Computing and Informatics, Vol. 39, 2020, 568–586, doi: 10.31577/cai_2020_3_568

NEW SOFTWARE TOOL FOR MODELLING AND CONTROL OF DISCRETE-EVENT AND HYBRID SYSTEMS USING PETRI NETS

Erik Kučera, Oto Haffner, Peter Drahoš, Ján Cigánek

Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
e-mail: {erik.kucera, oto.haffner}@stuba.sk

Juraj Štefanovič, Štefan Kozák

Faculty of Informatics
Pan-European University
Bratislava, Slovakia
e-mail: {juraj.stefanovic, stefan.kozak}@paneurouni.com

Abstract. The main aim of the proposed paper is to design a new software tool for modelling 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 which is 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 has been successfully verified in control of laboratory systems. It can also 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 tool can improve education and practice in the field of cyber-physical systems (Industry 4.0).

Keywords: Discrete-event systems, hybrid systems, system control, microcontroller, Firmata protocol

Mathematics Subject Classification 2010: 93C65

1 INTRODUCTION

Development of various 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]. As these operations are challenging and important for the final product, it is appropriate and necessary to create a model of the system [2, 3]. Development of control methods for discrete-event and hybrid systems belongs to modern trends in automation and mechatronics. A hybrid system is a combination of continuous-time and discrete-event systems. Control of such systems brings new challenges due to the necessity to join control methods of discrete-event systems (where the Petri nets formalism can be helpful) and classic control methods of continuous-time systems [4]. With good methodology and software modules, these approaches can be synergistically combined yielding an appropriate and unique control system that allows harmonizing discrete-event and continuous-time control methods (e.g. PID algorithms). Effective cooperation of these approaches allows to control hybrid systems. This approach is useful in systems which require using different control algorithms (for example PID controllers with different parameters) according to the state of the system. The concept of Petri nets is capable to manage these control rules in a very efficient, robust and well-arranged (graphical) way. This paper presents new Petri Net tools for modelling and control of discrete-event and hybrid systems. Case studies dealing with control of a laboratory fire alarm system and a DC motor are included.

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

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

The SW tool Snoopy [12] offers modelling based on many Petri nets classes like stochastic, hybrid, colour, 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.

Coloured Petri nets are used for modelling of automated storage and retrieval system in [13] and [14].

One of interesting research approaches is the Modelica language and the OpenModelica open-source tool. There is a library that supports modelling by Petri nets in this tool. One of the advantages of OpenModelica is that a PN model can be connected with other Modelica components. The first Petri net toolbox was introduced in [15], its extension is described in [16]; an important addition (called PNlib) including support of extended hybrid Petri nets for modelling of processes in biological organisms is described in [17] and [18]. However, this tool was devel-

oped 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 COM port communication support was missing.

The above survey has shown there is a lack of tools based on Petri net formalism that support control of real systems. As a result, it was necessary to develop an original solution for control of discrete and hybrid systems based on microcontrollers using Petri nets formalism.

2 DESCRIPTION OF DEVELOPED SW TOOL PN2ARDUINO

As a basis for the newly-developed SW tool, the PNEditor was chosen [19]. This tool is open-source. The developed extension of this tool is called PN2ARDUINO and was fully tested in [20] and [21]. The main topic of this paper includes 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 systems.

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

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

There are several Petri net-based control concepts. Petri net as a control logic has to 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 PC able to communicate with the microcontroller. Both approaches have both 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 modelled 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.

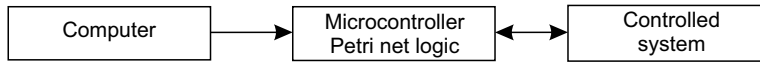


Figure 1. Basic scheme of proposed solution – Petri net's logic in microcontroller

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 [22]) 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.

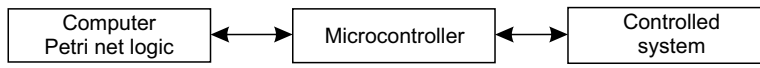


Figure 2. Basic scheme of proposed solution – Petri net's logic in PC

New software module PN2ARDUINO is based on the second approach. The Petri net runs on a personal computer. For communication between SW application and microcontroller, the Firmata protocol [22] has been used. Firmata is a protocol designed for communication between a microcontroller and a computer (or a mobile device like smartphone, tablet, etc.). It is based on MIDI messages [23]. This 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 in the possibility of using another microcontroller compatible with Firmata.

