CARLETON UNIVERSITY

# LITERATURE REVIEW ON THE APPLICATION OF RT-DEVS IN CYBER PHYSICAL SYSTEMS

Report submitted in partial fulfillment of the requirements for directed studies in

System and Computer Engineering

by

Mahya Shahmohammadimehrjardi

August 2021

## Table of Contents

ABSTRACT ..... iv

Chapter 1: Cyber Physical Systems ..... 1

**1.1 What are Cyber Physical Systems** ..... 1

**1.2 CPS Modeling and Simulation** ..... 1

1.2.1 Challenges in CPS ..... 2

1.2.2 Modeling Methods of CPS ..... 3

1.2.3 Programming models for CPS: ..... 7

Chapter2: DEVS and RT-DEVS ..... 10

**2.1 DEVS Formalism** ..... 10

2.1.1 Atomic model ..... 10

2.1.2 Coupled model ..... 11

**2.2 RT-DEVS Formalism** ..... 11

2.2.1 Atomic Model ..... 12

2.2.2. Coupled Model ..... 13

2.2.3. Previous Work in RT-DEVS ..... 13

2.2.4. DEMES ..... 14

2.2.5 RT-DEVS based Simulators ..... 16

**2.3 CPS Reliability in DEVS** ..... 21

2.3.1 DEVS-based Framework for Sensor Fusion ..... 21

2.3.2 Real-Time Fault Detection and Diagnosis of CPS Faults in DEVS ..... 23

**2.4. Imprecise DEVS (I-DEVS)** ..... 25

Chapter 3: Opportunity in Research in CPS Using RT-DEVS Approaches ..... 26

**3.1 Sensors and Actuators** ..... 26

**3.2 Network** ..... 26

**3.3 Verification and Validation**..... 27
**3.4 Software Development**..... 27
Chapter 4: Case Study (Fire Emergency System)..... 28
**4.1 Original Version of the Building Controller** ..... 29
**4.2 Advanced Version of the Fire Emergency System**..... 31
References..... 35

## ABSTRACT

Cyber Physical Systems (CPS) are composite of computers and physical systems that all work in harmony. It works as embedded computers monitor and control physical processes, usually with feedback loops, where physical processes affect computations and vice versa (E Lee 2015).

Now-a-days, Cyber Physical systems play an important role in every aspect of our lives. The application is so widespread that sometimes we do not even notice them. They are used in industrial control systems, smart grid, automatic air conditioning systems, autonomous automobile systems, aviation industry, space crafts, medical monitoring, robotic systems and basically in any electronic device that can work automatically. Coupled cooperation of software and hardware design make an CPS accessible for a certain purpose. Even though CPS is widely used, and the components are usually low cost, they are complex and software development have not been scaling up yet. The modeling part of these systems is the most important part of their design because it can lower the cost significantly and increase the reliability and accuracy of the system. However, modeling and simulating CPS play an important role and it is critical, because of the inheritance of CPS, modeling these systems is challenging and several modeling and simulation have been proposed.

Real-time Discrete Event System Specification (RT-DEVS) is a methodology has been proposed for modeling and simulating CPS. RT-DEVS is an extension of Discrete Event

System Specification (DEVS) formalism proposed by Bernard P. Zeigler in 1997. DEVS is a modular and hierarchical formalism used for modeling and analyzing discrete event dynamic systems that can be described using input events, output events, time advance function, transition functions and different states.

Different RT-DEVS tool kits such as RT-Cadmium are reviewed in this report. And at the end, an Emergency System as a case study of applying RT-DEVS to CPS is discussed.

# Chapter 1: Cyber Physical Systems

## 1.1 What are Cyber Physical Systems

Cyber Physical System (CPS) is an intersection of various disciplines. Norber Weiner, an American mathematician first introduced the term Cybernetics. He had a huge impact on the development of control systems theory. The term Cyber Physical System in rooted from the term Cybernetics [**1**].

CPS can be applied in areas as diverse as energy handling, healthcare, civil infrastructure, chemical process and embedded systems, software engineering, and control theory.

A CPS is an orchestration of computers and physical systems. Embedded computers monitor and control physical processes, usually with feedback loops, where physical processes affect computations and vice versa [**2**].

The CPS is constituted by several physical structures (PS) and their corresponding virtual models; sensors, actuators, controller and a network.

## 1.2 CPS Modeling and Simulation

Different methods are proposed for designing CPS. Since traditional methods profoundly relies on hardware and software design, CPS modelling and simulating techniques have drawn attention because of their low cost and high reliability. And that is why CPS is all about modeling. Models play a central role in all scientific and engineering disciplines.

Since CPS conjoins distinct disciplines, modeling becomes important. Because the models from distinct disciplines do not combine well [**2**]. In practice, this diversity leads to heterogeneous engineering processes, which are often difficult to combine and customize when planning a system [**3**].

### 1.2.1 Challenges in CPS

CPS software development is a complex job that causes many problems. Systems are logical and physically divided; they need to run on different platforms, meet specific execution times and address communication issues. Some of the main challenges are discussed in the following paragraphs.

#### *Representing a sophisticated system that balances between Continuous and discrete structures*

CPS engineering is located around the boundaries between discrete digital and continuous physical worlds, one of the most important features of modelling techniques is the processing of their possible continuous phenomena such as time and space. Even though, classic software engineering models support connectivity and automated verification, managing their continuous phenomena is often too limited for the CPS [**3**].

Quality design of CPS is relatively complicated because engineers need to find a balance between continuous and discrete structures using sophisticated system representations, interacting with physical and digital processes [**4**].

***Timing setup***

Calculations, networking, and physical changes must be properly synchronized to enable CPS to work as intended [**5**] since, different parts of the system may depend on each other and their execution must take place in a certain sequence and under certain circumstances [**3**].

***Connecting heterogeneous components and processes***

Since CPS is about modeling physical systems from different disciplines, it requires heterogeneous processes and components to interact and cooperate which is often practically problematic.

### 1.2.2 Modeling Methods of CPS

CPS Modeling techniques vary depending on the scientific field from which they are derived [**3**]. As CPS engineering is located around the boundaries between discrete digital and continuous physical worlds, one of the most challenging part of modeling these systems is the processing of continuous phenomena such as time and space [**3**]. Based on the approach used to address this issue, modeling methods can be classified into Discrete, Continues and Hybrid modeling methods.

***Discrete modeling methods***

One category of discrete modeling focuses on describing complex states and their relationships (Data schemas and object models) [**6**]. Alloy, TLA+, Object-Z, VDM-SL are some formalisms of these category [**6**].

Other category focuses on describing processes, where the primary focus is placed on the system state transitions and changes [**6**]. State machine forms and algorithm notations such as Process algebra (like CSP and FSP [**6**]), Transitions systems – Statecharts and Promela/Spin [**6**], PlusCal [**6**], Petri networks [**6**], Dynamic logic (based on JML specification) [**6**] and Reactive models [9]; are the foundations of this method [**3**].

***Continues modeling methods***

Some physical processes, such as mechanics, thermodynamics, and electromagnetism are described in differential equations which are continuous. Modelica language is used a combination of discrete units with shared variables; Differential equations of partial derivatives can be simplified by lump element models mapped into simple differential equations [**3**].

