Modeling, Simulation, and Control of Smart Homes Using Petri Nets

Azza K. Nabih[1], Mostafa M. Gomaa[2,*], Hossam S. Osman[2], Gamal M. Aly[2]

[1]*Software Engineering Competence Center, ITIDA,*
Smart Village, 6th October, Egypt 12577
azkamal@itida.gov.eg

[2] *Department of Computer and Systems Engineering, Ain Shams Univ.,*
Abbasia, Cairo, Egypt 11517
[*]*Mostafa.gomaa@eng.asu.edu.eg, hosman@itida.gov.eg, gmaly@itida.gov.eg*

Abstract

Smart home is a relatively new technology. Originally, smart home technology was used to control environmental systems such as lighting and heating; but recently the use of smart technology has been developed so that almost any electrical component within the home can be included in the system. Complex smart home applications include three levels of control: local control, discrete-event control, and supervisory control. The local control, at the lower level, handles the operating conditions of the continuous time physical variables (e.g. temperature, light, level, etc.). The discrete-event control is meant by the asynchronous events and accordingly issues the corresponding discrete actions, based on the evolution specification. The supervisory control, at the upper level, takes care of the resources allocation, activities coordination, and deadlock avoidance.

This paper proposes a technique that employs the Petri net tools to model, simulate, analyze, and control at the discrete-event level the smart home applications. Petri nets are proved to be suitable formal models that can be used to verify the operation of smart homes at the simulation level. Besides, it is very easy to get an executable version of the model for real-time implementation.

Keywords: *Smart Home, Petri Nets, State-Based Models, Discrete-Event Control.*

1. Introduction

System development usually starts with a high-level model and proceeds through a process of refinement, simulation or emulation, verification, implementation, and test. Much of the process used for smart home systems design is ad-hoc. By the time, all the gritty details of implementation are taken care with the original system description has pretty much been lost, causing a lack of design oversight and a surplus of one-time-only design artifacts.

To combat the ad-hoc nature of the smart home systems design process, it's required to utilize a suitable formal model that is able to handle smart home domain specific nature. Complex smart home applications include three levels of control: local control, discrete-event control, and supervisory control. The local control, at the lower level, handles the operating conditions of the continuous time physical variables. The discrete-event control handles the asynchronous events by issuing corresponding discrete actions, based on the evolution recipe. The supervisory control, at the upper level, achieves resources allocation, activities coordination, and deadlock avoidance.

In addition to having various control levels in the smart home, there are wide interactions between the different smart home modules, the environment, and the inhabitants. Moreover, smart home environments are dynamic in such a way that devices, systems, and services constituting the smart home can be changed at run time. In addition, several smart home systems are required to run concurrently.

The initial efforts towards formality were the state-based models (such as Finite State Machines). Though, they are proved to be inadequate. They suffer from the state explosion problems; any design flaws or mistakes could invalidate the entire model; any system specification changes could require tremendous effort to modify the design [1]; and they cannot explicitly model concurrency.

This paper employs the Petri net tools to model, simulate, analyze, and control smart home systems at the discrete-event level. Petri nets can model concurrency [2] and overcome other state machine problems. Petri net models provide the following advantages:

- Graphical representation of the system specification (modeling).
- Guarantee of consistency of the design and the ability to verify the system behavior at the simulation level before its actual implementation.
- The effectiveness of studying the properties of the system through the model.
- The ability to design discrete event controllers for the individual smart home subsystems and the ability to model concurrency of the various systems.
- The ability to design a supervisory controller that coordinates, allocates resources, avoids deadlock situations that are related to the smart home operation as a whole.
- Modularity of the design that provides better chances for reuse.
- The direct mapping of the model into source code which reduces the development time very much and the freedom to use any programming language according to programmers' experience and other system interconnections.

The contribution of this paper can be summarized as follows: *i)* proposing Petri net controllers that formally capture the complex specifications of the smart home domain; *ii)* proposing an algorithm that can be easily used for model simulation and realization; *iii)* verifying the proposed model and algorithm through applying it on a proposed smart home application, giving the simulation results that are analyzed to evaluate the model efficiency.

This paper is organized as follows: Section 2 presents a demonstration smart home that contains various modules that are required to run concurrently to provide a convenient life for the home inhabitants and increase their comfort. Section 3 presents the Petri net controllers of the proposed smart home modules and highlight the efficiency of the model to capture concurrency. Section 4 presents the proposed Petri net model simulation/realization algorithm. Section 5 gives the simulation results and Section 6 presents final conclusions.

2. Demonstration Smart Home System

Smart homes contain a large set of services that cooperate to simplify the life of inhabitants, to make energy saving, and to provide comfort and security solutions [3]. These services include, but are not limited to: resource repository and management (e.g. water, electricity, gas) [4], system security [5], home devices monitoring and control [5-6], preference model and service provision [7-8], remote management [9-12], elderly assistance [9], indoor and outdoor environment facilities [13-14]. Figure 1 presents a demonstration smart home, where various sensors and actuators are installed to constitute a cooperating environment that provides smart services to inhabitants. Table 1 lists the meaning of the symbols used to represent sensor and actuator types.

