applied sciences

MDPI

Article

New Software Tool for Modeling and Control of Discrete-Event and Hybrid Systems Using Timed Interpreted Petri Nets

Erik Kučera [1,*], Oto Haffner [1], Peter Drahoš [1], Ján Cigánek [1], Roman Leskovský [1] and Juraj Štefanovič [2]

[1] Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, 812 19 Bratislava, Slovakia; oto.haffner@stuba.sk (O.H.); peter.drahos@stuba.sk (P.D.); jan.ciganek@stuba.sk (J.C.); roman.leskovsky@stuba.sk (R.L.)

[2] Faculty of Informatics, Pan-European University, 821 02 Bratislava, Slovakia; juraj.stefanovic@paneurouni.com

* Correspondence: erik.kucera@stuba.sk

Received: 3 July 2020; Accepted: 21 July 2020; Published: 22 July 2020

Abstract: For the development of modern complex production processes in Industry 4.0, it is appropriate to effectively use advanced mathematical models based on Petri nets. Due to their versatility in modeling discrete-event systems, Petri nets are an important support in creating new platforms for digitized production systems. The main aim of the proposed article is to design a new software tool for modeling and control of discrete-event and hybrid systems using Arduino and similar microcontrollers. To accomplish these tasks, a new tool called PN2ARDUINO based on Petri nets is proposed able to communicate with the microcontroller. Communication with the microcontroller is based on the modified Firmata protocol hence, the control algorithm can be implemented on all microcontrollers that support this type of protocol. The developed software tool was successfully verified in control of laboratory systems. In addition, it can be used for education and research purposes as it offers a graphical environment for designing control algorithms for hybrid and mainly discrete-event systems. The proposed software tool can improve education and practice in cyber-physical systems (Industry 4.0).

Keywords: discrete-event systems; hybrid systems; cyber-physical systems; system control; microcontroller; Firmata protocol

1. Introduction

Development of cyber-physical systems is a complex discipline that includes many activities, e.g., system design, specification of required properties, implementation, testing and further development of the system [1]. These operations are important for the final product, therefore it is necessary to create a model of the system first [2,3]. Development of control methods for discrete-event and hybrid systems belongs to modern trends in automation and mechatronics. Hybrid systems in manufacturing processes combine time-driven and event-driven in automotive industry. Control of such systems is challenging due to the necessity to apply control methods both for discrete-event systems (where the Petri nets formalism can be helpful) and standard continuous-time control methods (e.g., PID (proportional-integral-derivative) algorithms) [4]. With an appropriate software development methodology, and software and hardware modules, these control methods can be synergistically combined to yield a proper and unique control system that allows joining discrete-event and continuous-time control methods for an effective hybrid systems control. This is useful in systems which require using different control algorithms depending on the state of the system—for example

proportional-integral-derivative (PID) controllers with different parameters. Using the Petri nets concept, these control rules can be managed in a very efficient, robust and visual way. In this article, new Petri net tools for modeling and control of discrete-event and hybrid systems are developed. Case studies dealing with control of a laboratory fire alarm system and a DC (Direct Current) motor are included.

During the research, it was necessary to look for projects and papers on modeling and control of discrete-event or hybrid systems using high-level Petri nets. An important point was to find out whether the existing research projects dealt just with the theory, or some open-source software tools to support modeling and control using high-level Petri nets were used or directly developed. For practical outcomes of the presented research it would be beneficial to find the latter type of research projects.

In papers [5,6] authors deal with usage of hybrid and color Petri nets for modeling traffic on crossroads and on highways. From these authors, there are also interesting projects in the field of manufacturing systems [7,8]. Unfortunately, it is not mentioned whether the results are only theoretical models, or they have been simulated using a SW (software) tool, or deployed in practice.

An interesting software tool named Visual Object Net++ that supports hybrid Petri nets was developed in [9]. There are a lot of papers mainly from an author of [10,11] describing capabilities of Visual Object Net++. However, this tool is not open-source and has not been further developed.

A SW tool Snoopy [12] offers modeling based on many Petri nets classes like stochastic, hybrid, color, music Petri nets, etc. Using this tool, many types of research in biology and chemistry are being solved. Unfortunately, the source code is not available.

Colored Petri nets are used for modeling of automated storage and retrieval systems in [13,14].

An interesting open-source Matlab tool for modeling with timed discrete, continuous and hybrid Petri nets is HYPENS [15]; however, it is no longer available for download. As described therein, a Petri net model in HYPENS can be defined only using matrices, graphical representation is not supported. This fact eliminates one of the advantages of Petri nets, support of newer versions of Matlab is questionable as well.

GPenSIM is a tool for modeling and simulation of discrete-event systems [16]. It offers many options for simulation and analysis of Petri net models. A big advantage is the support of timed Petri nets [16,17]. In the paper [18], GPenSIM is used for modeling of a flexible manufacturing system but not for its control. As in the previous case, the Petri net is represented only by matrices and cannot be defined graphically.

One of interesting research approaches is the Modelica language and the OpenModelica open-source tool. There is a library that supports modeling by Petri nets in this tool. One of the advantages of OpenModelica is that a Petri Net model can be connected with other Modelica components. The first Petri net toolbox was introduced in [19], its extension is described in [20]; an important addition (called PNlib) including a support of extended hybrid Petri nets for modelling of processes in biological organisms is described in [21,22]. However, this tool was developed primarily for the commercial tool Dymola and not for OpenModelica, so its extensibility and applicability in scientific research are limited. In 2015, the team that developed PNlib published a modified version of PNlib that partially worked in OpenModelica. Unfortunately, it was not possible to use OpenModelica for control purposes using microcontrollers because the communication (COM) port communication support was missing.

The above survey has shown that there is a lack of tools based on Petri net formalism to support control of real systems using a hardware control unit. In the presented article, an original software tool based on microcontrollers was developed for control of discrete-event and hybrid systems using Petri nets formalism.

2. Materials and Methods

As a basis for the newly developed SW tool, the open-source PNEditor was chosen [23]. In this editor it is possible to model systems using basic Petri nets. The software application is implemented

in Java. It is released under an open-source license. Its main advantage is well-structured code, clear design and direct support from developers. It also contains the possibility of determining the boundedness of the Petri net and it can inspect P (place) invariants and T (transition) invariants. The disadvantage is that it supports only basic Petri nets. As part of the solution of our research, it was necessary to add a support for timed Petri nets interpreted for control.

The developed extension of this tool is called PN2ARDUINO and was fully tested in [24,25]. The main aim of the proposed article is to present the developed software for control of discrete-event and hybrid systems verified on laboratory discrete-event and hybrid systems.

There are several Petri net-based control concepts. Petri net as a control logic must be connected with the controlled system (e.g., using a microcontroller). One of the main aspects of the control system design is the question whether the Petri net's logic should be stored in the microcontroller or in the personal computer (PC) able to communicate with the microcontroller. Both approaches have their advantages and disadvantages.

If the Petri net's logic is stored in the microcontroller, the main advantage is the control unit independence from the software application (program on a PC). The Petri net logic is modeled using a PC, and then the Petri net is translated into a program code which is loaded into the microcontroller. Afterwards, the PC and the microcontroller can be disconnected. Another advantage is the capability of real-time control. Disadvantages include limited computational and memory resources of the microcontroller, need for repeated program compiling and its uploading into the microcontroller (mainly during the development phase). The proposed solution is shown in Figure 1.