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

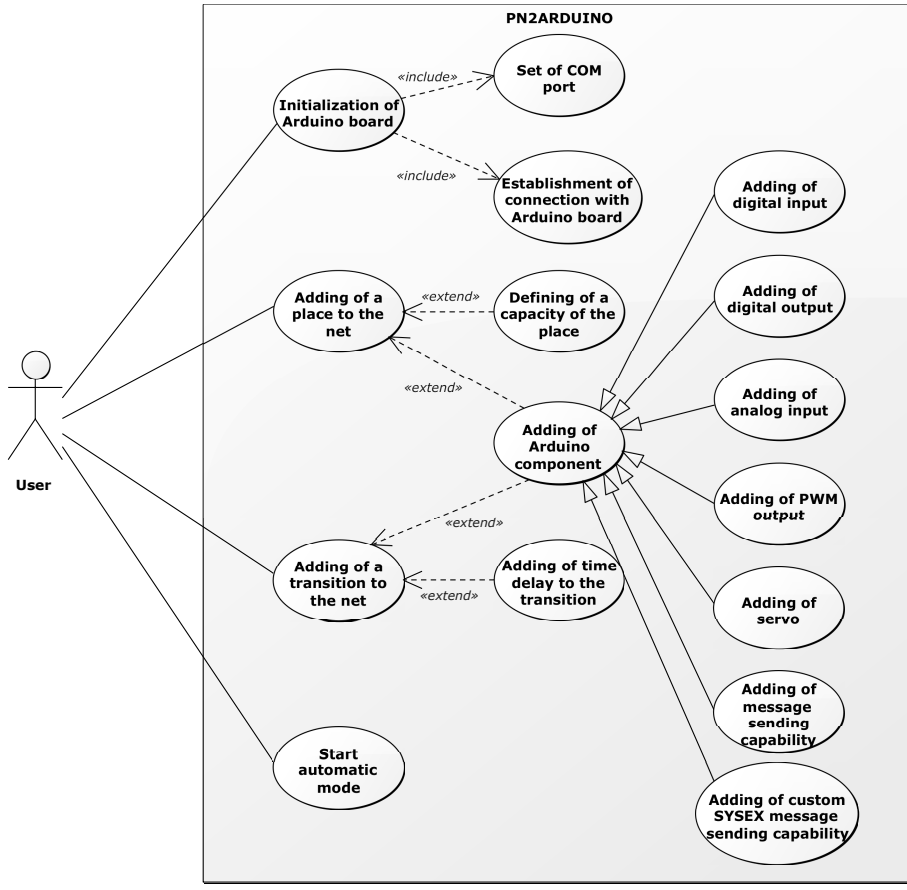


Figure 3. PN2ARDUINO – use-case diagram

types of Arduino components are supported: digital input and output, analog input, servo control, PWM output, message sending, and custom SYSEX message [22] sending.

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

As it was 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 has to 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

