Towards an agile distributed management system based on Petri Nets

Jose Jean Paul Zanlucchi de Souza Tavares[1,*,†], Jose Reinaldo Silva[2,†]

[1]Manufacturing Automated Planning Lab, Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Brazil

[2]Design Lab, Department of Mechatronics and Mechanical Systems, University of Sao Paulo, Brazil

Abstract

The post-pandemic society demands interactive structures requiring autonomous logistics and manufacturing systems that could be monitored and changed during distributed implementation. Different methods have been used to model requirements satisfying all the target system's demands while providing the best matching to an implementable framework. Two aspects are detached: i) the need to formalize requirements so that the interactive process (systems/user) could be modeled and verified during the requirements phase; ii) an implementable framework that fits the interactive model, adherent to the requirement formalization: a distributed discrete system. The application is directed at fitting logistics and manufacturing process requirements in a digital factory approach with many distributed physical sites and low connections. The development of a Petri Nets Management System (*PNMS*) can integrate artifacts inside the logistic by using Radio Frequency Identification - *RFID* - readers along these different sites, exchanging and storing Petri Net information inside a *RFID* tag, and using Petri Net inside RFID Database (*PNRD*) and/or inverse *PNRD* (*iPNRD*) as a contingent system. This paper presents the proposal of a *PNMS* with two distinct modes: Setup, which deals with a Petri Net Modeling Tools interface - based on *PNML* (Petri Net Markup Language), and a *RFID* interface; and Runtime, standing for real-time activities management and monitoring. Thus, the *PNMS* has an internal data structure integrating *RFID* and Petri Nets. It is adaptive and applicable to assist operational management, including *PNRD/iPNRD* supervision, integrating design and deployment. A case study illustrates the application of *PNRD/iPNRD* Arduino Library Management System (a.k.a. *PALMS*) based on *PNMS* concept.

1. Introduction

The evolution of logistics and manufacturing systems points to distributed and re-configurable modes with agile and precise control. A clear consequence is that such models should be visualized directly or through supervisory or twin systems. Recent events launched after the pandemic accelerated this tendency, motivated by reduced human mobility, minimizing crowding in production and service environments.

Holloway et al. [1] distinguish three methods for discrete event control systems: i) the R-W framework, based on the Supervisory Control System (SCT) theory [2]; ii) the logic

Algorithms Theories for the Analysis of Event Data and Petri Nets for Twin Transition 2023, June 26–27, 2023, Lisbon, Portugal

*Corresponding author.

†These authors contributed equally.

jean.tavares@ufu.br (J. J. P. Z. d. S. Tavares); reinaldo@usp.br (J. R. Silva)

control approach [3]; iii) and the behavior approach [4]. The logic controller approach aims to design a controller defining input-output behavior to achieve the desired behavior for the closed-loop system. This method can be applied to control simple processes, and simulation is necessary to validate the closed-loop behavior. The controlled behavior approach is preferable for designing complex manufacturing systems. It consists of modeling the desired closed-loop system behavior, namely, the joint model of the plant and controller, and then extracting the discrete event controller for implementation. This strategy allows properties of interest such as liveness, boundedness, and reversibility to be guaranteed or even previously analyzed using the desired closed-loop model. Most of these discrete event approaches still separate the control level from plants, making rapid configuration difficult or dealing with that without proper visualization or a delay between analysis and re-configuration. Therefore, there is a space for new approaches relying upon a hybrid schema where control information or preconditions are spread among the production system and the artifacts. The higher-level control is based on an open relationship between a target agent and the environment.

Many applications could benefit from a mixed distributed system environment, but we could mention two main areas where its suitability is straightforward and intuitive: logistics and healthcare. Of course, modern manufacturing - from the Industry 4.0 perspective - is also a target. For instance, logistic Radio Frequency Identification - RFID - tags could be spread in the environment guiding retrieval or insertion of new resources to avoid conflicts, using its internal data as the deadline for use - to perishing products. Such police could be changed without modifying the control of Automated Guided Vehicles or AGVs by changing the dynamic guiding in the RFID tags.

Thus the main contribution of this paper is to present how to manage to disperse information along a non-structured physical framework using Radio Frequency Identification (RFID) and Petri Net. This solution must fulfill these challenges once there is no guarantee of network connection in many places and agents (such as the logistics one) since they can have poor computational infrastructure on a lack of it.

As several initiatives emerged, based on technology representation [5][6], it raised a question on how to model and design hybrid and re-configurable environments with high visualization and agility to change valuable information. Once the target is dispersed, dynamic information and visualization are connected to a workflow, and Petri Nets are a suitable formal representation.

Petri Nets are suitable for modeling parallelism and concurrency. Petri Nets models can be formally verified by property analysis or Model Checking [7] [8] [9], which constitutes an additional advantage for the hybrid system approach. Some studies show how to directly integrate the Petri nets model with process monitoring or IoT (Internet of Things) devices via Client-Server communication [10]. Fig.1 presents a schema showing the different application domains for Petri nets, which justifies its choice to represent hybrid distributed environments based on resource concurrency and parallel activities. The interaction generates a continuous improvement structure: once the Model is updated, all the implementations reflect this change.

Many works relate RFID and Petri Net in different areas: quality management [11], logistic modeling [12], healthcare [13], design of flexible manufacturing cells [14], monitoring and control of assembly and disassembly systems [15], material management [16]. Most of these works propose a low-level connection between Petri Net and RFID, focusing on generating the Petri Net marking after reading tags.

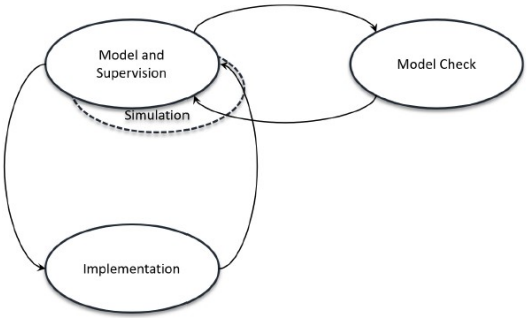


Figure 1: Petri nets application schema.