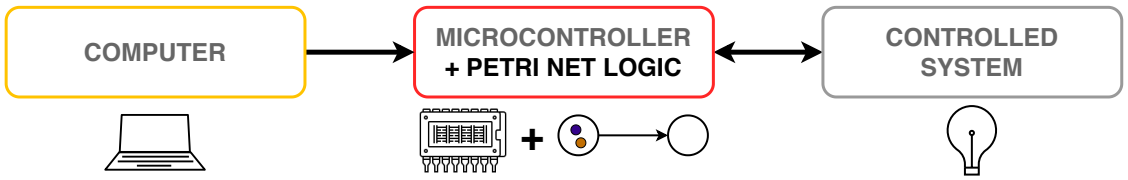

Figure 1. Basic scheme of proposed solution—Petri net's logic in microcontroller.

When the Petri net's control logic is stored on a PC in a specialized SW application, it is possible to control the system directly. In the microcontroller, only the program with communication protocol is stored. This communication protocol (in our case Firmata [26]) is used for communication between the PC and the microcontroller. This solution eliminates the necessity of recompiling and reuploading the program during the development. The next advantage is the elimination of restrictions on computing and storage resources because a PC has almost unlimited resources compared with a microcontroller. One of the disadvantages is that the control system cannot respond in real time. The proposed solution is shown in Figure 2. These differences are specified in Table 1.

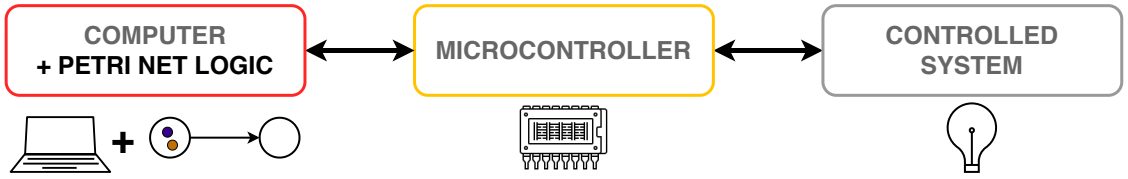

Figure 2. Basic scheme of proposed solution—Petri net's logic in personal computer (PC).

Table 1. Comparison of two concepts of system control using Petri nets.

Petri Net Logic in PC	Petri Net Logic in Microcontroller
limited capability of real-time control	real-time control
much more computation and memory resources available	limited computation and memory resources
code in microcontroller does not need recompiling	during development repeated compiling is needed
PC must be still online	independence of control unit

New software module PN2ARDUINO is based on the second approach (Figure 2) so the Petri net runs on a personal computer. For communication between SW application and microcontroller, the Firmata protocol [26] has been used. Firmata is a protocol designed for communication between a microcontroller and a computer (or a mobile device like a smartphone, tablet, etc.). It is based on MIDI messages [27]. MIDI (musical instrument digital interface) message is made up of 8-bit status byte which is generally followed by 1 or 2 data bytes. Firmata protocol can be implemented in firmware of various microcontrollers; mostly Arduino-family microcontrollers are used. On the PC, a client library is needed. These libraries are available for many languages like Java, Python, .NET, PHP, etc.

On the Arduino side, the Standard Firmata 2.3.2 version is used, the client application on the PC is based on Firmata4j 2.3.3 library which is programmed in Java. The advantage of using Firmata consists of the possibility of using another microcontroller compatible with Firmata.

PN2ARDUINO extends PNEditor with many features. For Petri nets modeling, there is a possibility of adding time delays to transitions, and capacity for places. Also, automatic mode of transitions firing was added for automatic control purposes as only manual mode is available in PNEditor.

In PN2ARDUINO, a new module is added to PNEditor to enable communication with the compatible microcontroller. This module consists of two parts. The first part establishes connection with the microcontroller by setting the COM port where the microcontroller is connected. The second part provides a capability of adding Arduino components to Petri net places and transitions. The following types of Arduino components are supported: digital input and output, analog input, servo control, PWM (pulse-width modulation) output, message sending, and custom SYSEX message [26] sending. Originally, SYSEX (System Exclusive) messages are model-specific messages for setting various parameters of a MIDI device. The idea for SYSEX is to have a second command space using the first byte after the SYSEX Start byte. The key difference is that the data can be of any size, rather than just 1 or 2 bytes for standard MIDI messages.

The use-case diagram of the developed SW tool is depicted in Figure 3, and the class diagram is shown in Figure 4.

As stated, transitions and places can be associated with Arduino components. Digital and analog inputs serve as enabling conditions for Petri net transitions. Digital and PWM outputs and messages are used as executors of the respective actions.

The interesting functionality is the capability of sending custom SYSEX messages. The user must enter a SYSEX command (`0x00-0x0F`) and optionally also the content of the message. The message is sent when the token comes to the place or when the transition is fired. SYSEX messages have been used e.g., in the proposed hybrid control example in the last section of this article. Here, the SYSEX message notifies the microcontroller that a different PID control algorithm is to be used. Then the PID algorithm is switched, and the controlled system remains stable.

A main window of PN2ARDUINO consists of a quick menu, main menu, canvas for Petri net modeling and log console. PN2ARDUINO supports two modes—a design mode and a control mode, the control mode can be manual or automatic.

First, it is necessary to initialize communication with Arduino (`Setup board` in the menu). Then it is possible to add Arduino component to the place or to the transition (Figure 5). The example of analog input is shown in Figure 6.

Time policy is also supported—it is possible to add time delay to the transitions which can be deterministic or stochastic. Using a deterministic time policy, the delay is entered by user during Petri net modeling as a specific integer that represents the duration of the delay. After the specified delay, the transition is fired. The stochastic time policy is based on entering two time frames. The first represents the shortest possible delay and the second represents the longest possible delay. These time frames are also entered directly by user during modeling process. The specific value of the delay is determined randomly in the range of this entered interval. During enabling a transition that has a defined delay, tokens from all entry places are consumed first. Subsequently, a timer is started. After the specified delay, tokens are generated to all exit places.

Figure 3. PN2ARDUINO—Use-case diagram.