Continues modeling methods are oftentimes used in control. And Simulink (Matlab) and SCADE are some of the common software tools for this purpose [**3**].

***Hybrid modeling methods***

Hybrid modeling methods combine both the discrete and continues approaches to create one holistic model [**3**]**.** Hybrid systems are digital real-time systems embedded in analog environments. Hybrid systems can be explained using an example; digital embedded control program for an analog plant environment like an airplane: the controller state moves discretely between control modes, and in each control mode, the plant state evolves continuously according to physical laws. Those systems combine discrete and continuous

dynamics [**7**]. Some examples of hybrid modeling methods are hybrid automata and timed automata as described below.

*Hybrid Automata*

Hybrid Automata is a formal model that combines discrete control graphs, usually called finite state automata, with continuously evolving variables. A hybrid automaton exhibits two kinds of state changes: discrete jump transitions occur instantaneously, and continuous flow transitions occur when time elapses. [**7**].

*Timed Automata*

Timed Automata is an important subset of hybrid machines [**3**]. Two characteristics of Timed Automata are accessibility problem and computational tracking that can be performed without any approximate values. Timed Automata is not only used in tools such as UPPAAL to check the logic and the ability of the machine using. It is also used as a semantic basis for component-based models such as BIP and EAST-ADL, which enable design and synchronization analysis at higher abstraction levels [**3**].

*Advantages and disadvantages of hybrid models*

The advantage is that an engineer can choose which dynamics to display on a continuous basis and which in discrete form. The disadvantages are the price of this flexibility, the complexity of syntax and semantics, as well as the difficulty of analyzing and linking hybrid models with other models [**3**].

These continuous, discrete and hybrid methods can be used in any modeling techniques since it is an approach to describe continuous phenomena. These methods can be applied in component and model-based methods, which are two novel modeling methods described in this section.

***Component modeling methods***

component-based software engineering is deployed by platform-dependent systems that can overcome CPS problems such as the logically and physically differences of the systems; different platforms required for the components and the fact that each component needs to meet specific execution. Component and Connector (C&C) models are widely used to design and develop CPS to display features and their logical interactions [**3**]. Simulink and LabView are tools used to apply C&C models in different fields.

***Model-based development***

model-based method is a powerful CPS development that mainly focuses on individual steps such as simulation, software synthesis, or verification [**3**]. A step by step holistic methodology for model-based CPS development – from abstraction to architecture and from concept to implementation is described in details in [**3**]. The steps of the model-based development lifecycle described in [**3**] is as follows:

i. Problem statement
ii. Physical process modeling
iii. Characterization of the problem
iv. Expression of control algorithm

v. Choice of computing models

vi. Determination of the hardware to be used

vii. Simulation

viii. Design

ix. Verification, validation and testing

### 1.2.3 Programming models for CPS:

Today's computing and networking methods are not suitable for CPS modelling, as they do not consider the time and concurrency of the physical system and have set promising research directions to better use computing and networking systems in the CPS codec [**3**]. One of these programming models proposed in [**8**] is Programming Temporally Integrated Distributed Embedded Systems (PTIDES) which reflects the physical concept of time to simulate a distributed real-time embedded system. Giotto is a programming language used for real time CPS [**8**]. Ptolemey 2 is another modelling and simulation environment which applies mostly in heterogeneous systems [**8**].

***Modeling and Simulating CPS using DEVS***

Discrete EVent System Specification [**9**] is a modular and hierarchical formal modeling language used to describe discrete event systems. There exists an abstract DEVS simulator, which is an algorithm to simulate DEVS models. The algorithm has been proven to be correct and closed under coupling making DEVS an ideal modeling and simulation framework for this application [**9**]

*Difference between a discrete time simulation and a discrete event simulation*

In discrete time systems, time is viewed as discrete variable where the time period is divided into equal sections and this concept is used in discrete time simulation as well. However, in discrete event systems, the system is composite of different components. Each component has its behavior which is define as an event of the system. The described events should work in an obligated order. Different modeling methods can be used for each event. In DEVS, the timing can be continuous during each event and it is the events that are discrete. DEVS also provides a formal background for modeling both discrete and continuous worlds [**10**].

*How DEVS solves the modeling challenges in the CPS*

Some of the modeling challenges in CPS was discussed in section 1.2.1. Here we are going to address those challenges and see how DEVS overcome these challenges. Representing a sophisticated system that balances between Continuous and discrete structures was one of those challenges that is solved in DEVS since DEVS counts as hybrid modeling. The time setup issue is solved by logical and timing correctness rely on DEVS mathematical theory. Also, because of the fact that different modeling techniques can be used for each event separately, heterogeneous components and processes can easily connect.

*Advantages*

DEVS has well-defined concepts for coupling of components and hierarchical, modular model composition [**11**]. The original models can be part of the final product. DEVS formalism is a rigorous mathematical theory that can be used for logical and timing

correctness that increases the model reliability [**11**]. Modeling and simulation using DEVS decrease the cost of the final product by saving the resources.

*Disadvantages*

Since DEVS is a relatively new technique for modeling and simulation, there are still not many platforms built for this method. In section 2.2.5, we are going to discuss number of these platforms, their algorithm and their limitations.

# Chapter2: DEVS and RT-DEVS

## 2.1 DEVS Formalism

Discrete Event System Specification is a modeling and simulating technique that models and analyze discrete event dynamic systems. In DEVS methodology, a model of a real system consists of *Atomic* and *Coupled* models [**10**].

### 2.1.1 Atomic model

Atomic models describe the behavior of a given component. Atomic models have a current state, a time that they will stay in that state unless interrupted, input ports, and output ports.

An atomic model is described as follows [**10**].

$$\text{AM} = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

where

$X$: a set of external input event types

$Y$: an output set

$S$: a sequential state set

$\delta_{ext}$: $Q \times X \rightarrow S$ , the external state transition function, where $Q = \{(s,e)/s \in S, e \in [0, ta(s)]\}$ and $e$ is the elapsed time since the last state transition;

$\delta_{int}$: $S \rightarrow S$ , the internal state transition function;

$\lambda$: $S \rightarrow Y$ , the output function;

$ta$: $S \rightarrow R^{+}_{0,\infty}$, the time advance function.

### 2.1.2 Coupled model

Coupled models are used to link groups of models (atomic or coupled). The outputs of one DEVS model can be passed into the inputs of another DEVS model, and the coupled model is used make these links.

The coupled model is described as [**10**]:

$$\text{CM} = < X, Y, \text{D}, \{M_i\}, \{I_i\}, \{Z_{ij}\}, \text{select} >$$

where

$X$ is the set of input events;

$Y$ is the set of output events;

D is an index for the components of the coupled model; $\forall i \in \text{D}$,

$M_i$ is a basic DEVS model;

$I_i$ is the set of influencees of model i and $\forall j \in I_i$;

$Z_{ij}$ is the i to j translation function, where $Z_{ij}$: $Y_i \rightarrow X_j$

## 2.2 RT-DEVS Formalism

DEVS formalism does not have the ability to model the real time systems where execution time is essential like in avionic control systems or breaking systems in autonomous cars

where it is critical for some task to be done within a deadline. There are two approaches to this issue in different studies. One approach that we discuss here is by presenting an extension of DEVS formalism [**12**]; RT-DEVS, which provides the possibility of executing real time models by adding time interval function (time-windows). Time windows restrict the simulation time to be completed within a real time [**13**]. In RT-DEVS we also have atomic and couple models.

