Modelling and Control of Discrete-Event Systems Using Petri Nets and Arduino Microcontrollers

Submitted: 17th March 2020; accepted: 10th August 2020

Erik Kučera, Oto Haffner, Roman Leskovský

DOI: 10.14313/JAMRIS/3-2020/29

Abstract:
The main aim of proposed article is the design of new software system for modelling and control of discrete-event and hybrid systems using Arduino and similar microcontrollers. In this paper we propose a new tool. It is based on Petri nets and it is called PN2ARDUINO. It offers a capability of communication with the microcontroller. Communication with the microcontroller is based on modified Firmata protocol so control algorithm can be implemented on all microcontrollers that support this type of protocol. The developed software tool was successfully verified for control of laboratory systems. It can also be used for education and also for research purposes as it offers a graphical way for designing control algorithm for hybrid and mainly discrete-event systems. Proposed tool can enrich education and practice in the field of cyber-physical systems (Industry 4.0).

Keywords: *Hybrid systems, Petri-nets, Microcomputer-based control, Motor, Discrete-event dynamic systems*

1. Introduction

Development of various systems is a complex discipline that includes many activities, e.g. system design, a specification of required properties, implementation, testing and further development of the system. As these operations are challenging and important for the final product, it is appropriate and necessary to create a model of the system. Development of control methods of discrete-event and hybrid systems belongs to the modern trends in automation and mechatronics. Hybrid system is a combination of continuous and discrete event systems. Control of such systems brings new challenges because it is necessary to join control methods of discrete event systems (where formalism of Petri nets can be helpful) and classic control methods of continuous systems. With good methodology and software module, these approaches can be synergistically combined. This will give us an appropriate and unique control system that allows harmonizing discrete event control methods with the methods of control of continuous systems (e.g. PID algorithms). Effective cooperation of these approaches allows to control hybrid system. This method would be useful in systems where it is necessary to use different control algorithms (for example PID controllers with different parameters) according to the state of the system. The concept of Petri nets is capable of covering a management of these control rules in a very efficient, robust and well-arranged (graphical) way. This paper is aimed to present new Petri Net tools for modelling and control of discrete-event and hybrid systems. Case studies for control of laboratory fire alarm system and DC motor are also presented.

In papers [1] and [2] authors deal with usage of hybrid and colour Petri nets for modelling of crossroads and traffic on highways. From these authors, there are also interesting projects from the field of manufacturing systems [3] and [4]. Unfortunately, it is not mentioned whether the results are only theoretical models or have been simulated using a SW tool or deployed in practice.

In [5] author developed an interesting software tool that supports hybrid Petri nets named Visual Object Net++. There a lot of papers (mainly from Romanian author [6] and [7]) that describes capabilities of Visual Object Net++. This tool is not open-source and it is not further developed.

Software tool Snoopy [8] offers modelling using many classes of Petri nets like stochastic, hybrid, colour, music Petri nets, etc. Using this tool, many types of research in the field of biology and chemistry are being solved. Unfortunately, the source code is not available.

In [9] and [10] Coloured Petri nets are used for control of computer model of automated storage and retrieval system.

As an interesting way of research, a Modelica language and open-source tool OpenModelica appeared. There is a library that supports modelling by Petri nets in this tool. One of the advantages of OpenModelica is that PN model can be connected with other components of Modelica. The first Petri net toolbox was introduced in [11]. An extension of this toolbox was described in [12]. The greater addition to the toolbox was made by the German author who enriches it by a support of extended hybrid Petri nets for modelling of processes in biological organisms [13] and [14]. This tool was developed primarily for commercial tool Dymola and not for OpenModelica, so applicability in scientific research and extensibility is limited. During 2015 the team that developed PNlib published modified version of PNlib that partially worked in OpenModelica. Unfortunately, it was not possible to use OpenModelica for control purposes using microcontrollers because of lack of COM port communication support.

Wolfram SystemModeler is easy-to-use modelling and simulation program for cyber-physical systems [15]. It is based on Modelica language, too, but it is a proprietary software. Great advantage of Wolfram SystemModeler is a support of communication with serial port. There a free extension ModelPlug that provides connection of simulation with microcontrollers that supports Firmata protocol [16]. Unfortunately Wolfram SystemModeler supports only StateGraph [17] and not High-level Petri nets (PNlib). The solution can be an integration of PNlib library to Wolfram SystemModeler. Attempts of this integration appeared in [18]. Unfortunately these attempts were not successful due to the various complications with SystemModeler's interpreter from Modelica to C++ language.