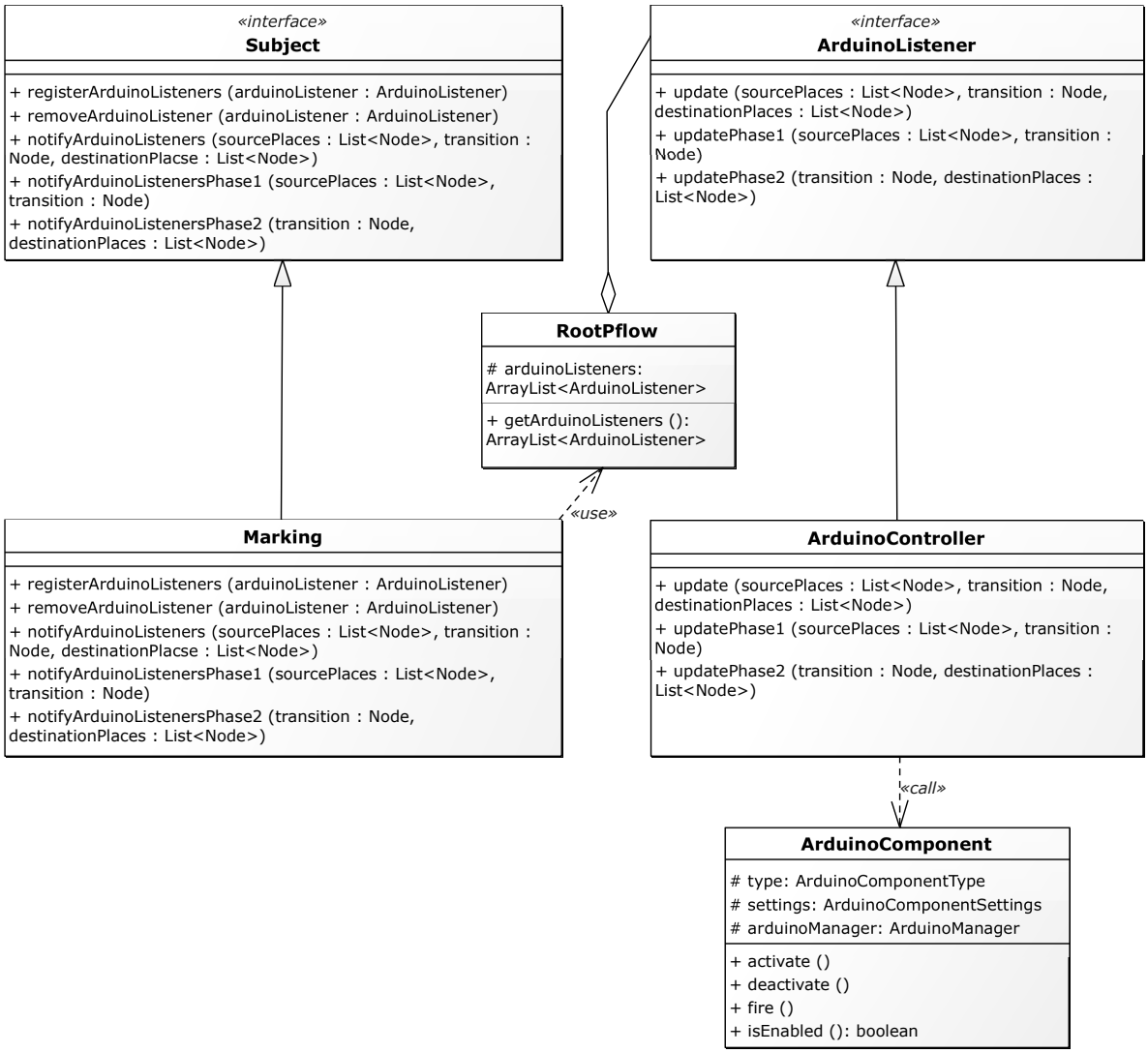

Figure 4. PN2ARDUINO—Class diagram.

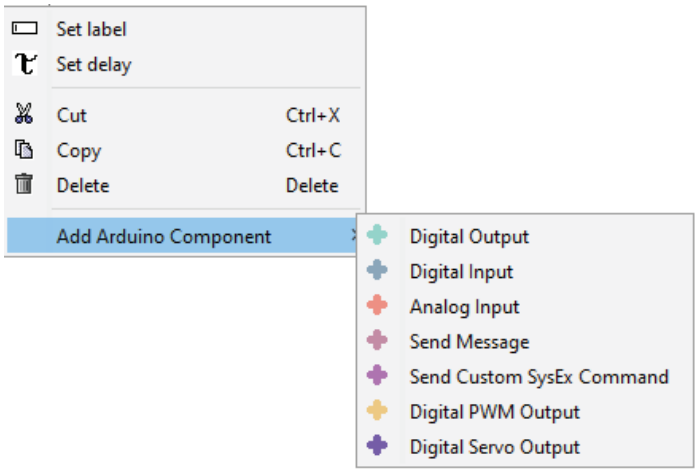

Figure 5. PN2ARDUINO—Adding of Arduino component.

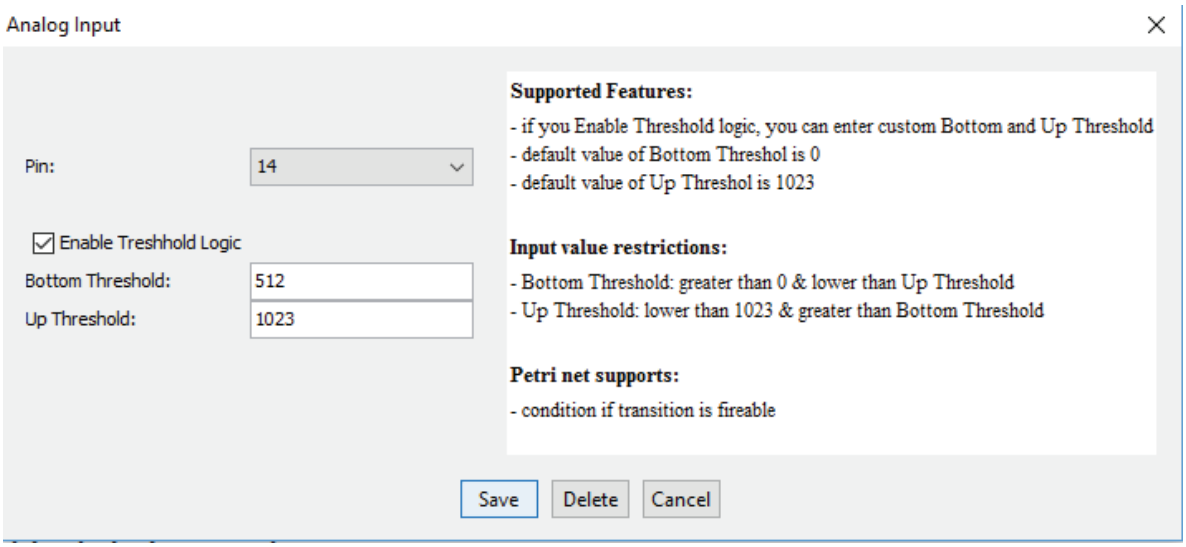

Figure 6. PN2ARDUINO—Analog input.

3. Results

3.1. Case Study: Control of Laboratory Discrete-Event System

For verification of the developed software tool and discrete-event systems control method it was necessary to design a laboratory model of such a system. A fire alarm model was built. The scheme can be seen in Figure 7.

The fire alarm model consists of an active buzzer, photoresistor, three resistors and an NPN transistor. The NPN transistor is mandatory for active buzzer connection. The LED of Arduino in pin 13 is also used. A photoresistor was used instead of a smoke sensor to simplify the experiment.

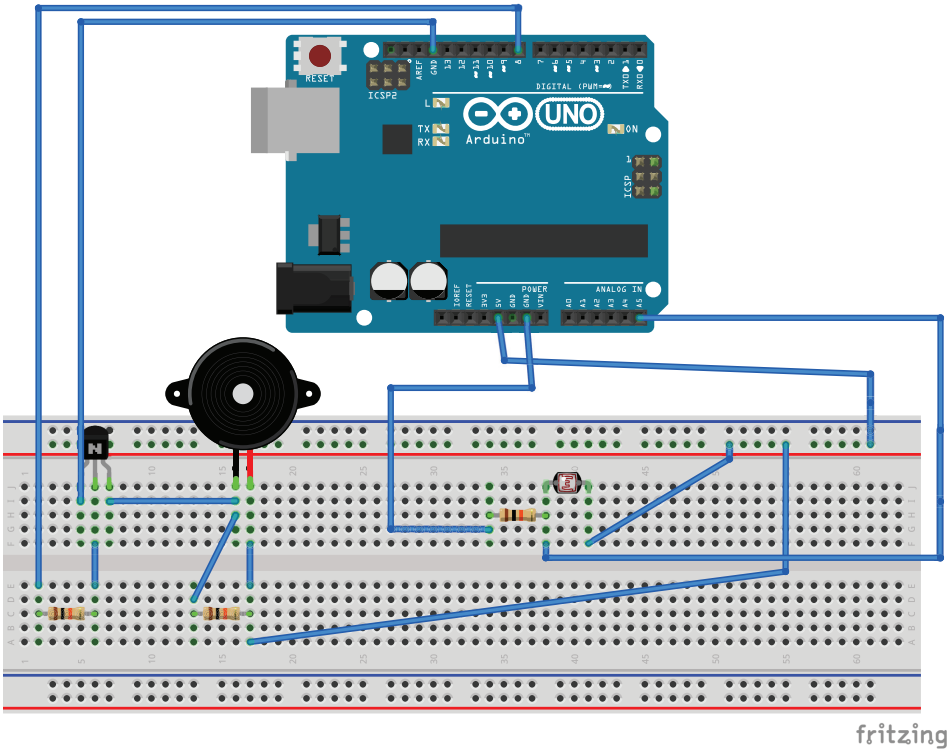

Figure 7. The scheme of laboratory model of fire alarm.