A question arises about how to fit logistics and manufacturing process requirements, including the dynamic of new revisions, production schedule re-planning, and process updates. Elementary Petri Nets inside RFID Database (a.k.a.*PNRD*) [17] uses RFID tags and readers as a dispersed data structure, based on Petri Nets matrix equation, along with logistics passive agents. For active agents, the inverse *PNRD*, a.k.a. *iPNRD*, changes the Petri nets data structure between RFID readers and tags compared to the original one. With these approaches, RFID tags can store process information trought Petri net data structure, and interact to the RFID reader transmitting its internal data, and allowing RFID reader to calculate the RFID tag next state. Both can provide automatic identification of non-conformities in the expected and stored process. [18] presented the *PNRD* and *iPNRD* integration for the block world domain with three blocks. It showed that the robotic arm could solve some inconsistencies by itself. However, this solution cannot deal with a more complex model, integrating flow with automatic planning. Even in the case which requires automated planning assistance, it is possible to translate the plan in a Petri net as presented by [19]. Regardless of whether the *PNRD* and *iPNRD* can integrate logistics and manufacturing process requirements, there is a lack of management of new versions and process updates.

This paper presents a *PNRD/iPNRD* Management System, supporting the *PNMS* concept. The *PNMS* has two distinct modes: one called *Setup* that deals with the Petri Nets Modeling Tools interface - based on *PNML* (Petri Net Markup Language) [20] [21], and also with *RFID*; and another named *Runtime*, which deals with *PNRD/iPNRD* real-time monitoring and supervising, following state calculus results, including exception identification.

Based on *PNMS* concept, this paper proposes a framework called *PNRD/iPNRD* Arduino® Library Management System - *PALMS*. *PALMS* has two distinct interfaces: the first is related to PN modeling tools based on *PNML* files [1]; the second cares about *Runtime* mode, monitoring the RFID tag according to state calculus results, and providing feedback to PN modeling tools through *PNML* file updating. *PALMS* modifies the original *PNRD/iPNRD* Arduino® library to include a *TCP/IP* communication and memory card access in order to propitiate data exchange with an *MQTT* (Message Queuing Telemetry Transport) Broker.

[1] *PALMS* has an internal data structure that deals with RFID and PN relationship, assisting RFID tag initial marking in *PNRD* format and Arduino® trigger vector programming

The following section reviews the *PNRD/iPNRD*. Section 3 presents *PNMS* concept. A *PALMS* implementation example is shown in section 4. Conclusions are in Section 5, followed by the acknowledgment and references.

2. *PNRD/iPNRD*: the base for a network independent shop-floor distributed communication

An RFID system is composed of tags (usually sticky on objects) and readers. Readers have one or more antennas (up to 4) that emit a signal at a certain frequency. The tag is composed of an integrated circuit coupled to an antenna. When stimulated by the frequency of a reader antenna, the tag responds to the information stored internally. This information is generally unambiguous identification; however, there may be room for additional data. The signal transmitted by the tag is received by the reader's antenna, which decodes it. The reader can be inserted into a communication network.

A Petri net inside RFID Database or *PNRD* is a formal data structure grounded in elementary Petri Nets, which stores Petri Net state equation into RFID tags (marking M_k and adjacent matrix A^t) and RFID reader (trigger vector u_k). An RFID tag could be abstractly related to a single marking on *PNRD*, and each reader can represent a Petri nets transition [18].

PNRD was developed to be a network independent shop-floor distributed communication that automatically identifies and monitors passive agents, such as commercial items, parts, logistics units, and physical products. The target process is associated with the intended behavioral model formally represented by Petri Nets. Operationally, along the physically distributed RFID readers, tag data can represent a single marking. Following the Petri Nets next state calculus, this marking is automatically updated during the tag data capture from the RFID reader. As the PNRD data is previously stored, this calculus and marking updating is independent of network connection. This feature allows the real-time evaluation of whether the next calculated marking is suitable, depending on its results. In this sense, a suitable marking should respect the tag-token bi-univocal unique relationship, which means each tagged object cannot have any negative component in its marking. A marking with a negative component is called *PNRD* exception.

A conflict in the *PNRD* approach arises when the same label, in the *PNRD* context means the same RFID reader/antenna, is related to more than one transition for the same RFID tag identification and previous marking. In this case, it is necessary to apply a decision algorithm to define which transition should be chosen based on additional data or software based on a specific case as presented in [17]. This conflict can also be represented as a stochastic choice.

The iPNRD [18] evolved from the PNRD structure to provide a framework with distributed knowledge information. As the active agents are normally embedded with a microcontroller, it becomes impracticable to associate their identity with only one RFID tag, as it is more reasonable to place a RFID reader in agents similar to mobile robots. In order to fit the *PNRD* approach in the control and monitoring of active agents, it is necessary to swap the storage locations of the state and incidence matrices of the Petri nets. This approach was called inverse *PNRD* or *iPNRD* as presented in [18]. In the original *iPNRD*, the RFID tag stores information about the firing vector u_k, while the adjacent matrix A^t and the current marking M_k are associated with

the RFID Reader/Antenna.

Fig.2 summarizes how the terms of Petri nets state calculus parameters are originally associated with the RFID components in both approaches.

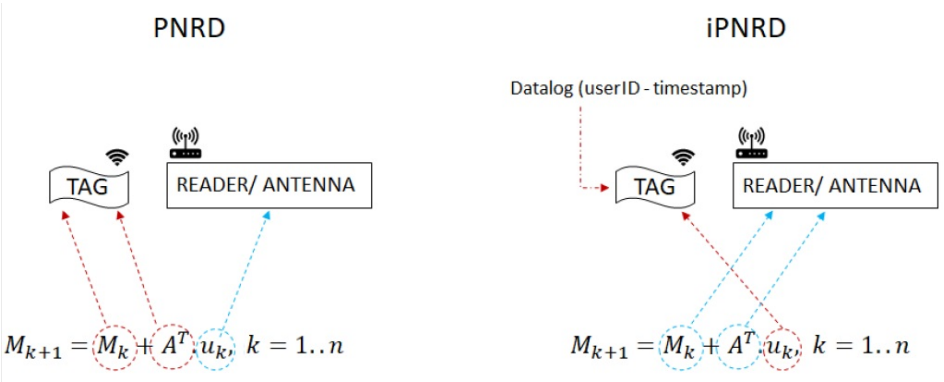


Figure 2: Original Petri nets state equation association with *PNRD/iPNRD*

PNRD can be integrated with *iPNRD* to generate a more autonomous system, as well as, *iPNRD* from distinct active agents can be integrated through smart environment [18].
A *PNRD/iPNRD* Arduino ◦ library [22] assists the generation of stand-alone programs in each device Arduino. This tool applied to the logistic and manufacturing process evokes a question concerning how to manage *PNRD/iPNRD* gadgets in a distributed arrangement with Petri nets modeling tools. This library allows three different types of Petri nets state equation components to be stored in RFID tags and readers. This paper works with the original *PNRD/iPNRD* data structure.
RFID systems can fail, such as RFID tag reading and writing issues, preventing M_k update and generating false exceptions after this point. Despite the fact the RFID tag ID can be integrated with an external Database, in order to structure a contingent system preventing network malfunctioning, a *PNRD/iPNRD* Management System could inform in advance RFID readers of the new marking, that was not stored in the RFID tag, avoiding false exception identification.