2. Description of Developed SW Tool PN2ARDUINO

As it was realized that there is no complex SW solution to support control of discrete event and hybrid system by microcontrollers using High-level Petri nets, it was necessary to develop it. As a basis for such software, PNEditor [19] was chosen. This tool is open-source. The developed extension of this tool is named PN2ARDUINO and is fully tested in [20] and [21]. The main topic of this paper is an introduction to this developed software that can be used for control of discrete event and hybrid systems and its verification on laboratory discrete-event and hybrid system.

There are more concepts of control using Petri nets. Petri net as a control logic is necessary to connect with the controlled system (e.g. using 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 into the PC (which can communicate with microcontroller). Both approaches have their advantages and disadvantages.

If the Petri net's logic is stored in the microcontroller, the main advantage is the independence of control unit from the software application (program on PC). The Petri net logic is modelled using PC, and then the Petri net is translated into program code which is loaded into the microcontroller. Then PC and microcontroller can be disconnected. The advantage is also the capability of control in real time. Disadvantages are limited computational and memory resources of the microcontroller. Following disadvantage is the need of repeating compiling and uploading the program into the microcontroller (mainly during development phase). The proposed solution is shown in Fig. 1.

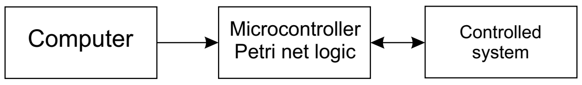


Fig. 1. Simple scheme of proposed solution - Petri net's logic in microcontroller

When the Petri net's control logic is stored in specialized SW application on PC, this solution gives an opportunity to control the system directly from it. In the microcontroller, only the program with communication protocol is stored. This communication protocol (in our case it is Firmata [16]) is used for communication between PC and microcontroller. This solution eliminates the necessity of recompiling and reuploading the program during development. The next advantage is the elimination of restrictions on computing and storage resources because PC has (in comparison with microcontroller) almost unlimited resources. One of the disadvantages is that the control system cannot react in real time. The proposed solution is shown in Fig. 2.

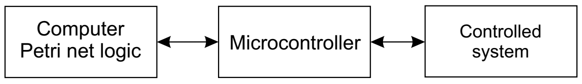


Fig. 2. Simple scheme of proposed solution - Petri net's logic in PC

In Table 1, these differences are specified.

New software module PN2ARDUINO was based on the second approach. The Petri net runs on the personal computer. For communication between SW application and microcontroller, the protocol Firmata [16] was used. Firmata is a protocol that is designed for communication between microcontroller and computer (or mobile device like a smartphone, tablet, etc.). This protocol can be implemented in firmware of various microcontrollers. Mostly Arduino-family microcontrollers are used. On PC the client library is needed. These libraries are available for many languages like Java, Python, .NET, PHP, etc. Firmata protocol is based on MIDI messages [22].

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

PN2ARDUINO extends PNEditor with many features. For Petri nets modelling, there is a capability of adding time delay to transitions and capacity for places. Also, automatic mode of firing transition was added for automatic system control purposes as only manual mode was present in PNEditor.

PN2ARDUINO brings a new communication module to PNEditor. This module communicates with the compatible microcontroller. This module consists of two parts. The first one provides the creation of connection with the microcontroller, so it sets COM port where the microcontroller is connected. The second part provides the implementation of a capability of adding Arduino components to Petri net's places and transitions. These types of Arduino components are supported: digital input and output, analog input, servo control, PWM output, message sending, custom SYSEX message [16] sending.

In Fig. 3, the use-case diagram of developed SW tool can be seen. Class diagram is shown in Fig. 4.

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

The interesting functionality is a capability of sending custom SYSEX messages. The user must enter SY-