Next, the behavior of the system must be defined. When the photoresistor detects an excessive lighting (experimentally determined as an input value greater than 799 on the analog pin of Arduino Uno which resolution is from 0 to 1023) the intermittent tone of the buzzer is turned on. This tone alternates with LED lighting. When the value on the analog pin drops below 800, the sound and light effects stop. This is repeated cyclically.

Initial marking of modeled timed Petri Net interpreted for control (or sometimes known as timed interpreted Petri net) in PN2ARDUINO is shown in Figure 8.

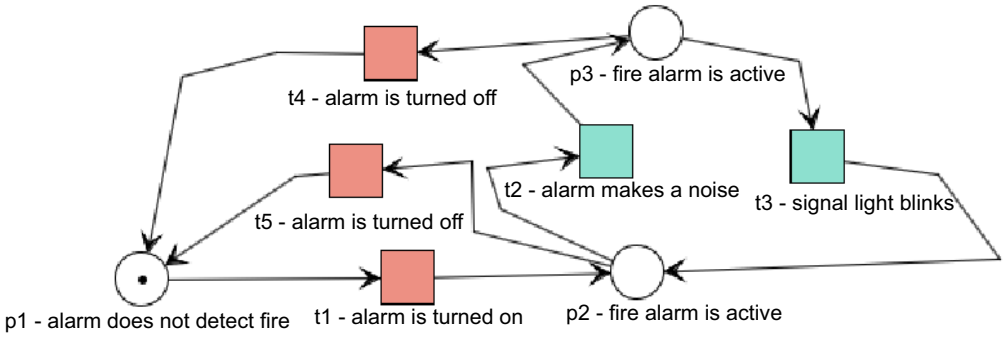

Figure 8. Petri Net for fire alarm (initial marking).

Places of the Petri net (Figures 8–10) correspond to the following states:

- p_1—alarm does not detect fire
- p_2 and p_3—alarm is active (fire was detected)

Transitions of Petri net (Figures 8–10) correspond with the following actions/events:

- t_1—alarm is turned on
- t_2—alarm makes a noise
- t_3—signal light blinks
- t_4 and t_5—alarm is turned off

The token in place p_1 corresponds to the state when the fire alarm is not activated because the photoresistor has not detected the light intensity threshold.

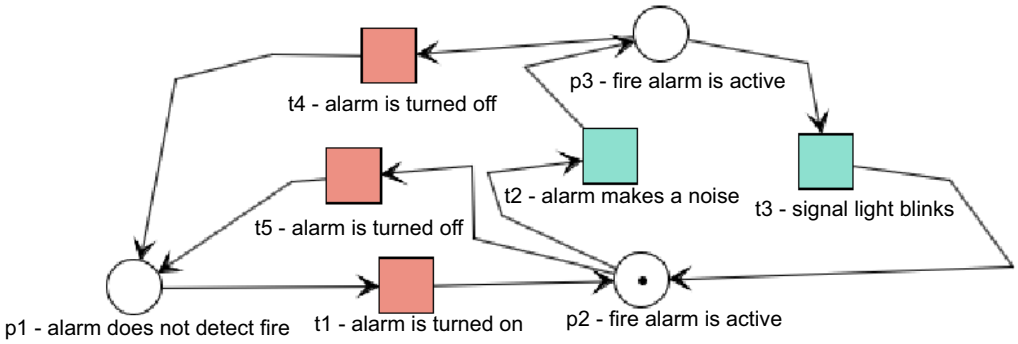

Figure 9. Petri Net for fire alarm (t_1 is fired).

When a value greater than 799 is detected on the analog pin of Arduino, the transition t_1 is fired. This transition is associated with Arduino component *Analog Input* where the range of input values is set. The transition is enabled depending on this range.

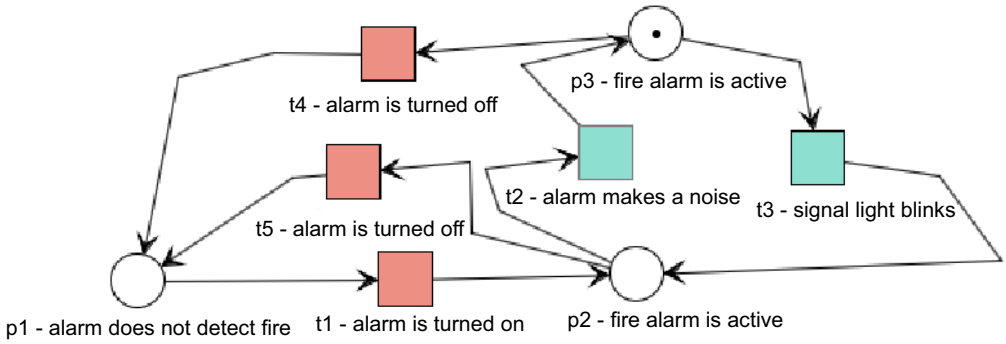

Figure 10. Petri Net for fire alarm (t_2 is fired).

In Figure 9, the token is in the place p_2. Transition t_2 is associated with Arduino component *Digital Output* (pin 8 in this case) where the buzzer is connected. This transition has also associated the function of a time delay (2 s) which means that the transition firing (and buzzer sound effect) lasts for 2 s.

In Figure 10, the token is in the place p_3 (Figure 10). Transition t_3 is associated with Arduino component *Digital Output* (pin 13 in this case) where the build-in LED is connected. The time delay is set to 1 s, i.e., the LED diode is turned on for 1 s.

This process is repeated cyclically, and stops when the value on the analog pin drops under 800. Then the transition t_4 or t_5 is fired and the token moves to the place p_1 when the fire alarm does not detect the fire.

The reachability graph for described Petri net is shown in Figure 11.

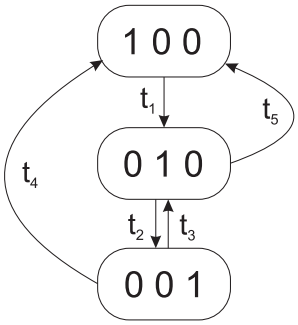

Figure 11. Reachability graph.

We can conclude that the ability of discrete-event control with PN2ARDUINO was successfully verified and can be generalized for other applications.

3.2. Case Study: Control of Laboratory Hybrid System

To verify the proposed software tool for hybrid systems control it was necessary to find an appropriate laboratory model. A DC motor with encoder was chosen; its parameters are in Table 2. An incremental encoder is used to measure speed for the feedback. The measured speed of the DC is the process value.

The DC motor was connected to Arduino Uno using a motor shield module based on dual full bridge driver L298. Using the motor shield, it is possible to independently control speed and direction of rotation of the DC motor. For speed measurement the hardware interruptions functionality of Arduino Uno has been used.

The speed of the motor is set by a pin denoted "PWM A". When the input is set to "PWM = 255" the Arduino program shows 186 rpm which approximately corresponds with parameters as stated by the manufacturer.

The next step was measurement of the steady state I/O characteristics. The input (armature) voltage ranges from 0 V to 5 V which corresponds to a PWM signal from 0 to 255 (8-bit resolution), the sampling period was chosen—0.05 s.

Table 2. Specification of DC motor.

