 && (4.25)\\
\boldsymbol{Post}(p_8, t_8)(c_4) &= 1 \otimes c_4 && (4.26)\\
\boldsymbol{Post}(p_8, t_8)(c_6) &= 1 \otimes c_6 && (4.27)
\end{aligned}$$

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 is marked, cf. with Table 4.2.

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.

4.3 Arduino compliant SICPN compiler

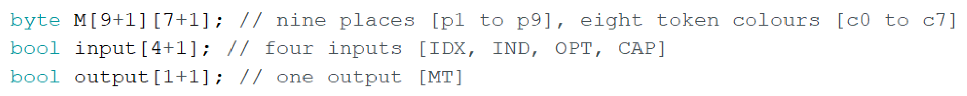
Refer to section 3.5 to remember the two aspects to run a SICPN within a control board.

This thesis is focused on explaining how to implement it within an Arduino Mega board, which is a microcontroller equipped with a 16 MHz ATmega2560 CPU, 4kB of SRAM, and 256kB of Flash memory using C++ language. This model has 33 places and 39 transitions and the used memory in Arduino is just a small fraction of its capacity. Thus, the scalability is possible considering the increase of the model complexity. Regarding computational complexity, Arduino did not use much of its processing capacity.

4.3.1 Memory allocations and function definitions

To implement the Token-Player in Arduino, one should define the SICPN model considering the net structure, colours and signal interpretation (inputs and outputs) according to the expected functionality and behaviour of the studied system.

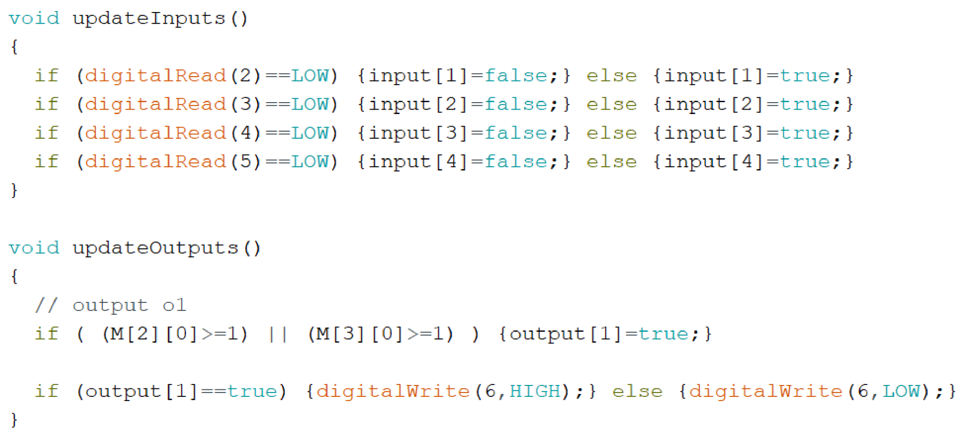
Once defined, the first step is to declare a matrix whose inputs are the *number of places* and the *number of token colours* as a byte, and the input and output signals as boolean vectors (Figure 4.5).



```
byte M[9+1][7+1]; // nine places [p1 to p9], eight token colours [c0 to c7]
bool input[4+1]; // four inputs [IDX, IND, OPT, CAP]
bool output[1+1]; // one output [MT]
```

Figure 4.5 – Screenshot of the Arduino IDE, displaying the variable declaration.

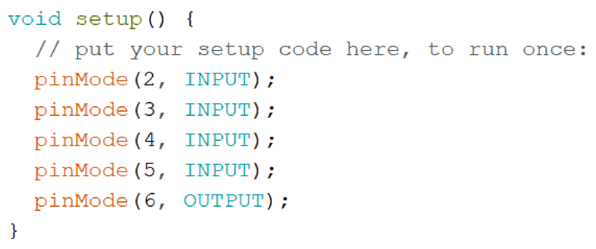
Then, *updateInputs()* and *updateOutputs()* functions are built (Figure 4.6). During the loop cycle, these functions will read physical inputs and store its values in the Input Image Table (input array) and to write the Output Image Table (output array) to the physical outputs. At the beginning of the loop, the system's inputs are read and copied into an input vector to be considered during that cycle because in each cycle the input signals can vary. In the end of the loop, all the calculated outputs are stored into an output vector and transmitted to the system to be executed when appropriated.



```
void updateInputs()
{
  if (digitalRead(2)==LOW) {input[1]=false;} else {input[1]=true;}
  if (digitalRead(3)==LOW) {input[2]=false;} else {input[2]=true;}
  if (digitalRead(4)==LOW) {input[3]=false;} else {input[3]=true;}
  if (digitalRead(5)==LOW) {input[4]=false;} else {input[4]=true;}
}

void updateOutputs()
{
  // output o1
  if ( (M[2][0]>=1) || (M[3][0]>=1) ) {output[1]=true;}

  if (output[1]==true) {digitalWrite(6,HIGH);} else {digitalWrite(6,LOW);}
}
```

Figure 4.6 – Screenshot of the Arduino IDE, displaying the input and output functions.

Setup configuration (Figure 4.7) associates Arduino pins with the input/output signals.



```
void setup() {
  // put your setup code here, to run once:
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, OUTPUT);
}
```

Figure 4.7 – Screenshot of the Arduino IDE, displaying the setup configuration.

4.3.2 Token-Player implementation

It is noticiable that the Token-Player is different for each SICPN model. It was constructed in an organised and clear way to render it adaptable to any other SICPN model in study.

Now that the initial configuration is defined, the behaviour of the Token-Player is exemplified showing the implementation only for the station PI-1 (Figure 4.3) of the FESTO

FMS station, including the *table*, *rotation request* and *rotation* itself leading the workpiece to the next station.

The firing transition rules and the dynamic behaviour is summarised as follows.