Tab. 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

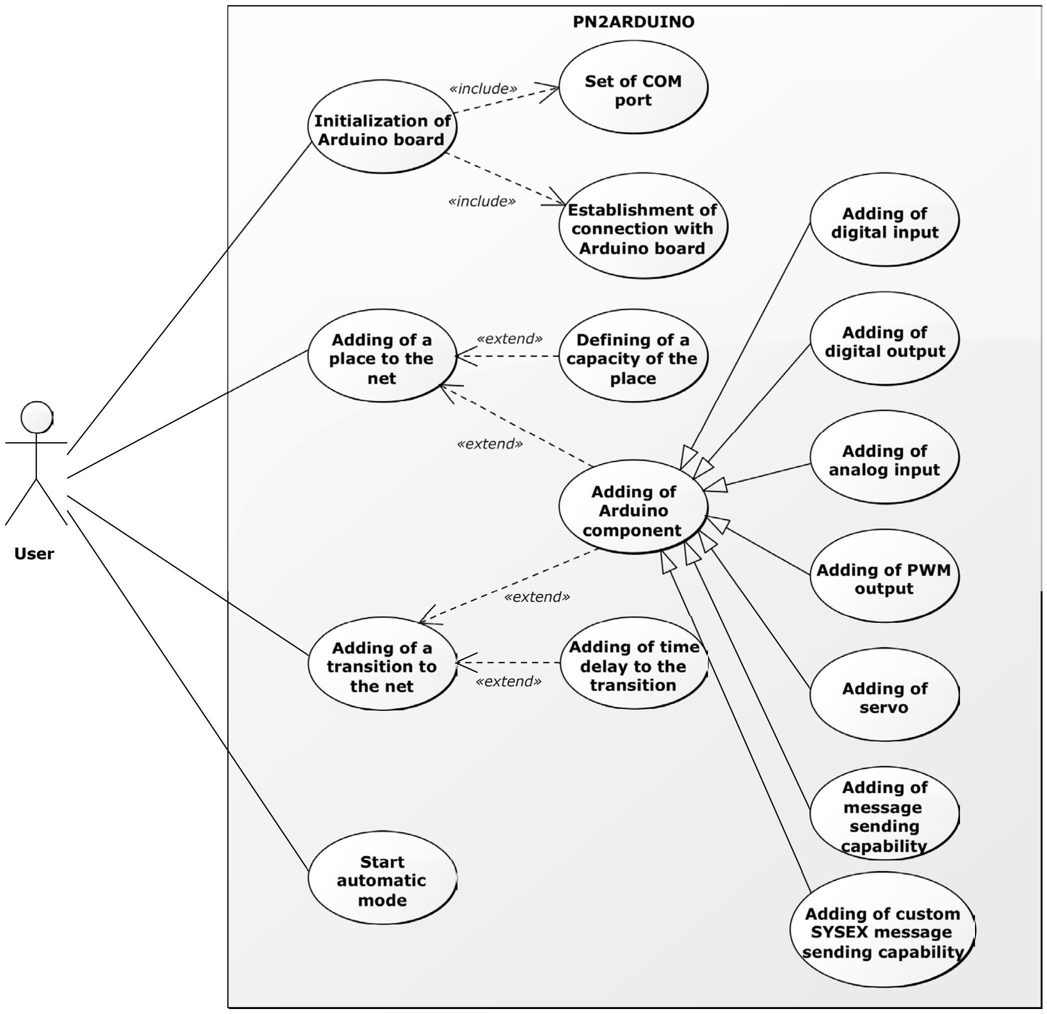

PN2ARDUINO
User
Initialization of Arduino board
«include»
Set of COM port
«include»
Establishment of connection with Arduino board
Adding of a place to the net
«extend»
Defining of a capacity of the place
«extend»
Adding of Arduino component
«extend»
Adding of a transition to the net
«extend»
Adding of time delay to the transition
Start automatic mode
Adding of digital input
Adding of digital output
Adding of analog input
Adding of PWM output
Adding of servo
Adding of message sending capability
Adding of custom SYSEX message sending capability

Fig. 3. PN2ARDUINO - Use-case diagram

SEX 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. For example, SYSEX messages are used in the proposed example of hybrid control in the last section of the paper. Here, the SYSEX message notifies the microcontroller that a different PID algorithm should be used for system control. Then 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 modelling and log console. PN2ARDUINO supports two modes - design mode and control mode. Control mode is manual and automatic.

Firstly, 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 the transition (Fig. 5). The example of analog input can be seen in Fig. 6.

Time politics are also supported. To the transitions, it is possible to add time delay which can be deterministic or stochastic.

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

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

This model consists of an active buzzer, photo-resistor, three resistors and NPN transistor. NPN transistor is mandatory for active buzzer connection. The LED of Arduino in pin 13 is also used. Photo-resistor was used instead of the smoke sensor because of the less complicated feasibility of experiment.

