BacNet and Analog/Digital Interfaces of the Building Controls Virtual Testbed

Thierry Stephane Nouidui, Michael Wetter, Zhengwei Li, Xiufeng Pang, Prajesh Bhattachayra, Philip Haves

November 2011

DISCLAIMER

BACNET AND ANALOG/DIGITAL INTERFACES OF THE BUILDING CONTROLS VIRTUAL TESTBED

Thierry Stephane Nouidui[1], Michael Wetter[1], Zhengwei Li[2], Xiufeng Pang[1], Prajesh Bhattacharya[1], Philip Haves[1]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA, U.S.A.

[2]Georgia Institute of Technology, Atlanta, GA, U.S.A.

ABSTRACT

This paper gives an overview of recent developments in the Building Controls Virtual Test Bed (BCVTB), a framework for co-simulation and hardware-in-the-loop.

First, a general overview of the BCVTB is presented. Second, we describe the BACnet interface, a link which has been implemented to couple BACnet devices to the BCVTB. We present a case study where the interface was used to couple a whole building simulation program to a building control system to assess in real-time the performance of a real building. Third, we present the ADInterfaceMCC, an analog/digital interface that allows a USB-based analog/digital converter to be linked to the BCVTB. In a case study, we show how the link was used to couple the analog/digital converter to a building simulation model for local loop control.

INTRODUCTION

This paper describes two interfaces that have been implemented in the Building Controls Virtual Test Bed (Wetter, 2011). The BCVTB is a modular framework, developed at the Lawrence Berkeley National Laboratory. The BCVTB is based on the Ptolemy II software from UC Berkeley (Brook et. al., 2007). Ptolemy II is an open-source software framework supporting experimentation with actor-oriented design. Actors are software components that execute concurrently and communicate through messages sent via interconnected ports. A model is a hierarchical interconnection of actors. The BCVTB allows users to couple different simulation tools during the run-time and also to couple tools with hardware through the BACnet and the analog/digital interfaces. The BCVTB allows advanced users to

- define new Heating, Ventilating, and Air Conditioning (HVAC) components and systems in a modular, hierarchical, object-oriented, equation-based graphical modeling environment and couple them to whole building simulation programs,
- simulate innovative new HVAC system and control architectures for which models do not yet exist in off-the-shelve building simulation programs,
- analyze dynamic effects in HVAC systems, modeled in an object-oriented, equation-based graphical modeling environment such as Modelica (Fritzson et al., 1998), and their local and supervisory control loops, modeled in a modeling environment such as MATLAB/Simulink (Mathworks, 2010), Modelica or Ptolemy II, and
- simulate virtual experiments prior to full-scale testing in a laboratory or a real building in order to determine the range of required boundary conditions, the type of experiments that need to be conducted and, for example, to improve a control logic in simulation where iterations can be made faster than in an actual experiment.

In this paper, we describe the interfaces for hardware-in-the-loop experimentation that have been integrated in the BCVTB. The first interface is the BACnet (ASHRAE, 2004) interface. This interface allows users to couple the BCVTB with BACnet compliant building control systems. This paper shows an application where a whole building simulation program was linked to a building control system using the BACnet interface. The coupling of the building with the control system allowed the use of measured data to assess in real-time the performance of the building.

This paper also presents the analog/digital interface that has been implemented in the BCVTB. This interface allows the BCVTB to be linked to a USB-based analog/digital converter. We show in a case study how the link was used to couple a virtual building to a particular commercial product in order to perform a hardware-in-the-loop simulation.

BACNET

Introduction

The BACnet interface has been developed to allow a link between the BCVTB and real control system hardware. For the coupling, two new actors have been implemented and added to the BCVTB library:

- An actor, called *BACnetReader* that can read from BACnet devices.
- An actor, called *BACnetWriter* that can write to BACnet devices.

The *BACnetReader* and the *BACnetWriter* use the open source BACnet protocol stack developed by Karg (2009) and shipped with the BCVTB installation. These two actors require a configuration file that specifies the BACnet devices, the object types and the property identifiers with which data are to be exchanged.