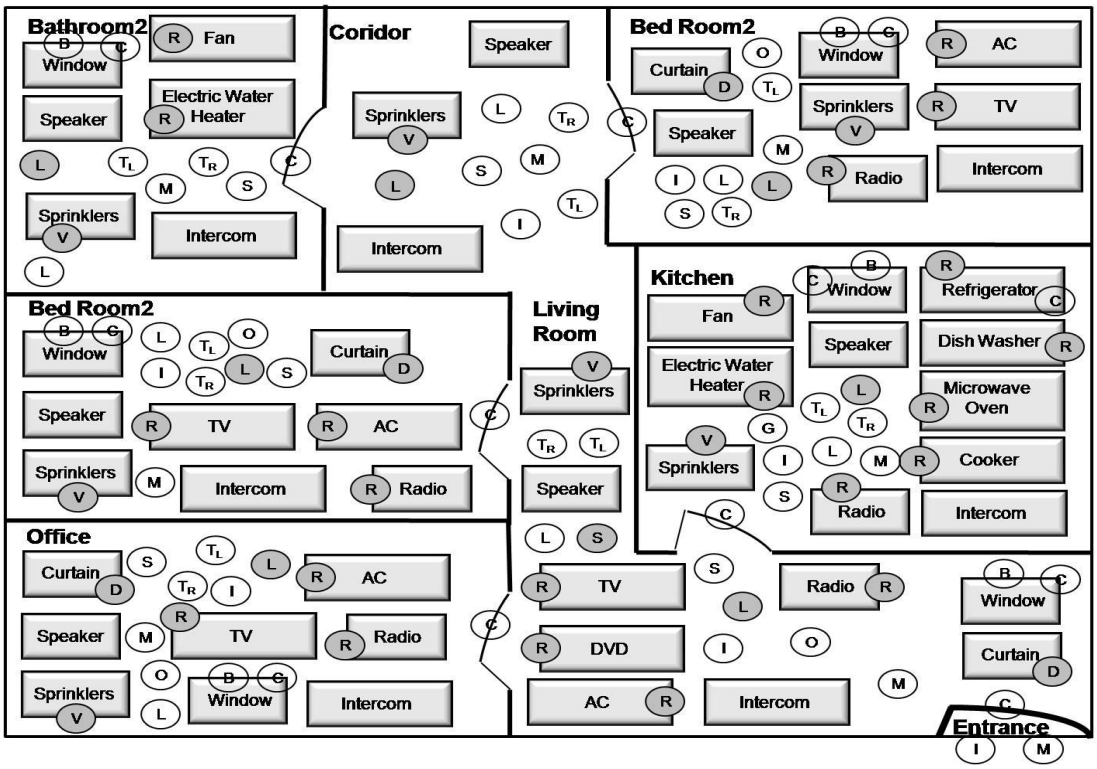

Figure 1 Proposed Demonstration Smart Home.

Table 1 List of Smart Home Sensors and Actuators.

Symbol	(M)	(T_L)	(T_R)	(G)	(C)	(I)
Meaning	Motion Detector	Temperature Level Sensor	Temperature Rate Sensor	Gas Detector	Contact Switch	Image Sensor
Symbol	(B)	(L)	(S)	(L_h)	(R)	(O)
Meaning	Glass Break Sensor	Light Sensor	Smoke Detector	Level Switch	Rain Sensor	Occupancy Sensor
Symbol	(L)	(D)	(R)	(S)	(V)	
Meaning	Lights	DC Motor	Relay	Siren/Horn	On/Off Valve	

Smart home guarantees the security and safety of inhabitants through utilizing information from temperature level sensors, temperature rate sensors, and smoke detectors to detect the possibility of fire in any place and take immediate actions. Sirens are installed everywhere to alarm home inhabitants and sprinklers will be opened automatically upon ensuring the existence of fire. In the kitchen, where there is a possibility of gas leakage, a gas detector is installed. Gas detection system keeps an open eye so that it detects any gas leakage and takes immediate action by closing the

gas valve and alarming the home inhabitants. There are image sensors and motion detectors in all places to identify any visitor and check whether he is authorized or not. Glass break sensors are also installed on all windows to monitor their safety.

Smart home provides energy saving solutions through installing occupancy sensors in all places so that they can provide information on whether the place is occupied or not. This information will be utilized in home systems that consume power like lighting and HVAC, so that lights and air conditioners (ACs) will be turned off automatically in the non occupied places. Systems like lighting, curtain, and climate will consider environmental conditions for their operation to help saving more energy.