### 2.2.1 Atomic Model

An atomic model in RT-DEVS is described below:

$$RTAM =< X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ta, ti, \psi, A >$$

Where

X, S, Y, $\delta_{int}$, $\lambda$, $ta$: same as the original

$\delta_{ext}: Q \times X \rightarrow S$ , An external transition function

Where Q is the total state set of $M = \{(s, e) | s \in S \; and \; 0 \leq e \leq ti(s)|_{max}\}$

$ti$: a time interval function

$\psi$: an activity mapping function

$\mathcal{A}$: a set of activities

With constrains

$$\psi: S \rightarrow \mathcal{A}$$

$ti : S \rightarrow R_{0,\infty}^{+} \times R_{0,\infty}^{+}$,

Where $ti(s)\big|_{min} \leq t(a) \leq ti(s)\big|_{max}$, $ti(s)\big|_{min} \leq ta(s) \leq ti(s)\big|_{max}$, $s \in S$,

$a = \psi(s) \in \mathcal{A}$, and $t(a)$ is the execution time of an activity $a$.

$$\mathcal{A} = \left\{a | t(a) \in R_{0,\infty}^{+}, a \notin \{X?, Y?, S =\}\right\}$$

Where X? is the action of receiving data from X, Y? is the action of sending data through Y, and S = is the action of modifying a state in S

### 2.2.2. Coupled Model

Coupled RT-DEVS is described as:

$$RTCM =< D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$$

Where,

$D$, $I_i$, $Z_{i,j}$: same as the original

$M_i$: a component basic RT-DEVS model

### 2.2.3. Previous Work in RT-DEVS

In [**14**], an introduction to the discrete event system specification formalism and its application for modeling and simulating cyber-physical systems by using a line-tracking robot as a case study is given. A line-tracking robot is designed to follow a track identified by a dark line, and to get back on track if the trail is not detected. The IR Sensor receives actions as input and sends signals as output in real-time.

In [**11**], a light rail controller prototype is built using DEMES methodology for modeling and implementation. Light rails are a form of urban trains operate at a high capacity with an exclusive right-of-way. In simple words, they operate by receiving light signals. The main task of Railway is to transfer the passengers from one station to another and they stop whether there are passengers boarding or disembark from the stations or not. The proposed light rail controller prototype in [**11**], controls light rail stops by taking into the account whether there are passengers to load and unload at the stations or not.

In [**9**] RTDEVS/CORBA Environment is proposed by Zeiglar et la in 2003. DEVS Executive [**12**] is another RTDEVS software proposed by Kim and Park in 1997. In 2009 and 2010 tools for real-time system modeling have been proposed in [**15**], [**16**] respectively. In [**17**] and [**18**] are two works have been done by ARSLAB in developing $ECD^{++}$ software tools for modeling and simulating real-time systems using RT-DEVS.

In [**20**], sensor fusion technique using DEMES is proposed for more reliability in CPS. In [**13**], real-time fault detection and diagnostic DEVS-based method for CPS is proposed. A DEVS-based method for imprecise computation to handle the overload computation issue in hard real-time systems is introduced in [**21**].

### 2.2.4. DEMES

DEMES uses modeling and simulation and formal methods as a new technique for developing embedded software. Discrete-Event Modeling of Embedded Systems (DEMES) is an abstract and intuitive DEVS-based methodology independent of underlying simulators, hardware, and middleware [**13**].

The steps of applying DEMES methodology to model a real-time system are shown in the figure. 1 [**13**].

1. Formal specifications (DEVS, Bond Graphs, etc.) are used to model the System of Interest (a real-time system and its environment).
2. These models subsequently transformed into TA (Time Advance function) and verified using model-checking tools.
3. Simultaneous to this verification step, same models are used to test the components in a simulated DEVS environment.
4. The physical environment can be simulated as well as the components.
5. The real-time system model under particular loads is also simulated together with components and the environment.
6. After testing mentioned submodels, they can be applied incrementally into the target platform.
7. At the testing step, we can take advantage of simulation environments with fast, risk free performance as mentioned in the next section.

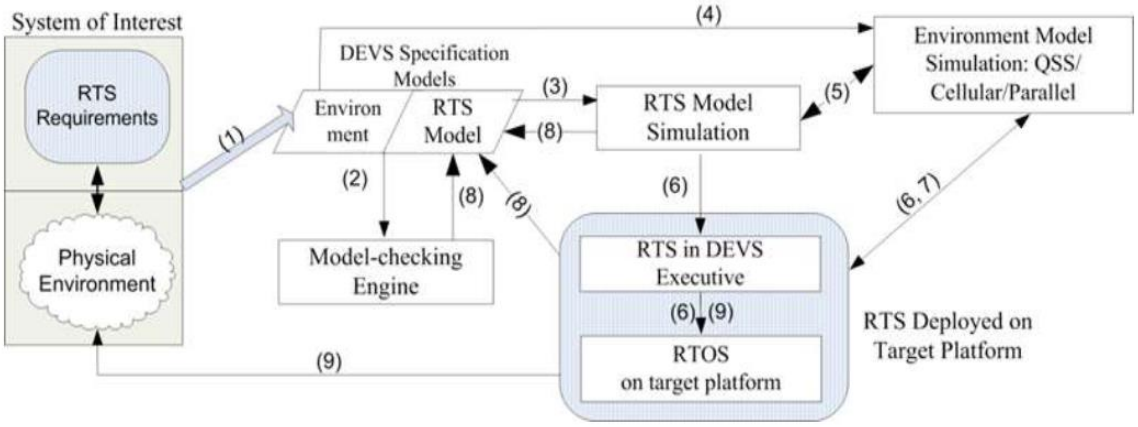


**Fig. 1. DEMES methodology to model a real-time system [13]**

8. Changes are done incrementally in a loop that provides a consistent set of apparati throughout the development cycle.
9. When the real-time system and every model deployed in the target platform is completely tested, the looping stops [**13**].

E-CD++ [**22**], [**24**] is a simulation platform that facilitates the application of DEMES. It provides a platform for models to be defined according to the DEVS formalism andimplemented in real-time. For deploying the models on hardware, the tool allows for the generation of binary files that can be interfaced with input/output devices through the ARM MBED Library. More details about this study are provided in the next section.

### 2.2.5 RT-DEVS based Simulators

In 1997, DEVS Executive was the first RT-DEVS based framework was proposed by T. G. Kim and K. H. Park to define and simulate real time systems [**12**]. The purpose of this work was to develop a RT-DEVS based simulation platform (DEVS Executive) to be

applied for a seamless real-time software.

The DEMES methodology has been used to develop several tools and simulators. DEVSJAVA [**15**], RTDEVS/CORBA [**9**], PowerDEVS [**5**], E-CD++ [**17**], [**18**] and RT-Cadmium [**15**] are some of these environments designed for modeling and simulation real-time systems. In the next paragraphs, more details about E-CD++ and how it provides a basis for developing RT-Cadmium is given.