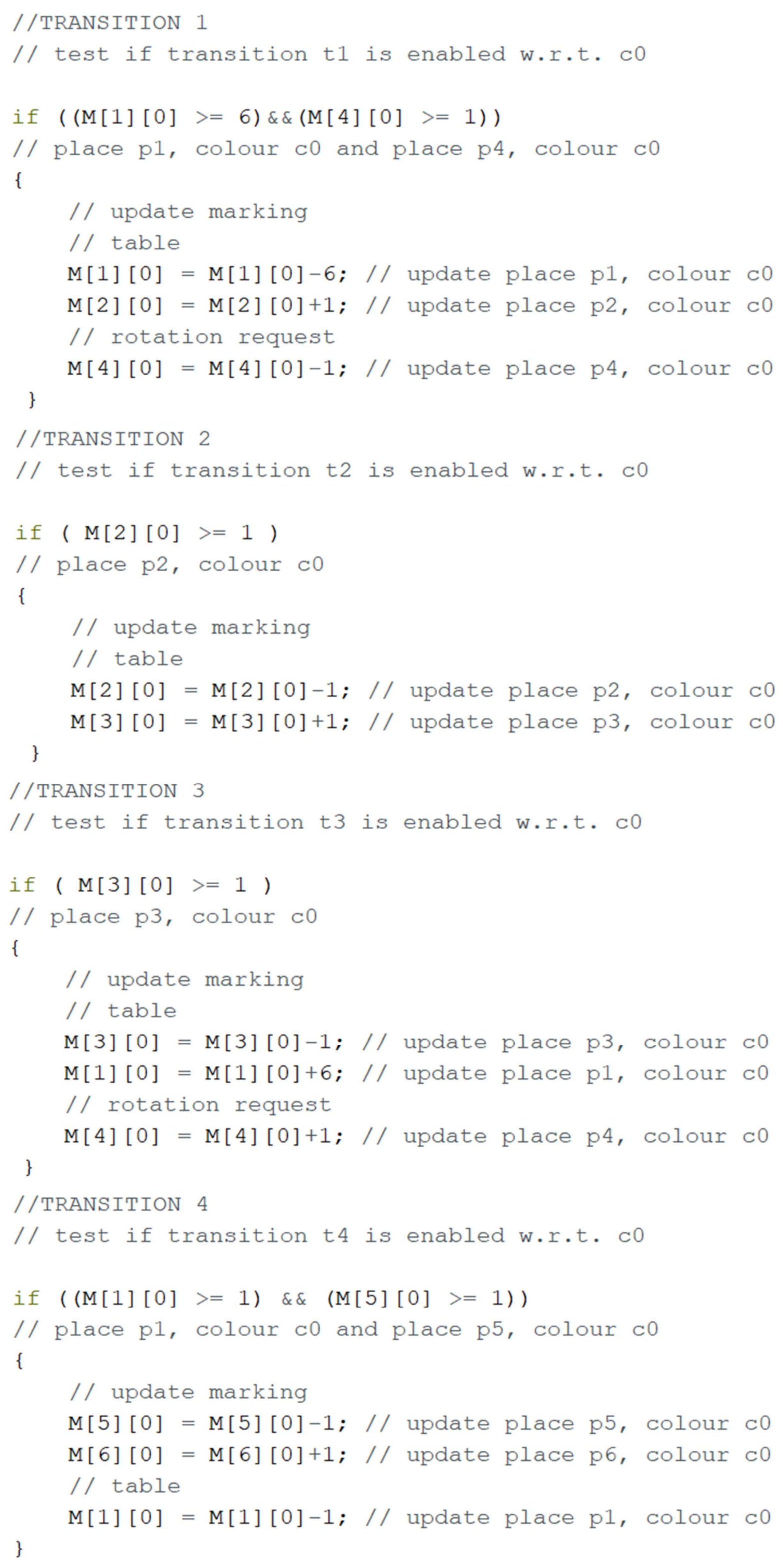
- Check if net structure and colour conditions are satisfied
  - Check if signal interpretation conditions are satisfied (if any)
    - Remove pre-place tokens
    - Add post-place tokens
    - Update outputs

The addressing of the marking is given by M[$\#$ of the place][$\#$ of the token colour]. This notation is applied when checking if the net and colouring are satisfied to enable the transition with respect to a specific colour and also when updating the marking after a transition fires removing tokens with Pre and adding tokens with $Post$.

First, transitions t_1 to t_4 (Figure 4.8) represent the table and the arrival of a workpiece with unidentified material into PI-1 station. Transitions t_1 verifies conditions (i) at least 1 token in p_4 (rotation request) and (ii) 6 tokens in p_1 (all stations have finished their processes), to start the table rotation. Transition t_2 represents the process of rotating the table and t_3 is when the rotation finishes, returning the tokens to p_1 and p_4. Finally, transition t_4 verifies if there is 1 token in p_5m which represents the resource that guarantees the PI-1 station is empty and ready to receive an unidentified workpiece.

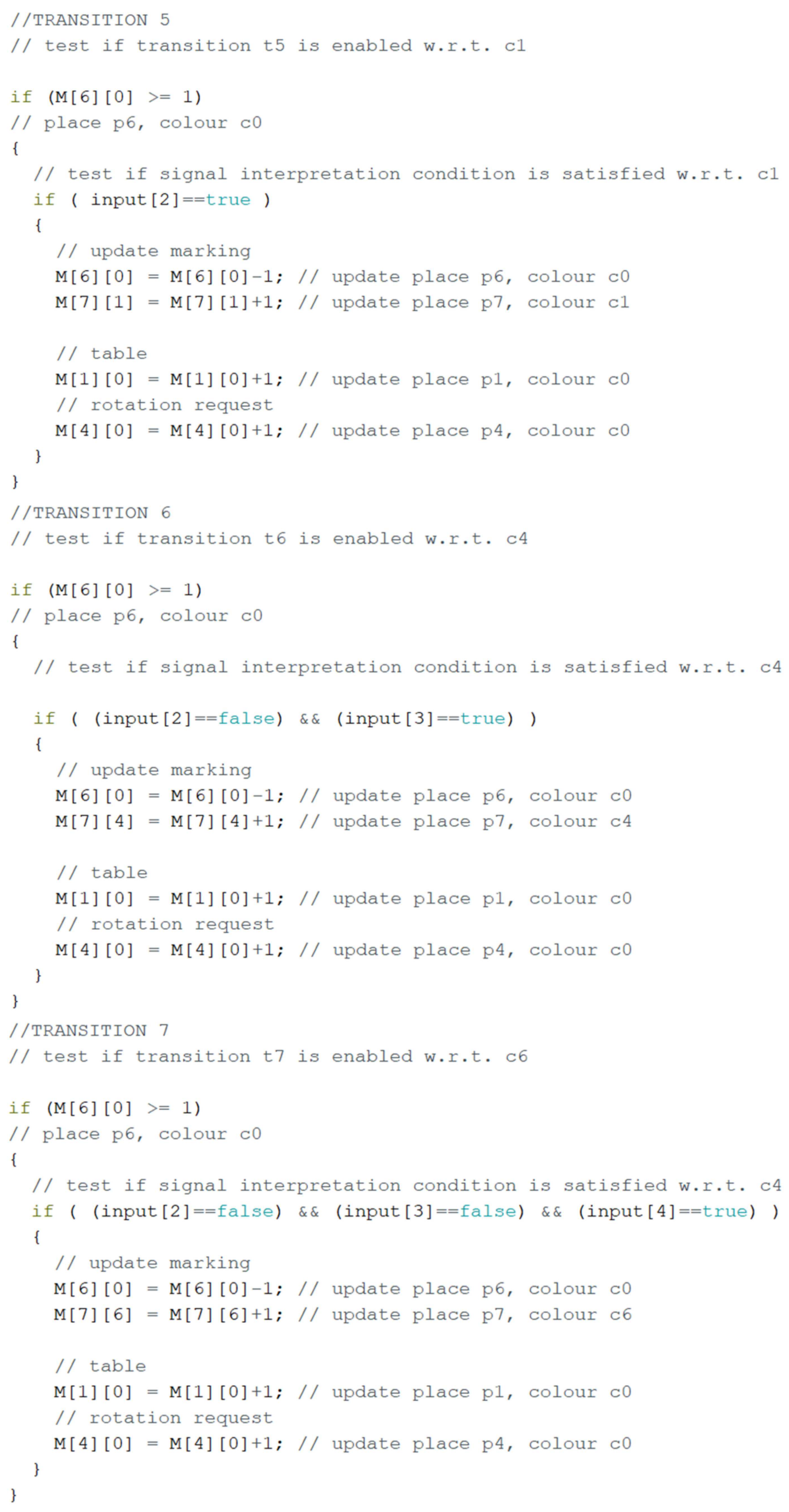
Once the workpiece enters the PI-1 station in place p_6, transitions t_5 to t_7 (Figure 4.9) are responsible to identify the workpiece's material. This recognition is done by using CAP, IND and OPT sensors. Each material has its own sensor combination logic as shown in Table 4.1 and equations (4.3) for metal, firing transition t_5; (4.4) for red plastic, firing transition t_6; and (4.5) for black plastic, firing transition t_7.

