12-31-2017

Development of Economic Water Usage Sensor and Cyber-Physical Systems Co-Simulation Platform for Home Energy Saving

Joe Singer
Santa Clara University, jsinger@scu.edu

Santa Clara University

Department of Mechanical Engineering

December 31st, 2017

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Joe Singer

ENTITLED

DEVELOPMENT OF ECONOMIC WATER USAGE SENSOR AND CYBER-PHYSICAL SYSTEMS CO-SIMULATION PLATFORM FOR HOME ENERGY SAVINGS

BE ACCEPTED IN PARTIAL FILFILLMENTS OF THE REQUIREMENTS FOR THE DEGREE

OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

HOHYUN LEE, PhD, THESIS ADVISOR

WALTER YUEN, PhD, THESIS READER

DRAZEN FABRIS, PhD, CHAIRMAN OF DEPT.

Development of Economic Water Usage Sensor and Cyber-Physical Systems Co-Simulation Platform for Home Energy Saving

By
Joe Singer

MASTERS THESIS

Submitted to the Department of Mechanical Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Master of Science in Mechanical Engineering

500 El Camino Real, Santa Clara, CA 95053

2017

Abstract

In this thesis, two Cyber-Physical Systems (CPS) approaches were considered to reduce residential building energy consumption. First, a flow sensor was developed for residential gas and electric storage water heaters. The sensor utilizes unique temperature changes of tank inlet and outlet pipes upon water draw to provide occupant hot water usage. Post processing of measured pipe temperature data was able to detect water draw events. Conservation of energy was applied to heater pipes to determine relative internal water flow rate based on transient temperature measurements. Correlations between calculated flow and actual flow were significant at a 95% confidence level. Using this methodology, a CPS water heater controller can activate existing residential storage water heaters according to occupant hot water demand. The second CPS approach integrated an open-source building simulation tool, EnergyPlus, into a CPS simulation platform developed by the National Institute of Standards and Technology (NIST). The NIST platform utilizes the High Level Architecture (HLA) co-simulation protocol for logical timing control and data communication. By modifying existing EnergyPlus co-simulation capabilities, NIST's open-source platform was able to execute an uninterrupted simulation between a residential house in EnergyPlus and an externally connected thermostat controller. The developed EnergyPlus wrapper for HLA co-simulation can allow active replacement of traditional real-time data collection for building CPS development. As such, occupant sensors and simple home CPS product can allow greater residential participation in energy saving practices, saving up to 33% on home energy consumption nationally.

Acknowledgements

Portions of this publication and research efforts are made possible through the support of NIST via federal award #70NANB17H039.

Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States. Certain commercial products are identified in order to adequately specify the procedure; this does not imply endorsement or recommendation by NIST, nor does it imply that such products are necessarily the best available for the purpose.

I would also like to acknowledge Santa Clara University for financial support, and Michael Simmons for designing and manufacturing the clamps.

Additionally, thank you to Dr. Hohyun Lee for his excellent advising, and Chenli Wang for his feedback and support.

Lastly, thank you to my family and friends who have supported and motivated me through the thesis writing process.

Table of Contents

List of Figures

List of Tables

Nomenclature

C_p	specific heat [J/kg·K]
dx	axial unit length [m]
h	convection heat transfer coefficient [W/m^2·K]
k	thermal conductivity [W/m·K]
M	thermal mass, $\rho \cdot C_p \cdot A/dx$, [J/ m^2·K]
Nu_w	Nusselt number of internal water
Pr	Prandtl number
R	thermal resistance [K/W]
Re	Reynolds number
r	pipe radius [m]
T	measured pipe surface temperature [°C]
T_w	internal water temperature [°C]
T_∞	measured ambient air temperature [°C]
t	time [s]
U	internal overall heat transfer coefficient [W/m^2·K]
U_∞	external overall heat transfer coefficient [W/m^2·K]

Greek Letters

μ	viscosity [kg/m·s]
ρ	density [kg/m^3]

Subscripts

$(\cdot)_c$	cold inlet pipe conditions
$(\cdot)_h$	hot outlet pipe conditions
$(\cdot)_w$	internal water condition
$(\cdot)_\infty$	ambient conditions
$(\cdot)_1$	first point in data set
$(\cdot)_2$	last point in data set

Abbreviations

CPS	Cyber-Physical System
DOE	Department of Energy
FMU	Functional Mockup Unit
HLA	High Level Architecture
IDF	Input Data File
IEEE	Institute of Electrical and Electronics Engineers
NIST	National Institute of Standards and Technology
RTI	Run-Time Infrastructure
SEED	Simulation Environment Distributor
UCEF	Universal CPS Environment for Federation

