DESIGN AND IMPLEMENTATION OF A BUILDING CONTROL SYSTEM IN REAL-TIME DEVS

Ben Earle, Kyle Bjornson, Joseph Boi-Ukeme, Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON, Canada
{joseph.boiukeme, ben.earle, kyle.bjornson}@carleton.ca, gwainer@sce.carleton.ca

ABSTRACT

Buildings occupy about 40% of the world's energy consumption, accounting for 30% of total CO_2 emissions. The motivation to reduce energy consumption and associated greenhouse gas emissions from buildings has led to increased interest in building automation. Real-Time Discrete Event Modelling and simulation presents an efficient way to design control systems for conservation of energy in buildings. We present a Building Controller built using the E-CD Boost library, that uses a DEVS model to control the lights and emergency systems of a building based on room occupancy and other inputs. This is a step in the development of model-based control systems to optimize energy consumption in buildings.

Keywords: Embedded Systems, DEVS, Embedded-CD boost, Building Controller**.**

1 INTRODUCTION

In developed countries, buildings account for about 20-40 percent of primary energy use nationwide (Pérez-Lombard et al., 2008, Shakih et al. 2014) and about 25-30 percent of the world's CO_2 emissions (Shakih et al., 2014, UNEP-SBCI, 2009). As the problem of energy performance in the building sector becomes more important, there is a need to improve processes for sustainably managing a building over its entire lifecycle —including design, construction, and operation.

Reduction of energy consumption in buildings would result in a correlated decrease in Green House Gas (GHG) emissions (Kate Randolph et al., 2014). To tackle the problem of building emissions, there have been efforts to optimize new building designs as well as improve existing buildings for energy management in order to reduce emissions (Soares et al., 2017).

In order to manage energy and hence reduce emissions effectively, there must be a way to control the operations of a building, for example, reducing the air conditioning when there is no one in the room (Chu et al.,2017). Such a control system would utilize computers, microprocessors and communication links to ensure proper operation of the building to meet the occupant's expectation while reducing energy consumption. (Levermore, G. J, 2000).

There have been several methods proposed for developing building control systems, including conventional controllers, agent-based controllers, model-based controllers etc. Model-based controllers have gained attention among researchers over the last decade because of its flexibility in terms of testing and reusability (Shakih et al., 2014). Discrete Event Modeling for Embedded Systems (DEMES) is a structured model-driven technique used to simplify embedded system development, increasing robustness and code reusability, all while decreasing the time spent developing (Wainer 2003, Niyonkuru and Wainer 2016). The DEMES method shows very good prospects, it promises model continuity which means preserving the model specification as much as possible during the development process and allows original models to be

deployed to the target hardware. The use of DEMES in the practical implementation of embedded systems especially with regards to smart building control systems and home automation is limited in literature.

In previous work, DEVS methodology for Real-Time (RT) systems using E-CD boost was presented with a simple case of a line follower robot (Niyonkuru and Wainer 2016). In this paper, using the line follower models as a starting point, we apply this methodology to a more complex case study in order to evaluate the performance and limitation of the DEVS methodology for RT systems. We propose the application of RTDEVS in the design and implementation of a smart building control system. This system will be modeled in DEVS, these models would be implemented in E-CD Boost and then the models would be gradually replaced with real hardware components while observing the behavior of our system in comparison with expected behavior.

2 BACKGROUND

Over the years several relevant smart building control systems have been proposed. (Nedelcu et al. 2009) designed and implemented a system in LabView to integrate different wireless technologies with wireless nodes to monitor various ambient conditions including temperature, lighting, and humidity. A similar system was also proposed by (Fernández-Caramés 2015) with the same tools but included intelligent power outlets to minimize power consumption.

(Baraka et al. 2013) proposed a design that incorporated a network of sensors and actuators on an Arduino that retrieved information from the environment. This method made use of low-cost controllers and offered the advantage of reduced cost when compared to the LabView method.

There have also been smart building control systems that were designed to work with android devices (Kumar 2014). With a Wi-Fi connection, individual devices can be connected to the internet and the user can control appliances remotely in order to minimize energy consumption.

There has been significant development in terms of hardware quality and sophistication of control systems. However, most if these systems require rigorous design methods and significant cost implications, they are not reusable or interoperable (Suárez-Albela 2016) meaning the same effort would be needed for every new control system design. Another challenge is that the testing of these systems is complicated.