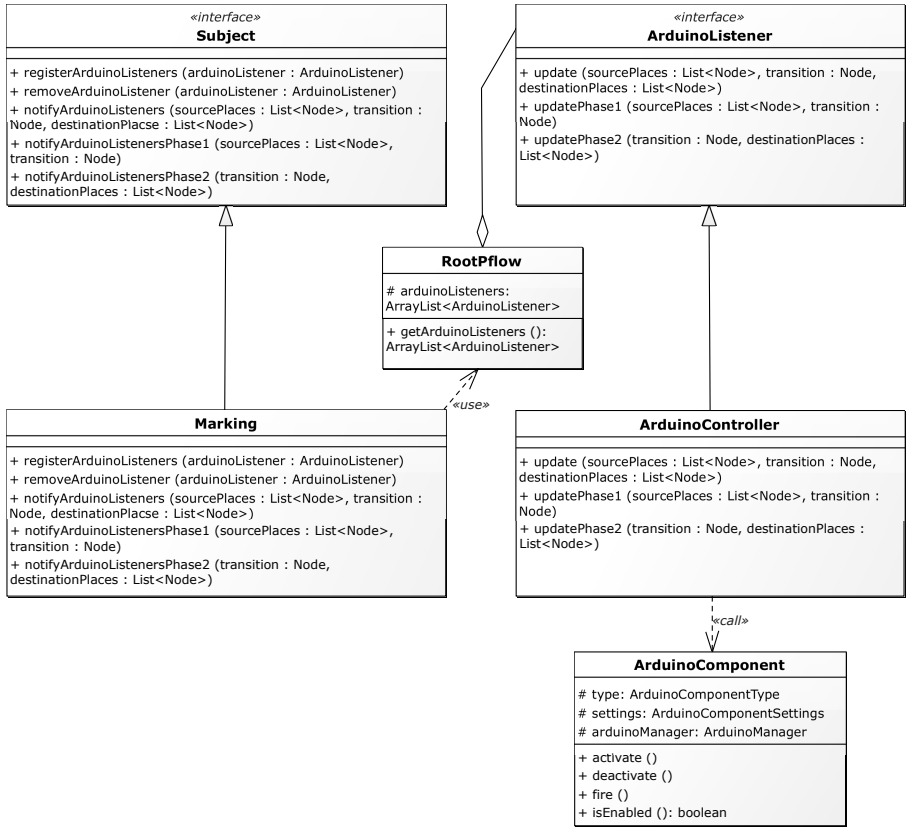


Figure 4. PN2ARDUINO – class diagram

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 modelling and log console. PN2ARDUINO supports two modes – a design mode and a control mode, the control mode can be manual or 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 to the transition (Figure 5). The example of analog input is shown in Figure 6.

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

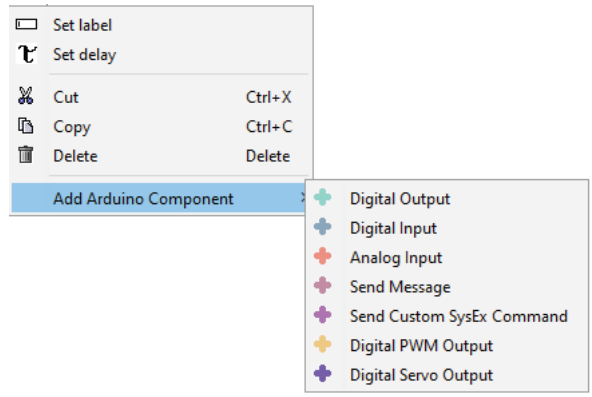


Figure 5. PN2ARDUINO – adding of Arduino component

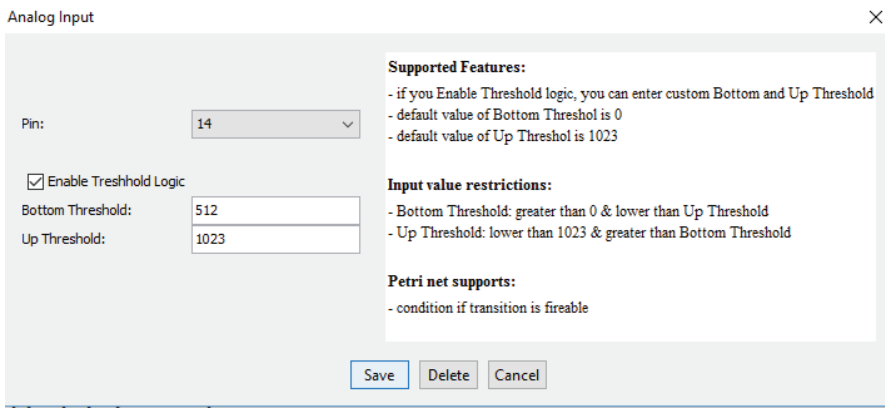


Figure 6. PN2ARDUINO – analog input

3 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, photo-resistor, 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 photo-resistor was used instead of a smoke sensor to simplify the experiment.