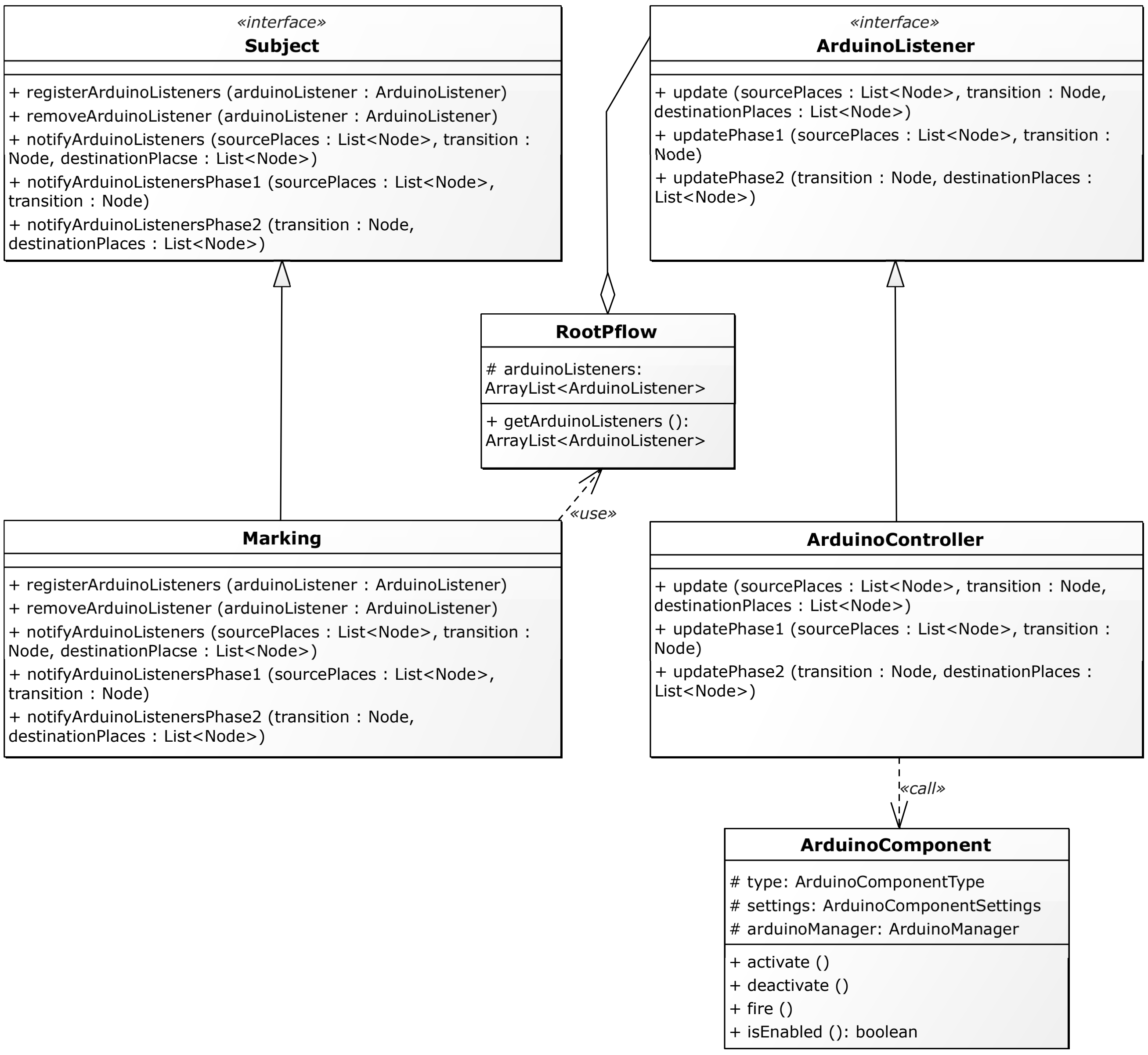

«interface»
Subject
+ registerArduinoListeners (arduinoListener : ArduinoListener)
+ removeArduinoListener (arduinoListener : ArduinoListener)
+ notifyArduinoListeners (sourcePlaces : List<Node>, transition : Node, destinationPlacse : List<Node>)
+ notifyArduinoListenersPhase1 (sourcePlaces : List<Node>, transition : Node)
+ notifyArduinoListenersPhase2 (transition : Node, destinationPlaces : List<Node>)
«interface»
ArduinoListener
+ update (sourcePlaces : List<Node>, transition : Node, destinationPlaces : List<Node>)
+ updatePhase1 (sourcePlaces : List<Node>, transition : Node)
+ updatePhase2 (transition : Node, destinationPlaces : List<Node>)
RootPflow
# arduinoListeners: ArrayList<ArduinoListener>
+ getArduinoListeners (): ArrayList<ArduinoListener>
«use»
Marking
+ registerArduinoListeners (arduinoListener : ArduinoListener)
+ removeArduinoListener (arduinoListener : ArduinoListener)
+ notifyArduinoListeners (sourcePlaces : List<Node>, transition : Node, destinationPlacse : List<Node>)
+ notifyArduinoListenersPhase1 (sourcePlaces : List<Node>, transition : Node)
+ notifyArduinoListenersPhase2 (transition : Node, destinationPlaces : List<Node>)
ArduinoController
+ update (sourcePlaces : List<Node>, transition : Node, destinationPlaces : List<Node>)
+ updatePhase1 (sourcePlaces : List<Node>, transition : Node)
+ updatePhase2 (transition : Node, destinationPlaces : List<Node>)
«call»
ArduinoComponent
# type: ArduinoComponentType
# settings: ArduinoComponentSettings
# arduinoManager: ArduinoManager
+ activate ()
+ deactivate ()
+ fire ()
+ isEnabled (): boolean