Actuators Conditions	
Rated voltage	6.0 V (DC)
Temperature range	−20 °C ~+60 °C
Humidity range	0–90%
No-load characteristics	
No-load current	≤200 mA
No-load speed	185 ± 10% rpm
Load characteristics	
Rated load	0.0883 N · m
Rated current	≤550 mA
Rated speed	135 ± 10% rpm
Starting torque	0.4413 N · m
Locked-rotor current	≥2.0 A

The resulting steady state I/O characteristics is in Figure 12.

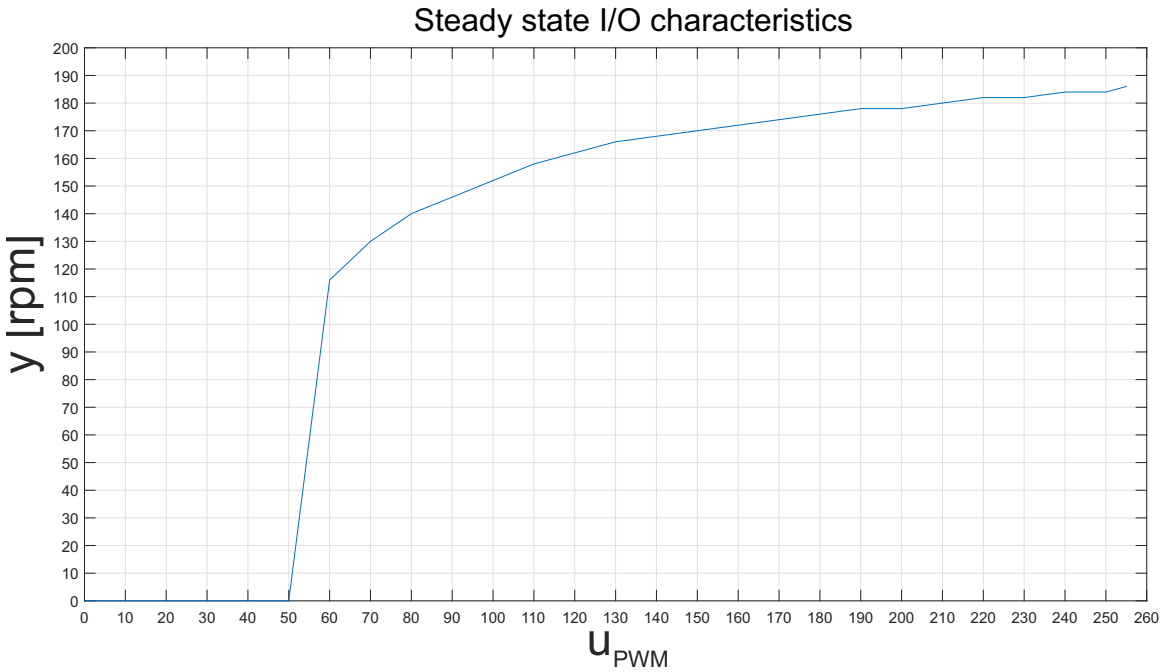

Figure 12. Steady state I/O characteristics of DC motor.

To be able to use linear dynamic models, the working points had to be chosen from linear parts of the I/O characteristics Two working points have been chosen (u_{P_1},y_{P_1}) and (u_{P_2},y_{P_2}) where:

$$u_{P_1} = 80 \rightarrow y_{P_1} = 140 \text{ rpm} \quad (1)$$

$$u_{P_2} = 170 \rightarrow y_{P_2} = 174 \text{ rpm} \quad (2)$$

Since the controller has been designed for a real system (fast dynamics, large noise, uncertainties), the controller parameters were tuned using practice-oriented design methods. The designed PID controller was implemented using the Arduino PID Library—[28]. Creation of a PID class object has the following syntax:

```
PID (&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)
```

- `Input`: controlled variable (`double`), rpm of the motor
- `Output`: control variable (`double`), in this case input voltage—PWM (0-255)
- `Setpoint`: setpoint (`double`), desired rpm of the motor
- `Kp, Ki, Kd`: tuning parameters (`double>=0`)
- `Direction`: Either DIRECT or REVERSE—determines which direction the output will move when faced with a given error.

The closed-loop scheme is in Figure 13. The monotype font was used for variables names in the Arduino program. The `Setpoint` is the desired speed (rpm). Due to the used data type in the Arduino program (`long`), multiples of ten of the setpoint are used. Using an extra order enables us to deal with an equivalent of a number with one decimal place. `Output` is the control variable ranging between 0 and 255 (8 bits) corresponding to the input voltage between 0 V and 5 V (PWM 0-255).

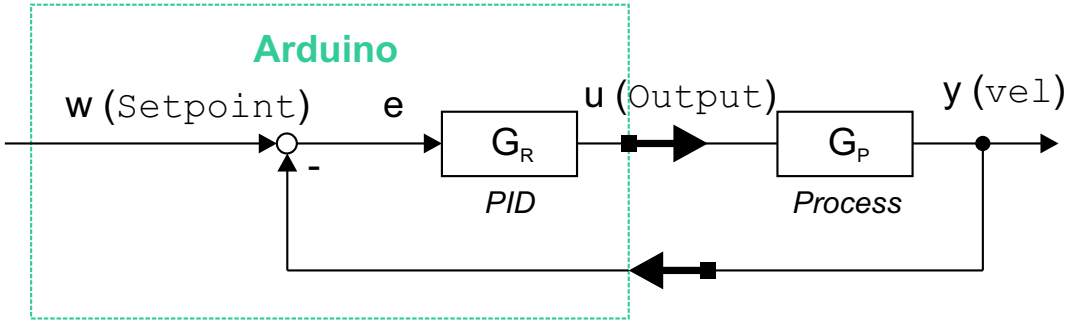

Figure 13. Block diagram of PID (proportional-integral-derivative) controller in a feedback loop.

Experiments showed that designing an effective PID controller for higher speeds (2nd working point: $\omega = 176$ rpm) is not complicated. A satisfactory control performance can be achieved by tuning individual PID controller parameters.

$$G_R = P + \frac{I}{s} + Ds \tag{3}$$

However, it was not so easy to design an effective controller in the 1st working point ($\omega = 140$ rpm); mostly, the closed-loop system was not stable. Hence, lower P and I values had to be used. It was expected that this controller will be effective also in the 2nd working point however with a worse performance (too long settling time). To switch between different control algorithms in individual working points, the proposed software for control of hybrid systems using Petri Nets can be used.

For the 2nd working point, a PID controller with parameter values $P = 0.83$; $I = 5$; $D = 0.005$ was designed. The closed-loop step response is shown in Figure 14 whereby a PWM step from 176 to 186 was realized in $t = 5$ s. The settling time is 1.1 s (considering a tolerance ± 2 rpm). This controller was tested also in the 1st working point (for a step from 140 rpm to 146 rpm). It is obvious from the corresponding step response in Figure 15 that this controller does not provide a satisfactory performance.