Next, the behaviour of the system has to be defined. When the photoresistor detects an excessive lighting (experimentally determined as an input value greater

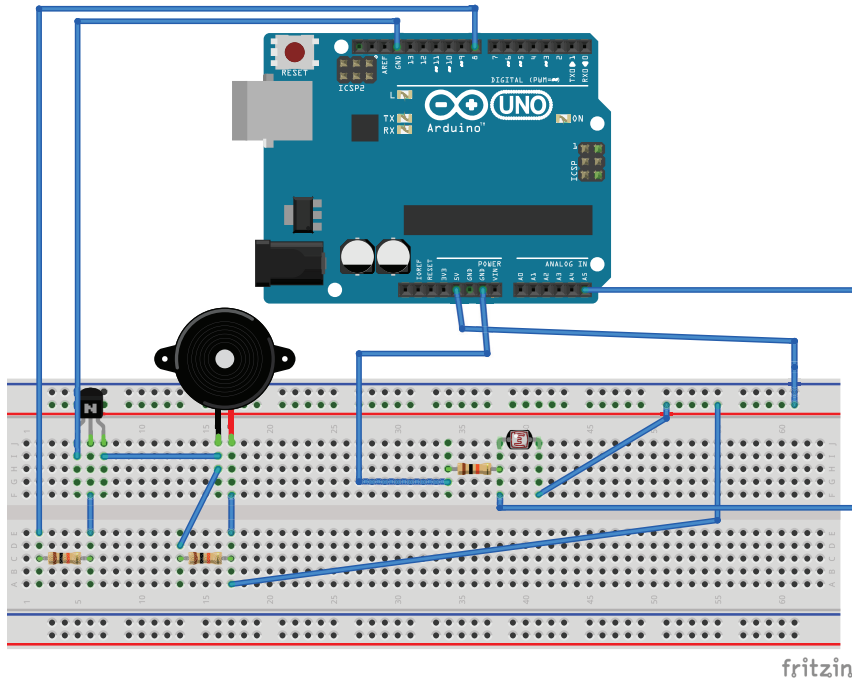


Figure 7. The scheme of laboratory model of fire alarm

than 799 on the analog pin of Arduino Uno which resolution is from 0 to 1 023) 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.

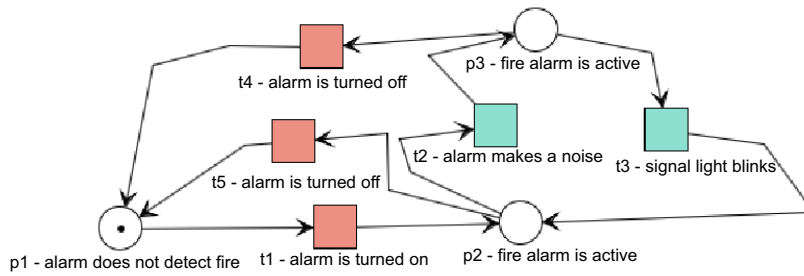


Figure 8. PN for fire alarm (initial marking)

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

Places of the Petri net (Figure 8 – Figure 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 (Figure 8 – Figure 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 photo-resistor has not detected the light intensity threshold.

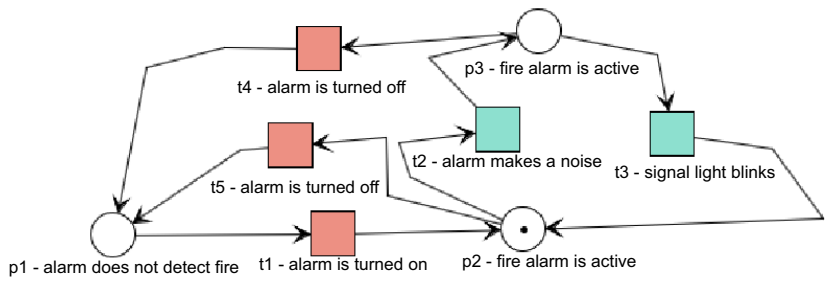


Figure 9. PN for fire alarm (t_1 is fired)

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.

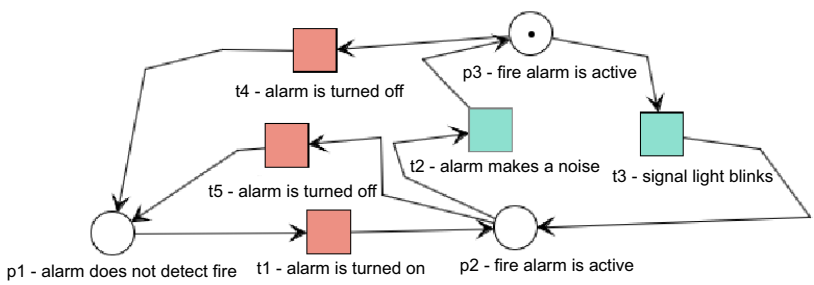


Figure 10. PN for fire alarm (t_2 is fired)

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 seconds) which means that the transition firing (and buzzer sound effect) lasts for 2 seconds.