Chapter 1: Introduction

Energy usage in the United States is an essential part of daily life. Often taken for granted, energy is regularly used in common sectors such as transportation, industrial plants, and commercial/residential buildings. A report by Lawrence Livermore National Laboratory and the United States Department of Energy (DOE) suggests that 81% of national consumed energy sources from natural gas, coal, and petroleum [1]. Each of these fossil fuel sources leaves a large carbon footprint, releasing abundant amounts of CO_2 into our atmosphere upon use [2], leading to increasing atmospheric temperatures [3, 4]. Based on the two 59-point data sets of CO_2 and temperature [5], represented in Figure 1.1, linear regression analysis indicates a linear correlation of 0.95. Alternatively speaking, the probability that 59 data points of two uncorrelated variables reaching the same level of correlation is 0.05%, resulting in significant linear correlation evidence between atmospheric CO_2 and temperature.

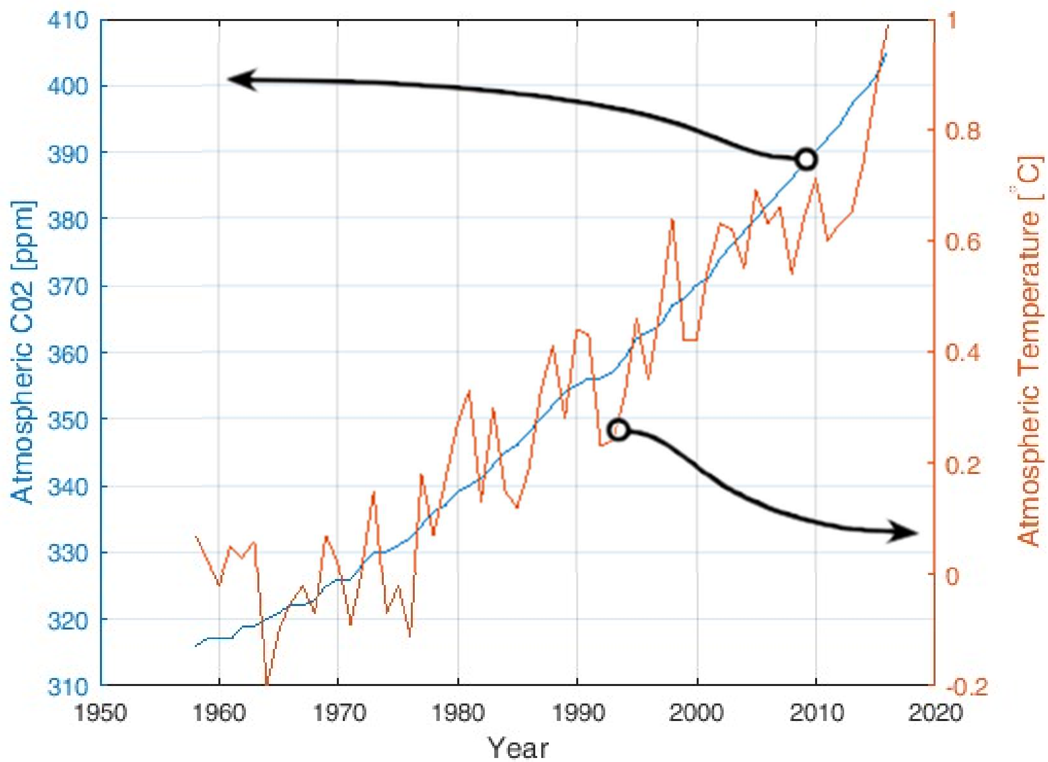


Figure 1.1: Carbon dioxide (CO2) production form fossil fuels is dissipated into the atmosphere. Atmospheric CO2 (measured in parts per million (ppm)) significantly correlates to increasing atmospheric temperatures [5].

Residential homes in particular posses energy saving potential to reduce the national carbon footprint. Represented in Figure 1.2, 69% of home energy consumption sources from fossil fuels [1], where 35% of the consumed energy is wasted. Potential energy losses contributing to this waste can be attributed to Carnot efficiencies of heating devices, home energy loss due to environment temperature differences, and inefficient or unnecessary activation of home appliances.

Appliance efficiencies and energy loss can be improved with newer building materials and more efficient appliances. However, buildings and their incorporated systems are rarely updated, where homes are not fully replaced for an average of 65-70 years [6, 7]. Automobiles, for comparison, are complex systems that have become increasingly efficient over the years [8]. These new efficient automobile technologies can more regularly be incorporated because vehicles have a relatively shorter lifespan of 5-10 years [9, 10]. At these rates, a more immediate energy saving impact for the residential sector can be achieved by appropriately controlling activation of home appliances.