Transitions t_8 and t_9 of the *Rotation* are synchronised with the *Table*. While t_8 controls the start of the table rotation (Figure 4.10), which signal interpretation condition is shown in (4.21) for any kind of material; t_9 controls the finish of the table rotation (Figure 4.11), leading the workpiece, if any, from PI-1 to PI-2 and allowing a new cycle of the loop.



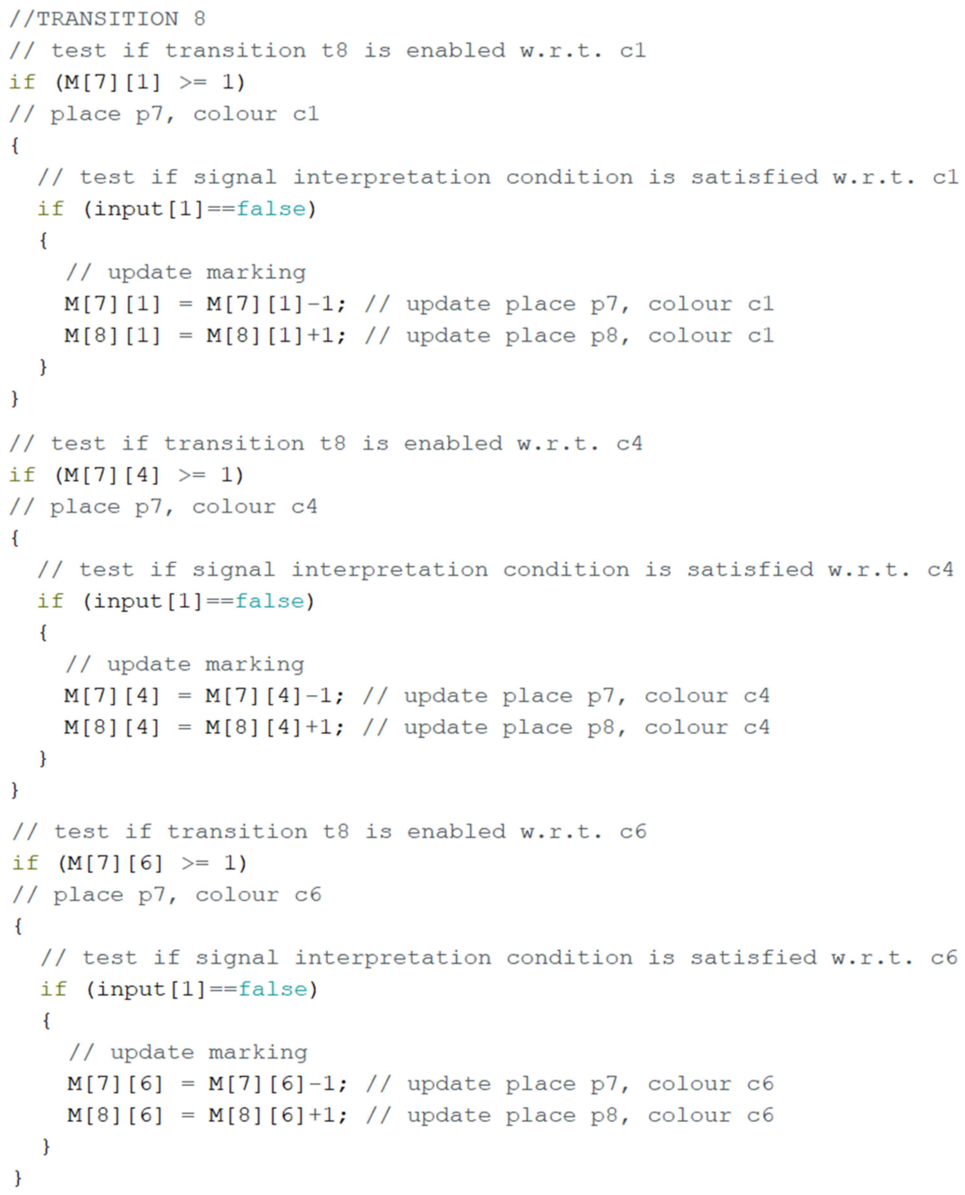
```
//TRANSITION 1
// test if transition t1 is enabled w.r.t. c0

if ((M[1][0] >= 6)&&(M[4][0] >= 1))
// place p1, colour c0 and place p4, colour c0
{
    // update marking
    // table
    M[1][0] = M[1][0]-6; // update place p1, colour c0
    M[2][0] = M[2][0]+1; // update place p2, colour c0
    // rotation request
    M[4][0] = M[4][0]-1; // update place p4, colour c0
 }
//TRANSITION 2
// test if transition t2 is enabled w.r.t. c0

if ( M[2][0] >= 1 )
// place p2, colour c0
{
    // update marking
    // table
    M[2][0] = M[2][0]-1; // update place p2, colour c0
    M[3][0] = M[3][0]+1; // update place p3, colour c0
 }
//TRANSITION 3
// test if transition t3 is enabled w.r.t. c0

if ( M[3][0] >= 1 )
// place p3, colour c0
{
    // update marking
    // table
    M[3][0] = M[3][0]-1; // update place p3, colour c0
    M[1][0] = M[1][0]+6; // update place p1, colour c0
    // rotation request
    M[4][0] = M[4][0]+1; // update place p4, colour c0
 }
//TRANSITION 4
// test if transition t4 is enabled w.r.t. c0

if ((M[1][0] >= 1) && (M[5][0] >= 1))
// place p1, colour c0 and place p5, colour c0
{
    // update marking
    M[5][0] = M[5][0]-1; // update place p5, colour c0
    M[6][0] = M[6][0]+1; // update place p6, colour c0
    // table
    M[1][0] = M[1][0]-1; // update place p1, colour c0
}
```

Figure 4.8 – Screenshot of the Arduino IDE, displaying the transitions t_1 to t_4 of Figure 4.3.