Due to the increased complexity, scalability and heterogeneity of home automation systems, alternative approaches are needed to address these considerations. Model-based design methods provide solutions, but they are still limited to the initial stages and have limited practical implementation in literature. Real-Time Discrete Event Systems specifications (RTDEVS) methodology offers a unique way of making embedded system design simpler through formal modeling and direct application of the models to the real world. These models are re-useable and can reduce design time for future projects. RTDEVS is superior to other modeling techniques in this domain because it facilitates system verification because of its solid mathematical foundation, this is a challenge for other modeling techniques. (Niyonkuru and Wainer 2016)

DEVS (Discrete Event Systems specifications) is a modeling paradigm based on general systems theory. A real system modeled using DEVS is composed of atomic and coupled models. Systems whose state can change upon reception of an external input event or due to the expiration of a time delay can be specified in DEVS. (Zeigler, Kim, and Prähofer 2000). DEMES is based on DEVS theory. The DEMES cycle starts by defining the relationship between the system requirements and its physical environment. The system is then modeled using formal DEVS. The models can then be formally validated and simulated by stubbing the sensor input data. When the simulations are satisfactory the stubbed sensors can be replaced with the actual sensors on the target platform (Niyonkuru and Wainer 2016).

Embedded-CD Boost is a library written in C++ that facilitates the execution of DEVS models which are also defined in C++. The code can then be compiled for use on an embedded platform which makes use of the C++ Boost libraries. E-CD Boost is based on DEVS and is suitable for Real-time embedded systems

because it provides a rich structural representation of components and a formal means to explicitly specify timing which is central to the Real-time system (Niyonkuru and Wainer 2015). For ease of use, the ARM MBED libraries are used for interfacing input/output devices. MBED uses a specific device's hardware while allowing the core program to be somewhat hardware agnostic. (Niyonkuru and Wainer 2015).

2.1 Hardware Description

The target system used for this work is a Nucleo F411RE, with an ARM Cortex M4 processor and 512kB of flash. This board is ideal for this project for a number of reasons; It has ample storage for the E-CD Boost project which compiles to approximately 175kB, it offers Arduino connectivity making it easier to use ST connectors, it is MBED enabled which allows us to move our model controllers to different microcontrollers without compromising the model, and allows for on-board debugging and programming. The Nucleo board also has Arduino compatible headers, with 5 analog I/O ports and 13 digital I/O ports. The Nucleo's Arduino pinouts are shown in on the left Figure 1. (STM Electronics 2018)

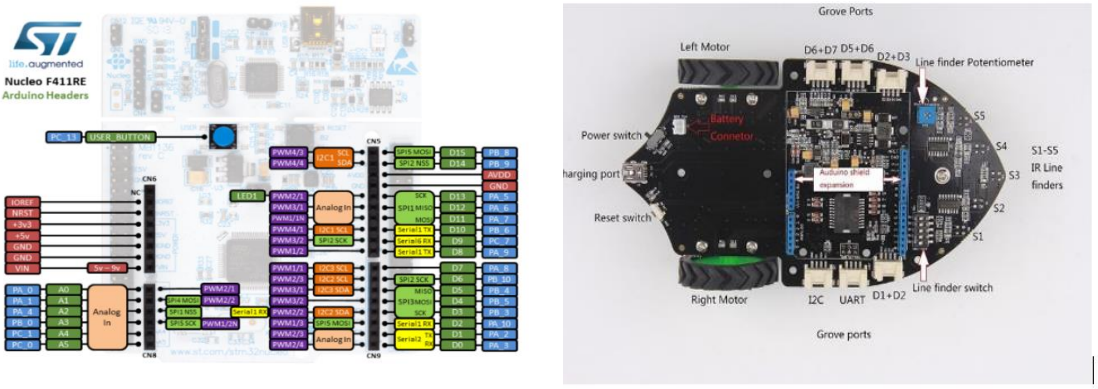

Figure 1: Nucleo Arduino pinouts (STM Electronics 2018) and Seeed Shield-Bot (Wikistudio 2018).

In Figure 1, the image on the left shows the pinouts for the Nucleo board while the image on the right shows the Seeed bot. All the sensors and actuators used in this project will be connected to the pins on the Nucleo board either directly or through the grove ports on the Seeed bot. The Nucleo board was mounted on a Seeed Shield-Bot, shown in Figure 1 on the right. This is to use its battery as well as its IR sensors to determine room occupancy.

Eight LEDs are used, four white LEDs, two red LEDs, and two green LEDs along with current limiting resistors. The LEDs represent room lighting and emergency exits. A Grove Light Sensor is used to detect ambient lighting in a room, a switch to represent a fire alarm, and a potentiometer to represent a thermometer. The potentiometer output is identical to the temperature sensor but simplifies testing.