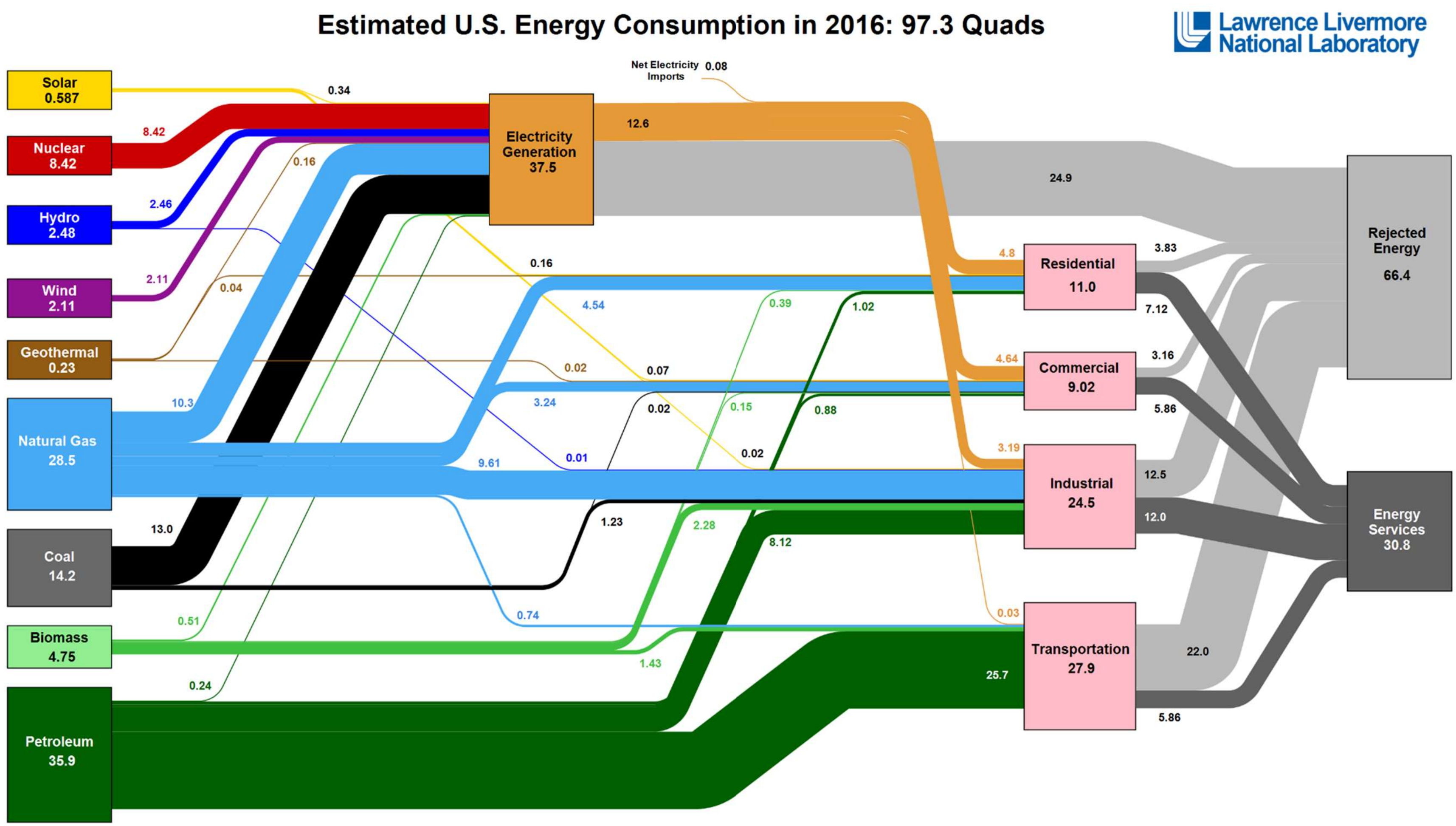


Figure 1.2: Energy generation to consumption flow chart as produced by Lawrence Livermore National Laboratory and the Department of Energy. Numbers have units of quadrillion BTUs. Residential energy consumption accounts for about 11% of national energy consumption [1]. Reproduced without permission.

The recent paper by Nguyen and Aiello [11] suggested that energy conscious behaviors in residential homes can lead to significant energy savings without need for home replacements. Such behaviors can achieve 33% and 50% energy savings, respectively compared to the design point and compared to those demonstrating wasteful behaviors, shown in Figure 1.3. Unfortunately, the typical home occupant does not consciously control appliances, such as water heaters or HVAC, for optimized energy usage [12] as it involves high amounts of effort. Incorporating automated intelligence into home appliances can allow existing and potentially wasteful devices to exhibit energy conscious savings. Using information from Figure 1.2, and assuming all residential homes are capable of reducing consumed energy by 33% through home intelligence, an upwards of 3.6 quadrillion BTUs of energy can be saved nationally on an annual basis in the United States' residential sector. This amount of energy savings would effectively lower fossil fuel demand, lowering the nation's carbon footprint.