***E-CD++***

E-CD++ is a real-time simulator developed based on CD++ (a DEVS-based simulation framework) and RT-CD++ which is an extension of CD++ for real-time simulations [**22**]. CD++ virtual time-advance function has been transformed into real-time in RT-CD++, and provides an RT simulation platform for verification for real-time models [**22**]. A layered approach to E-CD++ development framework is illustrated in figure. 2.

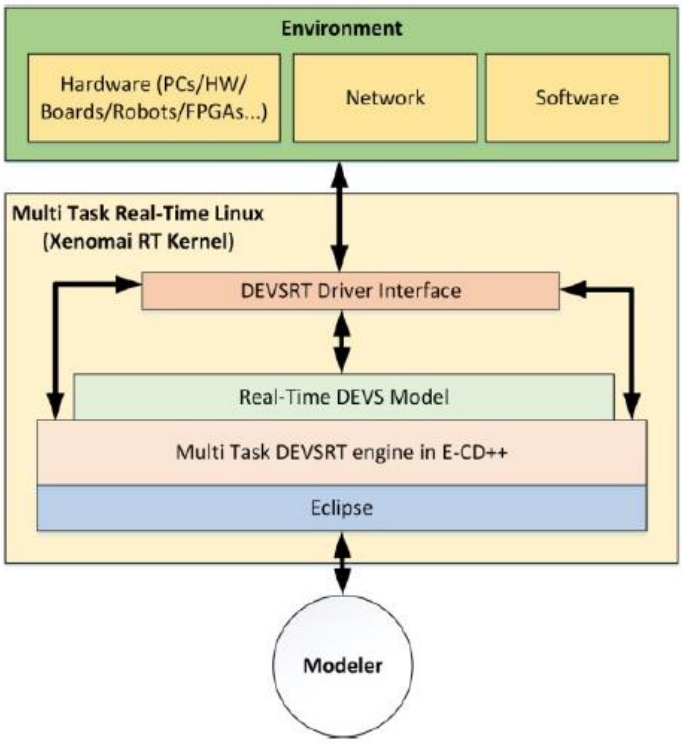


**Figure 2. E-CD++ Layers [17]**

An embedded platform interacts with the environment and the modeler defines models using a high-level DEVS language combined with C++ code if required. Afterwards, the DEVS Real-Time (DEVSRT) plays its part and execute these Real-Time models. provide the Input and Output ports of E-CD++ models provide the DEVSRT Drive Interface layer for model replacement corresponding to the external entities. The base middleware which is Xenomai RT Kernal and the runtime objects are imported to this platform as RT tasks. The models established by modeler and the driver objects is then combined with the E-CD++ core objects to make the whole combination compiled and produce an executable.

The Eclipse IDE layer provides the developer with the graphical development of models which allows for the automatic model generation. By using the IDE, the Generic Graphical Advanced environment for DEVS modeling and simulation (GGAD) provides the developer to take advantage of using a graph-based representation to specify models hierarchy, interconnections and features [**22**].

***RT-Cadmium***

RT-Cadmium is designed based on the E-CD++. Besides the requirements that are met in the E-CD++ design such as backward compatibility with all existing Cadmium models, conserving the DEVS simulator structure, having identical behavior when simulated and deployed on the target, there were some other features that required enhancement such as interchangeability of simulation and deployment, Platform portability and handling Asynchronous events.

These advancements and features are applied in RT-Cadmium which are discussed below in details [**19**].

*Interchangeability of simulation and deployment*

Based on the DEMES workflow, DEVS models are developed incrementally rapidly. As a result, a perfect tool for DEMES is a tool that can easily go back and forth between the testing step and deploying it to the hardware. Since E-CD++ is designed to be used in a waterfall development fashion; which means the new E-CD++ must be made for reusing and adding some features to make it compatible for the target platform after the initial modeling and simulation is done in E-CD++ and the user was satisfied with it. As it is

obvious, this workflow is not feasible, and most users rather skip simulating their system completely. In RT-Cadmium, the user program the project and that is why it can be used for simulation and the target platform, which is one of purpose in DEMES methodology. RT-Cadmium is also equipped with visualizations tools to make debugging simulation process, and runtime model verification easier [**19**].

*Platform portability*

A good software should be portable to most the common platforms so it can be used ubiquitously. The RT-Cadmium is hardware independent and as the result portable to many platforms with only three requirements: a C++17 Compiler for the target platform, a real-time Clock class relating simulation time to microseconds and DEVS models that save Input/output drivers. Therefore, the RT-Cadmium can be applied for many embedded platforms whereas, the E-CD++ was written specifically only for ARM platforms [**19**].

*Asynchronous event handler*

Normally, embedded system development platforms should be able to handle asynchronous events, however, many standard DEVS simulators such as E-CD++ does not have this ability. All DEVS events are required to be defined at run time before the model goes to sleep. When a DEVS model goes to sleep, the time when it should be woken up is always been told to the simulator in advance. However, this is not possible for an asynchronous model to tell the real-time engine when it will generate its next output. To tackle this issue, RT-Cadmium uses an observer pattern to add support for asynchronous

events, while maintaining the top down coordinator tree structure of the DEVS abstract simulator. This method allows the asynchronous atomic model to notify the DEVS engine of the event at the top coordinator level [**19**].

### 2.3 CPS Reliability in DEVS

CPS composites of sensors, actuators, controllers and a network and errors in each part of these systems can occur. Failure in a system occur when the system/component is unable to perform a required function according to the tasks while a fault is a condition that could lead to failure [**23**], and that is why it is important to tackle faults early so that they do not lead to failure [**13**]. Since in CPS, heterogeneous components from different disciplines work together, the possibility of a fault and as a result caused a failure increases dramatically. Therefore, detecting and solving a fault in early stages matters.

In couple of studies, various techniques have been proposed to increase CPS's reliability. In this section we review two of the DEVS-based methods proposed to lower the error in these systems. In [**20**], a DEVS-based framework for sensor fusion has been proposed to increase the data accuracy collected by sensors in a DEVS platform. And in [**13**] fault detection and diagnostic in CPS has been done utilizing DEVS formalism. In following paragraphs, more details about these studies are given.

#### 2.3.1 DEVS-based Framework for Sensor Fusion

Sensor fusion is a method used to combine sensor data and improve the description of the physical property measured by these sensors [**20**]. In this paper a novel generic framework for

implementing sensor fusion in DEVS has been proposed so that, the benefits of sensor fusion can be utilized in the modeling process of complex and heterogeneous systems such as CPS [**20**]. There are different sensor faults which can occur because of different factors. Faults such as bias, out of range, distortion as a result of noise can happen in an individual sensor and fault such as data-centric faults and system view faults that occur in a sensor network [**20**]. The proposed sensor fusion framework in [**20**], works for all applications, it provides a collection of different fusion algorithms, receive a set of inputs and give a single output immediately, the output is identical to the single sensor output and the framework uses unique identifiers and timestamp for keep tracking of outputs [**20**]. As shown in the sensor fusion block in figure. 3, it consists of two layers; fusion layer and sensor layer. The sensor layer is formed by a number of sensors measuring the same variable. These sensors can be different kinds or the same kinds.