3 METHODOLOGY

The objective of this project is to implement a Real-Time Building Control System using RTDEVS. We have implemented two controllers to create a smart building system. A description of the system of interest is shown in Figure 2. The architecture of the system includes two rooms fitted with lights, motion detectors, heat detector, an ambient light sensor, and a fire alarm switch. In line with DEMES, after the system of interest is identified, we model them in DEVS as atomic and coupled models, then the models are checked against expected behavior by running simulations. These models are then executed on the target hardware with the aid of a real-time executive. A summary of the modeling process includes conceptual model development, formal modeling with DEVS, implementation of the models in E-CD Boost and deploying the models directly to the target hardware.

3.1 Model Description

A conceptual model of the system of interest is shown in Figure 2. The control model is designed to control the amount of lighting and the model response in case of an emergency. The system would turn on lights in a room depending on the ambient light and occupancy. The emergency system includes a fire alarm and a heat detector located by two doors. Each door has two signs, one indicating that it is a safe path to exit and the other showing that it is unsafe. Although the controller cannot be seen in Figure 2, the logic behind the operation of the controllers is explained in Figure 3 and Figure 4.

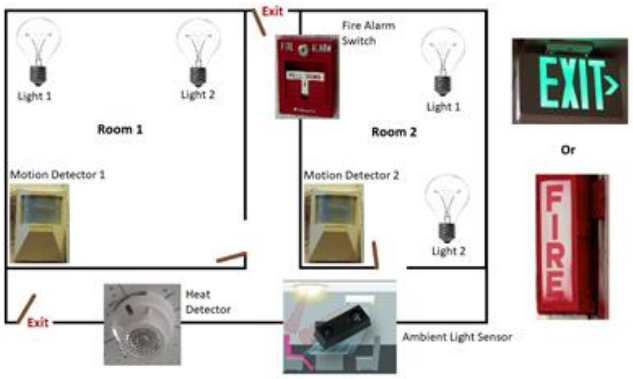

Figure 2: Conceptual Model of System.

.

The logic of the light control model can be explained with the state diagram in Figure 3, while that for the emergency control is explained in Figure 4.

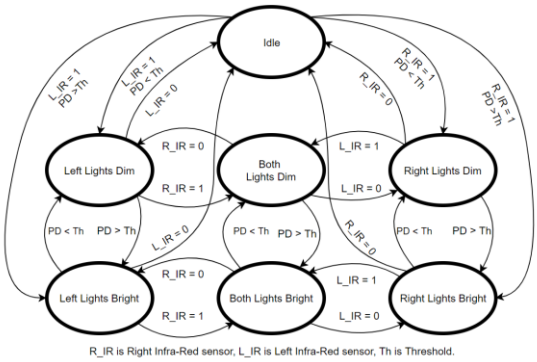

Figure 3: State Diagram for the Room Light Control Model.

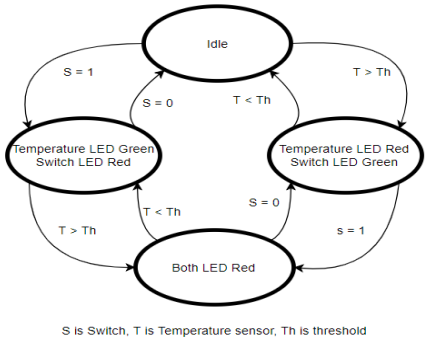

Figure 4: State Diagram for the Emergency Control Model.

Figure 3 shows the state diagram for the light control model. It has seven states shown as circles and several transitions indicated by the arrows. The lights are either entirely OFF, left ON, right ON, or both ON. The

model will monitor the IR Sensor that detects room occupancy to determine which lights should be turned ON. These transitions are represented in the diagram as the X_IR signals, which are digital inputs to the system. There is also a Photo Detector (PD) that will decide if the lights should be dim or bright based on the ambient lighting in the room. The PD is an analog sensor, so it is compared to a threshold, if it is above the threshold then any occupied room would have bright lights, if below then the lights would be dim. The logic of the emergency control model can be explained with the state diagram in Figure 4.

The state diagram shown in Figure 4 describes the four states for the emergency control model. The model starts in the idle state, with all the lights OFF. The transitions through the states on the left side of the image start when the switch is closed. The system will then enter an alarm state, with the switch's red LED and the temperature sensor's green LED lit up. If the temperature sensor then reports a voltage above the threshold it would sound an alarm, causing its light to switch from green to red.

3.2 Controller Model Description

This section describes the structure of the coupled models, which includes; the top model, the emergency model, and the room lighting. The top model is shown in Figure 5.