Reading from BACnet

The *BACnetReader* actor reads an xml configuration file to determine what data it needs to read from BACnet devices. This configuration file specifies BACnet object types and their child elements. A child element can be either another object type or a BACnet property identifier. The xml has the following syntax:

```
<?xml version="1.0" encoding="utf-8"?>
<BACnet>
  <!-- Child elements are not shown. -->
</BACnet>
```

The above BACnet element requires at least one child element of the form

```
<Object Type="Device" Instance="123">
  <!-- Child elements are not shown. -->
</Object>
```

where the elements and attributes have the following values:

- The name of the element needs to be set to *Object*.
- The attribute *Type* needs to be set to *Device*.
- The attribute *Instance* needs to be set to its instance number, which is a unique number that is assigned at the discretion of the control provider.

Any *Object* element can contain other *Object* elements and other *PropertyIdentifier* elements.

The *Object* elements can have values for the attribute *Type* such as *Analog Input*, *Analog Output*, *Analog Value*, *Binary Input*, *Binary Output*, *Binary Value*, *Calendar*, *Command*, *Device*, *Event Enrollment*, *File*, *Group*, *Loop*, *Multi State Input*, *Multi State Output*, *Notification Class*, *Program*, *Schedule*, *Averaging*, *Multi State Value*, and *Trend Log*. A complete list of attributes can be found in Chapter 12 of the BACnet Standard (ASHRAE, 2004).

Each of these object types has its own set of properties. These properties are declared in the element *PropertyIdentifier* which has one attribute called *Name*.

The following code shows an example of a configuration file that was used to read data from BACnet device.

```
<?xml version="1.0" encoding="utf-8"?>
<BACnet>
 <Object Type="Device" Instance="637501">                    (1)
  <PropertyIdentifier Name="Local_Date"/>                    (2)
  <Object Type="Analog Input" Instance="1">                  (3)
        <PropertyIdentifier
            Name="Object_Identifier"/>                       (4)
        <PropertyIdentifier
            Name="Units"/>                                   (5)
        <PropertyIdentifier
            Name="Present_Value"/>                           (6)
  </Object>
  <Object Type="Analog Output" Instance="2">                 (7)
    <PropertyIdentifier
        Name="Present_Value"/>
  </Object>
  </Object>
 <Object Type="Device" Instance="637502">
  <Object Type="Analog Input" Instance="1">
        <PropertyIdentifier
            Name="Present_Value"/>
  </Object>
  <Object Type="Analog Output" Instance="3">
    <PropertyIdentifier
        Name="Present_Value"/>
  </Object>
 </Object>
</BACnet>
```

The numbered items have the following functionalities:

(1) declares the BACnet device of the control system.

(2) declares a BACnet property identifier of the device with instance number 637501. This statement will cause the *BACnetReader* to read the local date from the device.

(3) (7) declare BACnet object types that are children of the device object type with instance number 637501.

(4)(5)(6) declare BACnet property identifiers of the device with instance number 1. These statements will cause the *BACnetReader* to read its object identifier, its units and its present value.

Writing to BACnet

The *BACnetWriter* actor can write to BACnet devices. It provides the *WritePropertyService* that is specified in Section 15.9 of the BACnet Standard (ASHRAE, 2004). The configuration file that is read by this actor is almost identical to the one used for the *BACnetReader*. The only difference is that the xml elements of type *PropertyIdentifier* have the additional attributes *ApplicationTag*, *Priority* and *Index*.

- The parameter *ApplicationTag* specifies the data format that is used to send the value to the BACnet device. Possible entries are *NULL*, *BOOLEAN*, *UNSIGNED_INT*, *SIGNED_INT*, *REAL*, *DOUBLE*, *OCTET_STRING*, *CHARACTER_STRING*, *BIT_STRING*, *ENUMERATED*, *DATE*, *TIME*, *OBJECT_ID*, *MAX_BACNET_APPLICATION_TAG*.
- The parameter *Priority* sets the priority of the *write* operation. Allowed entries are any integers from 0 to 16. The highest priority is 1 and the lowest is 16. If *Priority* 0 is given, then no priority is sent.
- The parameter *Index* is the index number of an array. If this parameter is -1, the index is ignored and the entire array is referenced.