3. *PNMS - PNRD/iPNRD* Management System Concept

The main idea of the *PNRD/iPNRD* Management System or *PNMS* is to intermediate PN model with a logistics and manufacturing automated system. In this sense, the *PNMS* receives a PNML (Petri net Markup Language) file from the PN modeling tool; it generates and transfers a *Setup file* to the Implementation, based on *PNRD/iPNRD* approach; it receives M_{k+1} and exception from *PNRD/iPNRD* devices; it generates a *Runtime PNML* file; and it can store *PNRD/iPNRD* history to be integrated with process mining tools optionally. Fig.3 presents the correspondent *PNMS* concept.

As informed before, the *PNMS* has two modes: *Setup* and *Runtime*. The *Setup* mode manages both PN modeling tool and *PNRD/iPNRD* application connection. From the PN model, *PNMS* receives the *PNML* file related to the process to be implemented in *PNRD/iPNRD*. The PNML

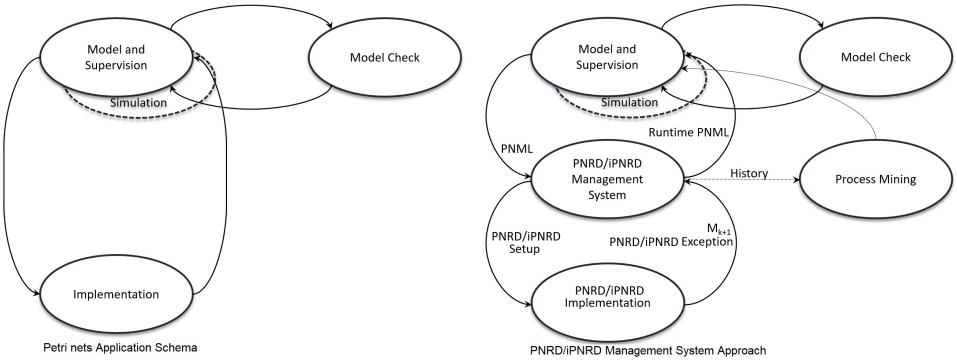


Figure 3: *PNMS* Concept.

File is converted to PN state equation and additional data, depending on the model. *PNMS* manages the connections from PN transitions to each RFID reader, including RFID reader network connection. Another feature of the Setup mode is the generation of a *PNRD/iPNRD Setup* file including RFID tag initial marking and RFID reader trigger vector distribution. On one hand, this file structures *PNRD* tag initial marking and adjacent matrix, and the RFID reader triggers vector data; on the other hand, *iPNRD* Reader initial marking and adjacent matrix, and the RFID tag triggers vector data. *PNMS* transfers these files to the *PNRD/iPNRD* applications dispersed along the logistic and manufacturing process. *PNRD/iPNRD* updating parameters are sent during this data exchange.

The *PNMS Runtime* mode can monitor asynchronously several *PNRD/iPNRD* next state calculus messages containing RFID tag id, new marking vector, transition firing id, exception info, and additional data. If an exception is identified, *PALMS* is warned. As discussed in [18], RFID components can malfunction, and, in this case, a false exception occurs. For instance, if an RFID tag is not able to be written, the new marking is not stored, and all the *PNRD/iPNRD* applications must be informed about actual RFID tag marking. Thus, the *PNMS* has to monitor false exceptions. Another feature is the *PNRD/iPNRD* tag and reader historical database. In the *PNMS Runtime* mode, the RFID tag and marking relationship, and also the RFID reader transition one, is managed. Based on this information, the new marking and the triggered transition are updated in the *Runtime PNML* file to be sent to a PN supervising system. To explain the *PNMS* concept, the next section presents the implementation of *PNRD/iPNRD* Arduino® Library Management System - *PALMS*.

4. *PALMS - PNRD/iPNRD* Arduino ® Library Management System

PALMS or *PNRD/iPNRD* Arduino Library Management System is a middleware able to integrate Petri net modeling tools (through the use of *PNML* exported file) with *PNRD/iPNRD*. The PALMS version proposed in this paper allows managing as many Arduinos Mega® as available through MQTT connection. As presented in Figure 4, PALMS internal structure has two modes, which means, *Setup* and *Runtime*. Although *PALMS* can deal with both *PNRD/iPNRD*, this paper focuses

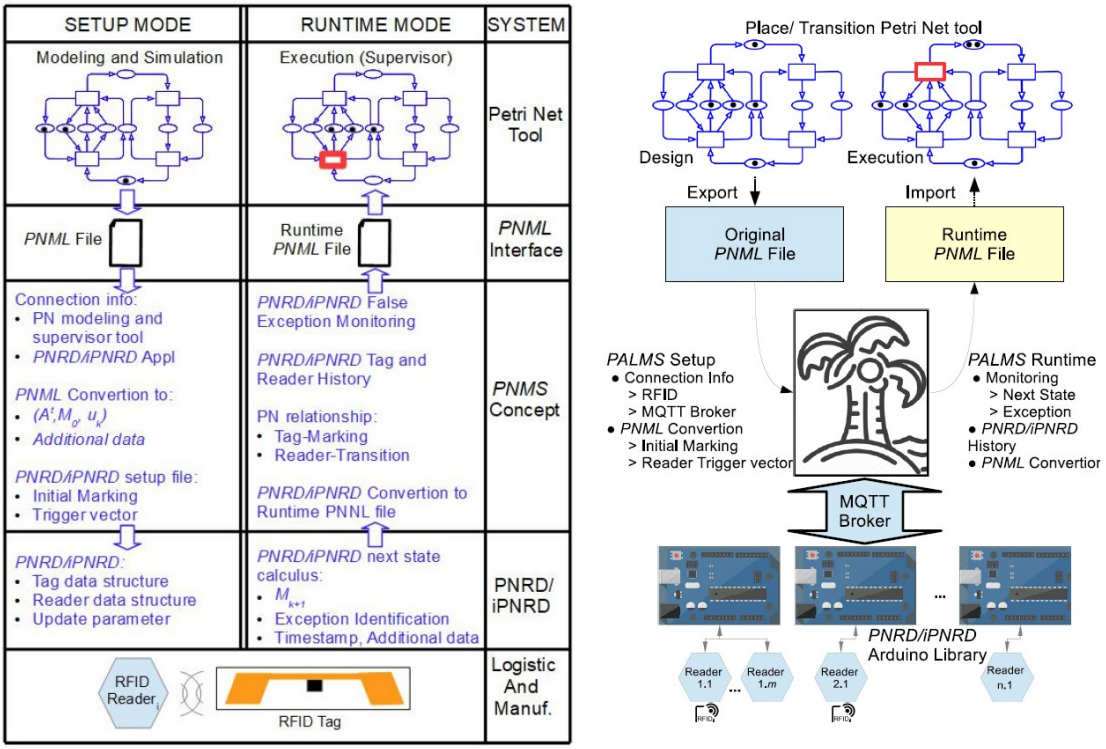


Figure 4: *PNMS* internal schema and *PALMS* General Schema.

on only *PNRD*.