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 second, i.e. the LED diode is turned on for 1 second.

This process is repeated cyclically, and it 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.

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

4 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.

Actuators Conditions	
Rated voltage	6.0 V (DC)
Humidity range	0 %–90 %
Temperature range	$-20\,^{\circ}\mathrm{C} \sim +60\,^{\circ}\mathrm{C}$
No-Load Characteristics	
No-load current	≤ 200 mA
No-load speed	$185 \pm 10\,\%$ rpm
Load Characteristics	
Rated load	0.0883 N.m
Rated current	≤ 550 mA
Rated speed	$135 \pm 10\,\%$ rpm
Starting torque	0.4413 N.m
Locked-rotor current	≥ 2.0 A

Table 2. Specification of DC motor

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.

The resulting steady state I/O characteristics is in Figure 11.

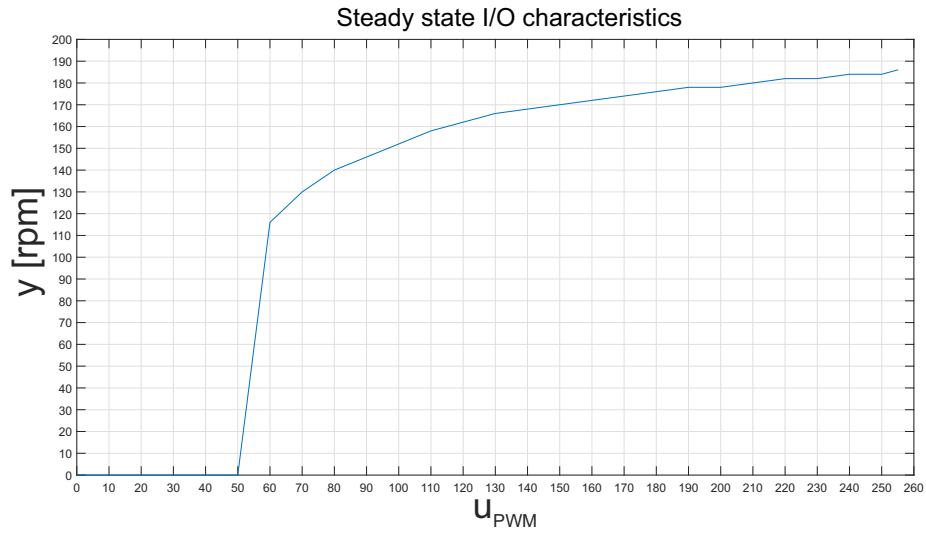


Figure 11. Steady state I/O characteristics of DC motor

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:

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

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

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 [24]. 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.

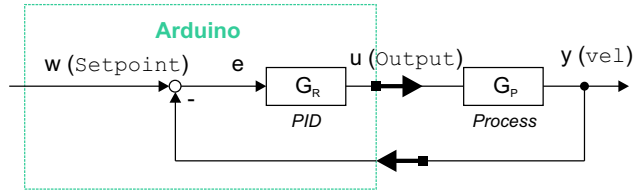


Figure 12. Block diagram of PID controller in a feedback loop

The closed-loop scheme is in Figure 12. 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 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. Step response (closed loop) – 2nd working point

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 13 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 14 that this controller does not provide a satisfactory performance.