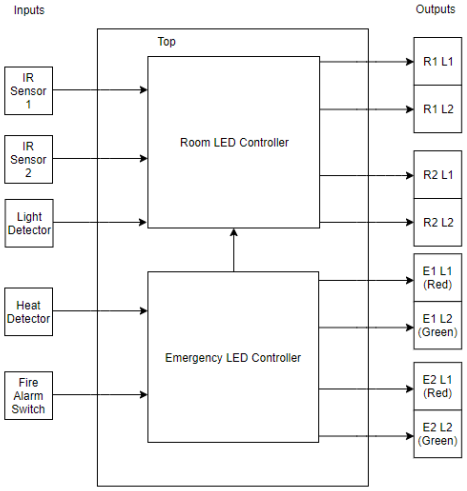

Figure 5: Top Model.

Figure 5 shows the top model which consists of two controllers, the room LED controller which is responsible for the room lighting and the emergency LED controller which is responsible for emergency control. There are five inputs to the model and eight outputs as shown. The operation of the components of the top model would be discussed in a later section of this paper. The Emergency LED Controller is made up of several atomic models as shown in Figure 6.

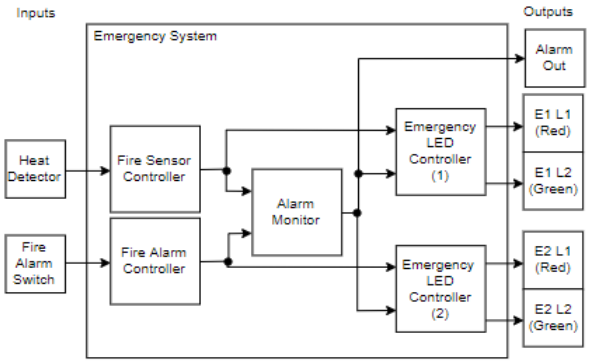

Figure 6: Emergency Control Model.

The Emergency control model shown in Figure 6 consists of five atomic models including fire sensor controller, fire alarm controller, alarm monitor, and emergency LED controllers. There are two inputs to the model and five outputs. The two Sensor Controller blocks interpret the input and determine if an alarm should be raised. The Alarm Monitor block does an OR operation on the Sensor Controller outputs, raising the alarm when any sensor triggers an alarm. The Emergency LED Controller turns on the red LED in response to a sensor alarm. If its sensor is off and the Alarm Monitor's alarm is triggered, then the green light is turned on. If neither of the input alarms are triggered thenboth lightsare turned off. The Alarm Out port goes to the Room LED Controller, shown in Figure 5.

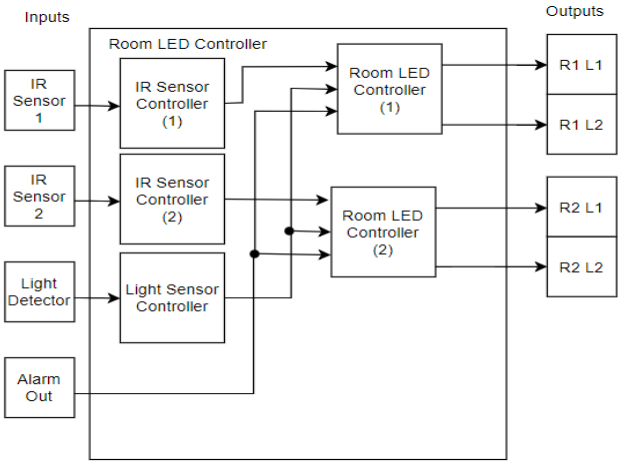

Figure 7: Room Light Control Model.

The Room Light model shown in Figure 7 consists of five atomic models including three sensor controllers and to LED controllers. There are four inputs to the model and four outputs from the model, the IR Sensor Controller will read a 0 when a line is present (the room is occupied) and output accordingly. The Light Sensor Controller checks if the ambient light in the room is above a threshold and sends an output depending on whether the room is dark or bright. Room LED Controller turns both lights on if the Alarm Out port is triggered, or if the room is occupied and the light sensor says it is dark. If the room is occupied and the light sensor says it is bright then it turns on one light.

3.3 Circuit Design

In addition to the major hardware discussed in section 2, external circuitry was designed. This includes the LED circuits for the LED controller and Emergency Controller.

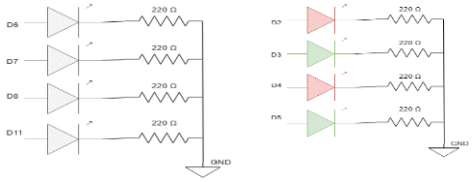

Figure 8: Room LED Controller Circuit and Emergency LED Controller Circuit.