For user comfort, contact switches are installed on all windows and doors, so that home owner can check for their open/close status and change it remotely. Relays are attached to all devices to be able to control them remotely by just sending the required signal to the relay that will open/close the device accordingly. Curtains will also have smart controllers. By installing DC motors for curtains, both curtain angel and height can be changed by button presses. Intercoms will be there in all places to facilitate communication between inhabitants within the home. Multimedia devices will be distributed in appropriate places, different multimedia zones will exist, with each zone having its own multimedia playlist and controls.

3. Control of Smart Home Modules Using Petri Nets

Smart home services can be organized in a form of modules. Each module is responsible for certain functionality. E.g. lighting module is responsible for the smart control of the smart home lights. Fire module is responsible for alarming the home inhabitants of any possible fires within the home. In such way, smart home can have different modules as follows: lighting system, fire system, curtain control system, intercom system, climate control system, appliance control system, gas detection system, multimedia control system, remote access system, and security system. In case of the existence of a garden, irrigation module can be easily added to the set of smart home modules. Petri nets can model the different smart home modules, their interactions, and their dynamic behavior.

Being able to model dynamic nature of smart home, the whole Petri net model of smart home modules can be easily maintained by adding/removing modules to/from the overall smart home as required within specific homes and according to specific user preferences.

Preset user modes require the execution of tasks from different modules concurrently. The "*wakeup mode*" for example comprises the following concurrent tasks: automatically control light dimming level, curtains, and AC according to environmental conditions and place occupancy, run zone multimedia playlist, disable remote access, and stop security system. This scenario involves the following smart home modules: lighting system, curtain control system, climate control system, multimedia control system, remote access system, and security system. Same, scenarios has been proposed for "*leave home*" mode, "*back home*" mode, "*relaxing evening*" mode, and "*go to bed*" mode.

To show the power of Petri net modeling, details of the smart home modules will be presented as per the concurrency required for the "*leave home*" mode, which is represented as the event (E2) in the modules' Petri net models.

The "*leave home*" scenario comprises seven concurrent tasks: operate fire system, turn off lights, close curtains, turn off AC, stop multimedia playlist, enable remote access, and operate security system. This scenario involves seven smart home modules: fire system, lighting system, curtain control system, climate control system, multimedia control system, remote access system, and security system. Due to paper space limitation, only Petri net

models of the fire system, the climate control system, and the remote access system will be presented to highlight the concept of concurrency. The idea can be further extended to any number of concurrent modules.

3.1. Fire System

The fire system is divided into 13 similar fire sub-modules, covering all the home parts. In each fire sub-module, the fire system monitors temperature level, temperature rate change, and smoke in order to take the appropriate actions to protect the smart home against the fire. Having high temperature level with temperature rate increase, then alarms are launched immediately. If the temperature rate is still in increase or a smoke is detected, then automatic fire extinguisher is started, as these indications ensure the existence of fire. The home owner has the option to enable or disable the fire system, besides it is enabled/disabled as a part of the scenario of some selected user modes. This system is enabled automatically upon "*leave home*" mode.

Figure 2 shows the Petri net model/controller of a sub-module within the fire system, which satisfies the specifications above, noting that the Petri net controllers of the 13 sub-modules are run concurrently. In Figure 2, the set of places can be classified into two main categories as follows:

- *Places* representing *partial states*: Pf1: the fire sub-module has normal temperature; Pf2: there is elevated temperature; Pf6: fire system is enabled; Pf7: fire system is disabled. A token in a place, representing a partial state, indicates this partial state.
- *Places representing operations*: Pf3: launch fire alarm; Pf4: launch automatic fire extinguisher; Pf5: start timer. A token in a place representing an operation indicates that this operation is started, while a non-token indicates that this operation is stopped.

All the transitions of the fire sub-module are synchronized ones. The description of the events, associated to the transitions, is given in Table 2. Concerning the events Ef1 and Ef2, in Table 2, the "*certain value*" is a predetermined temperature value assigned depending on the fire sub-module nature. Similarly the "*certain threshold*" concerning Ef5.

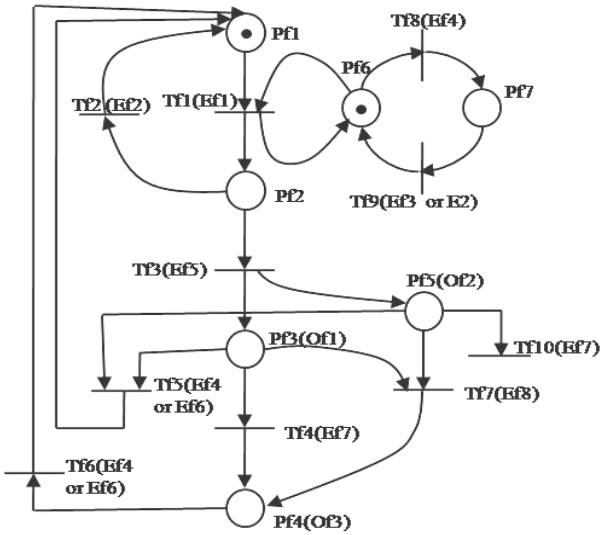

Figure 2 Petri Net Model/Controller of a Fire Sub-module.