Figure 1 and *Figure 2* show the Ptolemy II system models that use the *BACnetReader* and the *BACnetWriter* actors to read and write properties from and to BACnet devices. A detailed description of how to configure these actors can be found in the BCVTB manual (BCVTB Documentation, 2011).

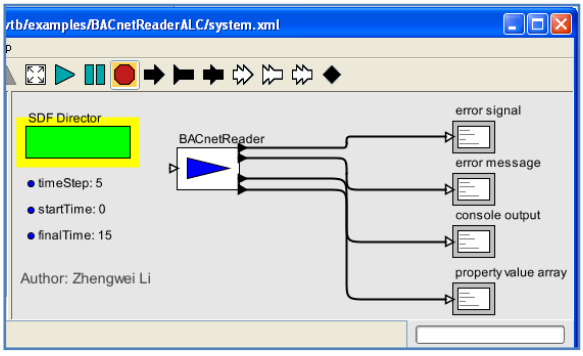

Figure 1: Ptolemy II system model that uses the BACnetReader actor

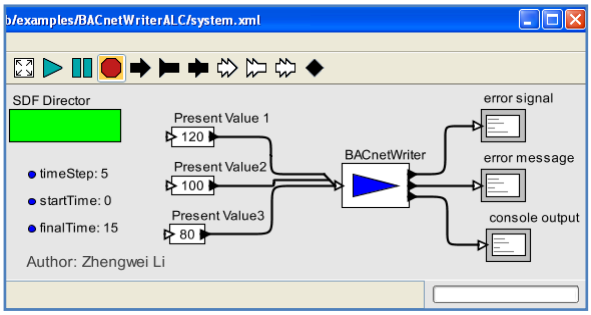

Figure 2: Ptolemy II system model that uses the BACnetWriter actor

Application

A real-time whole building performance monitoring tool that uses the *BACnetReader* has been implemented at a site in the Chicago, IL, area. The facility is a two-story building with a gross floor area of 70,000 ft^2 (6503 m^2). About 80% of the floor area serves as a drill deck for personnel training and ceremonies. The rest of the building is used for administration and is lightly occupied.

There are two 100-ton (352 kW) air-cooled chillers and associated chilled water pumps providing cooling for the whole building. Steam for heating is supplied by a campus-wide distribution system and converted to hot water locally. The drill deck is served by two single-zone Variable Air Volume (VAV) Air Handling Units (AHU). The office area is served by one VAV AHU with VAV terminal units. There is a classroom in the building, which is served by a single-zone VAV AHU.

A commercial Energy Management System (EMCS) is installed in this building. In order to accommodate the needs for real-time energy simulation, additional sensors were installed to provide the measurements for outdoor air dry bulb and relative humidity, wind speed and direction, global solar irradiation, direct normal solar irradiation and diffuse solar irradiation. To compare the simulation results with the actual performance of the building, sub-systems such as lighting, plug-loads and chillers, dedicated electrical power sub-meters were installed to measure the lighting power, plug load power and chiller power. The requirements for sensor accuracy were taken from Gillespie et al. (2007).

A whole-building simulation model representing the design intent of the envelope, the HVAC system, the lighting system and the control system was created for EnergyPlus (U.S. Department of Energy, 2011). The model was then calibrated based on EMCS trend data collected between April and July, 2010 (O'Neill et al., 2011). Since the EMCS uses a proprietary communication protocol, a BACnet server was installed and connected to the EMCS so that the *BACnetReader* in the BCVTB can communicate with the EMCS. The real-time simulation is launched by

starting the BCVTB through the Graphical User Interface (GUI) or through the console. The latter allows use of the BCVTB in an automated workflow or in a window-less system.

Figure 3 shows the overall system architecture that was implemented at this facility. It consists of two sub-systems: (i) the EMCS that serves as the data acquisition system and (ii) the real-time simulation environment that integrates the EnergyPlus simulation, a PostgreSQL database and the *BACnetReader*, and synchronizes the simulation to real-time. The sub-systems reside in two different computers connected using a Local Area Network (LAN).

There are two processes running in parallel. The *BACnetReader* acquires the relevant EMCS data through an Ethernet connection. The sampling interval is 5 minutes. The data is then passed to the PostgreSQL database. The EnergyPlus simulation program establishes the communication with the server that is launched by the BCVTB. At each time step, the EnergyPlus simulator receives the weather data as inputs and advances the model by one time step. Once the simulation is finished for that time step, the BCVTB writes the results to the database.