The fire alarm is represented with a simple switch. Pin D12 is used as an active low input and is set up with an internal pull-up resistor. Closing the switch grounds the port and activates the alarm. The temperature sensor was modeled for this case study as a 10K Ohm potentiometer as mentioned previously.

Figure 9: Fire Alarm Circuit and Temperature Sensor Circuit.

The Grove light sensor was interfaced with port A5 which outputs an analog value between 0 and 5V. The A/D converter on the board enables this pin to be read and determines the light intensity on the sensor.

4 IMPLEMENTATION

This section describes the steps taken to implement the software in E-CD Boost and how the models are replaced by hardware.

4.1 Software Implementation

In this section we describe the steps taken to implement the models in E-CD Boost. Figure 10 shows a summary of the steps required for implementing the software.

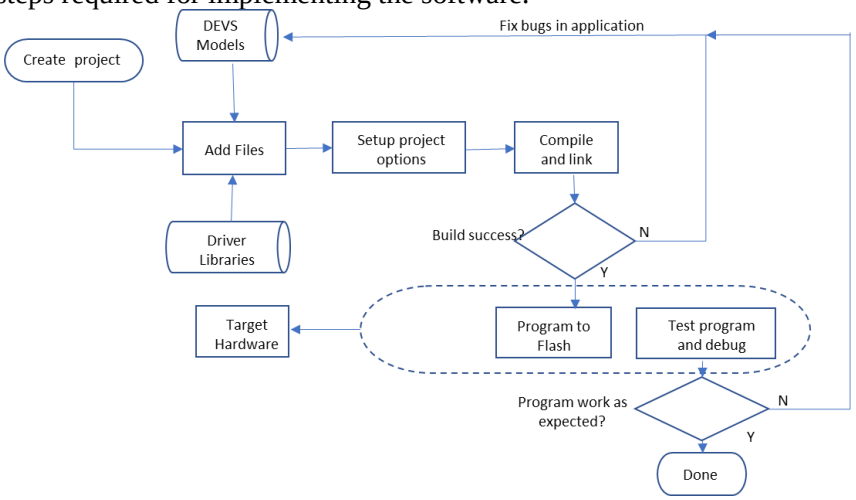

Figure 10: Software Implementation Steps.

A new project is created and then the driver libraries for ARM Cortex M4 (MBED library) is added to the project. We declare the pins using the MBED library and set the enable pin for the motor low, allowing us to use pins D10-D13 for other components other than the motors. We define input controller models (Temperature Sensor Controller, Light Sensor Controller, and the Digital Input Controller) and define ports and drivers. After the drivers have been defined and instantiated, the DEVS models are added to the project, the E-CD Boost tool already has a RT executive which allows us to run the project in different modes. All source codes are compiled and if the build is successful the program image is uploaded to the flash memory of the microcontroller.

The first stage of development was creating the models for state transitions. Once the models were validated using stubbed inputs, we connected the DEVS models with the hardware. The driver interfaces were used to map hardware sensors and actuators to model inputs and outputs. These port definitions that connect the physical port drivers to the DEVS model are implemented after all the required ports have been instantiated. After this, the root model which defines the links between the control unit coupled model and the systems I/O port driver is created. The snippet below shows the port definitions that connect the physical port drivers to the DEVS model. First, all the required ports are instantiated.

```
// Input ports
auto light_left = make_port_ptr<LIGHT_IN_LEFT<Time,
Message>>();
auto light_right = make_port_ptr<LIGHT_IN_RIGHT<Time,
Message>>();
auto ambient_light = make_port_ptr<AMBIENT_LIGHT_IN<Time,
Message>>();
auto fire_alarm = make_port_ptr<FIRE_ALARM<Time,
Message>>();
auto temperature = make_port_ptr<TEMPERATURE_IN<Time,
Message>>();
```

```
// Output ports
auto room1 = make_port_ptr<ROOM1_OUT<Time,
Message>>();
auto room2 = make_port_ptr<ROOM2_OUT<Time,
Message>>();
auto emerg1 = make_port_ptr<EMERGENCY1_OUT<Time,
Message>>();
auto emerg2 = make_port_ptr<EMERGENCY2_OUT<Time,
Message>>();
```

The snippet below shows the root Model which defines the links between the control unit coupled model and the systems I/O port drivers.

```
Time initial_time{Time::currentTime()};
erunner<Time, Message> root {ControlUnit,
//External Input Coupling                    //External Output Coupling
```

```
{light_left,rctrl1},
{ambient_light,rctrl1},
{light_right,rctrl2},
{ambient_light,rctrl2},
{fire_alarm,actrl2},
{fire_alarm,actrl1},
{temperature,actrl1},
{temperature,actrl2},
{room1,rctrl1},
{room2,rctrl2},
{emerg1,actrl1},
{emerg2,actrl2},
```

The above declaration links the components (switches, LEDs and sensors) with the control model acting as a controller on the target hardware. The control model used on the target hardware is same one used to control the DEVS models in the DEVS simulator. This allows us to test and develop the control system independent of the hardware. Once the control model is complete it is linked with the port drivers and deployed using the real sensors and actuators.