Table 2 List of Events Associated to Synchronized Transitions of a Fire Sub-Module.

Event	*Description*	*Event*	*Description*
E2	Leave home mode	Ef5	Temperature rate >= "*certain threshold*"
Ef1	Temperature >= "*certain value*"	Ef6	Reset
Ef2	Temperature < "*certain value*"	Ef7	Smoke existence
Ef3	Operate fire system	Ef8	Timeout
Ef4	Stop fire system		

3.2. Climate Control System

Climate control system is the part of the smart home system that provides smart control of the smart home air conditioning systems to provide the maximum possible comfort and effective usage of such systems within the smart home. It turns on/off ACs according to the occupancy of places, leading to energy saving. Besides, ACs settings are automatically adjusted according to environmental conditions in order to increase user comfort and participate in the energy saving goal. Still the user has the option to manually control ACs; to turn them on/off, or to change their settings according to his preferences. ACs are turned on/off as part of some user mode scenarios. The "*leave home*" mode turns ACs off; the "*wakeup*", "*back home*", and "*relaxing evening*" modes automatically control the ACs according to occupancy and environmental conditions. Figure 3 shows the Petri net model of a climate control sub-module that satisfies the listed specifications above. In Figure 3, the set of places can be classified into two main categories as follows:

- *Places* representing *partial states*: Pm1: initial AC settings, Pm4: AC is operating in manual mode, Pm5: AC is operating in automatic mode.

- *Places* representing *operations*: Pm2: turn off AC through local AC control system; Pm3: operate local AC control system; Pm6: start timer; Pm7: operate local AC control system.

All the transitions of the climate control sub-module are synchronized ones. The description of the events, associated to the transitions, is given in Table 3.

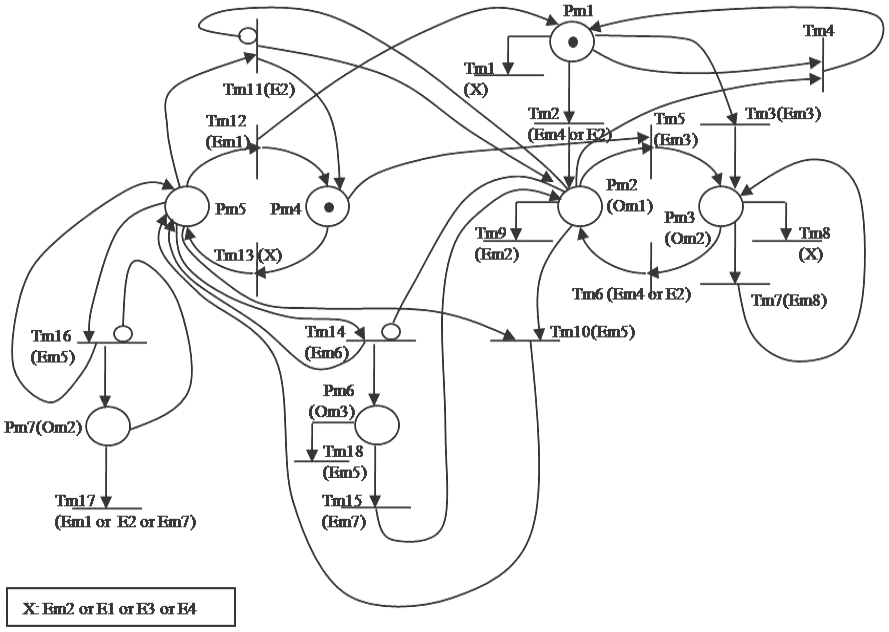

Figure 3 Petri Net Model/Controller of a Climate Control Sub-module.

Table 3 List of Events Associated to Synchronized Transitions of a Climate Control Sub-module.

Event	*Description*	*Event*	*Description*
E1	"*Wakeup*" mode	Em3	Turn on AC
E2	"*Leave home*" mode	Em4	Turn off AC
E3	"*Back home*" mode	Em5	Occupied place
E4	"*Relaxing evening*" mode	Em6	Non-occupied place
Em1	manual mode	Em7	Timeout
Em2	automatic mode	Em8	Change AC settings (Temp–Fan speed …)

3.3. Remote Access System

Remote access system is the part of the smart home system that enables home inhabitants to monitor and control their homes from outside the home, after appropriate user authentication process. Unauthorized access trials will be reported to the home owner. The home owner has the option to enable or disable this feature. This feature is enabled automatically as part of the "*leave home*" mode scenario, and disabled automatically as part of "*wakeup*" and "*back home*" mode scenarios. Figure 4 shows the Petri net model of the remote access system that satisfies the listed specifications above. In Figure 4, the set of places can be classified into two main categories as follows:

- *Places* representing *partial states*: Pr1: no one is logged in to the smart home system remotely, Pr5: remote access of the smart home system is enabled, Pr6: remote access of the smart home system is disabled.