The existing EMCS has about 1,200 control points including both physical points and virtual points such as control set points. About 700 control points that are relevant to the proof-of-concept demonstration have been made accessible through the BACnet server. The *BACnetReader* in the BCVTB reads these data points and sends them to the database. Several manual checks have been conducted by exporting the data from the database to a spreadsheet and comparing to the trending data in the EMCS over one month period. The comparison shows that the data stored in the database match exactly the trending data in the EMCS.

Figure 4 shows a comparison of the simulated and measured building total electric power from October 26 to October 31, 2010. The blue line represents the simulation results while the red line represents the actual measurements. As can be seen from the chart in *Figure 4*, significant differences between the simulated and measured performance were recorded. Further analysis of the electric power indicated that different chiller operation strategies were the cause of the considerable performance deviations. The difference was due to the chiller being scheduled to be off after October 15 in the EnergyPlus model, which was based on design documents, while in reality, chiller ON/OFF was only controlled by outside air temperature. Simulation showed that free cooling was sufficient to handle the building cooling load.

ANALOG/DIGITAL INTERFACE

Introduction

An interface between the BCVTB and USB-based analog/digital converters has been developed. Since the interfaces to such devices are specific to particular product lines, it was necessary to use a particular product for the first implementation and the USB-1208LS (Measurement Computing Corporation, 2011) was selected. Two actors have been implemented to access *read- and write-* functions that are provided in a Dynamic Link Library (DLL) by the manufacturer. This allows the BCVTB to be linked to this analog/digital converter. This approach can be extended to different hardware. Similar to the BACnet interface, two actors have been implemented:

- An actor, called *ADInterfaceMCCReader* that can read from the analog/digital converter.
- An actor, called *ADInterfaceMCCWriter* that can write to the analog/digital converter.

USB-1208LS

The bus-powered USB-1208LS is an analog and digital input/output interface to any USB port. This module provides eight single-ended or four differential analog inputs with 12-bit resolution. When configured for single-ended mode, each analog input has 11-bit resolution, due to restrictions imposed by the analog/digital converter. In this mode, the input signal is referenced to signal ground and the input range is ±10V.When configured for differential mode, each analog input has 12-bit resolution. In this mode, the input signal is measured with respect to a reference signal and the input range must remain between −10V to +20V range. The USB-1208LS offers sample rates up to 1.2 kilosamples/sec. In addition to the analog inputs, the unit provides two 10-bit analog outputs.

Calibration

Prior to using the USB-1208LS, its analog inputs and outputs need to be calibrated. This can be done with the *InstaCal* (Quick Start Guide, 2010) program which shipped with the converter. *Insta*Cal is an installation, configuration, calibration, and test program for use with the analog/digital converter. With *Insta*Cal, you can change device configuration settings, calibrate analog inputs and outputs, and test the device's analog channels.

In the remainder of the paper the terminologies

- *analog/digital converter* will refer to the USB-1208LS that encapsulates an analog to digital and a digital to analog converter,

- *analog to digital converter* will refer to the process of reading data from the USB-1208LS, and
- *digital to analog converter* will refer to the process of writing data to the USB-1208LS.

Reading from the analog/digital converter

The *ADInterfaceMCCReader* is an actor that reads an xml configuration file to determine what data needs to be read from the analog to digital converter.

The xml-configuration file has the following syntax:

<?xml version="1.0" encoding="utf-8"?>
<ADInterfaceMCC>
<!-- Child elements are not shown. -->
</ADInterfaceMCC>

The above element *ADInterfaceMCC* must have at least one child element of the form

<Object BoardNumber = "0" ChannelNumber = "0" ChannelGain = "2" ChannelOptions = "0" ApplicationTag = "read"/>

where the elements and attributes have the following values:

- The element name needs to be set to *Object*.
- The attribute *BoardNumber* is the board number associated with the board used to collect the data.
- The attribute *ChannelNumber* is the analog to digital (A/D) channel number.
- The attribute *ChannelGain* is the A/D range code.
- The attribute *ChannelOptions* is reserved for future use by the hardware manufacturer and should be set to zero.
- The attribute *ApplicationTag* is a value that needs to be set to *read*.