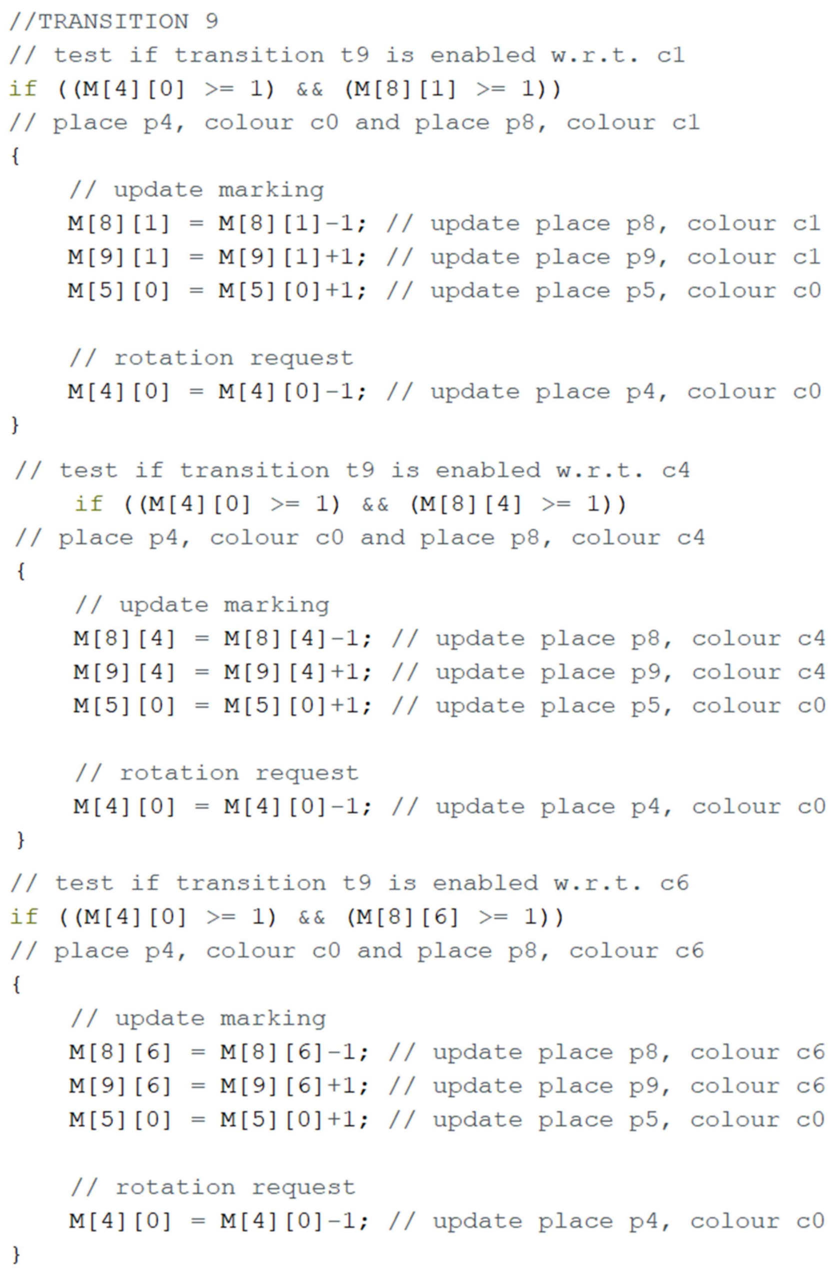
```
//TRANSITION 5
// test if transition t5 is enabled w.r.t. c1

if (M[6][0] >= 1)
// place p6, colour c0
{
  // test if signal interpretation condition is satisfied w.r.t. c1
  if ( input[2]==true )
  {
    // update marking
    M[6][0] = M[6][0]-1; // update place p6, colour c0
    M[7][1] = M[7][1]+1; // update place p7, colour c1

    // table
    M[1][0] = M[1][0]+1; // update place p1, colour c0
    // rotation request
    M[4][0] = M[4][0]+1; // update place p4, colour c0
  }
}
//TRANSITION 6
// test if transition t6 is enabled w.r.t. c4

if (M[6][0] >= 1)
// place p6, colour c0
{
  // test if signal interpretation condition is satisfied w.r.t. c4

  if ( (input[2]==false) && (input[3]==true) )
  {
    // update marking
    M[6][0] = M[6][0]-1; // update place p6, colour c0
    M[7][4] = M[7][4]+1; // update place p7, colour c4

    // table
    M[1][0] = M[1][0]+1; // update place p1, colour c0
    // rotation request
    M[4][0] = M[4][0]+1; // update place p4, colour c0
  }
}
//TRANSITION 7
// test if transition t7 is enabled w.r.t. c6

if (M[6][0] >= 1)
// place p6, colour c0
{
  // test if signal interpretation condition is satisfied w.r.t. c4
  if ( (input[2]==false) && (input[3]==false) && (input[4]==true) )
  {
    // update marking
    M[6][0] = M[6][0]-1; // update place p6, colour c0
    M[7][6] = M[7][6]+1; // update place p7, colour c6

    // table
    M[1][0] = M[1][0]+1; // update place p1, colour c0
    // rotation request
    M[4][0] = M[4][0]+1; // update place p4, colour c0
  }
}
```

Figure 4.9 – Screenshot of the Arduino IDE, displaying the transitions t_5 to t_7 of Figure 4.3.



```
//TRANSITION 8
// test if transition t8 is enabled w.r.t. c1
if (M[7][1] >= 1)
// place p7, colour c1
{
  // test if signal interpretation condition is satisfied w.r.t. c1
  if (input[1]==false)
  {
    // update marking
    M[7][1] = M[7][1]-1; // update place p7, colour c1
    M[8][1] = M[8][1]+1; // update place p8, colour c1
  }
}
// test if transition t8 is enabled w.r.t. c4
if (M[7][4] >= 1)
// place p7, colour c4
{
  // test if signal interpretation condition is satisfied w.r.t. c4
  if (input[1]==false)
  {
    // update marking
    M[7][4] = M[7][4]-1; // update place p7, colour c4
    M[8][4] = M[8][4]+1; // update place p8, colour c4
  }
}
// test if transition t8 is enabled w.r.t. c6
if (M[7][6] >= 1)
// place p7, colour c6
{
  // test if signal interpretation condition is satisfied w.r.t. c6
  if (input[1]==false)
  {
    // update marking
    M[7][6] = M[7][6]-1; // update place p7, colour c6
    M[8][6] = M[8][6]+1; // update place p8, colour c6
  }
}
```

Figure 4.10 – Screenshot of the Arduino IDE, displaying the transition t_8 of Figure 4.3.



```
//TRANSITION 9
// test if transition t9 is enabled w.r.t. c1
if ((M[4][0] >= 1) && (M[8][1] >= 1))
// place p4, colour c0 and place p8, colour c1
{
    // update marking
    M[8][1] = M[8][1]-1; // update place p8, colour c1
    M[9][1] = M[9][1]+1; // update place p9, colour c1
    M[5][0] = M[5][0]+1; // update place p5, colour c0

    // rotation request
    M[4][0] = M[4][0]-1; // update place p4, colour c0
}
// test if transition t9 is enabled w.r.t. c4
    if ((M[4][0] >= 1) && (M[8][4] >= 1))
// place p4, colour c0 and place p8, colour c4
{
    // update marking
    M[8][4] = M[8][4]-1; // update place p8, colour c4
    M[9][4] = M[9][4]+1; // update place p9, colour c4
    M[5][0] = M[5][0]+1; // update place p5, colour c0

    // rotation request
    M[4][0] = M[4][0]-1; // update place p4, colour c0
}
// test if transition t9 is enabled w.r.t. c6
if ((M[4][0] >= 1) && (M[8][6] >= 1))
// place p4, colour c0 and place p8, colour c6
{
    // update marking
    M[8][6] = M[8][6]-1; // update place p8, colour c6
    M[9][6] = M[9][6]+1; // update place p9, colour c6
    M[5][0] = M[5][0]+1; // update place p5, colour c0

    // rotation request
    M[4][0] = M[4][0]-1; // update place p4, colour c0
}
```

Figure 4.11 – Screenshot of the Arduino IDE, displaying the transition t_9 of Figure 4.3.

4.4 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. MATLAB is employed to simulate the plant dynamics, while Arduino is employed to implement the controller depicted in Figure 4.2. The control loop was closed using two-way UART serial communications between the Arduino and MATLAB, as illustrated in Figure 4.12.

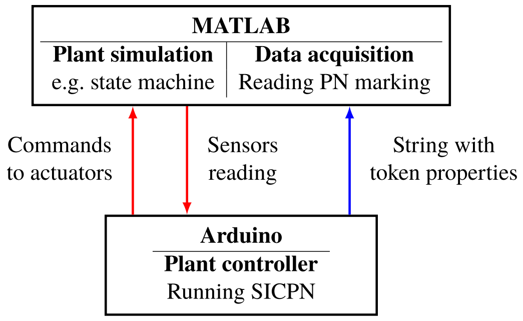


Figure 4.12 – Scheme for MATLAB and Arduino communication.