Fig. 4. PN2ARDUINO - Class diagram

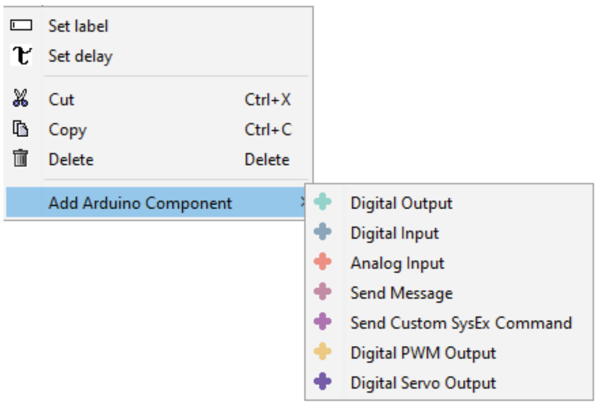
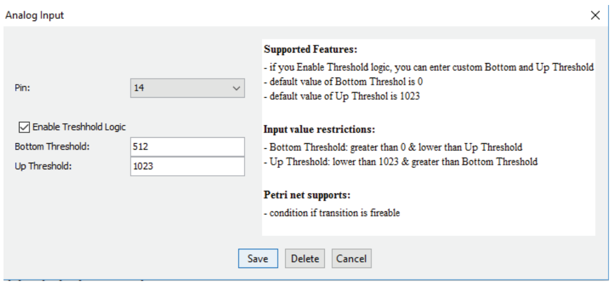


Fig. 5. PN2ARDUINO - Adding of Arduino component

Then the behaviour of the system must be defined. When the photoresistor detects an excessive lighting (it was experimentally determined as input value greater than 799 on the analog pin of Arduino Uno which


Analog Input

Pin: 14

[✓] Enable Treshhold Logic

Bottom Threshold: 512

Up Threshold: 1023

Supported Features:
- if you Enable Threshold logic, you can enter custom Bottom and Up Threshold
- default value of Bottom Threshol is 0
- default value of Up Threshol is 1023

Input value restrictions:
- Bottom Threshold: greater than 0 & lower than Up Threshold
- Up Threshold: lower than 1023 & greater than Bottom Threshold

Petri net supports:
- condition if transition is fireable

Save Delete Cancel

Fig. 6. PN2ARDUINO - Analog input

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 lowers below 800, these sound and light effects stop. This is repeated cyclically.

Initial marking of modelled timed Petri Net interpreted for control (or sometimes called as interpreted timed Petri net) in PN2ARDUINO is shown in Fig. 8.

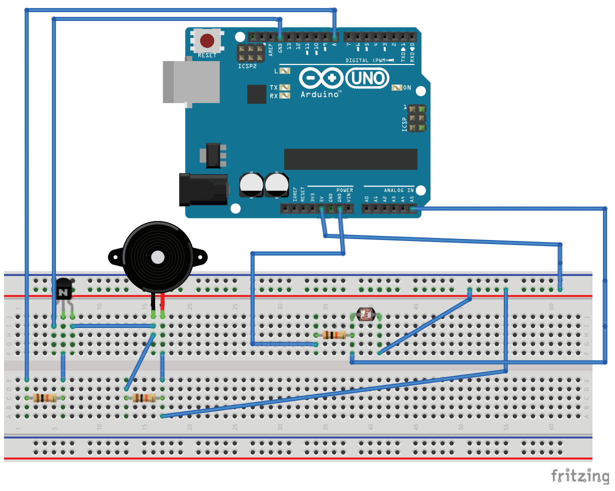


Fig. 7. The scheme of laboratory model of fire alarm

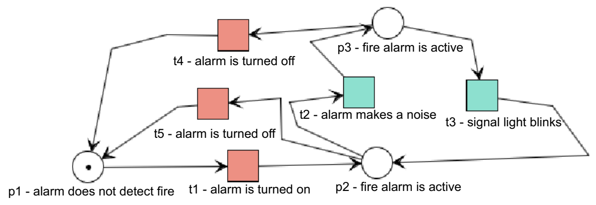


Fig. 8. PN for fire alarm (initial marking)

Places of Petri net (Fig. 8 - Fig.10) corresponds with these states:
- p_1 - alarm does not detect fire
- p_2 and p_3 - alarm is active (fire was detected)

Transitions of Petri net (Fig. 8 - Fig.10) corresponds with these 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 is in place p_1 which corresponds with the state when the fire alarm is not activated because the photo-resistor does not detect light intensity threshold.