A detailed description of these attributes can be read from the BCVTB manual.

To read data from the analog to digital converter, the *ADInterfaceMCCReader* actor calls an executable program that is in the *bcvtb/lib/adInterfaceMCC-stack* directory. The low level implementation of this function is

java -jar adInterfaceReader.jar BoardNumber ChannelNumber ChannelGain ChannelOptions

where the program arguments are replaced by the values specified in the xml configuration file.

Writing to the analog/digital converter

The *ADInterfaceMCCWriter* is an actor that is similar to the *ADInterfaceMCCReader***.** In contrast to the reader, the *ADInterfaceMCCWriter* needs as inputs the data that should be written to the digital to analog converter. Its xml configuration file has the same syntax as the configuration file of the reader:

<?xml version="1.0" encoding="utf-8"?>
<ADInterfaceMCC>
<!-- Child elements are not shown. -->
</ADInterfaceMCC>

The above element *ADInterfaceMCC* has identical syntax to the child element of the *ADInterfaceMCCReader* with the only differences being that

- the attribute *ChannelNumber* is the digital to analog (D/A) channel number,
- the attribute *ChannelGain* is the D/A range code, and
- the attribute *ApplicationTag* is a value that must be set to *write*.

Similar to the *ADInterfaceMCCReader*, the adInterfaceMCC-stack provides the following function to write to the digital to analog converter:

java -jar adInterfaceWriter.jar BoardNumber ChannelNumber ChannelGain ValueToBeWritten ChannelOptions

The program argument *ValueToBeWritten* is the value that will be written to the digital to analog converter.

Application

Figure 5 shows an example where the *ADInterfaceMCCReader* and the *ADInterfaceMCCWriter* have been used to couple a building simulation model with the analog/digital converter.

This application demonstrates the capability to link the BCVTB to analog devices, such as sensors or actuators that can be used in a hardware-in-the-loop simulation.

Some of the actors used for the simulation are numbered. This case study emulates an application where the BCVTB is used for hardware-in-the-loop simulation.

In this configuration, the *ADInterfaceMCCReader* (1) reads the current value of a voltage signal that comes from a test board (*Figure 6*) and is applied to the input of the analog to digital converter. In a real application, this signal could have been generated by a power meter measuring plug loads in a real building. This voltage signal is then scaled to represent the measured plug load power (2), added to the heating power (3) and fed into a room model (4). The room model is a surrogate for the real building. A proportional controller (5) varies the heating power

to regulate the simulated room air temperature at a desired set-point. The room model computes the new room air temperature and sends its value to the *ADInterfaceMCCWriter* (6), which then controls the digital to analog converter, producing an output voltage that is proportional to the room air temperature. This voltage signal can then be used for further applications.

Another application, illustrated in *Figure 7*, is to use the analog/digital interface for hardware-in-the-loop control. The room air temperature in a real building (1) is measured with a temperature sensor (2). The voltage from the sensor (3) is input to the analog to digital converter (4) and the digital value is sent to a building simulation program (6) by using the analog/digital interface (5). The role of the building simulation program is to compute the actuator position that will be needed to keep the room air temperature at a desired set-point. The analog/digital interface (7) maps the required actuator position to a voltage value and controls the digital to analog converter (8), which then generates the voltage signal (9), which is then applied to an actuator (10) in the real building.

SUMMARY

This paper described two interfaces that have been recently added to the BCVTB. The first interface is the BACnet interface. This interface is used to couple the BCVTB with BACnet compliant building control systems. The second interface is an interface to a USB-based analog/digital converter for linking the BCVTB to an analog/digital data acquisition device. Two application cases have been presented.
The BACnet and the analog/digital interfaces extend the applicability of the BCVTB to read and write real measured data. These data can then be used in other tools for real-time assessment of the performance of real buildings. Additionally, the BACnet and the analog/digital interfaces also extend the capability of the BCVTB to operate in hardware-in-the-loop mode.

ACKNOWLEDGEMENT

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

REFERENCES

ASHRAE. 2004. ANSI/ASHRAE Standard 135-2004, BACnet - A Data Communication Protocol for Building Automation and Control Networks. ISSN 1041-2336.