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, the Arduino scan cycle is set to 1 second, enabling safe bidirectional serial communication.

4.4.1 Modelling the FESTO MPS Testbed Using SFC

To test the 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, the Sequential Function Chart (SFC) language is utilised 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, with each one depicting different stations of the system:

• Figure 4.13 represents the input-output response of the rotating table. Here, the table is modelled as a circular buffer with 6 positions (cf. with Figure 4.1) 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 4.13). 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 4.2.4 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 4.2 is marked, the workpiece in position PI-6 is ejected using an ejection module.

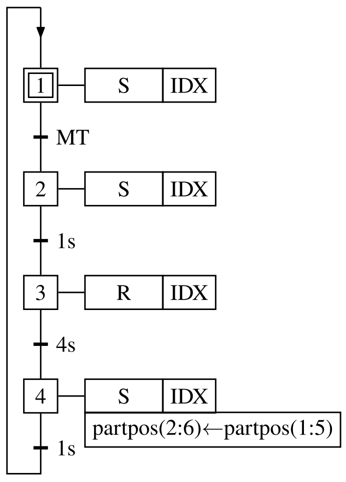


Figure 4.13 – SFC modelling of the plant's rotating table.

• Figure 4.14 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, which lasts 3 seconds, leading to step {3}. Depending on the test results, the SFC transitions to 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 4.15 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 $\{1\}$.

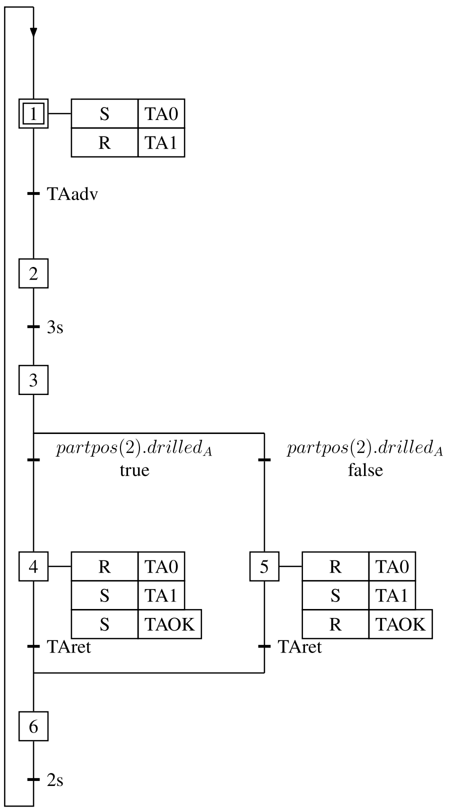


Figure 4.14 – SFC modelling of the Testing A station (resp. Testing B station).

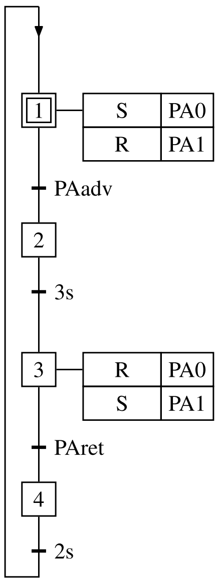


Figure 4.15 – SFC modelling of the Processing A station (resp. Processing B station).

4.4.2 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 4.2. For this purpose, the compiler discussed in section 4.3 has been written in C++.

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

```
 1: System configuration                              ▷ Arduino setup
 2:     Define IO vector
 3:     Begin serial port connection
 4:     Define initial marking $M_0$
 5: while true do                                     ▷ Arduino loop
 6:     SICPN Token-Player (net dynamics)
 7:     Arduino write outputs $O$
 8:     Communication (serial port)
 9:         MATLAB reads $O$ from the Arduino
10:         MATLAB calculates $I$
11:         MATLAB sends back $I$ to Arduino
12:     Arduino reads inputs $I$
13: end while
```

Algorithm 1 – Closed-loop tests' main steps.

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

1. The first test aims to track the operation of a not-drilled metal workpiece among all stations, while verifying the correctness of the operations it undergoes.
2. The second test aims to track the operation 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, this second test will take into account parallel simultaneous operations.

Table 4.4 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 4.4 – 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 4.5 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 4.5 – 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.

4.4.2.1 Acquiring and plotting data

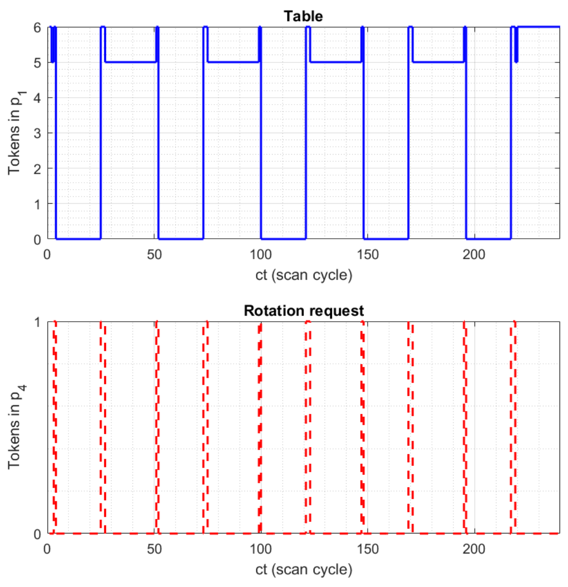
Among the many verification tests we conducted to verify the correct design and implementation of the proposed SICPN controller, two plots are of particular interest for this analysis.

The first is to track the markings of places p_1 and p_4, which relate the condition of a processing operation being completed for a given block, represented as $\boldsymbol{m}_1 = 6 \otimes c_0$, with the condition of *rotation request*, corresponding to $\boldsymbol{m}_4 \geq 1 \otimes c_0$.

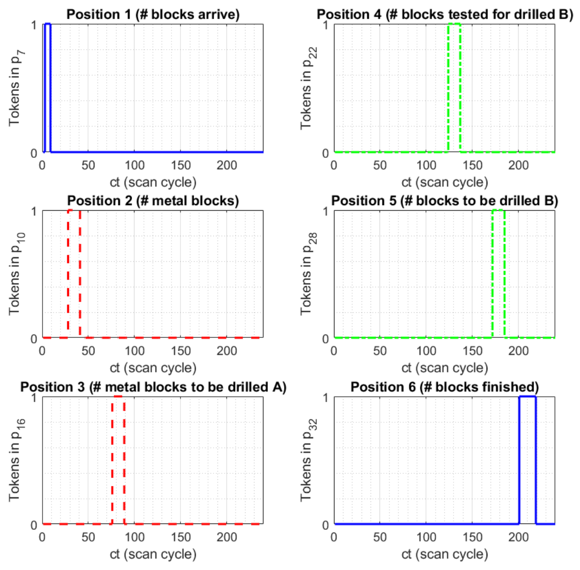
The second is to track the markings of places p_6, p_9, p_{15}, p_{21}, p_{27}, and p_{33}, which indicate when each block enters a new process station.

4.4.2.2 Test 1: One metal block on the table

One can observe the results of Test 1 in Figure 4.16. More precisely, in Figure 4.16a, it shows that the rotation request is synchronized with the table rotation. Furthermore, when the metal block leaves the table, it stands still waiting for the next block to arrive, i.e., $\boldsymbol{m}_1(t) = 6 \otimes c_0$ for all $t > 220$ time units. Figure 4.16b shows that the metal block passes through all positions, as expected.