On one hand, as presented by Figure 5 *PALMS Setup* opens a *PNML* file from a PN Modeling tool. This file is converted as *PNRD/iPNRD* data structure and sends it to *PNRD/iPNRD* Application following MQTT subscribe/publish approach, an 1 to *N* relationship. *PALMS Setup* also manages MQTT and Reader connection, and it generates initial marking and RFID Reader trigger vector files. It is necessary to inform the RFID Reader name, its number of antennas (up to 3), and its corresponding Arduino IP information. Each transition is automatically linked with RFID Reader-Antenna following the sequence of the transition in the setup.palms file. If the transition antenna relationship is distinct, the setup.palms file can be manually edited. The *PNRDInfo.pnrd* file is generated for each RFID reader, and the *pnrd-initTag.pnrd* file is created to assist in the initial recording of the RFID tag. In this version of *PALMS* there is no tag initial recording management system, required when the process has more than one RFID tag with different Petri nets initial marking.

On the other hand, *PNRD/iPNRD* Applications send the next state calculus result to *PALMS Runtime* mode which consolidates this information. *PALMS Runtime* monitors each RFID tag's next state and exceptions, it stores *PNRD/iPNRD* history and it generates *Runtime PNML* file to be exported to a PN Supervisory tool (see Figure 6). This mode is still in development and it was implemented only the Setup MQTT one. Another version of *PALMS* exchanging file using *ftp* was already implemented integrating tag history in the *PNML* file. More detail about *PALMS*

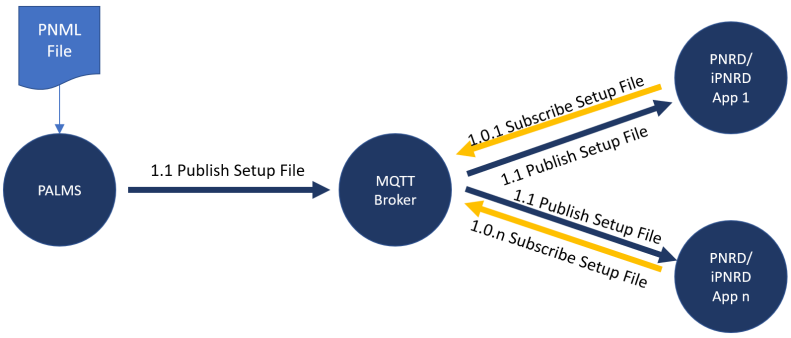


Figure 5: *PALMS* Setup MQTT Schema.

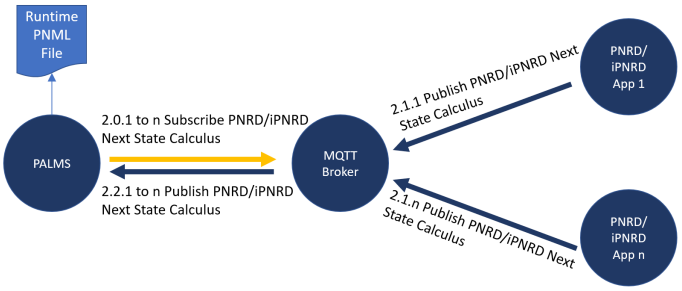


Figure 6: *PALMS* Runtime MQTT Schema.

ftp[2]. Both *ftp* and *MQTT* initiatives can be in use of *PALMS* and follow the *PNMS* concept.

4.1. *PALMS* Implementation

PALMS[3] was implemented in Phyton 3.8. *PALMS* was tested in Linux and Windows OS. Figure 7 presents *PALMS* GUI showing *Setup* mode behind and a window with the *PNML* file or *Setup* file opening options.

PNRD/iPNRD Arduino ® Library had to be updated to open files on an SDCard in order to be able to update the internal settings files. Another change refers to the inclusion of libraries for MQTT communication, which requires that Arduinos ® have Ethernet shield available. PALMS-MQTT assumes setups based on Arduinos complemented with Ethernet shields and SD cards. Each Arduino Mega ® can connect up to three PN532 RFID readers.

4.2. *PALMS Setup* Mode, case study

After *PALMS* installation, the MQTT Broker must be connected.
The first action in *PALMS* is to open a *PNML* file related to the *PNRD/iPNRD* process to be

[2]https://github.com/MAPL-UFU/palms
[3]https://github.com/MAPL-UFU/PALMS-MQTT

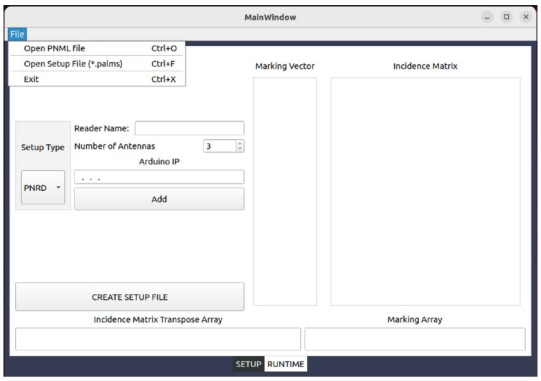


Figure 7: *PALMS* GUI

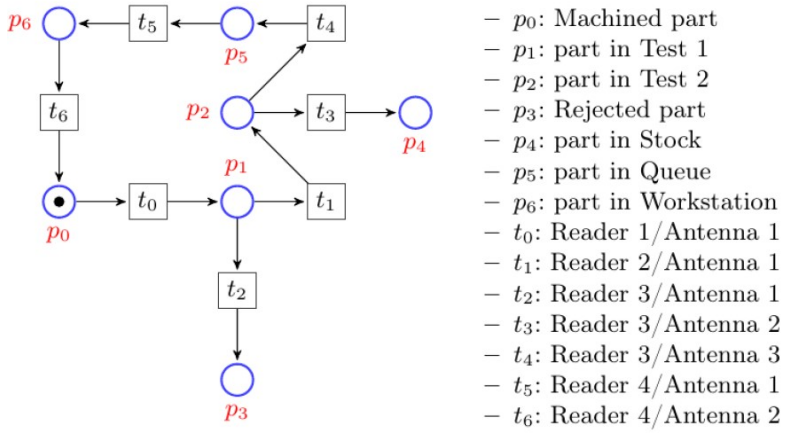


Figure 8: *PNRD* process example.

implemented. This paper focuses on *PNRD* approach because it represents better passive logistic and manufacturing objects.