BCVTB Documentation. 2011. Online available at: http://simulationresearch.lbl.gov/bcvtb/releases/1.0.0/doc/manual/index.xhtml [last accessed: 05/20/2011].

Brook C., Lee E., Liu X., Neuendorffer S., Zhao Y., Zheng H. 2007. Ptolemy II – heterogeneous concurrent modeling and design in Java. Technical report No. UCB/EECS-2007-7, Berkeley, CA: University of California at Berkeley.

Fritzson P. and Engelson V. 1998. "Modelica, A Unified Object-Oriented Language for System Modeling and Simulation." In Proceedings of ECOOP'98 (the 12th European Conference on Object-Oriented Programming).

Gillespie K. L. Jr, Haves P., Hitchcock R. J., Deringer J. J., Kinney K. 2007. A Specification Guide for Performance Monitoring System. Berkeley, CA: Lawrence Berkeley National Laboratory, http://cbs.lbl.gov/performance-monitoring/specifications/ [last accessed: 05/20/2011].

Karg S. 2009. http://bacnet.sourceforge.net/ [last accessed: 05/22/2011].

Mathworks. 2010. MATLAB/Simulink. Online available at: http://www.mathworks.com [last accessed: 05/20/2011].

Measurement Computing Corporation. 2011. Online available at: http://www.mccdaq.com/ [last accessed: 05/20/2011].

O'Neill, Z. D., B. Eisenhower, S. Yuan, T. Bailey, S. Narayanan and V. Fonoberov. 2011. Modeling and Calibration of Energy Models for a DoD Building. Accepted. ASHRAE 2011 Annual Meeting. Montreal, Québec, Canada. June 25–29, 2011.

Quick Start Guide. 2010. Online available at: http://www.mccdaq.com/PDFs/Manuals/DAQ-Software-Quick-Start.pdf [last accessed: 05/22/2011].

U.S. Department of Energy (DOE). 2011. EnergyPlus. Online available at http://apps1.eere.energy.gov/buildings/energyplus/energyplus_about.cfm[last accessed: 05/20/2011].

Wetter M. 2011. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed, *Journal of Building Performance Simulation*, First published on: 11/05/2010 (iFirst), DOI: 10.1080/19401493.2010.518631.