4.2 Hardware Implementation

For the hardware implementation, physical devices replace models and the exact hardware that replace each of the described models in section 3 is described in this section. Before that, a brief description of ports and hardware is done in the component description section.

4.2.1 Component Description

The Seeed Shield-Bot platform has five infrared sensors that on board that was utilized to model room occupancy detection. These sensors were connected to the A0, A1, A2, A3 and D4 headers on the Seeed Shield and can be disabled with onboard dip switches. For this study, dip switches 1 and 4 were enabled to use the IR sensors on A0 and A3, while switches 2, 3, and 5 were turned off.

Some simple external circuits were also needed as discussed in section 3. The room LEDs are driven through ports D6, D7, D8, and D11 skipping D9 and D10 as these are used to enable the motors on the Seeed Shield-Bot. The Emergency LED circuit is similar but utilizes ports D2, D3, D4 and D5, this time skipping D0 and D1 as these are used for USART debugging of the Nucleo board. This circuit also uses Red and Green LEDs as opposed to the white LEDs of the Room. The ports are and pins are shown in Figure 11.

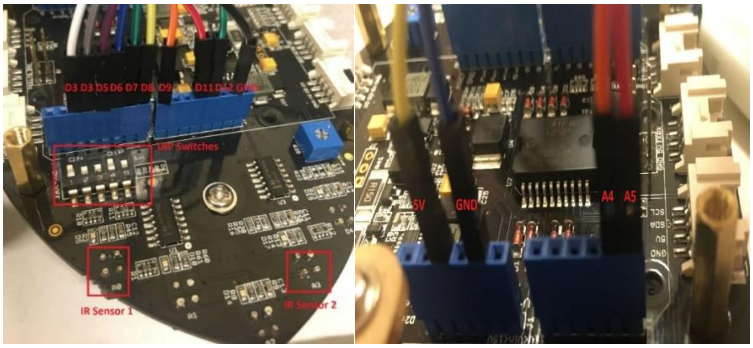

Figure 11: Seed Shield Showing IR sensors and Ports.

4.2.2 Light Sensor Controller

The light controller is made up of two sensors: a motion detector to know if the room is occupied and an ambient light sensor to determine how much light is required in the room. If the room is brighter then the light would be turned to fifty percent intensity. The two motion sensors are represented as the IR line finders on the shield bot. Room 1 occupancy is detected by the IR sensor on pin A0, Room 2 on A3. For simplicity, we use one ambient light sensor, this is shown in Figure 12. The system checks if the ambient light is above a threshold. If it is, it will consider the room to be bright. If the room is bright then only one LED

will be on in an occupied room. If the room is dark, then both LEDs will be on in an occupied room. The grove light sensor shown in Figure 12 outputs an analog value. This value is read using the onboard ADC and then passed as input to the sensor controller. Lighting that is above a certain threshold will mean that it is day. The threshold is determined experimentally using the hardware.

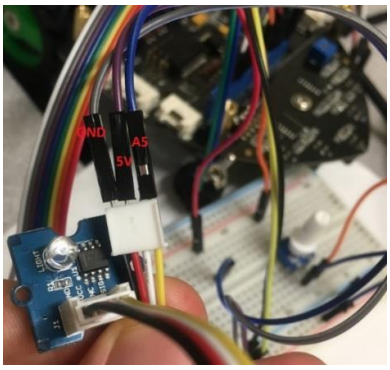

Figure 12: Ambient light sensors.

4.2.3 Emergency LED Controller

The emergency control system is made up of a red and green LEDs for each door and two emergency sensors. The first emergency sensor is a switch. The switch is connected to a digital input pin with a pull-up resistor, grounding it when active. The second is a heat sensor, raising the alarm when the temperature is above a safe threshold. The red LED will light up when the sensor's alarm near that door is triggered. The green LED will light up if the alarm is raised but not by the sensor near that door. The green LED shows that this is a safe path to exit. Additionally, when there is an alarm raised, all the lights in the room are turned on. This will allow for maximum vision in an emergency scenario. This controller checks if its alarm is raised, if so then it will display the red light, depicting an unsafe path. If the system alarm is on but the sensor is not, then it will light up green to show it is a safe exit path. If neither input is high the LED is off. The LED connections are shown in Figure 13.