To demonstrate the practical application of *PALMS*, consider the *PNRD* shown in Figure 8. Schematically this example depicts a machined part test system identified with RFID tags, and some part of the process holding RFID readers. The process in question tests a metal part *MachinedPart* for its cylindricity in place *PartInTest*1. If the cylindricity test rejects the piece then the same goes for recycling *RejectedPart*. If the part is approved, it goes to the roughness test in place *PartInTest*2 and, if it meets the specification, the part is directed to the stock *PartInStock*. If the part does not meet the roughness test, it goes into the reworked queue *PartInQueue*, and through a new machining process *PartInWorkstation* and it returns to the beginning of the test system. It is noticed that the input of each activity is monitored by an exclusive reader/antenna set. For example, the part that enters the cylindricity test system passes through Reader1/Ant1. There is one marking at $p0$, 7 places, and 7 transitions.

After *PALMS* setup loads the *PNML* file, automatically, *PALMS* extracts the Incidence Matrix

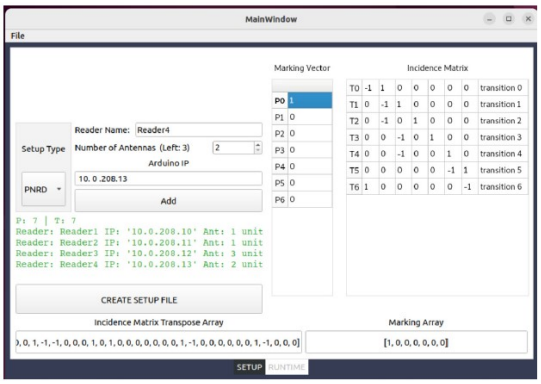


Figure 9: *PALMS* after *PNML* file upload in *PALMS Setup* Mode.

and token vector from it. In *PALMS* GUI, the Incidence Matrix has an additional column where it is possible to inform the correspondent transition label. For each transition, an RFID Reader must be associated. This association is automatically depending on the RFID reader/ antenna manual identification. In this example, Reader 1 and 2 have only one antenna, Reader 3 has 3 antennas, and Reader 4 has 2. Thus Reader1/antenna1 is the transition T0, Reader2/antenna1 is T1, Reader3/antenna1 T2, Reader3/antenna2 T3, and Reader3/antenna3 T4. Figure 9 shows Reader/antenna and transition association, and also Reader IP's info.

After finishing the automatic reader/antenna association with transition, there is a need to push the *CREATE SETUP FILE* button to generate the *PALMS Setup File*. Figure 10 shows this information. It is possible to notice the number of antennas in each Reader, the Reader IP, and the transition and antenna relationship are sequential, which means, that Reader 1 (IP: 10.0.208.10) has one antenna which is associated with "transition 0", Reader 2 (IP:10.0.208.11) has one antenna linked to "transition 1", and so one. This relationship can be manually updated directly in the *PALMS Setup File* and a new version of *PALMS* will assist this more properly.

As mentioned before, the *PALMS Setup* generates initial marking and RFID Reader trigger vector files, and it doesn't check the number of initial markings because it is dependent on the context. However, it is possible to receive an acknowledgment from each tag stored, and another release of *PALMS* will have this functionality.

4.3. *PALMS Runtime* mode, case study

Opening the *setup.palms* file, *PALMS* enables the *Runtime* mode. Figure 11 shows that this mode presents what approach is in use (*PNRD* in this example), the number of Readers, their correspondent antenna numbering, and IP information.

There are three buttons in the *Runtime* mode *TRANSFER PNRD SETUP, GET RUNTIME INFO*, and *GENERATE NEW PNML*.

TRANSFER PNRD SETUP button stores *PNRD* data in each RFID reader, and it is mandatory to be done before the process is running. Two distinct files are generated, one related to *PNRD* initial marking, and another to RFID Reader information. *PALMS* creates the folder *palmsSetup* inside *PNMLexamples*. Figure 12 presents the *palmsSetup* folder for the example. Can be seen

```
"pnmlFile": "/home/jean/PALMS/PALMS-MQTT/PNMLexamples/PNML_example1.pnml",
"qtdReaders": 4,
"readerListConfig": [
    {
        "IP": "10.0.208.10",
        "qtdAntenna": 1,
        "readerName": "Reader1"
    },
    {
        "IP": "10.0.208.11",
        "qtdAntenna": 1,
        "readerName": "Reader2"
    },
    {
        "IP": "10.0.208.12",
        "qtdAntenna": 3,
        "readerName": "Reader3"
    },
    {
        "IP": "10.0.208.13",
        "qtdAntenna": 2,
        "readerName": "Reader4"
    }
],
"transitionNames": [
    "transition 0",
    "transition 1",
    "transition 2",
    "transition 3",
    "transition 4",
    "transition 5",
    "transition 6"
],
```


Figure 10: *PALMS Setup File* for the *PNRD* process example.

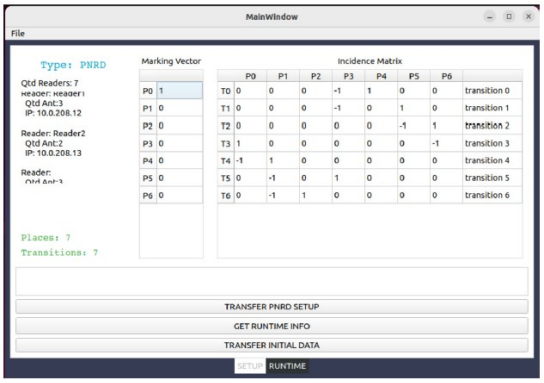
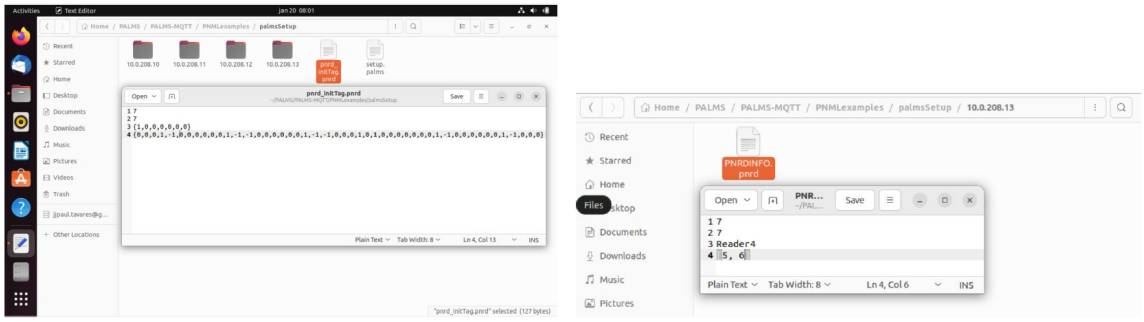


Figure 11: *PALMS Runtime* mode front end.

in four folders related to each RFID Reader IP and two files: *pnrd-initTag.pnrd* and *setup.pnrd*. In front of it *pnrd-initTag.pnrd* is opened with the number of places (7), transitions (7), initial marking vector$(1, 0, 0, 0, 0, 0, 0)$ and the Incidence Matrix in vector format (see Figure 12-a).