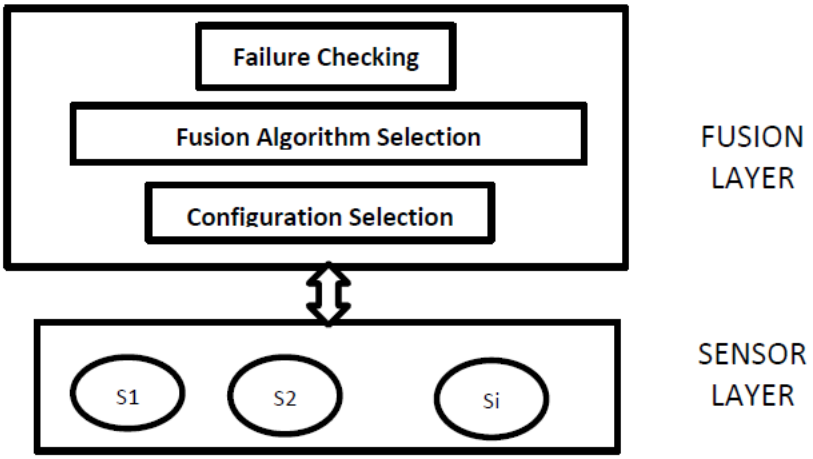


**Figure 3. Proposed Sensor Fusion Framework [20].**

The fusion layer has three blocks; Configuration Selection, Fusion Algorithm Selection and Failure Checking. In the Configuration Selection step, the configuration of sensors defines which is important because different systems with different applications requires different configuration for their sensors connections. The Configuration step is also the input for the algorithm step. In Fusion Algorithm Selection section, the algorithms are classified based on configuration and their objectives configured in the Configuration step. In the Failure Checking section, the quality of the fused sensor output at different time intervals are being checked to make sure the outputs are actually representing the variable they are measuring [**20**].

### 2.3.2 Real-Time Fault Detection and Diagnosis of CPS Faults in DEVS

In [**13**], a DEVS-based method for fault detection has been proposed to be applied when building a CPS. It is a challenging task to Fault Detection and Diagnosis (FDD) in CPS because of its inherent complexity and interconnection of heterogeneous components. An event-based generic scheme is proposed in [**13**] to properly identify faults and distinguish them from noise or uncertainty, isolate faults within a CPS, handle several faults, prevent major faults by detecting faults at subsystem levels, consume minimum resources and have fast computation [**13**]. The scheme is shown in figure. 4.

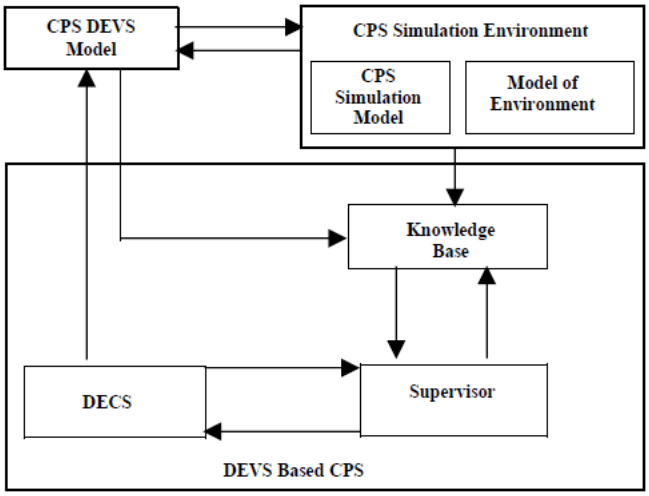


**Fig. 4. Proposed CPS FDD [13]**

CPS DEVS Model component, which is a model of the CPS using DEVS, test and populate the Knowledge Base with faults. After implementation of this model, it will be executed in CPS Simulation Environment. The model's behavior in various scenarios is captured at this stage. Based on the simulation results, we can go back to the CPS DEVS Model, modify the specifications if needed, and run new simulations. The CPS Simulation Environment is also used to study the model for possible faults. These studies would provide data to populate the Knowledge Base component, which is a database that holds information about the known set of faults that can occur in the CPS. Once we are satisfied with the simulation results, the CPS DEVS Model, which was already updated using the CPS Simulation Environment, is applied into the target platform and then transformed into the *DECS* (Discrete Event Control Software), which has the ability to violate the specification originally described by the model [**13**].

### 2.4. Imprecise DEVS (I-DEVS)

Imprecise-DEVS (IDEVS) is a DEVS-based model-driven approach that develops real-time and embedded applications that combines the dynamic advantages of the imprecise computation technique with the rigor of a formal modeling methodology [**21**]. This approach can be useful when testing under real conditions are impossible. This method is also very useful for hard real-time systems, where it is important for tasks to be done before certain deadlines. However, in overload circumstances, the chances that the operations could not meet the deadlines is very high and that is where the Imprecise Computation technique becomes helpful where the computations are categorized into mandatory and optional tasks. In I-DEVS, the optional and mandatory behaviors of the target system is defined by the designer [**21**]. I-DEVS is derived from RT-DEVS based methodology with a modification in P-DEVS (Parallel DEVS) atomic model formalism (adding a deadline and mandatory and optional conditions to the states).

## Chapter 3: Opportunity in Research in CPS Using RT-DEVS Approaches

Since our lives has been digitalized over the past decades, CPS application is seen in different day-to-day activities. CPS is applied in medical industry, entertainment industry and basically in any filed that requires automation. However, CPS bridges different disciplines and uncountable research in any engineering field can be done using CPS such as energy handling, mechanics, electronics etc., in this section, the focus is on the application of RT-DEVS in different aspects of CPS modeling and simulation.

### 3.1 Sensors and Actuators

Sensors and actuators are important components of the CPS and the accuracy of these components increases the accuracy and reliability of the system. Innovative works on actuators and sensors to command and collect data more efficiently using RT-DEVS such as the sensor fusion method proposed in [**20**] can be done in the future.

Also, there is a research potential to define different kinds of sensors and actuators in a CPS using RT-DEVS.

### 3.2 Network

Interaction between components in a CPS is done by its network. Networking can be done by novel techniques, such as Wi-Fi and Cloud. Since RT-DEVS can also make modeling and defining these network systems much feasible, several works can be done in this area.

### 3.3 Verification and Validation

CPSs are complex, heterogeneous prone to error systems. Studies in eliminating failure in CPS using DEVS approaches can be done. Such as smart fault detection and diagnostic with combination of artificial intelligence and DEVS methodology.

Also, there are not so many efficient computation techniques provided to tackle the issue of complex and numerous computation tasks in a CPS that must be done within a deadline with no error. RT-DEVS approach can be applied for to overcome this issues. I-DEVS [**21**] is one of the works has been done in this area, but there is still a huge absence of studies in this section.

### 3.4 Software Development

Overall, making advancements on RT-DEVS software tools that can be used for CPS can make a huge difference in modeling and simulation CPS. Works such as enhancing the Linux package in RT-Cadmium which is less developed and it is missing DEVS models encapsulating port drivers [**19**] can be a research opportunity for future work.

## Chapter 4: Case Study (Fire Emergency System)

In this chapter, first, a description on the previous Building Controller System developed by ARSLAB is given and then, the changes that has been applied to part of this system is discussed.