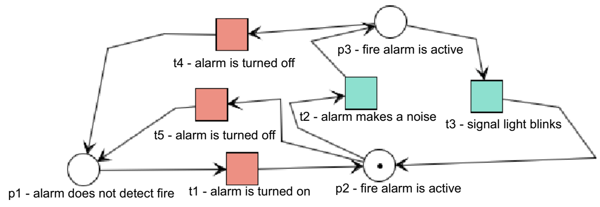


Fig. 9. PN for fire alarm (t_1 is fired)

At the time when the 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 a range of input values is set. This range determines when the transition is enabled.

Now the token is in the place p_2 (Fig. 9). Transition t_2 is associated with Arduino component *Digital Output* (in this case pin 8) where the buzzer is connected. This transition has also associated the function of time delay - 2 seconds. That means that transition firing (and sound effect of buzzer) lasts for 2 seconds.

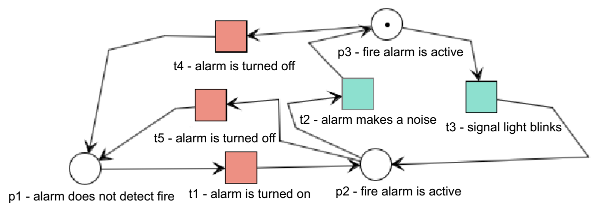


Fig. 10. PN for fire alarm (t_2 is fired)

Tab. 2. Specification of DC motor

Actuators conditions	
Rated voltage	6.0V (DC)
Temperature range	$-20°C \sim +60°C$
Humidity range	$0\% - 90\%$
No-load characteristics	
No-load current	$\leq 200mA$
No-load speed	$185 \pm 10\%$rpm
Load characteristics	
Rated load	$0.0883N.m$
Rated current	$\leq 550mA$
Rated speed	$135 \pm 10\%$rpm
Starting torque	$0.4413N.m$
Locked-rotor current	$\geq 2.0A$

Now the token is in the place p_3 (Fig. 10). Transition t_3 is associated with Arduino component *Digital Output* (in this case pin 13) where the build-in LED is connected. Time delay is set to 1 second. LED diode turns on for 1 second.

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

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

4. Case Study: Control of Laboratory Hybrid System

For verification of proposed software tool for hybrid systems control, it was necessary to design a laboratory model of such system. A DC motor with encoder was chosen. The encoder is used for feedback in the system because it is used for speed measurement. The actual speed of the DC is in is measured process value. See Table 2 for parameters of described DC motor.

DC motor was connected to Arduino Uno using the motor shield module. Arduino motor shield is based on dual full bridge driver L298. Using the motor shield, it is possible to independently control speed and motion direction of DC motor. The encoder in this motor is of incremental type. For speed measurement, it is necessary to use hardware interruptions functionality of Arduino Uno.

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

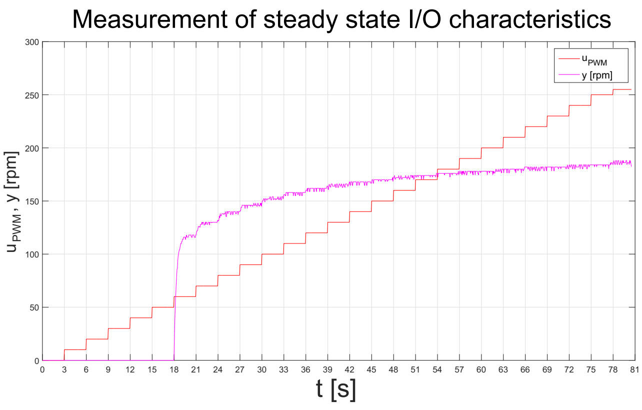


Fig. 11. Measurement of steady state I/O characteristics of DC motor

The next step was a measurement of steady state I/O characteristics. The input is a voltage supplied to the motor. These inputs are of size from 0V to 5V which corresponds with PWM signal from 0 to 255 (8-bit resolution). Sampling is 0.05 seconds. In Fig. 11 the process of measurement of steady state I/O characteristics is shown. The signal was filtered by 1-D median filter of 2nd order. Red line is input to the system (voltage or PWM). Output (rpm) is shown by magenta line. Steady state I/O characteristics is in Fig 12.