As an example of an RFID Reader file shown in Figure 12-b its information to Reader 10.0.208.13 IP (Reader4). This file stores the number of places (7), transitions (7), Reader name, and trigger vector ($[5, 6]$), meaning transitions 5 and 6. Each *PNRDInfo.pnrd* file is sent to the correspondent Arduino ®, and stored in the SD card. As consequence, as already informed, the original *PNRD/iPNRD* Arduino® Library Management System internal structure was updated to read this information. *PALMS* has two Arduino ® file example with this approach inside *startupArduinoFiles* folder, this means, *pnrd-initTag.ino* to initial marking, and *pnrd-reader,ino* to RFID Readers.

To send all these files, all Arduino's® Mega and their respective RFID Readers/Antennas must be physically connected and integrated with an Ethernet network. For example, Figure 13 shows



(a) Transfer *PNRD* Setup file. (b) *PNRD* Info file.

Figure 12: *PALMS* internal Files.

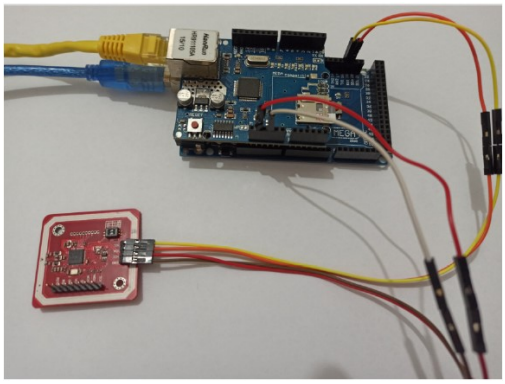


Figure 13: Bench with one Arduino Mega ® with one PN532 RFID readers example.

one Arduino® Mega with Ethernet Shield, SD card, and one PN532 RFID reader connected.

As informed before, in the *MQTT PALMS* version the *GET RUNTIME INFO* and *GENERATE NEW PNML* are still in development. The *FTP PALMS* version implemented the *GET RUNTIME INFO*, called *CONNECT* button. A sequence example shows Reader0, Reader1, and Reader0 transaction trigger for RFID tagId $80fa2243$. The first and second triggers had no error and they change the marking vector from $1,0,0,0,0,0,0$ to $0,1,0,0,0,0,0$ and, after it, to $0,0,1,0,0,0,0$. The last trigger points out an exception. Exception treatment is not implemented yet. Figure 14 shows this sequence of transition triggers.

The *GENERATE NEW PNML* button creates an updated *PNML* file for Petri net supervision. This feature is still under development.

4.4. Other PN Examples

Figure 15 shows a Petri net model with parallelism. The *PALMS Setup Mode* of the correspondent *PNML* file is at Figure 16. It can be noticed that each transition has a different label in the Incidence Matrix's last column.

Figure 17 shows a PN with only one transition which produces and destroys markings of

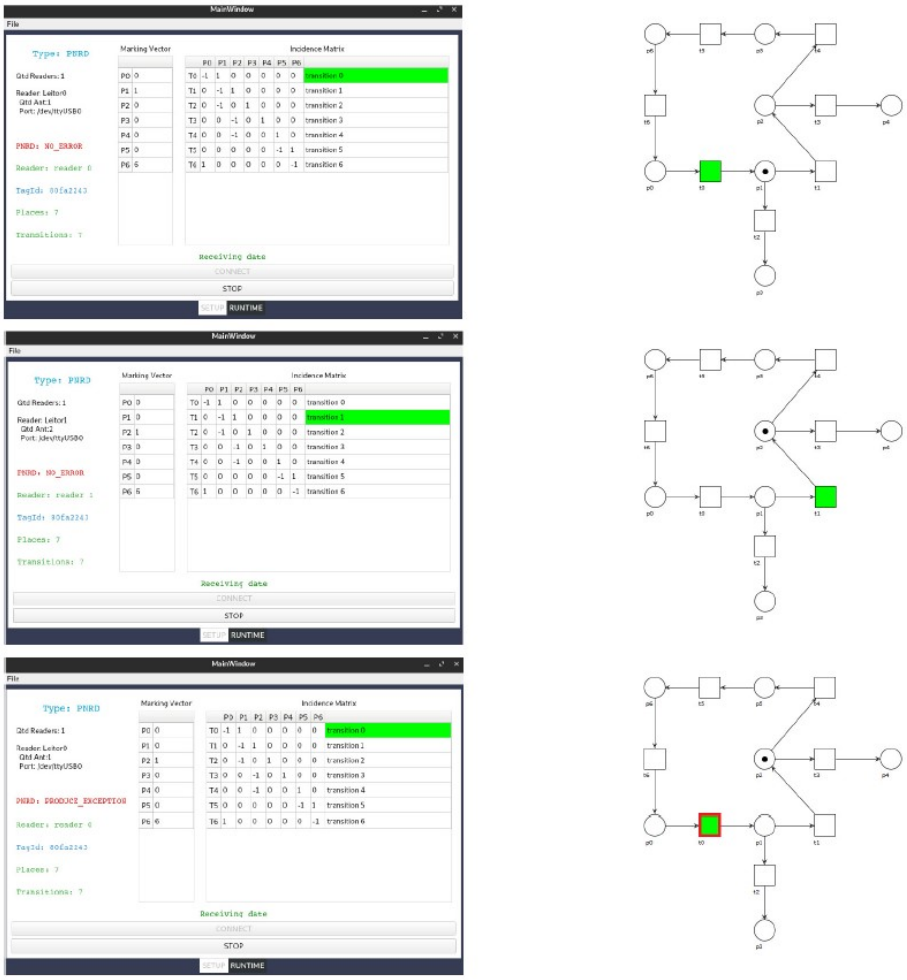


Figure 14: *PALMS Runtime* Experiment of Sequential Triggering.

a single place with 78 markings and its correspondent *PALMS Setup Mode*. It is possible to introduce the label rule as *'a'if fReader*1;*' b'if fReader*2 meaning that *Reader1* and *Reader2* are linked to the same RFID reader/antenna and each one has a different label. Another feature can be viewed in the Incidence Matrix which presents -1/1 inside to represent the effect of this transition. Figure 18 shows the correspondent *PALMS Runtime Mode*. This example can be applied in an unknown process with RFID devices along it. Label histories can assist a PN model generation through process mining techniques [13].