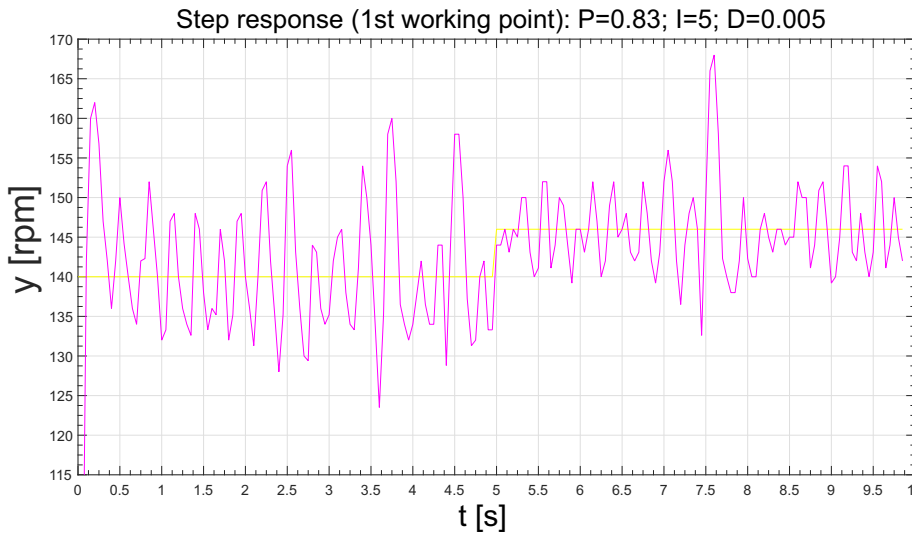


Figure 14. Step response (closed loop) – 1st working point – with inappropriate controller

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 15 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 16 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).

The analysis of the achieved results revealed the necessity to use different controllers in individual working points, or to design the controller using some ad-

Figure 15. Step response (closed loop) – 1st working point – under PID controller

vanced 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 17. Consider the above-mentioned DC motor which has to operate in two modes (working points). For effective setpoint tracking by the DC motor speed, controllers with different parameters have to 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 16. Step response (closed loop) – 2nd working point – under inappropriate controller

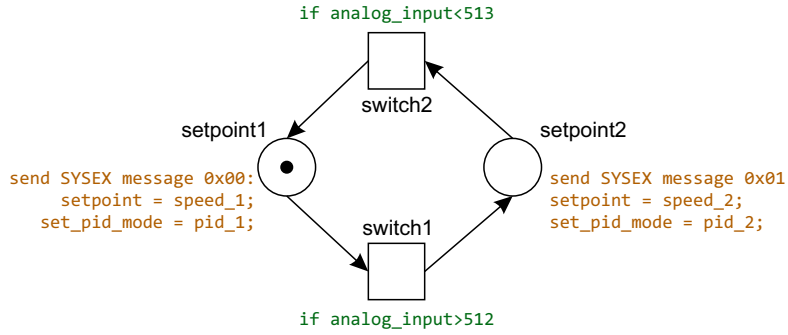


Figure 17. Control scheme for hybrid system using PN2ARDUINO

5 CONCLUSIONS

The paper presents a new software tool PN2ARDUINO extending the PNEditor to communicate with microcontrollers that support the Firmata protocol. This allows controlling discrete-event and hybrid systems using timed interpreted Petri nets with the developed software tool. The developed SW tool supports 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 comands. The main advantage of the developed SW tool is a possibility to control complex discrete-event and hybrid systems using the

benefits of Petri nets formalism which can support many challenging scenarios. The next research will focus on the concept of Petri nets based control with the control logic directly implemented in the microcontroller.

Acknowledgment

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.

REFERENCES

[1] Kharitonov, D.—Tarasov, G.—Golenkov, E.: Modeling of Object-Oriented Programs with Petri Net Structured Objects. Computing and Informatics, Vol. 36, 2017, No. 5, pp. 1063–1087, doi: 10.4149/cai_2017_5_1063.

[2] Babič, M.—Hluchý, L.—Krammer, P.—Matovič, B.—Kumar, R.—Kovač, P.: New Method for Constructing a Visibility Graph-Network in 3D Space and a New Hybrid System of Modeling. Computing and Informatics, Vol. 36, 2017, No. 5, pp. 1107–1126, doi: 10.4149/cai_2017_5_1107.

[3] Julia, S.—do Nascimento Vale, L.—Soares Passos, L. M.: Functional Testing Using Object WorkFlow Nets. Computing and Informatics, Vol. 35, 2016, No. 3, pp. 719–743.

[4] Szymczak, A.—Paszyński, M.—Pardo, D.—Paszyńska, A.: Petri Nets Modeling of Dead-End Refinement Problems in a 3D Anisotropic *hp*-Adaptive Finite Element Method. Computing and Informatics, Vol. 34, 2015, No. 2, pp. 425–457.