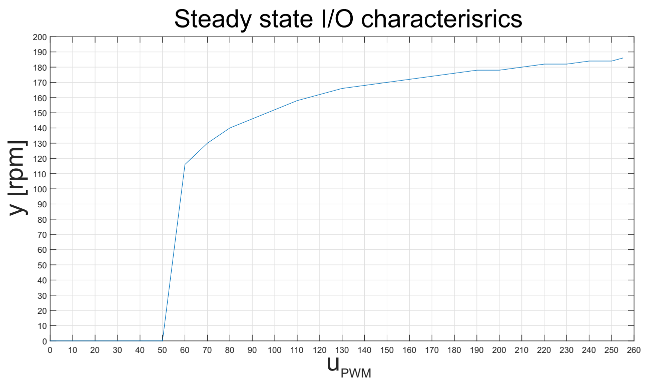


Fig. 12. Steady state I/O characteristics of DC motor

In the process of working points choosing it was necessary to choose points which meet the certain condition. This condition is that behaviour of the system must be close to the linear behaviour around these points. From I/O characteristics two values of input (u_{P_1} and u_{P_2}) and output (y_{P_1} and y_{P_2}) were chosen (these values will be our working points):

$$u_{P_1} = 80 \rightarrow y_{P_1} = 140rpm \tag{1}$$

$$u_{P_2} = 170 \rightarrow y_{P_2} = 174rpm \tag{2}$$

From solution analysis, it is obvious that for each working point it is necessary to use a different controller. One of the solutions is an option to switch between multiple controllers according to the working point - speed (rpm) of DC motor. It is possible to use developed software module PN2ARDUINO. It is possible to switch between controllers and setpoints using SYSEX messages. Arduino and other microcontrollers that support Firmata protocol can be used. Development and verification of this software module are one of the most interesting results of presented research.

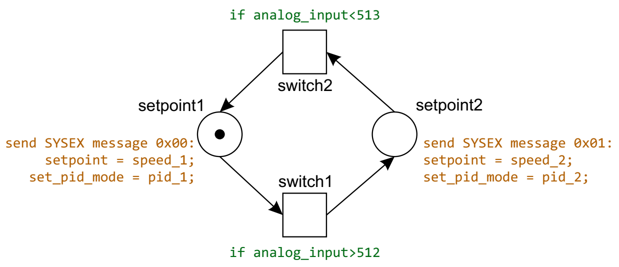


Fig. 13. Control scheme for hybrid system using PN2ARDUINO

For illustration see the scheme in Fig. 13. It is an example for a demonstration of proposed control method. Assume the mentioned DC motor. We require to operate it in 2 modes (working points or rpm). For effective settlement of speed value to the setpoint, controllers with different parameters are needed (different controller for each mode). We switch between rpm using potentiometer connected to the analog input of microcontroller Arduino Uno. The switching between controllers is provided by transitions of Petri net named *switch1* and *switch2* according to the input value from potentiometer. Input from the analog pin in Arduino is represented by value between 0 and 1023. As the threshold, a half value was used (512). In the moment when the token in Petri net is moved to the place named *setpoint1* or *setpoint2*, a SYSEX message is sent. This message ensures the execution of user defined program code on the Arduino side. In this case, the control algorithm is executed. An algorithm (PID controller) for continuous control is independent of Firmata messaging, so it provides real-time control.

The case study of hybrid systems control proposed a basic example. Researchers in the field of hybrid control design can use it for different and more complicated scenarios.

5. Conclusion

The article presents the new SW named PN2ARDUINO which extends PNEditor with the capability of communication with microcontrollers that supports protocol named Firmata. Then it is possible to control discrete-event and hybrid systems using timed interpreted Petri nets with developed SW tool. This tool uses the control paradigm when the microcontroller has implemented only the communication protocol. Petri net's control logic is stored in the personal computer which communicates with the microcontroller and sends control orders. The next research will focus on the concept of control with Petri nets where control logic will be directly implemented on the microcontroller.

AUTHORS

Erik Kučera* – Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, Slovakia, e-mail: erik.kucera@stuba.sk, www: www.uamt.fei.stuba.sk.

Oto Haffner – Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, Slovakia, e-mail: oto.haffner@stuba.sk, www: www.uamt.fei.stuba.sk.

Roman Leskovský – Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, Slovakia, e-mail: roman.leskovsky@stuba.sk, www: www.uamt.fei.stuba.sk.

*Corresponding author

ACKNOWLEDGEMENTS

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, by the Scientific Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under the grant VEGA 1/0819/17, and by the Tatra banka Foundation within the grant program Quality of Education, project No. 2019vs056 (Virtual Training of Production Operators in Industry 4.0).

REFERENCES

[1] M. Dotoli, M. Fanti, and G. Iacobellis, "A freeway traffic control model by first order hybrid petri nets". In: *2011 IEEE Conference on Automation Science and Engineering (CASE)*, 2011, 425–431, 10.1109/CASE.2011.6042526.