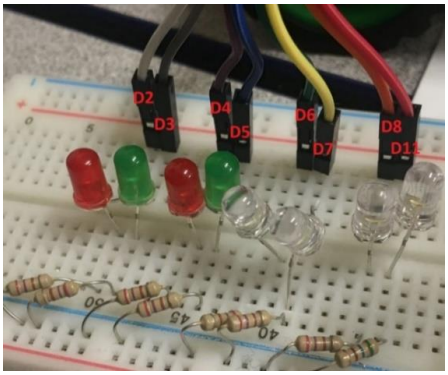

Figure 13: LED circuitry.

4.2.4 Temperature Sensor Controller and Alarm Circuitry

The Grove Temperature Sensor outputs an analog value. This value is read using the onboard ADC and then passed as input to the sensor controller. The Digital Input Controller is used for the fire alarm switch and the IR sensor, they handle the input and send an update to the system when it changes. The Alarm monitor ensures that the alarm is raised if any alarm sensor is triggered. The alarm circuitry is shown in Figure 14.

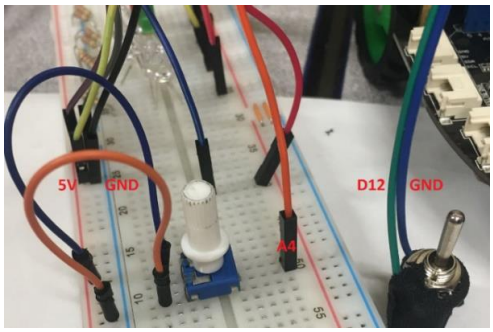

Figure 14: Alarm Circuitry.

5 RESULTS

In this section, we describe the experiments carried out to test the operation of the system and the results obtained under different sceneries. The results are shown in Table 1.

Table 1: Experiments and Result.

Test/Experiment	Input(s)	Result
Room 1 is occupied and there exists a bright ambient light	• Input to IR sensor associated with Room 1 • Light sensor above threshold	One light in Room 1 turns on
Room 1 is occupied and there is no ambient light	• Input to IR sensor associated with Room 1 • Light sensor below threshold	Two lights in Room 1 turn on
Room 1 and 2 are occupied and there exists a bright ambient light	• Input to IR sensor associated with Room 1 and Room 2 • Light sensor above threshold	One light in Room 1 turns on and one light in Room 2 turns on
Room 1 and 2 are occupied and there is no ambient light	• Input to IR sensor associated with Room 1 and Room 2 • Light sensor below threshold	Two lights in Room 1 turn on and two lights in Room 2 turn on
Fire alarm is pulled	• Fire alarm switch	Green light turns on at heat detector door, and red light turns on at fire alarm door indicating it is not a safe exit
Heat detector activated	• Temperature sensor above thresh-old	Green light turns on at fire alarm door, and red light turns on at heat detector door indicating it is not a safe exit
Both fire alarm and heat detector activated	• Fire alarm switch • Temperature sensor above the threshold	Both exit lights turn red indicating both exits are unsafe

The results are tabulated in three columns as shown in Table 1, the experiments are done on various inputs and the output behavior is observed. The results obtained from various experiments carried out show that for each input tested the output behavior observed is consistent with the expected behavior of the system.

6 CONCLUSION AND FUTURE WORK

In this work, we have successfully modeled a building control system and implemented it on the target hardware using the E-CD Boost software. This shows that the DEVS methodology can be used to develop a small control system to control some appliances in a building, this is a step in the right direction towards building more complex and efficient controllers. Although energy reduction is not covered in this work, a model of a building has been done and this would provide the foundation to run simulations of different

control algorithms and different sensor configurations to find out how to best optimize energy. The building control system can also be extended to include more functionalities like identifying occupants with their mobile devices and controlling building appliances accordingly

The method used is simple to implement and the models were preserved throughout the development process. It was also easy to compile models, test, debug and flash them into the hardware without much changes done to the original DEVS controller. The models used for a previous controller was the starting point for this project and the modifications to suit this project was trivial which further buttresses the point on reusability.

However, there was a limitation encountered when instantiating multiple instances of one atomic model. In ECD-Boost all the models subscribed to receive inputs would listen to all inputs and check if input's port matches the hardcoded port name for that individual model. If two models are instantiated, they will have the same port names and there will be no way to differentiate which message is for which model. A possible solution to this would be using the observer pattern where each output port has a list of input port listeners subscribed to get its messages. This way if many instances of a model exist it would be possible to separate the messages.