A model house controller has been designed in ECadmium for two rooms consists of an Emergency System and a LED controller as shown in figure. 5. There are two LEDs and a IR sensor in each room. There is also a fire alarm switch, a light sensor and a heat detector in total.

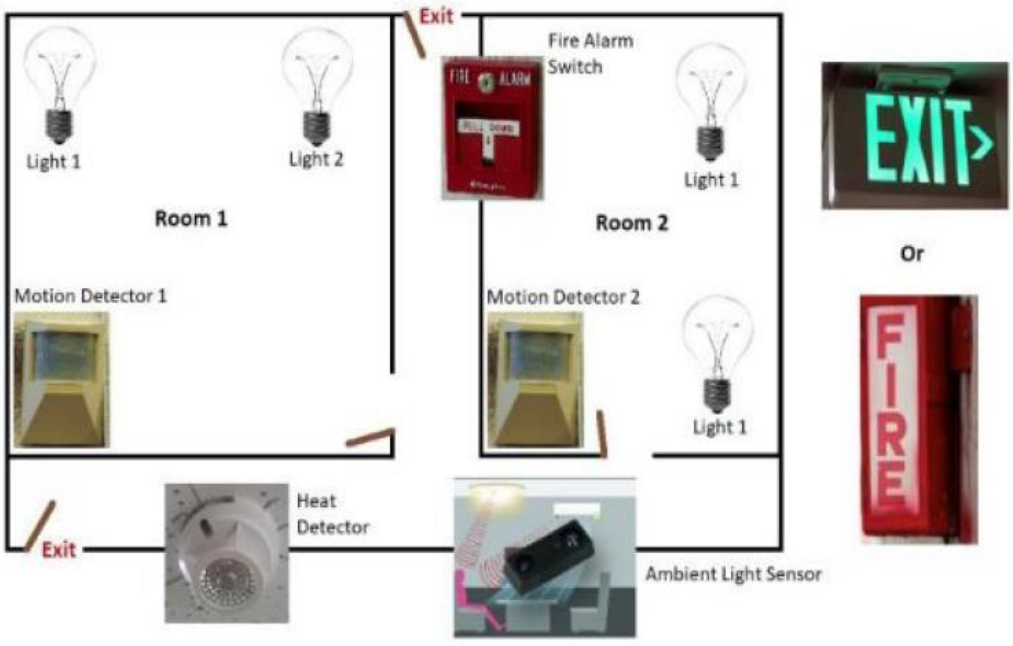


**Fig. 5. Building Controller Model**

### 4.1 Original Version of the Building Controller

The building controller system is consisting of two subsystems; the emergency system and a LED controller. The emergency system consists of two sensors; Fire Alarm Switch and Temperature Sensor as inputs. And 2 pairs of Red and Green LEDs (one set for each sensor) as outputs. The LED controller consists of a pair of IR sensor, and a light detector. The top model of the building controller system is shown in figure. 6.

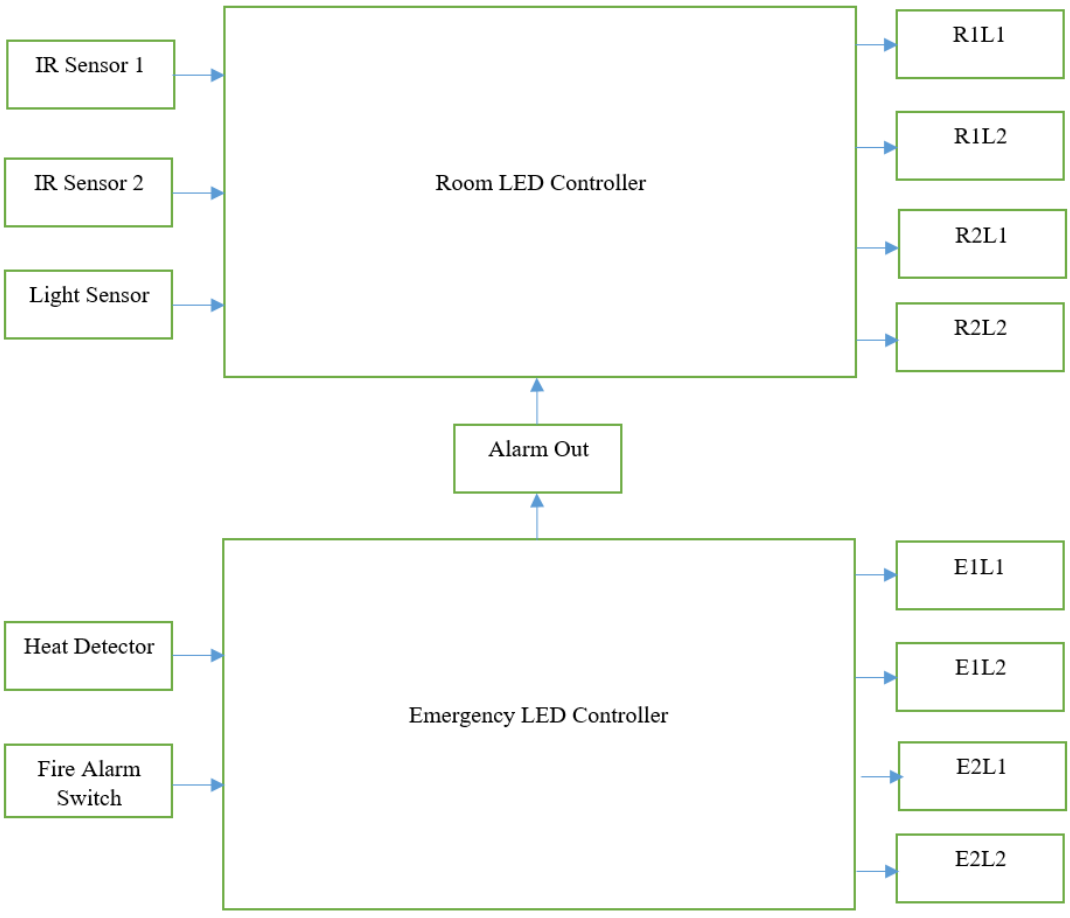


**Fig. 6. Top Modelof The Building Controller**

As shown in figure. 6, Alar out which is an output of the Emergency LED Controller counts as an output for the Room LED Controller. This top model is made of seven atomic models. Fire Sensor Controller, Fire Alarm Controller, Alarm Monitor, and Emergency LED Controller are atomic models of the Emergency LED Controller and IR Sensor Controller, Light Sensor Controller and Room LED Controller are atomic models used in the Room LED Controller System.

In this project, the main focus is on the Emergency LED Controller System and in the next section the changes that has been applied to this system is discussed. More information about the Room LED Controller System is provided in the link below.

https://github.com/ksuryakrishna/arslab/tree/master/TRIAL_SHIELD

The Emergency LED Controller System works based on the data received from sensors. If the Temperature Sensor reads a value that's greater than a threshold and the fire alarm switch is turned OFF, then the Red LED corresponding to Temperature Sensor turns ON (Green turns OFF) and the Green LED corresponding to the Fire Alarm Switch turns ON (Red turns OFF). Similarly, if the Fire Alarm Switch is turned ON and Temperature Sensor reads a safe room temperature, the Red LED corresponding to Fire Alarm Switch turns ON (Green turns OFF) and the Green LED corresponding to Temperature Sensor turns ON (Red turns OFF). In both these cases, an Alarm is sounded to denote that the people have to move out. If both the Temperature sensor and Fire Alarm Switch gives a safe reading, both pairs of Red and Green LEDs are turned OFF and alarm is not sounded.