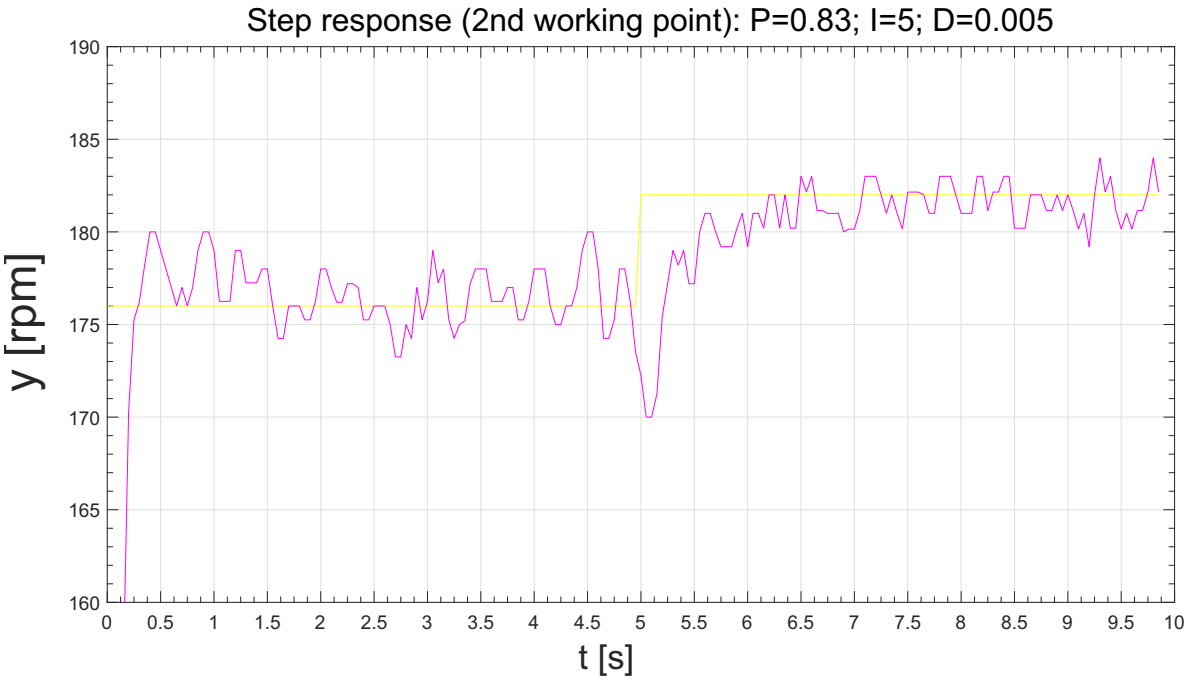

Figure 14. Step response (closed loop)—2nd working point.

Figure 15. Step response (closed loop)—1st working point—with inappropriate controller.

Hence, for the 1st working point a different PID controller was designed with $P = 0.0001$; $I = 1$; $D = 0.01$. The closed-loop step response in Figure 16 shows that the controller works properly (the settling time is 1.3 s). When applied in the 2nd working point, this controller again did not work properly, as expected (larger settling time achieved with a PID controller with smaller P and I). The achieved performance evaluated from the step response in Figure 17 is worse compared with the 1st controller ($P = 0.83$; $I = 5$; $D = 0.005$), the settling time 2.75 s is much larger than in case of the first controller (1.1 s).

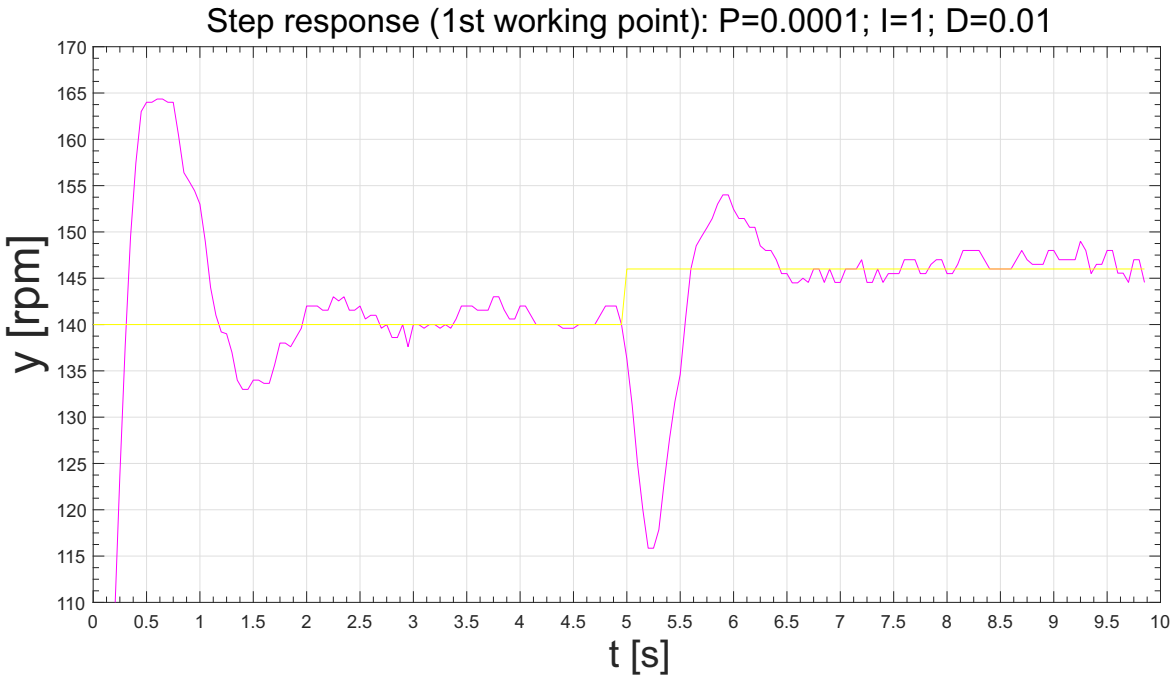

Figure 16. Step response (closed loop)—1st working point—under PID controller.

Figure 17. Step response (closed loop)—2nd working point—under inappropriate controller.

The analysis of achieved results revealed necessity to use different controllers in individual working points, or to design the controller using some advanced control design approach (robust, gain scheduled, switched). In case of switching between multiple controllers according to the working point it is possible to use the developed software module PN2ARDUINO. Switching between controllers and setpoints is based on SYSEX messages. Arduino and other microcontrollers that support Firmata protocol can be used. Development and verification of this software module are the most interesting results of the presented research.

A demonstration example of the proposed control method is in Figure 18. Consider the above-mentioned DC motor which must operate in two modes (working points). For effective setpoint

tracking by the DC motor speed, controllers with different parameters must be used (a different controller for each mode). Switching between individual working points is carried out using a potentiometer connected to the analog input of the Arduino Uno microcontroller. Switching between controllers is provided by transitions *switch1* and *switch2* of Petri net according to the input value from the potentiometer. Input from the analog pin in Arduino is represented by a value ranging between 0 and 1023. The mean value (512) was used as a threshold. In the moment when the token in Petri net is moved to the places *setpoint1* or *setpoint2*, a SYSEX message is sent. This message ensures the execution of a user-defined program code on the Arduino side, in this case the control algorithm. The (PID) algorithm for continuous-time control is independent of Firmata messaging, so it provides real-time control. The provided hybrid systems control case study is a basic example. Researchers in hybrid control design can use it for different and even more complicated scenarios.

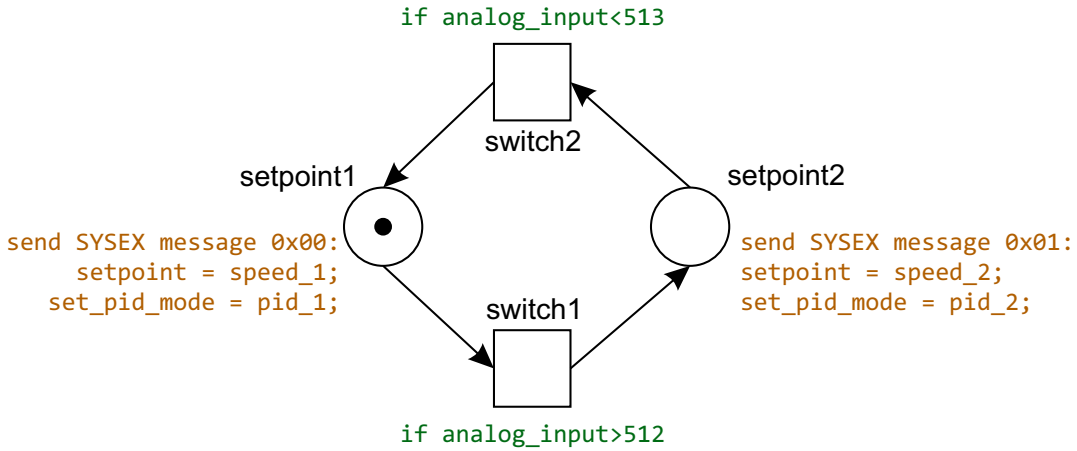

Figure 18. Control scheme for hybrid system using PN2ARDUINO.