(a) Table and request rotation

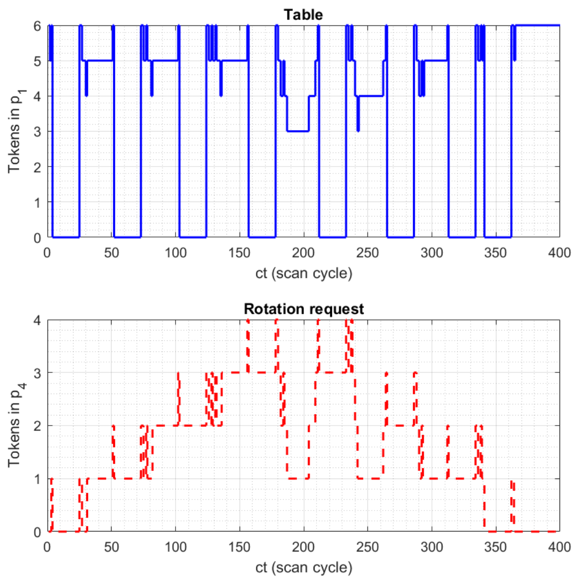


(b) Positions of the block

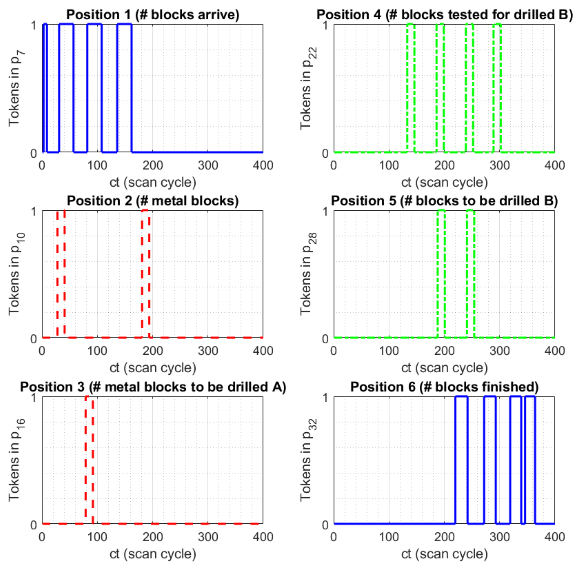
Figure 4.16 – Results of Test 1: One metal block on the table.

4.4.2.3 Test 2: Four different blocks on the table

One can observe the results of Test 2 in Figure 4.17. More precisely, Figure 4.17a 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, $\boldsymbol{m}_4$ 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 4.17b shows which blocks, according to their properties (see Table 4.4), are processed in each position as time increases. By inspection of the expected results illustrated in Table 4.5, it is verified that the plant is behaving correctly under the control of the proposed SICPN feedback control.



(a) Table and request rotation



(b) Positions of the blocks

Figure 4.17 – Results of Test 2: Four heterogeneous blocks on the table.

5 FINAL CONSIDERATIONS

5.1 Conclusion

In this thesis, a new Petri net, Signal-Interpreted Coloured Petri Net (SICPN), is derived by merging CPN and SIPN concepts. Adding colouring properties allows to treat tokens differently depending on the information stored in each colour, whereas adding interpretation signals (inputs and outputs) allows the system to receive external signals and execute commands when implemented in PLCs and microcontrollers.

The objective of this thesis is to create a methodology to use the SICPN framework to describe and model the system in study, then write the program of the modelled system using C-code Arduino language and then implement the (already tested) program in a microcontroller or in the real system.

As a novelty, the plant controller for feedback-control of DES is modelled in SICPN to run in Arduino to support feedback-based decision-making within the SICPN, highlighting the Token-Player structure.

A case study is implemented in a microcontroller using SICPN notation. The efficacy of this formalism is showcased in its application to control a Digital Twin of an extended configuration of the FESTO MPS modular processing station via an Arduino microcontroller and two-way UART serial communications.

The representativeness of this case study is proven by using the SICPN framework to simulate a FESTO MPS plant, which is relevant to show the applicability and reproducibility of the case study. The researchers can use this formalism to extend to other cases by defining the system modelling, its colour and signal-interpreted conditions. A point to highlight is that the implementation of the framework can be done in a Digital Twin simulation or even in real plants whether in PLC or microcontrollers, for educational and industrial purposes and applications.

Thus, the framework developed in this work properly describes the system presented in the case study and can be applied in several other cases, by modelling new systems with the SICPN notation.

5.2 Published paper

During my PhD, I had the opportunity to study a year abroad at University of Cagliari (UniCa), Italy, as an exchange student supervised by professors Dr. Carla Seatzu and Dr. Alessandro Pilloni. Their support and meaningful contributions to my research yielded the improvement of my studies and the publication of the following article:

Borges, M. U., Pilloni, A., Pontes, G. R., Seatzu, C., Lima II, E. J. Signal-Interpreted Coloured Petri Nets: A modelling tool for rapid prototyping in feedback-based control of discrete event systems, Control Engineering Practice, Volume 153, 2024, 106099, ISSN 0967-0661 https://doi.org/10.1016/j.conengprac.2024.106099

5.3 Future work

As future research themes, some relevant topics that should be considered are listed:

- 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. More elaborated examples could be constructed using a SICPN model that considers non-unitary arcs to be implemented in Arduino;
- Advancing the capabilities and applicability of SICPN-based control systems across various domains, thereby propelling the evolution of automation and decision-making processes. For example, the model could be extended in a timed setting in order to solve problems related to performance analysis. This approach consists of simulating and analysing systems' response instead of modelling a controller;
- Regarding more theoretical aspects, efforts to formalise well-known concepts in PNs such as reachability, liveness, observality, controllability, boundedness, determinism, etc., for SICPNs would also be undertaken. This task presents significant challenges since in SICPNs, transitions depend on exogenous inputs, rendering the standard definitions less applicable. Nonetheless, it would be of interest to explore the extension of these concepts from a closed-loop perspective. This involves modelling both the controller and the plant as coupled SICPNs, where the input of one is the output of the other and vice versa. In this sense, and by developing tailored composition rules between the two models, one could strive to formalise some closed-loop counterparts of these properties;
- As one knows, tokens can have different properties represented by colours, which cannot alter these properties within the token. Coloured Petri Nets (CPN) do not support methods execution, unlike Object Petri Nets (OPN). Based on this difference, a new PN - named Signal-Interpreted Object Petri Nets (SIOPN) - can be proposed. Thus, the advantage of defining a SIOPN is executing methods, which can compact even more the model of a system given its complexity.

Bibliography

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. *IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, 798–803. https://doi.org/10.1109/AIM52237.2022.9863367

Awad, H. (2018). Supervisory control systems: Theory and industrial applications. In *Petri nets in science and engineering* (pp. 93–109). IntechOpen. https://doi.org/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*(15), 7161. https://doi.org/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. https://doi.org/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*. https://doi.org/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. https://doi.org/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*(8), 256–261. https://doi.org/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*(4), 302–314. https://doi.org/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. https://doi.org/10.1109/ACCESS.2020.2977423

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems*. Springer. https://doi.org/10.1007/978-0-387-68612-7

Chen, C., & Hu, H. (2020). Extended place-invariant control in automated manufacturing systems using petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. https://doi.org/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*(2), 12–25.