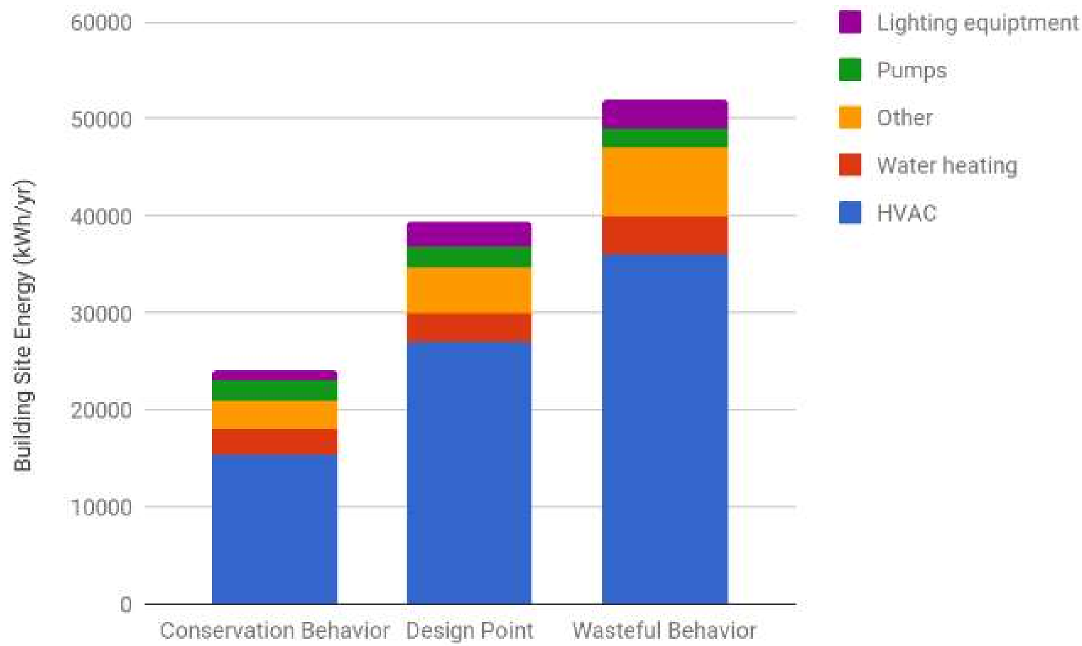


Figure 1.3: Differentiation between energy conscious and wasteful behaviors, as compared to the design point of select appliances [11].

1.1 Cyber-Physical Systems

Cyber-Physical Systems (CPS) are described as systems involving interactions between computation and physical components [13]. CPS are constantly taking input through sensors to monitor our physical world. Sensor information provides a knowledge basis for computational decision-making processes, where the results then allow action, operation, and control of other physical elements such as actuators, or more specifically, building appliances. This cycle of CPS operation, represented in Figure 1.4, can be used to integrate and interconnect systems to provide new functionalities for several economic sectors. To name a few, the monitoring and control functions of CPS extend to transportation, manufacturing, healthcare, buildings, and energy [14-17].

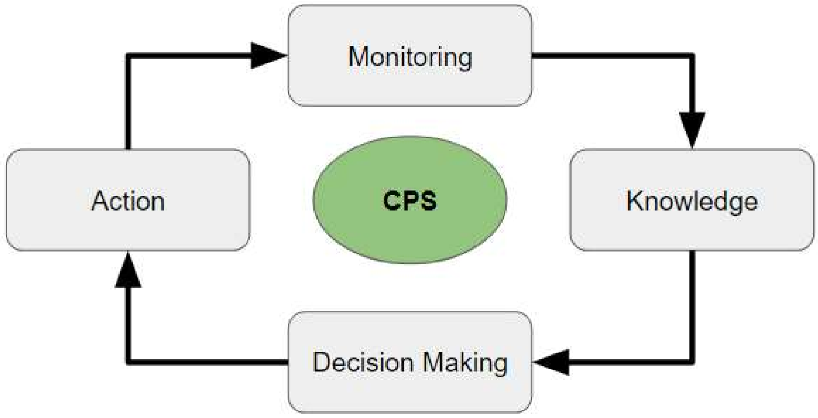


Figure 1.4: CPS development and use cycle, constantly collecting input, making decisions, and controlling output.