REFERENCES

Baraka, K., Ghobril, M., Malek, S., & Kanj, R. (2013). Low Cost Arduino/Android-Based Energy-Efficient Home Automation System with Smart Task Scheduling. *In Proceedings of the Fifth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, Madrid, Spain, 5–7 June 2013; pp. 296–301.

Chu, S., Cui, Y., & Liu, N. (2017). The path towards sustainable energy. *Nature materials, 16(1)*, p.16.

Fernández-Caramés, T.M. (2015). An Intelligent Power Outlet System for the Smart Home of the Internet of Things. *Int. J. Distrib. Sens. Netw*, pp. 1–11.

Kate, R., Doug, K., & Cole, R. (2014). Future Proofing Buildings: Resilience in The Age of Climate Change (*Research Sessions at Greenbuild*) pp.24-30.

Kumar, S. (2014). Ubiquitous Smart Home System Using Android Application *International Journal of Computer Networks & Communications*, 6, 33–42.

Levermore, G.J. (2000). Building energy management systems; application to low-energy HVAC and natural ventilation control. *E&FN Spon*. London, UK.

Nedelcu, A., Sandu, F., Machedon-Pisu, M., Aalexandru, M., & Ogrutan, P. (2009) Wireless-based Remote Monitoring and Control of Intelligent Buildings. *In Proceedings of the IEEE International Workshop on Robotic and Sensors Environments*, Lecco, Italy, 6–7; pp. 47–52.

Niyonkuru, D., & Wainer, G. (2016). A kernel for embedded systems development and simulation using the boost library. *In Proceedings of the Symposium on Theory of Modelling & Simulation Society for Computer Simulation International*, Pasadena, California, p.13.

Niyonkuru, D., & Wainer, G. (2015). Discrete-Event Modelling and Simulation for Embedded Systems. *Computing in Science & Engineering, 17(5)*, pp.52-63.

Pérez-Lombard, L., Ortiz, J. & Pout, C. (2008). A review on buildings energy consumption information. *Energy and buildings*, 40(3), pp.394-398.

Sbci, U.N.E.P. (2009). Buildings and climate change: summary for decision-makers. United Nations Environmental Programme, *Sustainable Buildings and Climate Initiative*, Paris, pp.1-62.

Shaikh, P.H., Nor, N.B.M., Nallagownden, P., Elamvazuthi, I. & Ibrahim, T. (2014). A review on optimized control systems for building energy and comfort management of sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34, pp.409-429.

Soares, N., Bastos, J., Pereira, L.D., Soares, A., Amaral, A.R., Asadi, E., Rodrigues, E., Lamas, F.B., Monteiro, H., Lopes, M.A.R. & Gaspar, A.R. (2017). A review on current advances in the energy and environmental performance of buildings towards a more sustainable built environment. *Renewable and Sustainable Energy Reviews*,77, pp.845-860.

St.com. (2018). NUCLEO-F446RE - STM32 Nucleo-64 development board with STM32F446RE MCU, supports Arduino and ST morpho connectivity - STMicroelectronics. [online] *Available at: https://www.st.com/en/evaluation-tools/nucleo-f446re.html* [Accessed 11 Dec. 2018].

Suárez-Albela, M., Fraga-Lamas, P., Fernández-Caramés, T.M., Dapena, A. and González-López, M., (2016). Home Automation System Based on Intelligent Transducer Enablers. *Sensors*, *16*(10), p.1595.

Wainer, G., & Giambiasi, N. (2001). Timed Cell-DEVS: modeling and simulation of cell spaces. *In Discrete event modeling and simulation technologies* (pp. 187-214). Springer, New York, NY.

Wiki.seeedstudio.com. (2018). Shield Bot V1.1. [online] *Available at: http://wiki.seeedstudio.com/Shield_Bot_V1.1* [Accessed 1 Dec. 2018].

Zeigler, B.P., Kim, T.G., Prähofer, H. (2000). Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems.

AUTHOR BIOGRAPHIES

KYLE BJORNSON is a fourth year undergraduate student in Computer Systems Engineering at Carleton University. His email address is kyle.bjornson@carleton.ca.

BEN EARLE is a fourth year undergraduate student in Computer Systems Engineering at Carleton University. After graduating this spring, he will start working on his M.Sc. at Carleton. His email address is ben.earle@carleton.ca.

JOSEPH BOI-UKEME is pursuing a PhD in Electrical and Computer Engineering at Carleton University where he researches on Cyber-physical Systems. His email address is joseph.boiukeme@carleton.ca.

GABRIEL WAINER is a Professor at the Department of Systems and Computer Engineering at Carleton University. He is a Fellow of the Society for Modeling and Simulation International (SCS). His email address is gwainer@sce.carleton.ca.