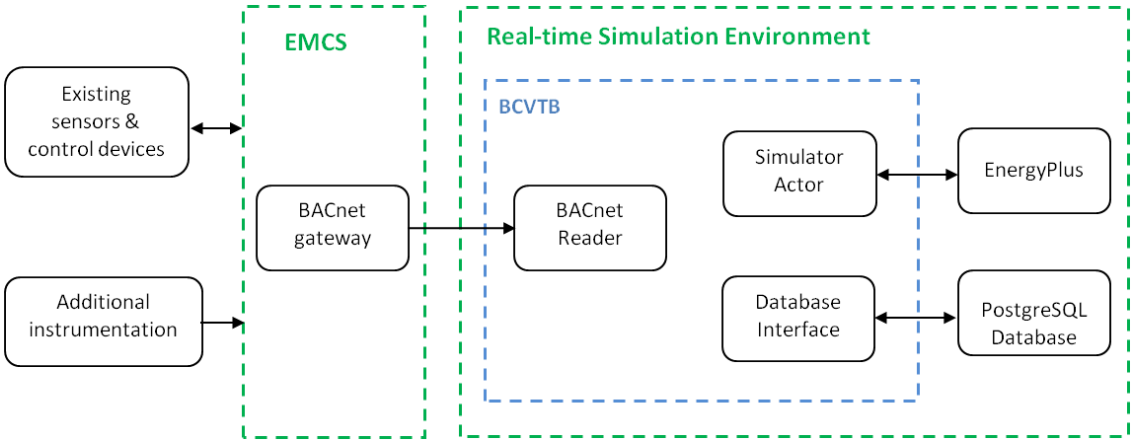

Figure 3: Overall system architecture

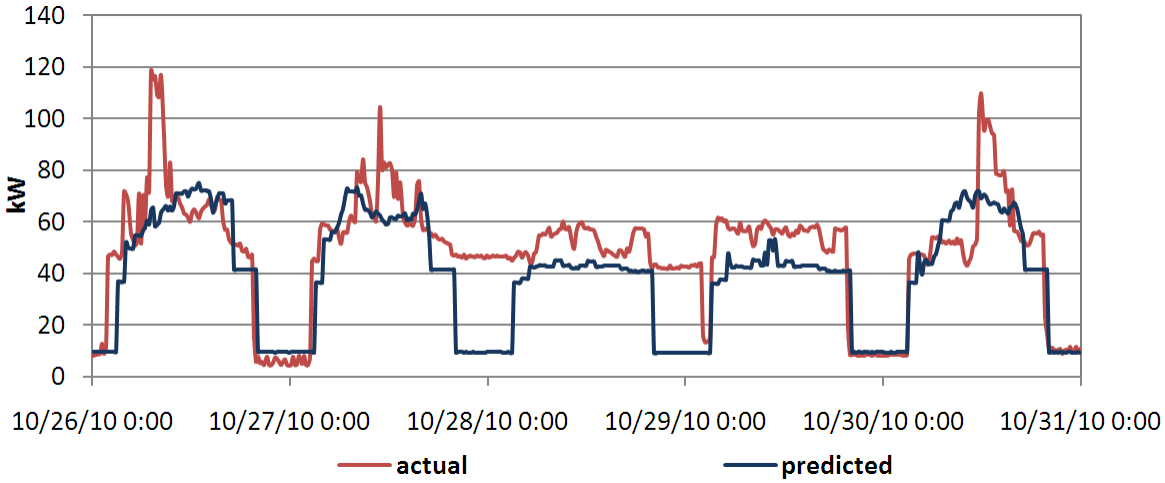

Figure 4: Simulated and measured building total electric power

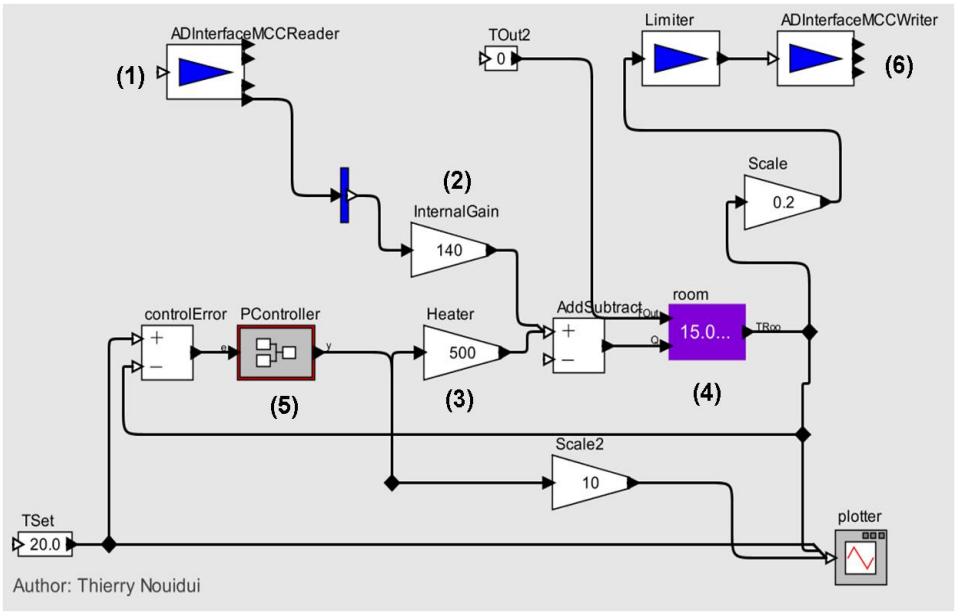

Figure 5: Ptolemy II system model that uses the ADInterfaceMCCReader and the ADInterfaceMCCWriter actors