Currently, CPS face several challenges including safety, security, performance, reliability, data transfer, and cost. Solutions to these challenges are currently being developed, but differ between domains, services, applications, and devices. For building CPS, progress has been made in attempt to develop reliable and high-performance systems called Energy Management Systems (EMS). EMS have been developed to automatically control lighting and HVAC in commercial buildings based on occupant activity. For example, Distech Controls has a variety of building management technologies capable of managing and optimizing energy efficiency and building comfort to save 30% of a building's energy consumption [18]. Though effective, these systems are expensive, causing a residential market penetration barrier for residential use.

Residential CPS development is difficult because buildings have many interconnected factors to account for. Figure 1.5 shows how home operation can be affected by many components such as home location, construction, and occupancy. These factors can vary depending on climate, age of the house, and occupants themselves, making robust CPS more complex. Often, this diversity of home operation cannot be captured in a single CPS component, but it may be required for CPS testing and validation. Such CPS experimentation necessitates interdisciplinary knowledge and real-time data collection, which requires significant amounts of time and resources [19]. Building CPS are not mainstream because high market costs stem from time and resources required for commercial CPS development and deployment.

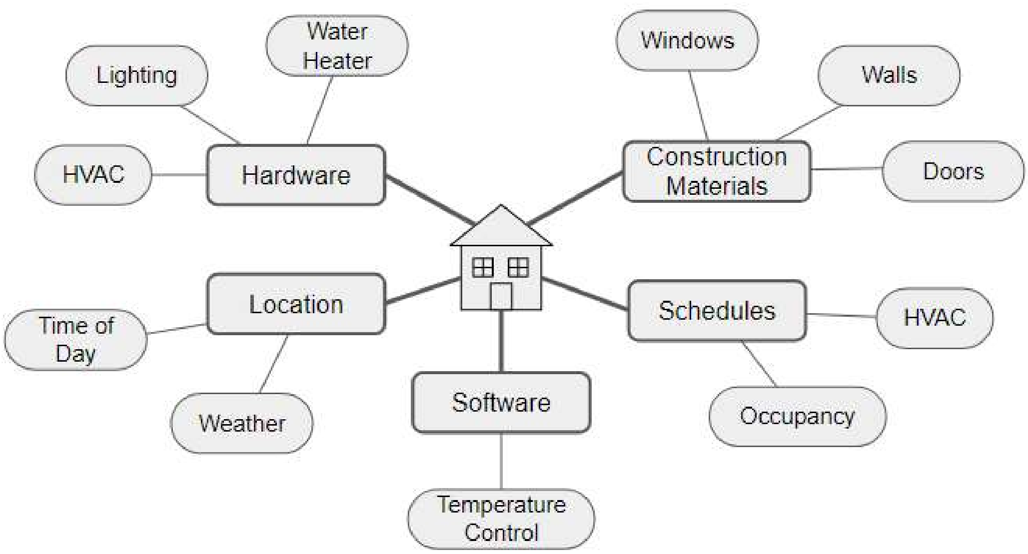


Figure 1.5: Many variables are involved in building energy consumption, and are often interconnected.

Some CPS products have made attempts to penetrate the residential market. Google's Nest Thermostat [20] is advertised as a smart thermostat which intelligently controls home HVAC. Though reasonably priced, the learning algorithms for HVAC operation rely on a certain level of remote controllability to properly function. Some complaints have been made about this cumbersome and manual process, and others claim they do not benefit from the advertised energy savings.

The market for smart homes is still in the early stages [19]. Other smart home approaches have been discussed [21-25], but have not been implemented. This thesis explores two CPS approaches for integration and home energy savings. First, a sensor is developed to provide residential hot water usage information automatically for existing gas or electric storage water heaters. Capitalizing on unique temperature traits of water heater inlet and outlet pipes, the temperature-based sensor can non-invasively detect internal flow. The market for such a sensor is analyzed, and determined to be appropriate for home CPS energy savings. Next, a detailed description is discussed for correlating water heater temperature change rate to internal water flow rate, as well as description for automated water draw detection methodology. Lastly, 33% less water heater energy consumption is achieved through a simple experiment based on automated sensor results.

The second approach for residential CPS improvement involves the integration of a building simulation software into a test platform to assist in CPS deployment. Existing CPS simulation platforms are evaluated for simple and low-cost CPS development. Next, an open-source building simulation software is chosen for its complex energy calculations necessary to evaluate building CPS performance and reliability. Integrating building simulations in a CPS platform allows for effective testing and validation of developing home CPS. To validate the described approach, a modeled HVAC system is simulated in the platform, and controlled by an external thermostat component. Showing no effect on building simulation results, the use of this open-source process can lead to cheaper, more effective home CPS.