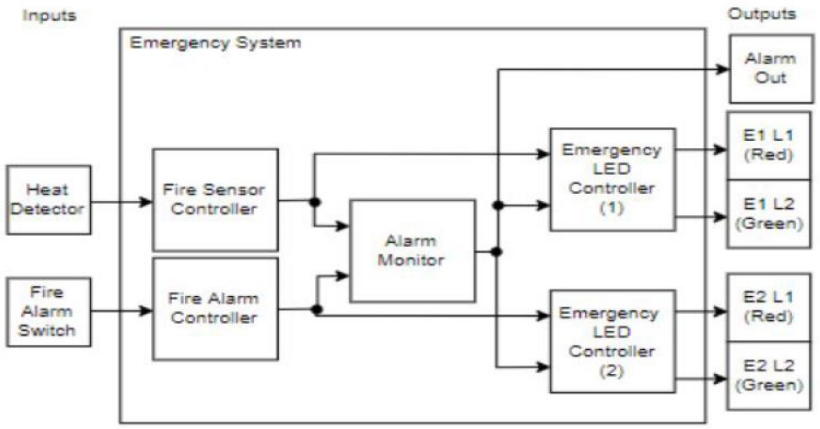


**Fig. 7. Block Diagram of the Original Version of the Emergency System**

The block diagram of the system is shown in figure. 7. The Fire Sensor Controller and the Fire Alarm Controller reads the value of the Heat Detector and Fire Alarm Switch respectively and pass it to Alarm Monitor and Emergency LED Controller 1 and 2. The logic of LED Controllers are OR. For instance, if Emergency LED Controller 1 receives a one from Fire Sensor Controller or Alarm Monitor, The E1L1 LED will go ON.

### 4.2 Advanced Version of the Fire Emergency System

The emergency system consists of four sensors; Fire Alarm Switch, Temperature Sensor and two Motion Sensors (PIR) as inputs. And 3 pairs of Red and Green LEDs (one set for each sensor) as outputs. One pair corresponding to the heat detector sensor, one pair allocated to the fire switch and one pair for both of PIR sensors; because even if in one room motion is detected, the red LED must turn on.

As shown in figure. 8, two PIR sensors and two LEDs corresponds to the sensors, two PIR sensor controllers and a PIR LED controller have been applied to the previous system.

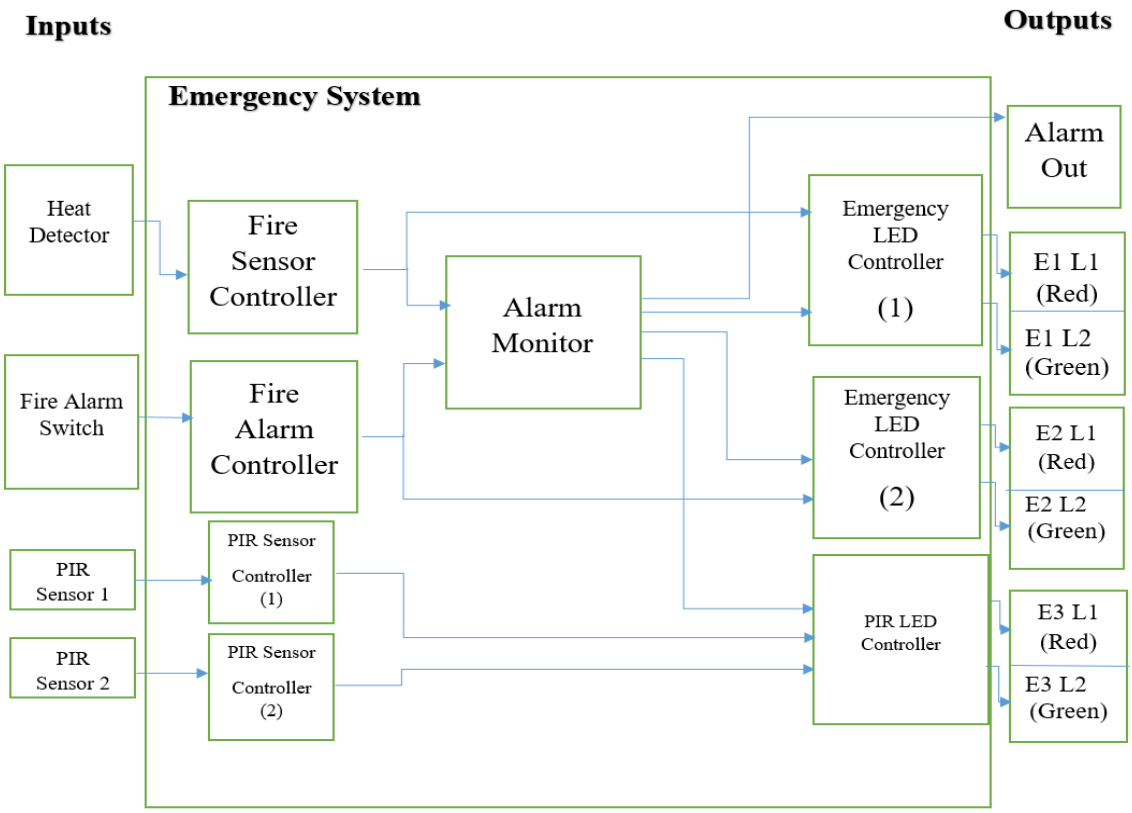


**Fig. 8. Block Diagram of the Advanced Version of the Emergency System**

There are two PIR sensors since we have two rooms, any motion is detected by these sensors.

In case of a fire, if an infant or a disabled person or a pet is trapped in the house, the pair of LEDs can be informative. If it is green, it indicates that no living being is inside the house and if it is red, it indicates that, the fire alarm has been turned on and someone is inside.

In cases described in section 4.1, when the alarm controller detects an emergency, either by the fire alarm switch or the temperature sensor, if the PIR sensors in any rooms or both rooms detects a motion, the LEDs corresponding to the PIR LED controller turns on (turns red). If an emergency detected but no motions are detected by the PIR sensors, them the LED Corresponding to the PIR LED controller stays off (green). Also, if no fire is detected and everything is normal, the PIR LEDs stays off even if there is motion detected in the house. In other words, the motion detector activates in case of a fire emergency.