5. Conclusion

In this paper, we presented the proposition of a new framework to manage distributed information related to the control and automation of discrete systems based on the concept of *PNRD/iPNRD* Management System - *PNMS*, integrating logistics and manufacturing processes

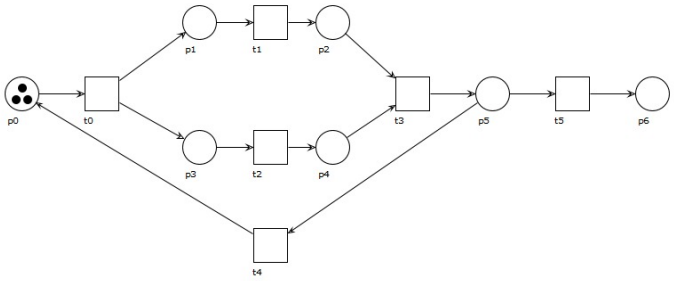


Figure 15: PN process example with parallelism.

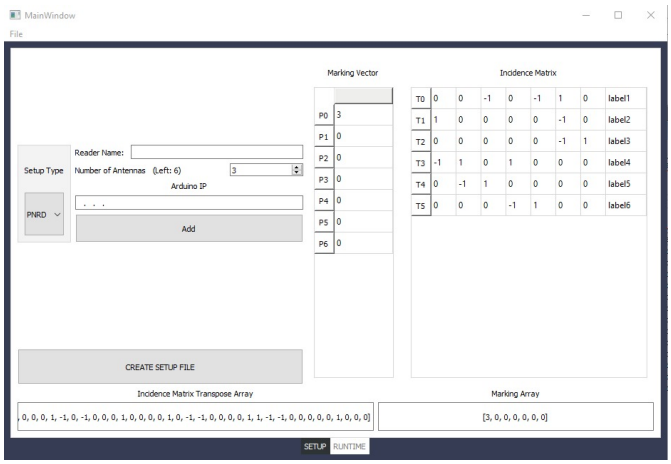


Figure 16: *PALMS Setup* for the PN process example with parallelism.

into a Petri Nets model. The *PNMS* concept requires RFID readers disposed into a local network connection. On the one hand, *PNRD/iPNRD* is network independent, and any failure in network connection cannot affect the shop floor implementation. Once the connection is restored, all data are updated in the Petri Net modeling/ supervising tool. This framework attends to logistics and manufacturing process requirements, dealing automatically with its dynamics, with include new versions of the production process, production schedule re-planning, and process updates. A software tool called *PALMS* (*PNRD/iPNRD* Arduino® Library Management System) is in development to provide practical use of the *PNMS* concept. *PALMS* software connects several Arduino® devices through MQTT communication, and it integrates model and execution grounded in *PNML* data sharing. This integration can generate a more robust and autonomous logistic and manufacturing process, reaching their requisites.

As can be viewed in [23], for 80 different logistics applications, 44 applied High-Level Petri nets (HLPN). Thus the current *PALMS* version can evolve to use HLPN, where real-time data *PNRD/iPNRD* can still be updated in Arduino.

Using *PALMS*, a *PNML* file can be updated depending on the tag's following state result during RFID data capture. A new feature can be implemented in PN tools related to the execution

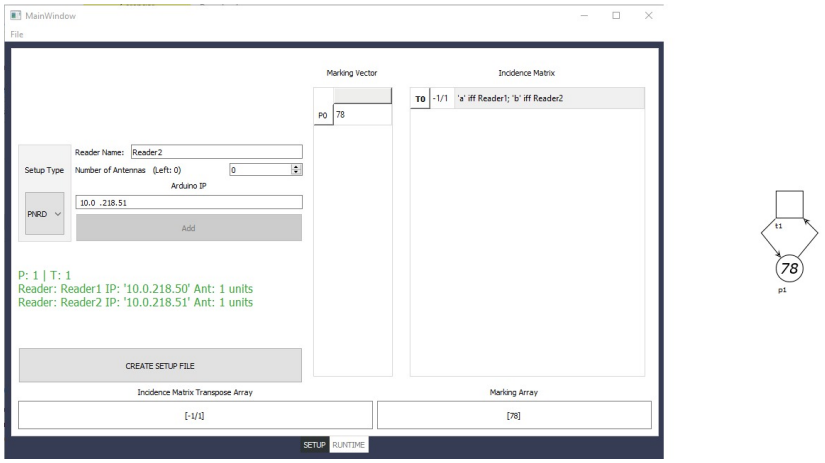


Figure 17: *PALMS Setup* for the PN with only one transition which produces and destroys markings of a single place.

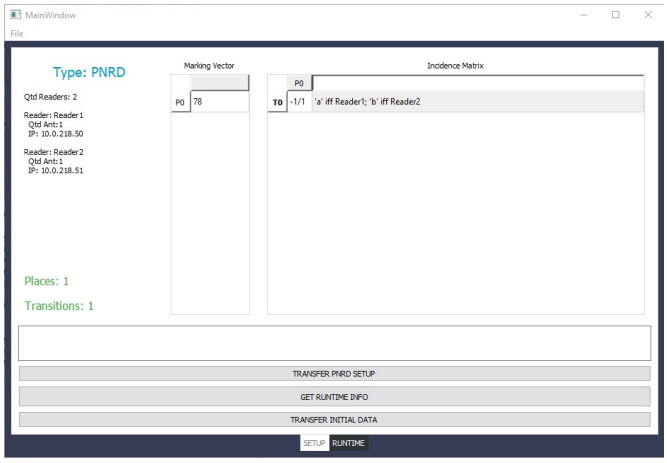


Figure 18: *PALMS Runtime* for the PN with only one transition which produces and destroys markings of a single place.

system, monitoring and supervising the physical system.
A brief case study for *PALMS*'s version was included, with *PNML* examples, to implement a *Runtime PNML* file, adding timestamps and historical *PNRD/iPNRD* database and an Adjacent matrix in GUI.
New developments are being planned to improve the framework, especially the relationship between high-level design and the implementation of communication in production environments and to fit IoT. The *PALMS* development is still in process, and new features will be added to fit *PNMS* concept. The next *PALMS* version will present two Adjacent Matrix instead of only one Incidence Matrix, it will have *MQTT PALMS RUNTIME* mode implemented, and the *PNMS* will be formalized using High-Level PN.