1.2 Water Heater Flow Sensing

Water heating is a very integral part of our daily lives. We use hot water to shower, clean our clothes, and wash our dishes. Though often kept out of sight, the DOE found water heaters account for 18% of total home energy consumption [26], shown in Figure 1.6.

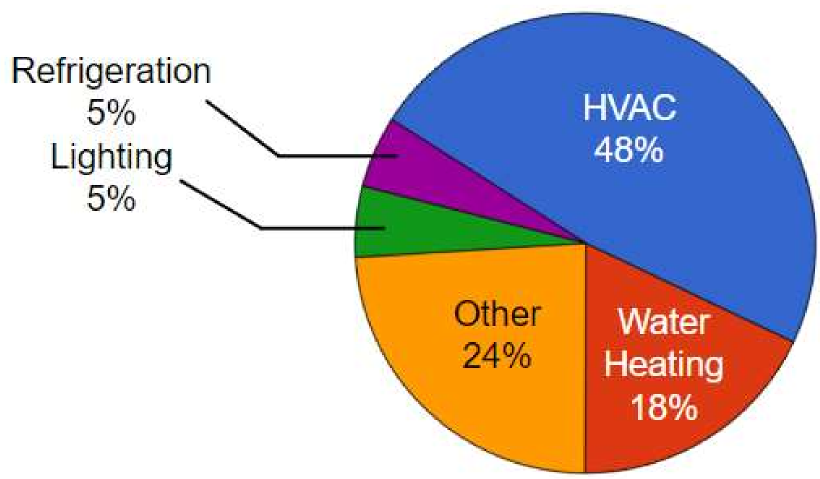


Figure 1.6: Distribution of residential energy consumption, as stated by the US Department of Energy. Water heating and HVAC contribute significantly [26].

Water heating is accomplished either by storage water heaters, tankless on-demand water heaters, heat pump water heaters, or solar water heaters [27]. Storage water heaters are most common but suffer from heat loss. Many people appreciate the efficiency of tankless water heaters, but may be required to purchase two due to limited flow rate. Heat pump water heaters are much more efficient than storage water heaters, but affect heating load on a home due to cold air exhaust. Choice of these heater varieties can depend on income, available space, local climate, and desire for energy savings. Table 1 below compares benefits and drawbacks of each type.

Table 1: Types of residential water heaters. Storage water heaters make up the majority of the market [27].

Type	Operation Basis	Pros	Cons	Amount in Use [%]
Storage	Internal heating elements from gas or electric sources. Constantly retains set-point.	Lower cost	Constant standby heat loss	97%
Tankless	Heats water on demand with high power heating elements	8-34% more efficient thank storage water heaters	Limited flow rate	3%
Heat Pump	Draws heat from air to heat stored water.	2-3 times more efficient than storage water heaters	Location dependent performance	<1%
Solar	Uses solar energy to heat stored water.	50% more efficient than gas or electric water heaters	Climate and weather dependent performance	<1%

The most common home water heaters by far are storage gas or electric water heaters, controlling about 97% of the market [28]. These water heaters rely on regular heating to constantly maintain available hot water. Though storage heaters are usually set to maintain 50°C-55°C, select appliances require different output water temperatures, shown in Table 2. Stored hot water causes two forms of heat loss. First, the heated storage tank suffers from standby heat loss throughout the day due to tank and environment temperature difference. Second, the excessively hot water is often mixed with colder water for desired output temperature causing losses through entropy generation. For an individual home, avoiding excessive and unnecessary heating can reduce these forms of heat loss and reduce consumed energy.

Table 2: Select home appliances require hot water at different (approximate) temperatures. Storage water heaters maintain temperatures to account for high temperature demand [29].

Appliance	Water Temperature Needed [°C]	Average Family Weekly Usage
Clothes Washer	55	7
Dish Washer	50	2-3
Shower	40	7-14
Sinks (hot water)	40	~50

Recent advancements in storage water heater research has led to more efficient products [30, 31]. Improved insulation, for example, can reduce heat losses by up to 45% [32]. Lifecycles are long (about 10 years), preventing home occupants from upgrading current systems. Making matters worse, newer storage systems are expensive, ranging between $500 and several thousand dollars, making a slow return on investment.

A more realistic and immediate impact can be achieved with an inexpensive add-on CPS device for existing water heaters. For such a device to effectively reduce the two forms of gas and electric water heater waste, existing unit activation should be controlled. Both water flow rate and draw regularity information can act as the basis for accurate water heater control. Draw regularity knowledge provides an activation schedule to avoid unnecessary heating and standby heat loss, while flow rate knowledge can provide necessary information to minimize excessive heating. Extraction of these two pieces of information is accomplished through flow rate sensing.