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

David, R. (1995). Grafcet: A powerful tool for specification of logic controllers. *IEEE Transactions on control systems technology*, *3*(3), 253–268. https://doi.org/10.1109/87.406973

de Mello, A. T. F., Barbosa, M. C., dos Santos Filho, D. J., Miyagi, P. E., & Junqueira, F. (2012). A transcription tool from petri net to CLP programming languages. *ABCM Symposium Series in Mechatronics—Vol. 5, Section IV—Industrial Informatics, Discrete and Hybrid Systems*, 781–790.

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*(4), 1–24. https://doi.org/10.1145/3322643

Dotoli, M., Fanti, M. P., Giua, A., & Seatzu, C. (2008). First-order hybrid petri nets. an application to distributed manufacturing systems. *Nonlinear Analysis: Hybrid Systems*, *2*(2), 408–430.

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

Fanti, M. P., Giua, A., & Seatzu, C. (2006). Monitor design for colored petri nets: An application to deadlock prevention in railway networks. *Control engineering practice*, *14*(10), 1231–1247. https://doi.org/10.1016/j.conengprac.2006.02.007

Farah, K., Chabir, K., & Abdelkrim, M. N. (2019). Colored petri nets for modeling of networked control systems. *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 226–230. https://doi.org/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. *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 15–20. https://doi.org/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*(45), 33–41. https://doi.org/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*(June), 2819–2823. https://doi.org/10.1109/ACC.2000.878725

Frey, G. (2002). *Design and formal analysis of petri net based logic controllers* [Doctoral dissertation, 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. *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 01–07. https://doi.org/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. *2019 Winter Simulation Conference (WSC)*, 1519–1533. https://doi.org/10.1109/WSC40007.2019.9004691

Giua, A., & Silva, M. (2018). Petri nets and automatic control: A historical perspective. *Annual Reviews in Control*, *45*, 223–239. https://doi.org/https://doi.org/10.1016/j.arcontrol.2018.04.006

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

Gomes, L., Barros, J. P., Costa, A., & Nunes, R. (2007). The input-output place-transition petri net class and associated tools. *2007 5th IEEE International Conference on Industrial Informatics*, *1*, 509–514. https://doi.org/10.1109/INDIN.2007.4384809

Gratie, D.-E., & Petre, I. (2014). Hiding the combinatorial state space explosion of biomodels through colored petri nets. *Annals of University of Bucharest*, *61*, 23–41.

Grobelna, I., & Karatkevich, A. (2021). Challenges in application of petri nets in manufacturing systems. *Electronics*, *10*(18), 2305. https://doi.org/10.3390/electronics10182305

Grobelna, I., & Szcześniak, P. (2022). Interpreted petri nets applied to autonomous components within electric power systems. *Applied Sciences*, *12*(9), 4772. https://doi.org/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*(11), 2856–2869. https://doi.org/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. https://doi.org/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*(3), 1–37. https://doi.org/10.1145/3449353

IEC. (2019, August). *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.

IEC. (2003, January). *61131-3: Programmable controllers-part 3: Programming languages* (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*(3), 317–336. https://doi.org/10.1016/0304-3975(81)90049-9

Jensen, K. (1987). Coloured petri nets. In *Petri nets: Central models and their properties* (pp. 248–299). Springer. https://doi.org/10.1007/978-3-540-47919-2_10

Jensen, K. (1992). Coloured petri nets, volume 1: Basic concepts. *EATCS Monographs on Theoretical Computer Science. Springer-Verlag*.

Jensen, K., & Kristensen, L. M. (2009). *Coloured petri nets: Modelling and validation of concurrent systems*. Springer Science & Business Media. https://doi.org/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*. https://doi.org/10.1080/09544828.2019.1630803

Klein, S., Frey, G., & Minas, M. (2003). PLC programming with signal interpreted petri nets. *International Conference on Application and Theory of Petri Nets, 2003*, 440–449. https://doi.org/10.1007/3-540-44919-1_27

Lewis, R. W. (1998). *Programming industrial control systems using IEC 1131-3*. 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*. https://doi.org/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*(4), 259–264. https://doi.org/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*(1), 438–450. https://doi.org/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*(2), 990–1002. https://doi.org/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. *Proceedings of the ESREL 2015 Conference*, *7*, 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*(6), 703–717. https://doi.org/10.1007/s40313-018-0411-x

Mathworks. (2024). Cell arrays. *MATLAB help.* www.mathworks.com/help/matlab/cell-arrays.html

Minas, M., & Frey, G. (2002). Visual PLC-programming using signal interpreted petri nets. *Proc. of IEEE American Control Conf.*, *6*, 5019–5024. https://doi.org/10.1109/ACC.2002.1025461

Miyagi, P. E., Hasegawa, K., & Takahashi, K. (1988). A programming language for discrete event production systems based on production flow schema and mark flow graph. *Transactions of the Society of Instrument and Control Engineers*, *24*(2), 183–190. https://doi.org/10.9746/sicetr1965.24.183

Miyagi, P., Camarinha-Matos, L., Santos Filho, D., Barata, J., & Arakaki, J. (1995). The application of enhanced mark flow graph in real time control systems. *IFAC Proceedings Volumes*, *28*(19), 147–154. https://doi.org/10.1016/S1474-6670(17)45073-7

Murata, T. (1989). Petri nets : Properties , analysis and applications. *Proceedings of the IEEE, Vol. 77*(4), 541–580. https://doi.org/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. https://doi.org/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*(12). https://doi.org/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*(3), 1208–1220. https://doi.org/10.1109/TASE.2020.2973996

Petri, C. A. (1962). *Kommunikation mit Automaten* [Doctoral dissertation, Technische Universität Darmstadt].

Petri, C. A. (1980). Introduction to general net theory. In *Net theory and applications* (pp. 1–39). Springer.

Ramadge, P. J., & Wonham, W. M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, *77*(1), 81–98. https://doi.org/10.1109/5.21072

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*(4), 557–565. https://doi.org/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. https://doi.org/10.1016/j.ress.2019.04.004

Silva, J. R., & Miyagi, P. E. (1995). PFS/MFG: A high level net for the modeling of discrete manufacturing systems. *International Conference on Information Technology for Balanced Automation Systems*, 349–362. https://doi.org/10.1007/978-0-387-34910-7_33

Silva, M. (2012). 50 years after the PhD thesis of Carl Adam Petri: A perspective. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, (1976), 13–20. https://doi.org/10.3182/20121003-3-MX-4033.00006

Silva, M. (2018). On the history of discrete event systems. *Annual Reviews in Control*, *45*, 213–222. https://doi.org/10.1016/j.arcontrol.2018.03.004

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*(1), 5–17. https://doi.org/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*(5), 1289–1296. https://doi.org/10.1016/j.net.2019.02.014

Sköldstam, M., Åkesson, K., & Fabian, M. (2007). Modeling of discrete event systems using finite automata with variables. *Proceedings of the IEEE Conference on Decision and Control, 2007*, 3387–3392. https://doi.org/10.1109/CDC.2007.4434894

Vázquez, C. R., Ramírez-Treviño, A., & Silva, M. (2014). Controllability of timed continuous petri nets with uncontrollable transitions. *International Journal of control*, *87*(3), 537–552. https://doi.org/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. https://doi.org/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*. https://doi.org/10.1155/2021/4005877

Xia, C. (2016). Property preservation of refinement for petri net based representation for embedded systems. *Cluster Computing*, *19*(3), 1373–1384. https://doi.org/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*(3), 1466–1473. https://doi.org/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. https://doi.org/10.1007/s00170-017-0800-5

ANNEX

Annex A: Multisets

In this section we recall some notation that will be useful in the following, when formally defining the colored PN model.