- *Places* representing *operations*: Pr2: check user authentication; Pr3: report home owner; Pr4: monitor and control home.

All the transitions of the remote access system are synchronized ones. The description of the events, associated to the transitions, is given in Table 4.

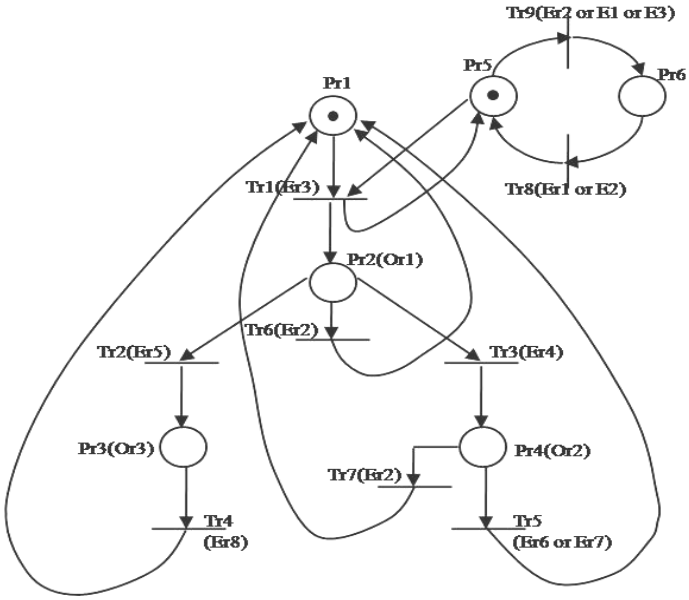

Figure 4 Petri Net Model/Controller of the Remote Access System.

Table 4 List of Events Associated to Synchronized Transitions of the Remote Access System.

Event	*Description*	*Event*	*Description*
E1	"*wakeup*" mode	Er4	Authorized user
E2	"*leave home*" mode	Er5	Unauthorized user
E3	"*back home*" mode	Er6	Logout
Er1	Enable remote access	Er7	Inactive timeout
Er2	Disable remote access	Er8	Reporting unauthorized user access is complete
Er3	Login		

4. Petri Net Model Simulation/Realization Algorithm

During simulation of the Petri net model, the events are simulated and the state transitions are observed (marking evolution). In real time, events acquisition is carried out and resources state is updated during Petri net evolution, and operations are launched according to place

markings. The proposed algorithm detailed hereafter can be used to verify a Petri net model at the simulation level or to have an executable form of the model for real time realization. The algorithm is modular and it can be easily implemented in any programming language. Details of the algorithm in pseudo code, concerning one Petri net model, are as follows:

1. **Initialization:**
   Assign the Petri net model incidence matrix *D* (encoding the Petri net structure) [15-16]; an event-transition matrix *ET* (*ET(T_i)(E_j)=1* if the transition T_i is synchronized with the event E_j; *ET(T_i)(E_j)=0* else); the Petri net initial marking vector M_0 (encoding initial state); a list of places having inhibitory arcs *PI* (place-transition couples making inhibitory arcs); and a list of places having self loops *PS* (place-transition couples making self loops). There are also a transition vector T (*T(T_i)=1* if T_i is enabled; *T(T_i)=0* else); an event vector E ((*E(E_i)=1* if the event E_i is active (or occurred); *E(E_i)=0* else); and a firing count vector *V* (*V(T_i)=1* if T_i is to be fired; *V(T_i)=0* else) which are initialized to zeros.

2. **Find all enabled transitions:**
   2.1. *FOR* each transition T_j in the transition vector *T*
   T(T_j)=1 // Set all transitions to the enabled state
   NEXT T_j
   2.2. *FOR* each transition T_j in the transition vector *T*
   FOR each place P_i in the marking vector *M* //*M[i]* equals the marking of the P_i
   IF (the place P_i is an input place to the transition T_j (where *D[P_i][T_j]<=-1 OR (D(P_i)(T_j)=0 AND* {*there is an entry [P_i, T_j] in the PS list OR there is an entry (P_i, T_j) in the PI list*})) *THEN*
   IF M(i) violates enabling condition of the transition T_j *THEN*
   T(T_j)=0; i.e. this transition is not enabled.
   END IF
   END IF
   NEXT P_i
   NEXT T_j