The current market of flow sensors vary between categories of either affordable and invasive (complex installation) [33-35], or expensive and non-invasive [36-42], compared in Table 3. Invasive sensors are placed in a passage of fluid flow and directly measure the flow rate. For example, Munir et al. used an in-line propeller for microcontroller flow detection [34]. Though accurate, the installation of a flow sensor to an existing piping network of a water heater often involves water drainage as well as replacement of tubes and fittings, which can be labor intensive and costly.

Table 3: Comparison of existing pipe flow sensors for water heater applications.

Type	Operation Basis	Pros	Cons	References
In Line	Directly measure fluid flow in piping network	Low cost, high accuracy	Requires installation and potential water heater drainage	[33-35]
Ultrasonic Sensor	Uses ultrasonic sound to get flow rate	Non-invasive, high accuracy	High costs	[40] [41]
Pressure Sensor	Detects pressure fluctuations upon water draw in piping network	Semi-invasive	Not water heater specific.	

Non-invasive sensors have more simple installation. Placed on the outer perimeter of a pipe, Tawackolina et al. evaluated an ultrasonic flow sensor for heat dependent accuracy [40], and Tasaka et al. investigated ultrasonic Doppler velocity profilers and their practical applications [41]. However, the technology used is often expensive (an order of magnitude of $1000 [42] for ultrasonic sensors) creating an impractical and undesirable reality of an acceptable return on investment. Additionally, these non-invasive methods typically have higher accuracy with more particles or bubbles in the fluid which do not regularly occur in simple water flow.

A unique characteristic of a water heater is that inlet and outlet pipes experience large temperature changes upon hot water draw events, as cold water replaces the drawn hot water. Though Nguyen notes principles for how heat transfer can affect a thermal micromachined flow sensor [35], the small size is not directly applicable for water heater use. For the larger application, the principle of energy conservation can be used to translate the pipe temperature change information into the flow rate of water moving through the water heater. As such, an economically viable compact package can measure flow rate using temperature sensors, without involving an invasive installation process. To the best of the author's knowledge, the proposed temperature-based approach has not been attempted, making this a unique concept to be investigated and explored.

Development of an affordable and easily installed flow rate detector can create a more widespread accessibility of usage pattern information for CPS control of currently installed gas or electric home water heaters. The second chapter of this thesis creates a foundation by presenting flow rate detection based on temperature changes of water heater pipes. Automated water draw detecting algorithms are developed for streamlined flow calculation processes. Incorporating relative flow rate knowledge and occupant hot water demand, a CPS can control water heating for increased energy saving capabilities.

1.3 Building Simulation Integrated within CPS Testing Environment

Effective home CPS are not readily available to consumers at a reasonable price. Time and resources required to harness building diversity from Figure 1.5 increase costs of CPS and de-incentivizes others from developing more CPS. A simple, low-cost CPS development method can allow for more robust designs. To accomplish such a feat, building simulations can be used to replace traditional processes of physical and real-time building data collection. Further, incorporating a building simulation with a developing CPS requires co-simulation friendly environments.

Currently there are several simulation environments available for CPS development. Poudel et al. developed a CPS testbed which can conduct electrical power control experiments in real-time [43]. Though their testbed integrates MATLAB/Simulink models [44], simulation is only limited to the power grid domain, and it cannot be applied directly to building CPS. Garraghan et al. proposed a service-oriented approach called SEED (simulation environment distributor) which is designed to simulate large scale CPS [45]. However, SEED is not a very user-friendly approach to CPS simulation, requiring knowledge of programming and virtual networks. Magnusson et al. developed a full simulation system call Simics [46], allowing a framework for firmware co-simulation. Their commercial product can connect many diverse devices, but are tuned more to software based systems, as opposed to physical components like those found in buildings.

Institute of Electrical and Electronics Engineers (IEEE) suggests a co-simulation standard known as the High Level Architecture (HLA) [47]. This set of rules provides a structure allowing simulations to describe their individual application for interoperability. The HLA describes individual simulation entities as *federates* and a collection of interconnected federates as a *federation*. A federation complying with the HLA allows data and information to be made available between all federates. Since the HLA is only a protocol, or standard, for data exchange, it requires a software to facilitate actual federation data transfer called Run-Time Infrastructure (RTI). RTI provides synchronous data exchange while accurately controlling time step progression between federates. Shown in Figure 1.7, implementation of the HLA/RTI can vastly improve interoperability and co-simulation between federates.