Definition .1. *Let D be a set. A* multiset *(resp.,* non negative multiset*) $\boldsymbol{\alpha}$ over D is defined by a mapping $\alpha : D \to \mathbb{Z}$ ($\alpha : D \to \mathbb{N}$) and is represented using a special symbol $\otimes$ as*

$$\boldsymbol{\alpha} = \sum_{d \in D} \alpha(d) \otimes d$$

where the sum is limited to the elements such that $\alpha(d) \neq 0$.

Let $\mathcal{Z}(D)$ (resp., $\mathcal{N}(D)$) denote the set of all multisets (resp., non negative multisets) over D.

The multiset $\boldsymbol{\varepsilon}$ is the empty multiset such that for all $d \in D,\ \varepsilon(d) = 0$. ■

Definition .2. *Given two multisets $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{Z}(D)$ and a number $a \in \mathbb{Z}$:*

- *The sum of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is denoted as $\boldsymbol{\gamma} = \boldsymbol{\alpha} + \boldsymbol{\beta}$ and is defined as $\forall d \in D : \gamma(d) = \alpha(d) + \beta(d)$.*
- *The difference of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is denoted as $\boldsymbol{\gamma} = \boldsymbol{\alpha} - \boldsymbol{\beta}$ and is defined as $\forall d \in D : \gamma(d) = \alpha(d) - \beta(d)$. Note that the difference of two non negative multisets may be negative.*
- *The product of $\boldsymbol{\alpha}$ and a is denoted as $\boldsymbol{\gamma} = a\ \boldsymbol{\alpha}$ and is defined as $\forall d \in D : \gamma(d) = a\ \alpha(d)$.*
- *We write $\boldsymbol{\alpha} \leq \boldsymbol{\beta}$ if $\forall d \in D : \alpha(d) \leq \beta(d)$.*

■

Now, given two sets D and D', let $\boldsymbol{F} : D \to \mathcal{Z}(D')$ be a function that associates to each element $d \in D$ a multiset on D':

$$\boldsymbol{F}(d) = \sum_{d' \in D'} F(d, d') \otimes d' \in \mathcal{Z}(D').$$

We can naturally extend this application to a function $\boldsymbol{F} : \mathcal{Z}(D) \to \mathcal{Z}(D')$ as follows.

Definition .3. *Given two sets D and D', a function $\boldsymbol{F} : D \to \mathcal{Z}(D')$, and a multiset $\boldsymbol{\alpha} \in \mathcal{Z}(D)$, we define*

$$\boldsymbol{F} \circ \boldsymbol{\alpha} \triangleq \sum_{d \in D} \alpha(d) \boldsymbol{F}(d) = \sum_{d \in D} \sum_{d' \in D'} \alpha(d) F(d, d') \otimes d' \in \mathcal{Z}(D')$$

i.e., *using the special symbol $\circ$, the linear combination with coefficients $\alpha(d)$ of the multisets $\boldsymbol{F}(d)$ over D' is denoted $\boldsymbol{F} \circ \boldsymbol{\alpha}$.* ■

A simple example will help to clarify the notation.

Example .1. *Let us consider the two sets $D = \{c_1, c_2\}$ and $D' = \{z_1, z_2, z_3\}$, and the multiset $\boldsymbol{\alpha}$ over D, where $\boldsymbol{\alpha} = 2 \otimes c_1 + 3 \otimes c_2$. Let $\boldsymbol{F}(c_1) = 4 \otimes z_1 + 5 \otimes z_2 + 2 \otimes z_3$ and $\boldsymbol{F}(c_2) = 3 \otimes z_1 + 2 \otimes z_2 + 2 \otimes z_3$ be two multisets over D'. Then, by definition,*

$$\begin{aligned} \boldsymbol{F} \circ \boldsymbol{\alpha} &= \textstyle\sum_{d \in \{c_1, c_2\}} \alpha(d)\boldsymbol{F}(d) \\ &= 2\boldsymbol{F}(c_1) + 3\boldsymbol{F}(c_2) \\ &= (2 \cdot 4 + 3 \cdot 3) \otimes z_1 + (2 \cdot 5 + 3 \cdot 2) \otimes z_2 + (2 \cdot 2 + 3 \cdot 2) \otimes z_3 \\ &= 17 \otimes z_1 + 16 \otimes z_2 + 10 \otimes z_3 \in \mathcal{Z}(D') \end{aligned}$$

■

We finally observe that it is possible to give a matrix representation of multisets and of functions over multisets.

Remark .1. *Given two sets D and D', let us arbitrary order their elements as follows: $D = \{d_1, \ldots, d_k\}$ and $D' = \{d'_1, \ldots, d'_{k'}\}$.*

A multiset $\boldsymbol{\alpha} \in \mathcal{Z}(D)$ can be represented by a vector:

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha(d_1) \\ \alpha(d_2) \\ \vdots \\ \alpha(d_k) \end{bmatrix} \in \mathbb{Z}^k.$$

Thus, given a function $\boldsymbol{F} : D \to \mathcal{Z}(D')$ for all $d \in D$ we can write

$$\boldsymbol{F}(d) = \begin{bmatrix} F(d, d'_1) \\ F(d, d'_2) \\ \vdots \\ F(d, d'_{k'}) \end{bmatrix} \in \mathbb{Z}^{k'}.$$

while its extension $\boldsymbol{F} : \mathcal{Z}(D) \to \mathcal{Z}(D')$ can be represented by the matrix

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}(d_1) & \boldsymbol{F}(d_2) & \ldots & \boldsymbol{F}(d_k) \end{bmatrix} \in \mathbb{Z}^{k' \times k}$$

and finally the multiset $\boldsymbol{F} \circ \boldsymbol{\alpha}$ can be computed with the usual matrix-vector product denoted by $\cdot$, i.e.,

$$\boldsymbol{F} \circ \boldsymbol{\alpha} = \begin{bmatrix} \sum_{i=1}^{k} \alpha(d_i) F(d_i, d'_1) \\ \sum_{i=1}^{k} \alpha(d_i) F(d_i, d'_2) \\ \vdots \\ \sum_{i=1}^{k} \alpha(d_i) F(d_i, d'_{k'}) \end{bmatrix} \in \mathbb{Z}^{k'}.$$

■

Example .2. *Let us go back to the Example .1. We can write*

$$\boldsymbol{F} = \left[\begin{array}{cc} \boldsymbol{F}(c_1) & \boldsymbol{F}(c_2) \end{array}\right] = \begin{array}{cc} \begin{array}{cc} c_1 & c_2 \end{array} & \\ \left[\begin{array}{cc} 4 & 3 \\ 5 & 2 \\ 2 & 2 \end{array}\right] & \begin{array}{c} z_1 \\ z_2 \\ z_3 \end{array} \end{array}$$

and thus

$$\boldsymbol{F} \circ \boldsymbol{\alpha} = \left[\begin{array}{cc} 4 & 3 \\ 5 & 2 \\ 2 & 2 \end{array}\right] \cdot \left[\begin{array}{c} 2 \\ 3 \end{array}\right] = \left[\begin{array}{c} 17 \\ 16 \\ 10 \end{array}\right].$$

■