6. Acknowledgement

This work is supported by CNPq, CAPES, FAPEMIG, and UFU. We special thanks Roger Henrique Carrijo de Paula, Thiago Souza Alves and Daniel Barbosa Pereira. This paper was partially supported by MEC- EMENDA 40640016.

References

[1] L. Holloway, B. Krogh, A. Giua, A survey of petri net methods for controlled discrete event systems, Discrete Event Dynamic Systems: Theory and Applications 7 (1997) 151–190.

[2] P. Ramadge, W. Wonham, Supervisory control of a class of discrete event processes, SIAM Journal on Control and Optimization 25 (1987) 206–230. doi:https://doi.org/10.1137/0325013.

[3] M. Zhou, F. DiCesare, D. Rudolph, Design and implementation of a petri net based supervisor for a flexible manufacturing system, Automatica 28 (1992) 1199–1208.

[4] I. Suzuki, T. Murata, A method for stepwise refinement and abstraction of petri nets, Journal of Computer and System Sciences 27 (1983) 51–76. doi:https://doi.org/10.1016/0022-0000(83)90029-6.

[5] R. Brennan, P. Vrba, P. Tichy, A. Zoitl, C. Snder, T. Strasser, V. Marik, Developments in dynamic and intelligent reconfiguration of industrial automation, Computers in Industry 59 (2008) 533–547. doi:https://doi.org/10.1016/j.compind.2008.02.001.

[6] G. Doukas, K. Thramboulidis, A real-time-linux-based framework for model-driven engineering in control and automation, IEEE Transactions on Industrial Electronics 58 (2011) 914–924. doi:10.1109/TIE.2009.2029584.

[7] K. Wolf, Petri net model checking with lola 2, in: V. Khomenko, O. H. Roux (Eds.), Application and Theory of Petri Nets and Concurrency, Springer International Publishing, Cham, 2018, pp. 351–362.

[8] L. M. Hillah, F. Kordon, Petri nets repository: A tool to benchmark and debug petri net tools, in: W. van der Aalst, E. Best (Eds.), Application and Theory of Petri Nets and Concurrency, Springer International Publishing, Cham, 2017, pp. 125–135.

[9] F. Kordon, L. Hillah, F. Hulin-Hubard, L. Jezequel, E. Paviot-Adet, Study of the efficiency of model checking techniques using results of the mcc from 2015 to 2019, Int J Softw Tools Technol Transfer 23 (2021) 931–952. doi:https://doi.org/10.1007/s10009-021-00615-1.

[10] D. Mourtzis, N. Milas, A. Vlachou, An internet of things-based monitoring system for shop-floor control, Journal of Computing and Information Science in Engineering 18 (2018). doi:10.1115/1.4039429.

[11] Y. Lv, C. Lee, H. Chan, W. Ip, Rfid-based colored petri net applied for quality monitoring in manufacturing system, Int J Adv Manuf Technol 60 (2012) 225–236. doi:https://doi.org/10.1007/s00170-011-3568-z.

[12] Y. Li, A. Oberweis, H. Zhang, Modelling and facilitating rfid-based collaborative logistics processes, International Journal of Organizational Design and Engineering 2 (2012) 85–105. doi:https://doi.org/10.1504/IJODE.2012.045907.

[13] C. Fernandez-Llatas, A. Lizondo, E. Monton, J.-M. Benedi, V. Traver, Process mining

methodology for health process tracking using real-time indoor location systems, Sensors 15 (2015) 29821–29840. URL: https://www.mdpi.com/1424-8220/15/12/29769. doi:10.3390/s151229769.

[14] K.-Y. Chen, Cell controller design for rfid based flexible manufacturing systems, International Journal of Computer Integrated Manufacturing 25 (2012) 35–50. doi:10.1080/0951192X.2010.523845.

[15] H. Sun, Z. Chang, R. Mo, Monitoring and controlling the complex product assembly executive process via mobile agent and rfid tags, Assembly Automation 29 (2009) 263–271. doi:10.1108/01445150910972949.

[16] E. Ngai, K. K. Moon, F. J. Riggins, C. Y. Yi, Rfid research: An academic literature review (1995–2005) and future research directions, International Journal of Production Economics 112 (2008) 510–520. doi:https://doi.org/10.1016/j.ijpe.2007.05.004, special Section on RFID: Technology, Applications, and Impact on Business Operations.

[17] J. J.-P. Z. d. S. Tavares, T. A. Saraiva, Elementary petri net inside rfid distributed database (pnrd), International Journal of Production Research 48 (2010) 2563–2582. doi:10.1080/00207540903564934.

[18] J. J.-P. Z. d. S. Tavares, G. D. A. Souza, PNRD and iPNRD integration assisting adaptive control in a block world domain, in: D. Moldt, E. Kindler, M. Wimmer (Eds.), Petri Nets and Software Engineering. International Workshop, PNSE'19, Aachen, Germany, June 24, 2019. Proceedings, volume 2424 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 73–92. URL: http://CEUR-WS.org/Vol-2424.

[19] J. R. Silva, J. M. Silva, T. S. Vaquero, Formal Knowledge Engineering for Planning: Pre and Post-Design Analysis, Springer International Publishing, Cham, 2020, pp. 47–65. doi:10.1007/978-3-030-38561-3_3.

[20] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, M. Weber, The petri net markup language: Concepts, technology, and tools, in: W. M. P. van der Aalst, E. Best (Eds.), Applications and Theory of Petri Nets 2003, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 483–505.

[21] E. Kindler, The ePNK: A generic PNML tool Users' and Developers' Guide for Version 1.0.0., Technical Report 2012-14, Technical University of Denmark, 2012.

[22] C. E. A. da Silva, J. J.-P. Z. d. S. Tavares, M. V. M. Ferreira, Arduino library developed for petri net inserted into rfid database and variants, in: V. Khomenko, O. H. Roux (Eds.), Application and Theory of Petri Nets and Concurrency, Springer International Publishing, Cham], pages=396–405, isbn=978-3-319-91268-4, 2018.

[23] G. Cavone, M. Dotoli, C. Seatzu, A survey on petri net models for freight logistics and transportation systems, IEEE Transactions on Intelligent Transportation Systems 19 (2018) 1795–1813. doi:10.1109/TITS.2017.2737788.