Block diagram of the PIR LED Controller based on the desired description explained above is shown in figure. 8. The truth table of the PIR LED Controller is shown in Table. I. Corresponding code for the logic above in the PIR LED Controller atomic model is shown in figure. 9. The project link is provided below.

https://github.com/mahyashah/RT-Cadmium_Building_PIR_Included

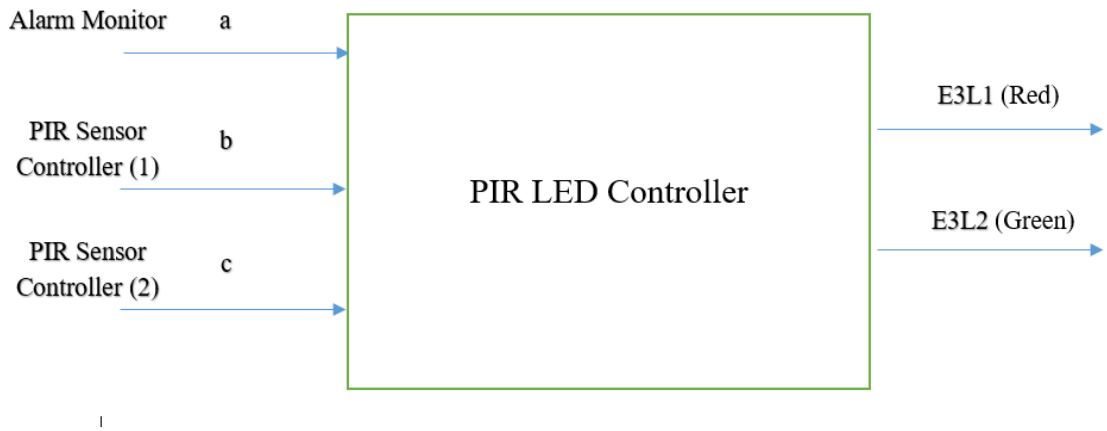


**Fig. 8. Bloch Diagram of the PIR LED Controller**

| a | b | c | R | G |
|---|---|---|---|---|
| 1 | x | 1 | 1 | 0 |
| 1 | 1 | x | 1 | 0 |
| 0 | x | x | 0 | 1 |

**Table I. Truth table of the PIR LED Controller**

```
  if(state.input_a && state.input_b){
    //cout << "Case 1 \n";
    state.output_1 = 1;
    state.output_2 = 0;
  }
  else if(state.input_a && state.input_c){
    //cout << "Case 2 \n";
    state.output_1 = 1;
    state.output_2 = 0;
    }
  else{
    //cout << "Case 3 \n"    ;
    state.output_1 = 0;
    state.output_2 = 1;
    }

  state.type = true;
}
```

**Fig. 9. Line of Codes in PIR LED Controller Atomic Model Corresponds to Its**

## References

[**1**] Wiener, N. Cybernetics: Or Control and Communication in the Animal and the Machine; MIT Press: Cambridge, MA, USA, 1948.

[**2**] E. Lee, "The Past, Present and Future of Cyber-Physical Systems: A Focus on Models," *Sensors*, vol. 15, no. 3, pp. 4837–4869, 2015.

[**3**] K. Babris, O. Nikiforova, and U. Sukovskis, "Brief overview of modelling methods, life- cycle and application domains of cyber-physical systems," *Applied Computer Systems*, vol. 24, no. 1, pp. 1–8, 2019.

[**4**] E. A. Lee, "CPS Foundations," *Proceedings of the 47th Design Automation Conference*, DAC '10, New York, pp. 737–742, 2010.

[**5**] E. A. Lee, "Cyber Physical Systems: Design Challenges," *In Proceedings of the 11th Symposium on Object Oriented Real-Time Distributed Computing*, IEEE Computer Society, Washington, pp. 363–369, 2008.

[**6**] I. Ruchkin, "Integration of Modeling Methods for Cyber-Physical Systems," PhD thesis, Institute for Software Research School of Computer Science, Carnegie Mellon University, Pittsburgh, 2018.

[**7**] J.-F. Raskin, "An introduction to hybrid automata," *Handbook of Networked and Embedded Control Systems*, pp. 491–517, 2005.

[**8**] K-D. Kim, and P. R. Kumar, "An Overview and Some Challenges in Cyber-Physical Systems," *Journal of the Indian Institute of Science,* vol. 93, no. 3, pp. 341–352, 2013.

[**9**] Cho, Y., Hu, X., and Zeigler, B. P. 2003. The RTDEVS/CORBA Environment for SimulationBased Design of Distributed RealTime Systems. Simulation 79, 4 (2003), 197210.

[**10**] G. A. Wainer, in *Discrete-Event Modeling and Simulation: A Practitioner's Approach (Computational Analysis, Synthesis, and Design of Dynamic Systems)*, 1st ed., vol. 483, CRC Press, pp. 35–72.

[**11**] J. Boi-Ukeme and G. Wainer, "Applying Modelling and Simulation for Development of Embedded Systems," 2019.

[**12**] T. G. Kim and K. H. Park, "A Real-Time Discrete Event System Specification Formalism for Seamless Real-Time Software Development," Oct. 1997.

[**13**] J. Boi-Ukeme, C. Ruiz-Martin, and G. Wainer, "Real-Time Fault Detection and Diagnosis of CPS Faults in DEVS," *2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys)*, 2020.

[**14**] G. A. Wainer, R. Goldstein, and A. Khan, "Introduction to the discrete event system specification formalism and its application for modeling and simulating cyber-physical systems," *2018 Winter Simulation Conference (WSC)*, 2018.

[**15**] B. Earle, K. Bjornson, C. Ruiz-Martin, and G. Wainer, "Development of a Real-time DEVS Kernel: RT-cadmium," *Spring Simulation Conference (SpringSim 2020)*, 2020.

[**16**] Bergero, F. and Kofman, E. 2010. PowerDEVS: a tool for hybrid system modeling and real-time simulation. *Simulation* 87, 12 (2010), 113132.

[**17**] Moallemi, M., and Wainer, G. 2013. Modeling and simulationdriven development of embedded realtime systems. *Simulation Modelling Practice and Theory*. 38, 0 (2013), 115131.

[**18**] Yu, H. Y., and Wainer, G. 2007. eCD++: an engine for executing DEVS models in embedded platforms. In *Proceedings of the 2007 Summer Computer Simulation Conference* (San Diego, CA, USA, July 15 - 18, 2007). SCSC '07. Society for Computer Simulation International, Vista, CA, 323-330.

[**19**] Furfaro, A. and Nigro, L. 2009. A development methodology for embedded systems based on RTDEVS. *Innovations in Systems and Software Engineering* 5, 2 (2009), 117127.

[**20**] J. Boi-Ukeme and G. Wainer, "A Framework for the Extension of DEVS with Sensor Fusion Capabilities," *Spring Simulation Conference (SpringSim 2020)*, 2020.

[**21**] M. Moallemi and G. A. Wainer, "I-DEVS: Imprecise Real-Time and Embedded DEVS Modeling," *2011 Spring Simulation Conference (SpringSim11)*, Apr. 2011.

[**22**] D. Niyonkuru and G. Wainer, "Towards a DEVS-based operating system," *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*,2015.

[**23**] X. S. Chen, K. Y. Revolorio, K. J. Baucom, A. Reina-Patton, and A. Christensen, "Recruitment and retention of Low-income COUPLES: Spanish- versus ENGLISH SPEAKING PARTICIPANTS," *PsycEXTRA Dataset*, 2012.

[**24**] D. Niyonkuru and G. Wainer, "Discrete Event Methodology for Embedded Systems."

*Computing in Science & Engineering* vol.17, no.5, pp.52-63, Sept-Oct. 2015.