As already stated, hybrid systems for Industry 4.0 manufacturing processes are a combination of time-driven and event-driven dynamics. Cyber-physical characteristics of production parts that undergo changes at various operation situations are described by time-driven control models, while the hybrid timing control of operations is described by event-driven models. We propose that manufacturing parts are characterized by cyber-physical states (e.g., velocity, position, pressure, temperature, etc.) subject to time-driven dynamics, and by temporal states (e.g., operation start and stop times) subject to event-driven dynamics. A hybrid system model for control of manufacturing processes represents the behavior of dynamical systems where the states can evolve continuously as well as instantaneously. Such systems arise when control algorithms that involve digital smart devices are applied to continuous-time systems, or due to the intrinsic dynamics (e.g., mechatronic systems with bumps, electrical and mechanical devices for switching control). Hybrid control may be used to improve performance and robustness compared to conventional control (PID), and hybrid dynamics may be unavoidable due to the interplay between digital and analog components of a complex manufacturing system.

An important class of discrete-event systems in practice are automated storage and retrieval systems (AS/RS). There is a big demand for new modeling and control methods of these systems in automotive industry in Slovakia. AS/RS are an important part of logistics. Presently, economic activity across continents is gaining unprecedented dimensions. The consumer lifestyle of most of the population requires production of large quantities of goods for a short time. The advantage of AS/RS consists especially in saving the human capital, and higher reliability. There is a big demand for new modeling and control methods of these systems in automotive industry in Slovakia (Volkswagen, Jaguar Land Rover, Peugeot-Citroen, KIA). The research about AS/RS, their examples and modeling of such systems using Petri nets can be found in [13,14].

The Smart Industry concept is a national initiative in Slovakia to transform and strengthen the industry using the Industry 4.0 methodology based on the latest research activities at universities and in companies mainly in automotive industry. Research and applications in this field are oriented mainly on small and medium-sized enterprises; one of important features is design of their optimal architectures and hybrid control of individual sub processes. An example of reference architecture of a workplace in which production is realized with minimal human intervention is shown in Figure 19.

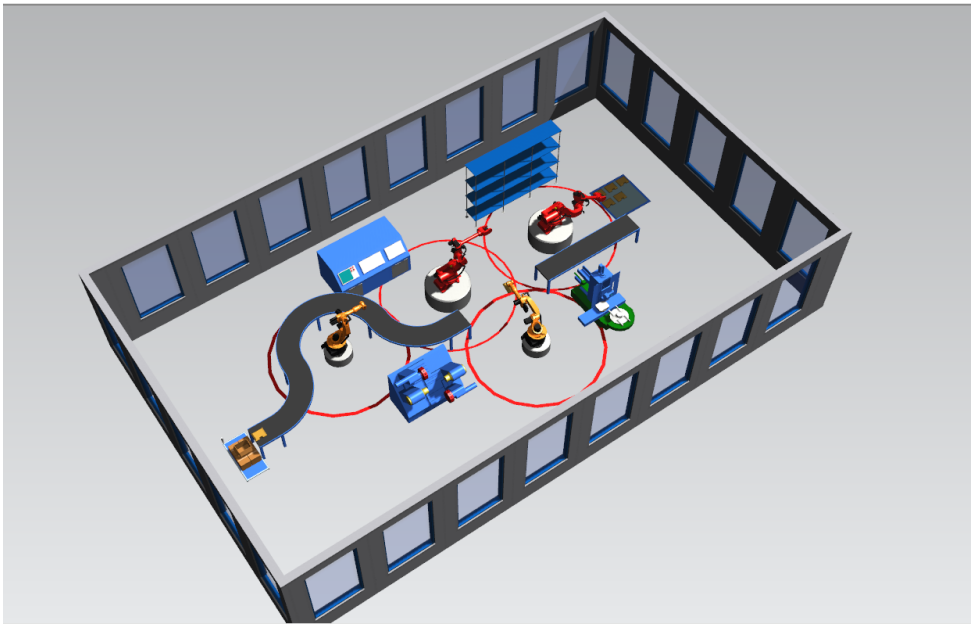

Figure 19. Possible architecture for small enterprises consists of storages, hybrid control, cognitive robots, CNC machines and laser quality testing.

4. Conclusions

The article presents an extension of the PNEditor named PN2ARDUINO; able to communicate with microcontrollers supporting the Firmata protocol. The new software tool enables us to control discrete-event and hybrid systems using timed interpreted Petri nets thus supporting the control paradigm when in the microcontroller only the communication protocol is implemented. Petri nets control logic is stored in the computer which communicates with the microcontroller and sends control commands. The main benefit of the developed SW tool is the possibility to control complex discrete-event and hybrid systems exploiting the advantages of Petri nets formalism which can support many challenging scenarios in modern manufacturing systems within Industry 4.0.

Scientific and application contribution of the research can be summarized in the following points:

- **Development of the software application PN2ARDUINO supporting modeling and control of discrete-event and hybrid systems**
  A new software application named P2ARDUINO based on the open-source Petri nets editor PNEditor was developed to support systems modelling by using timed interpreted Petri nets. Consequently, it enables us to realize control algorithms on Arduino-type microcontrollers as well as on other compatible microcontrollers which support the Firmata protocol used for communication between the computer and the microcontroller.
- **Verification of the newly developed application software system for control of discrete-event systems on a laboratory model**
  To verify and demonstrate possibilities to control discrete-event systems using the developed application P2ARDUINO, a laboratory fire alarm model was proposed. The Arduino-type microcontroller Arduino Uno was chosen for control purposes. The respective Petri net was

modeled in the original software PN2ARDUINO. Finally, control of the laboratory discrete-event system was demonstrated.

- **Development and verification of the developed hybrid system control methodology based on high-level Petri net**
  To verify the developed hybrid system control methodology, it was necessary to set up a laboratory physical model of such type. A DC motor was chosen with an encoder realizing the feedback. Two working points on the steady state I/O characteristics were selected and two respective control algorithms (PID controllers) were designed. The optimal solution was to switch between the two control algorithms depending on the motor speed (working point). The developed software application PN2ARDUINO showed to be appropriate for this purpose. Switching between individual control algorithms (controllers) and setpoints was realized via SYSEX messages offered by the Firmata protocol. It is the author's original solution which can be considered to be an important scientific and application contribution. Using the Firmata protocol, SYSEX type messages and possibility to implement user code on the microcontroller side enable us to apply any control algorithm realizable by the microcontroller.

Scientific and application contributions as declared in the three above points describe the original modeling and control procedures and solutions for discrete-event and hybrid systems, and can further be modified for next research and practice in Industry 4.0.

The next research will focus on the Petri net-based control methodology with the control logic directly implemented in the microcontroller.

5. Patents

The system for control of discrete-event and hybrid systems using Petri nets mentioned in the article is protected by utility model number 7982 by Industrial Property Office of the Slovak Republic (https://wbr.indprop.gov.sk/WebRegistre/UzitkovyVzor/Detail/67-2017).