3. **Find first enabled unsynchronized transition:**
   3.1. Set the firing count vector (*V*) to zeros.
   3.2. *FOR* each transition T_j in the transition vector *T*
   IF (*T(T_j)=1*) *AND* (*ET(T_j,:)=0*) *THEN* //i.e. enabled-unsynchronized transition
   V[T_j] = 1
   EXIT FOR
   END IF
   NEXT T_j

4. *DO WHILE* (*V(:) != 0*) //i.e. there is an enabled unsynchronized transition T_j
   Fire this enabled unsynchronized transition T_j //by applying the Petri net state equation [16]
   Find all enabled transitions by calling step 2
   Find first enabled unsynchronized transition by calling step 3
   LOOP

5. Acquire events and update the event vector *E*

6. *FOR* each event (*E_j*) in the event vector *E*
   IF (*E(E_j)=1*) *THEN* // i.e. the event E_j is occurred

6.1. Set the firing count vector (V) to zeros.
FOR each transition T_i in the transition vector T
IF ($T(T_i)=1$) *AND* ($ET(T_i,E_j)=0$) *THEN*
$V(T_i) = 1$ //*i.e.* T_i is the first enabled synchronized transition with E_j
EXIT FOR
END IF
NEXT T_i
6.2. *DO WHILE* ($V(:) != 0$) //i.e. there is an enabled synchronized transition T_i
Fire this transition T_i
Find all enabled transitions by calling step 2
Find first enabled unsynchronized transition T_j, by calling step 3
Fire all enabled unsynchronized transitions by call step 4
Find first enabled synchronized transition T_i associated to the occurred event E_j, by calling step 6.1
LOOP
E(E_j) =0 //i.e. the event E_j is handled
END IF
NEXT E_j

5. Simulation Results

It is required to verify the proposed algorithm presented in the previous section and prove the efficiency of using Petri net to model smart home modules. Hence, an implementation of the algorithm is carried out using C++ programming language and it is used to verify the correctness and efficiency of the Petri net models of all the smart home modules mentioned in Section 2. The simulation results concerning a fire sub-module are given, as an example.

The initial state is indicated by the initial marking of the Petri net of Figure 2; i.e. the fire sub-module is activated and the home has normal temperature. The marking vector is a 7x1 vector; it is interpreted as:

$$M_k = [m_{Pf1} \quad m_{Pf2} \quad m_{Pf3} \ldots\ldots m_{Pf7}]^T \tag{1}$$

and thus,

$$M_0 = [1\ 0\ 0\ 0\ 0\ 1\ 0]^T \tag{2}$$

The incidence matrix D is of size 7x10, and is listed as follows:

$$\begin{aligned} &D(i,j)= -1, \text{ for } (i,j) \in \{(0,0), (1,1), (1,2), (2,3), (2,4), (2,6), (3,5), (4,4), (4,6), \\ &\qquad (4,9), (5,7), (6,8)\} \\ &D(i,j)= 1, \text{ for } (i,j) \in \{(0,1), (0,4), (0,5), (1,0), (2,2), (3,3), (3,6), (4,2), (5,8), \\ &\qquad (6,7)\} \\ &D[i,j]= 0, \text{ otherwise} \end{aligned} \tag{3}$$

The event-transition matrix ET is of size 10x9, and is listed as follows:

$$\begin{aligned} &ET[i,j]= 1, \text{ for } (i,j) \in \{(0,1), (1,2), (2,5), (3,7), (4,4), (4,6), (5,4), (5,6), (6,8), \\ &\qquad (7,4), (8,0), (8,3), (9,7)\} \\ &ET[i,j]= 0, \text{ otherwise} \end{aligned} \tag{4}$$

The list of places having inhibitory arcs PI has no entries. The list of places having self loops PS has one entry indicating the place Pf6 and the transition Tf1 as shown in Figure 2. The event vector is a 9x1 vector; it is interpreted as (c.f. Table 2):

$$E_k = [E_2 \quad E_{f1} \quad E_{f2} \quad E_{f3} \quad E_{f4} \quad E_{f5} \quad E_{f6} \quad E_{f7} \quad E_{f8}]^T \tag{5}$$

Figure 5 shows the simulation results in the form of markings and their associated actions as results of simulating the following sequence of events: E1: Temperature>="*certain value*", E2: Temperature rate>="*certain threshold*", E3: Timeout, E4: Reset, E5: Temperature>="*certain value*", E6: Temperature<"*certain value*", E7: Temperature>= "*certain value*", E8: Temperature rate>="*certain threshold*", E9: Smoke existence, E10: Stop fire system, E11: "*Leave home*" mode. Table 3 lists the event vectors, if the form of (5), and Table 4 lists the marking vectors, in the form of (1).