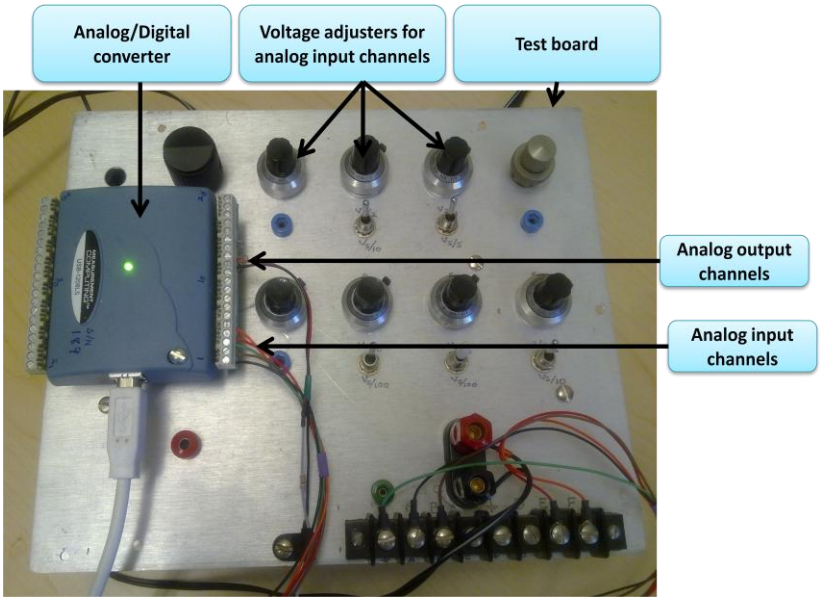

Figure 6: Test board with the USB-based analog/digital converter (USB-1208LS)

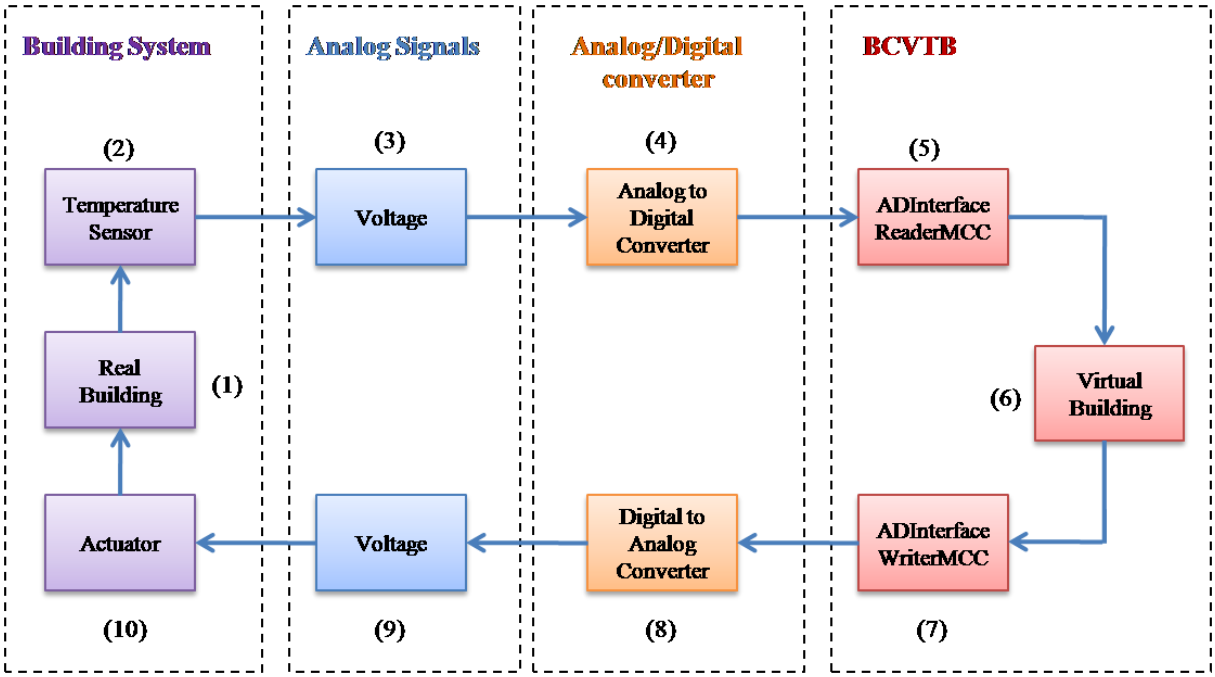

Figure 7: Example of an application that could use the BCVTB for hardware-in-the-loop