[2] M. Fanti, G. Iacobellis, A. Mangini, and W. Ukovich, "Freeway Traffic Modeling and Control in a First-Order Hybrid Petri Net Framework". In: *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, 2014, 90–102, 10.1109/TASE.2013.2253606.

[3] M. Dotoli, M. Fanti, and A. Mangini, "Fault monitoring of automated manufacturing systems by first order hybrid Petri nets". In: *2008 IEEE International Conference on Automation Science and Engineering, CASE*, 2008, 181–186, 10.1109/COASE.2008.4626493.

[4] N. Costantino, M. Dotoli, M. Falagario, M. P. Fanti, and A. M. Mangini, "A model for supply management of agile manufacturing supply chains", *International Journal of Production Economics*, vol. 135, no. 1, 2012, 451 – 457, 10.1016/j.ijpe.2011.08.021.

[5] H. Matsuno, A. Doi, R. Drath, and S. Miyano, "Genomic object net: object oriented representation of biological systems", *Genome Informatics*, vol. 11, 2000, 229–230, 10.11234/gi1990.11.229.

[6] M. A. Drighiciu and G. Manolea, "Application des reseaux de petri hybrides a l'etude des systemes de production a haute cadence". 2010.

[7] M.-A. Drighiciu and D. C. Cismaru, "Modeling a Water Bottling Line Using Petri Nets", *Annals of the University of Craiova, Electrical Engineering series*, 2013.

[8] C. Rohr, W. Marwan, and M. Heiner, "Snoopy - a unifying Petri net framework to investigate biomolecular networks", *Bioinformatics*, vol. 26, no. 7, 2010, 974–975, 10.1093/bioinformatics/btq050.

[9] E. Kucera, M. Niznanska, and S. Kozak, "Advanced techniques for modelling of AS/RS systems in automotive industry using High-level Petri nets". In: *16th International Carpathian Control Conference (ICCC)*, 2015, 261–266, 10.1109/CarpathianCC.2015.7145085.

[10] E. Kucera, O. Haffner, and S. Kozak, "Modelling and control of AS/RS using Coloured Petri nets". In: *2016 Cybernetics & Informatics (K&I)*, 2016, 1–6, 10.1109/CYBERI.2016.7438532.

[11] P. J. Mosterman, M. Otter, and H. Elmqvist. "Modeling Petri Nets as Local Constraint Equations for Hybrid Systems Using Modelica", 1998. `https://www.modelica.org/publications/papers/scsc98fp.pdf`, Accessed on: 2020.11.30.

[12] S. Fabricius and E. Badreddin, "Modelica library for hybrid simulation of mass flow in process plants". In: *Proceedings of the 2nd International Modelica Conference, Oberpfaffenhofen, Germany*, 2002, 225–234.

[13] S. Proß and B. Bachmann, "A Petri Net Library for Modeling Hybrid Systems in OpenModelica". In: *Proceedings of the 7th International Modelica Conference*, 2009, 454–462, 10.3384/ecp09430014.

[14] S. Proß and B. Bachmann, "PNlib-An Advanced Petri Net Library for Hybrid Process Modeling". In: *Proceedings of the 9th International MODELICA Conference*, 2012, 10.3384/ecp1207647.

[15] S. Wolfram, *Mathematica: ein System für Mathematik auf dem Computer*, volume 2, Addison-Wesley, 1994.

[16] H.-C. Steiner, "Firmata: Towards Making Microcontrollers Act Like Extensions of the Computer". In: *The International Conference on New Interfaces for Musical Expression (NIME)*, 2009, 125–130.

[17] M. Otter, K.-E. Årzén, and I. Dressler, "StateGraph–A Modelica Library for Hierarchical State Machines". In: *Modelica 2005 Proceedings*, 2005, 569–578.

[18] P. Cesek. "DEDS control system based on Petri nets and microcontrollers (in slovak)", 2016. M.Sc. Thesis, Slovak University of Technology in Bratislava, 2016 (in Slovak).

[19] M. Riesz, M. Seckár, and G. Juhás, "PetriFlow: A Petri Net Based Framework for Modelling and Control of Workflow Processes". In: *ACSD/Petri Nets Workshops*, 2010, 191–205.

[20] A. Cesekova. "Control of laboratory discrete event systems", 2016. M.Sc. Thesis, Slovak University of Technology in Bratislava, 2016 (in Slovak).

[21] E. Kucera. "Modelling and control of hybrid systems using High-level Petri nets", 2016. PhD Thesis, Slovak University of Technology in Bratislava, 2016, (in Slovak).

[22] MIDI Association. "Summary of MIDI 1.0 Messages", 2016. Accessed on: 2020.11.30.