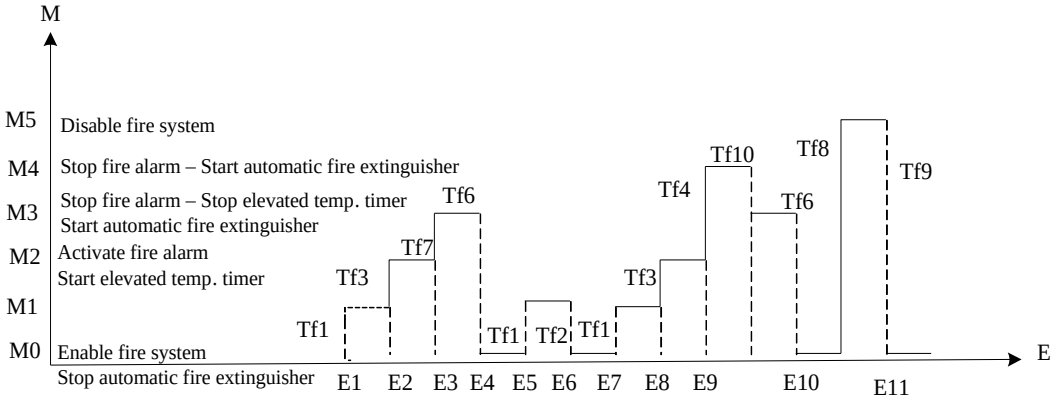

Figure 5 Fire Sub-system Simulation Results.

Table 5 Event Vectors Used for fire Sub-system Simulation.

E_k	*Its content*	E_k	*Its content*
E_1	$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$	E_7	$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$
E_2	$[0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$	E_8	$[0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$
E_3	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T$	E_9	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T$
E_4	$[0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T$	E_{10}	$[0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$
E_5	$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$	E_{11}	$[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$
E_6	$[0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$		

Table 6 Reachable Marking Vectors Obtained by Fire Sub-system Simulation Scenario.

M_k	*Its content*	M_k	*Its content*
M_0	$[1\ 0\ 0\ 0\ 0\ 1\ 0]^T$	M_3	$[0\ 0\ 0\ 1\ 0\ 1\ 0]^T$
M_1	$[0\ 1\ 0\ 0\ 0\ 1\ 0]^T$	M_4	$[0\ 0\ 0\ 1\ 1\ 1\ 0]^T$
M_2	$[0\ 0\ 1\ 0\ 1\ 1\ 0]^T$	M_5	$[1\ 0\ 0\ 0\ 0\ 0\ 1]^T$

The simulation results show that the model is responding as expected, which verifies the model and the proposed algorithm. To study the Petri net model properties, a reachability graph can be obtained. It allows for studying properties such as [15-16]:

- boundedness (a Petri net is set to be k-bounded if the number of tokens in any place is always less or equal to k);
- *safety* (a Petri net is safe if it is 1-bounded);
- liveness (this implies that for all markings M, which are reachable from the initial marking M0, it is ultimately possible to fire any transition in the net by progressing through some firing sequence);
- *deadlock* (it is a marking such that no transition is enabled);
- *conflict* whether structural or effective; etc.

Reachability graph is based on the enumeration of all possible markings reachable from the initial marking M_0. In the reachability graph, each node is labeled with a marking; arcs are labeled with transitions. The root node of the tree is labeled with an initial marking M_0. From the obtained reachability graph, it is concluded that the fore sub-module has no deadlock state; all reachable states are admissible; state sequence according to events is admissible.

6. Conclusions

This paper presents a new formalism utilizing Petri net tools to model, analyze, and implement smart home applications. The Petri nets are effective in handling the smart home specific nature including scalability, extendibility, and the concurrency of the smart home systems. The Petri net model of the smart home provides a graphical representation of the smart home specification that are easy to be understood and modified to reflect specific user preferences. In addition, Petri net models can be reused in the context of different smart home configurations. For an effective utilization of the Petri net power, an algorithm is proposed. The algorithm can be implemented in any programming language to simulate and verify the Petri net models and to have an executable version of the models for real time implementation.

The power of Petri nets and the proposed algorithm is proved and demonstrated in modeling the automation systems of a proposed smart home and verifying its behavior at the simulation level. Only results related to a fire sub-module are presented due to paper space limitation. Modeling of concurrency of smart home systems is presented in the context of activating specific user modes.

As a future work, real time implementation of the smart home models is to be done based on the proposed algorithm. It is expected that it will be very easy and straight forward task as the algorithm implementation used for models simulation will be reused. A real interaction with sensors and actuators is to be considered instead of simulating their behavior.

References

[1] N. Wu and M. Zhou, "System Modeling and Control with Resource-Oriented Petri Nets", Taylor and Francis Group, LLC, USA, 2010.

[2] S. Loke, S. Smanchat, S. Ling, and M. Indrawan, "Formal Mirror Models: an Approach to Just-in-Time Reasoning for Device Ecologies", International Journal of Smart Home, Vol. 2, No. 1, 2008, pp. 15-32.