Author Contributions: E.K. proposed the idea in this paper and prepared the software application; J.Š., O.H., P.D. and J.C. designed the experiments; E.K. and P.D. performed the experiments; O.H., R.L. and E.K. analyzed the data; E.K. wrote the paper; O.H., P.D., R.L. and J.C. edited and reviewed the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic, KEGA 038STU-4/2018 and KEGA 016STU-4/2020, and by the Slovak Research and Development Agency APVV-17-0190.

Acknowledgments: We would like to thank to Alžbeta Češeková for helping with programming the implementation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kharitonov, D.; Tarasov, G.; Golenkov, E. Modeling of Object-Oriented Programs with Petri Net Structured Objects. *Comput. Inf.* **2017**, *36*, 1063–1087. [CrossRef]
2. Prilandita, N.; McLellan, B.; Tezuka, T. Modeling Autonomous Decision-Making on Energy and Environmental Management Using Petri-Net: The Case Study of a Community in Bandung, Indonesia. *Challenges* **2016**, *7*, 9. [CrossRef]
3. Kaid, H.; Al-Ahmari, A.; Li, Z.; Davidrajuh, R. Intelligent Colored Token Petri Nets for Modeling, Control, and Validation of Dynamic Changes in Reconfigurable Manufacturing Systems. *Processes* **2020**, *8*, 358. [CrossRef]
4. Fonseca i Casas, P.; Lijia Hu, D.; Guasch i Petit, A.; Figueras i Jové, J. Simplifying the Verification of Simulation Models through Petri Net to FlexSim Mapping. *Appl. Sci.* **2020**, *10*, 1395. [CrossRef]
5. Dotoli, M.; Fanti, M.; Iacobellis, G. A freeway traffic control model by first order hybrid petri nets, in Automation Science and Engineering (CASE). In Proceedings of the 2011 IEEE International Conference on Automation Science and Engineering, Trieste, Italy, 24–27 August 2011; pp. 425–431. [CrossRef]

6. Fanti, M.; Iacobellis, G.; Mangini, A.; Ukovich, W. Freeway traffic modeling and control in a first-order hybrid petri net framework, Automation Science and Engineering. *IEEE Trans. Autom. Sci. Eng.* **2014**, *11*, 90–102. [CrossRef]
7. Dotoli, M.; Fanti, M.; Mangini, A. Fault monitoring of automated manufacturing systems by first order hybrid petri nets. In Proceedings of the 2008 IEEE International Conference on Automation Science and Engineering, Arlington, VA, USA, 23–26 August 2008; pp. 181–186. [CrossRef]
8. Costantino, N.; Dotoli, M.; Falagario, M.; Fanti, M.P.; Mangini A.M. A model for supply management of agile manufacturing supply chains. *Int. J. Prod. Econ.* **2012**, *135*, 451–457. [CrossRef]
9. Matsuno, H.; Doi, A.; Drath, E.; Miyano, S. Genomic object net: Object oriented representation of biological systems. *Genome Inf. Ser.* **2000**, *11*, 229–230. [CrossRef]
10. Drighiciu, M.A.; Manolea, G. Application des Reseaux de Petri Hybrides a L'etude des Systemes de Production a haute Cadence. 2010. Available online: http://mail.ace.ucv.ro/sintes11/Volume1/2MECHATRONICS/M10_Drighiciu_Adrian_2.pdf (accessed on 1 May 2020).
11. Drighiciu, M.A.; Cismaru, D.C. Modeling a water bottling line using petri nets, Annals of the University of Craiova. *Electr. Eng. Ser.* **2011**, *37*, 110–115.
12. Rohr, C.; Marwan, W.; Heiner, M. Snoopy—A unifying petri net framework to investigate biomolecular networks. *Bioinformatics* **2010**, *26*, 974–975. [CrossRef] [PubMed]
13. Kucera, E.; Niznanska, M.; Kozak, S. Advanced techniques for modelling of as/rs systems in automotive industry using high-level petri nets. In Proceedings of the 2015 16th International Carpathian Control Conference (ICCC), Szilvasvarad, Hungary, 27–30 May 2015; pp. 261–266.
14. Kucera, E.; Haffner, O.; Kozak, S. Modelling and control of as/rs using coloured petri nets. In Proceedings of the 2016 Cybernetics & Informatics (K&I), Levoca, Slovakia, 2–5 February 2016; pp. 1–6.
15. Fausto, S.; Giua, A.; Seatzu, C. HYPENS: A Matlab tool for timed discrete, continuous and hybrid Petri nets. In *International Conference on Applications and Theory of Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2008.
16. Davidrajuh, R. *Modeling and Simulation of Discrete Event Systems with Petri Nets: A Hands-On Approach with GPenSIM*; VDM Verlag: Saarbrücken, Germany, 2009.
17. Krenczyk, D.; Davidrajuh, R.; Skolud, B. An Activity-Oriented Petri Net Simulation Approach for Optimization of Dispatching Rules for Job Shop Transient Scheduling. In Proceedings of the International Joint Conference SOCO'17-CISIS'17-ICEUTE'17, León, Spain, 6–8 September 2017; Springer: Cham, Switzerland, 2017; pp. 299–309.
18. Davidrajuh, R.; Skolud, B.; Krenczyk, D. Performance Evaluation of Discrete Event Systems with GPenSIM. *Computers* **2018**, *7*, 8. [CrossRef]
19. Mostermany, P.J.; Ottery, M.; Elmqvistz, H. Modeling Petri Nets as Local Constraint Equations for Hybrid Systems Using Modelica. 1998. Available online: http://citeseer.ist.psu.edu/359408.html (accessed on 10 May 2020).
20. Fabricius, S.; Badreddin, E. Modelica library for hybrid simulation of mass flow in process plants. In Proceedings of the 2nd Citeseer International Modelica Conference, Oberpfaffenhofen, Germany, 18–19 March 2002; pp. 225–234.
21. Pross, S.; Bachmann, B.; Stadtholz, A. A petri net library for modeling hybrid systems in openmodelica. In Proceedings of the 7th International Modelica Conference, Como, Italy, 20–22 September 2009.
22. Pross, S.; Bachmann, B. Pnlib-an advanced petri net library for hybrid process modeling. In Proceedings of the 9th International MODELICA Conference, Munich, Germany, 3–5 September 2012.
23. Riesz, M.; Seckar, M.; Juhas, G. Petriflow: A petri net based framework for modelling and control of workflow processes. In Proceedings of the Workshops of the 31st International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (PETRI NETS 2010) and of the 10th International Conference on Application of Concurrency to System Design (ACSD 2010), Braga, Portugal, 21–25 June 2010; Volume 827, pp. 191–205.
24. Češeková, A. Control of Laboratory Discrete Event Systems. Master's Thesis, Slovak University of Technology in Bratislava, Bratislava, Slovakia, 2016. (In Slovak)
25. Kucera, E. Modelling and Control of Hybrid Systems Using High-Level Petri Nets. Ph.D. Thesis, Slovak University of Technology in Bratislava, Bratislava, Slovakia, 2016. (In Slovak)
26. Steiner, H.C. Firmata: Towards making microcontrollers act like extensions of the computer. In Proceedings of the NIME, Pittsburgh, PA, USA, 3–6 June 2009; pp. 125–130.

27. MIDI Association: Summary of Midi Messages. 2016. Available online: https://www.midi.org/specifications/item/table-1-summary-of-midi-message (accessed on 10 May 2020).
28. Comnes, B.; La Rosa, A. *Arduino Pid Example Lab*; Portland State University: Portland, OR, USA, 2013.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).