[5] Dotoli, M.—Fanti, M. P.—Iacobellis, G.: A Freeway Traffic Control Model by First Order Hybrid Petri Nets. 2011 IEEE Conference on Automation Science and Engineering (CASE), 2011, pp. 425–431, doi: 10.1109/CASE.2011.6042526.

[6] Fanti, M. P.—Iacobellis, G.—Mangini, A. M.—Ukovich, W.: Freeway Traffic Modeling and Control in a First-Order Hybrid Petri Net Framework. IEEE Transactions on Automation Science and Engineering, Vol. 11, 2014, No. 1, pp. 90–102, doi: 10.1109/TASE.2013.2253606.

[7] Dotoli, M.—Fanti, M. P.—Mangini, A. M.: Fault Monitoring of Automated Manufacturing Systems by First Order Hybrid Petri Nets. IEEE International Conference on Automation Science and Engineering (CASE 2008), 2008, pp. 181–186, doi: 10.1109/COASE.2008.4626493.

[8] Costantino, N.—Dotoli, M.—Falagario, M.—Fanti, M. P.—Mangini, A. M.: A Model for Supply Management of Agile Manufacturing Supply Chains. International Journal of Production Economics, Vol. 135, 2012, No. 1, pp. 451–457, doi: 10.1016/j.ijpe.2011.08.021.

[9] Matsuno, H.—Doi, A.—Drath, R.—Miyano, S.: Genomic Object Net: Object Oriented Representation of Biological Systems. Genome Informatics, Vol. 11, 2000, pp. 229–230.

[10] Drighiciu, M. A.—Manolea, G.: Application des Reseaux de Petri Hybrides a l'Etude des Systemes de Production a Haute Cadence. 2010 (in French).

[11] Drighiciu, M.—Cismaru, D.: Modeling a Water Bottling Line Using Petri Nets. Annals of the University of Craiova, Electrical Engineering Series, 2013, No. 37, pp. 110–115.

[12] Rohr, C.—Marwan, W.—Heiner, M.: Snoopy – A Unifying Petri Net Framework to Investigate Biomolecular Networks. Bioinformatics, Vol. 26, 2010, No. 7, pp. 974–975, doi: 10.1093/bioinformatics/btq050.

[13] Kučera, E.—Nižnanská, M.—Kozák, Š.: Advanced Techniques for Modelling of AS/RS Systems in Automotive Industry Using High-Level Petri Nets. 2015 16th International Carpathian Control Conference (ICCC), IEEE, 2015, pp. 261–266, doi: 10.1109/CarpathianCC.2015.7145085.

[14] Kučera, E.—Haffner, O.—Kozák, Š.: Modelling and Control of AS/RS Using Coloured Petri Nets. 2016 Cybernetics & Informatics (K & I), IEEE, 2016, pp. 1–6, doi: 10.1109/CYBERI.2016.7438532.

[15] Mosterman, P. J.—Otter, M.—Elmqvist, H.: Modeling Petri Nets as Local Constraint Equations for Hybrid Systems Using Modelica. Available at: `https://www.modelica.org/publications/papers/scsc98fp.pdf`, 1998.

[16] Fabricius, S. M. O.—Badreddin, E.: Modelica Library for Hybrid Simulation of Mass Flow in Process Plants. Proceedings of the 2nd International Modelica Conference, Oberpfaffenhofen, Germany, Citeseer, 2002, pp. 225–234.

[17] Pross, S.—Bachmann, B.: A Petri Net Library for Modeling Hybrid Systems in OpenModelica. Proceedings 7th Modelica Conference, Como, Italy, 2009, pp. 454–462, doi: 10.3384/ecp09430014.

[18] Pross, S.—Bachmann, B.: PNlib – An Advanced Petri Net Library for Hybrid Process Modeling. Proceedings of the 9th International Modelica Conference, Munich, Germany, 2012, pp. 47–56, doi: 10.3384/ecp1207647.

[19] Riesz, M.—Sečkár, M.—Juhás, G.: PetriFlow: A Petri Net Based Framework for Modelling and Control of Workflow Processes. In: Donatelli, S., Kleijn, J., Machado, R. J., Fernandes, J. M. (Eds.): Recent Advances in Petri Nets and Concurrency. CEUR Workshop Proceedings, Vol. 827, 2012, pp. 191–205.