[3] D. Cook and S. Das, "How Smart are Our Environments? An Updated Look at The State of The Art", ElSevier Journal of Pervasive and Mobile Computing, Vol. 3, 2007, pp. 53-73.

[4] J. Chan, "A Smart Home Embedded Computer System on Programmable Chips", A master dissertation, Ryerson University, Toronto, Ontario, Canada, 2006.

[5] M. Al-Qutayri, H. Barada, S. Al-Mehairi, and J. Nuaimi, "A Framework for an End-to-End Secure Wireless Smart Home System", SysCon 2008-IEEE International Systems Conference, Montreal, Canada, 2008, pp. 1-7.

[6] L. Zhang, H. Leung, and K. Chan, "Information Fusion Based Smart Home Control System and Its Application", IEEE Transactions on Consumer Electronics, Vol. 54, No. 3, 2008, pp. 1157-1165.

[7] Z. Lin and L. Fu, "Multi-user Preference Model and Service Provision in a Smart Home Environment", Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering Scottsdale, AZ, USA, Sept 22-25, 2007, pp. 759-764.

[8] J. Choi and D. Shin, "Research and Implementation of the Context-Aware Middleware for Controlling Home Appliances", IEEE Transactions on Consumer Electronics, Vol. 51, No. 1, 2005, pp. 301-306.

[9] M. Chan, D. Este`ve, C. Escriba, and E. Campo, "A review of smart homes-Present state and future challenges", ElSevier Journal of Computer Methods and Programs in Biomedicing, 2008, pp. 55-81.

[10] D. Pishva and K. Takeda, "Product-Based Security Model for Smart Home Appliances", IEEE A&E Systems Magazine, 2008, pp. 32-41.

[11] M. Aiello and S. Dustdar, "Are our Homes Ready for Services? A Domotic Infrastructure Based on The Web Service Stack", ElSevier Journal of Pervasive and Mobile Computing, Vol. 4, 2008, pp. 506-525.

[12] P. Wang, H. Jiang, W. Shi, and M. Cheng, "Design and Realization of Remote Control in Smart Home System", International Conference on Communication Software and Networks, 27-28 Feb., 2009, pp. 13-15.

[13] K. Jeong, "Design and Operation of Smart Homes in U.S.A. and Korea", A master dissertation, Purdue University, West Lafayette, Indiana, 2009.

[14] T. Lin, C. Hsu, T. Chun, and S. Cheng, "OSGi-Based Smart Home Architecture for Heterogeneous Network", ICST 3rd International Conference on Sensing Technology, Nov. 30 - Dec. 3, 2008, pp. 527-532.

[15] R. David and H. Alla, "Discrete, Continuous, and Hybrid Petri Nets", Springer, 2005.

[16] B. Hrúz, M. C. Zhou, "Modeling and control of discrete-event dynamic systems with Petri nets and other tools", *Springer*, 2006.

Authors

Azza K. Nabih is an R&D Engineer in *Software Engineering Competence Center* (SECC), ITIDA, Smart Village, Egypt. She is also an M. Sc. candidate at *Computer and Systems Engineering* department, Ain Shams University, Cairo, Egypt. She received her B. Sc. degree in Computers and Systems Engineering in 2004 from Ain Shams University. Her research interests include embedded systems applications especially multimedia, smart home technology, modeling smart home services, and utilizing wireless communication protocols for smart home networking.

Mostafa M. Gomaa is an Associate Professor at *Computer and Systems Engineering* department, Ain Shams University, Cairo, Egypt. He received his B. Sc. and M. Sc. degrees in Electrical Engineering in 1989 and 1993 respectively, from Ain Shams University. He received M. Sc. and Ph. D. degrees in Systems Engineering in 1994 and 1997 respectively from Grenoble University, Grenoble, FRANCE. His research interests include fault detection and diagnosis, intelligent control systems, hybrid systems modeling, simulation and control, Petri net theory and applications.

Hossam S. Osman is an Associate Professor at *Computer and Systems Engineering* department, Faculty of Engineering, Ain Shams University, Cairo, Egypt. He received his B. Sc. and M. Sc. degrees in Electrical Engineering from Ain Shams University. He received his Ph. D. degree in electrical and computer engineering from Queen's University, CANADA. He has 25+ years experience in Computer Engineering. His research interests include embedded systems, neural networks, and pattern classification.

Gamal M. Aly is a Professor at *Computer and Systems Engineering* department, Faculty of Engineering, Ain Shams University, Cairo, Egypt. He obtained his B. Sc. and M. Sc. degrees in Electrical Engineering from Ain Shams University. He received his Ph.D. in Electrical Engineering, University of Calgary, CANADA in 1972. Dr. Aly is also the Chairman of *Software Engineering Competence Center* (SECC) and Chief Editor of the *International Journal of Software Engineering* (IJSE). His research interests include Control Systems, Modeling and Simulation, Software Engineering, Expert Systems, and Parallel Processing.