[20] Češeková, A.: Control of Laboratory Discrete Event Systems. Master's thesis, Slovak University of Technology in Bratislava, 2016 (in Slovak).

[21] Kučera, E.: Modelling and Control of Hybrid Systems Using High-Level Petri Nets. Ph.D. Dissertation, Slovak University of Technology in Bratislava, 2016 (in Slovak).

[22] Steiner, H.: Firmata: Towards Making Microcontrollers Act Like Extensions of the Computer. Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), 2009, pp. 125–130, doi: 10.5281/zenodo.1177689.

[23] MIDI Association: Summary of Midi Messages. Available at: `https://www.midi.org/specifications/item/table-1-summary-of-midi-message`, 2016.

[24] Comnes, B.—La Rosa, A.: Arduino PID Example Lab. Portland State University, 2013.

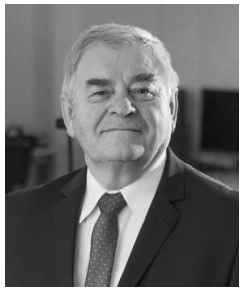
Erik Kučera graduated from the Slovak University of Technology, Faculty of Electrical Engineering (FEI STU) in Bratislava in 2013 and obtained his Ph.D. in mechatronic systems in 2016. He works at the Institute of Automotive Mechatronics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia. His focus is mainly on modern information and communication technologies and their use in the context of fourth industrial revolution Industry 4.0. This includes e.g. internet of things, virtual and mixed reality, cloud computing, new microcontrollers, etc.

Oto Haffner graduated from the Slovak University of Technology, Faculty of Electrical Engineering (FEI STU) in Bratislava in 2013 and obtained his Ph.D. in mechatronic systems in 2016. He works at the Institute of Automotive Mechatronics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia. His focus is mainly on modern machine vision methods and their use in the context of fourth industrial revolution Industry 4.0. This includes e.g. artificial intelligence, deep learning, etc.

Peter Drahoš graduated from the Slovak University of Technology, Faculty of Electrical Engineering (FEI STU) in Bratislava in 1985 and obtained his Ph.D. in automation and control in 2003. His main research interests include smart material actuators, sensors and automatic control. He is also interested in industrial communication systems. Since 2012, he is with the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, holding now the position of Associate Professor.

Ján Cigánek received his diploma and Ph.D. degree in automatic control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 2005 and 2010, respectively. He is now Assistant Professors at the Institute of Automotive Mechatronics FEI STU in Bratislava. His research interests include optimization, robust control design, computational tools, and hybrid systems.

Juraj Štefanovič graduated from the Faculty of Electrical Engineering of the Slovak University of Technology in Bratislava, Slovakia in 1987 in microelectronic engineering, and received his Ph.D. in applied informatics from the Slovak University of Technology in 2000. He was a researcher at the Slovak Academy of Sciences until 1992. Since 1992 he was with the Slovak University of Technology in Bratislava, the Faculty of Electrical Engineering/Faculty of Informatics and Information Technologies at the Department of Informatics until 2014. Currently he is with the Institute of Applied Informatics at the Faculty of Informatics, Pan-European University in Bratislava as Vice-Dean for Research and International Relations. He is also the Executive Editor of International Journal of Information Technology Applications (ITA) and he is a manager of regional competition of First Lego League (robotic competition for youth teams) in Bratislava-Petržalka. His research interests include modelling and simulation – discrete dynamic models and complex models, his teaching activity covers also the field of operating systems and design for usability.



Štefan Kozák obtained his M.Sc. from the Slovak University of Technology in Bratislava in 1970 and his Ph.D. in technical cybernetics from the Slovak Academy of Sciences in 1978. He worked at the Institute of Technical Cybernetics, Slovak Academy of Sciences, in the field of control algorithms design and was Leader of a Research Team at the Institute of Applied Cybernetics in Bratislava. Since 1984 he was with the Department of Automatic Control Systems at the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava. Currently he is with the Institute of Automotive Mechatronics, Slovak University of Technology in Bratislava. His research interests include system theory, linear and nonlinear control methods, numerical methods and software for modeling, control, signal processing and embedded intelligent systems. He published more than 220 research papers in conference proceedings and international journals, and he organized several IFAC events held in Slovakia.