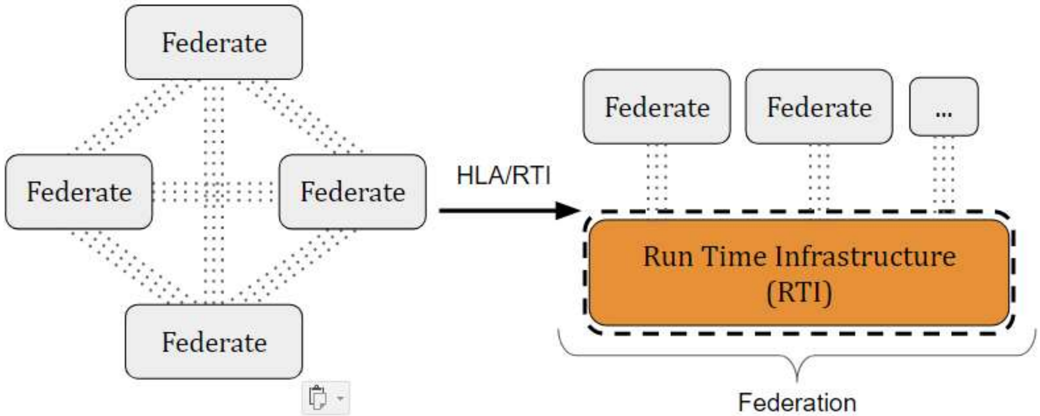


Figure 1.7: Rather having individual connectivity between simulation entities (federates), the RTI environment can more simply connect federates for co-simulations.

The National Institute of Standards and Technology (NIST) developed an open-source CPS experiment and testing environment called Universal CPS Environment for Federation (UCEF) [48] which utilizes HLA. Its graphical user interface is designed to make co-simulations and experiments for CPS product simple and available. UCEF can integrate various simulation entities (federates) sourced from different development environments, which has traditionally been challenging to accomplish. UCEF leverages the IEEE's HLA standard for its communication protocol, implemented by the Portico RTI [49], to achieve logical time progression and data transfer within a federation. So far, UCEF is still under development phase and only supports Java federates, but yields high potential for low-cost CPS experimentation and validation processes.

An open-source building simulation tool called EnergyPlus [50] can complement UCEF functionality by exchanging building simulation information. EnergyPlus is a widely used tool created by the DOE, and can model building energy consumption by performing complex calculations at sub-hourly time steps. The software calculation capabilities incorporate building parameters from Figure 1.5 such as user activity, HVAC systems, building composition, and more. Known for its robust capabilities, EnergyPlus building simulations can replicate building information typically measured for CPS development. Replacing traditionally collected physical data with a simulated model in EnergyPlus, UCEF co-simulation can accelerate home CPS development.

The third chapter of this thesis integrates EnergyPlus into UCEF for CPS co-simulation. An EnergyPlus model will communicate building information to the RTI using a UCEF Java federate. To verify co-simulation capability, an HVAC set-point algorithm implemented in another Java federate will receive environment temperature from EnergyPlus and return HVAC set-points to EnergyPlus. Intelligent set-point control of an HVAC system can significantly reduce energy consumption in a residential building, providing a good use case for the developed platform. Further, other simulators integrated with UCEF can expand HVAC controllability to include pre-heating or pre-cooling a collection of homes to reduce excessive power draw during peak demand [24]. Enabling UCEF co-simulation with EnergyPlus can allow for an established process to develop low-cost CPS for reduced residential energy consumption.

Chapter 2: Water Heater Flow Sensing

Water heaters account for approximately 18% of residential home energy consumption. Gas and electric water heaters are most common, but are often overheated, causing standby heat loss and potential entropy generation in attempt to reach desired output temperature. Sufficient inlet and outlet pipe temperature changes are experienced by these water heaters upon hot water draw events. Energy conservation is evaluated for water heater inlet and outlet pipes to correlate temperature change rate to internal water flow rate. Calculated flow information provides a CPS water heater controller with sufficient information to control water heater activation for reduced energy consumption. Differentiating between high and low flow rates can assist control for water heater activation. This chapter explores a flow rate sensor using temperature measurements as well as methodologies for automated water draw detection for CPS water heater control.

2.1 Flow Rate Approach

Energy conservation evaluation around a water heater pipe surface can convert temperature change to flow rate. Calculating relative flow rate can qualitatively differentiate high and low draws. A water heater pipe is explored and evaluated for heat transfer, where axial conduction along a pipe is assumed negligible for calculation simplicity.

Exploring temperature patterns of a water heater, it was found that with no flow, the inlet and outlet pipes will gain energy and rise in temperature due to close proximity of the heated and stored water. During a water draw event, when an occupant draws hot water, water flowing through the cold inlet and hot outlet pipes will respectively cool down or heat up the pipe temperature. The rate of temperature change depends on the flow intensity and thermal insulation. For calculation purposes, the cold inlet pipe will exclusively be analyzed. The energy balance equation for this scenario is given in Eq (1),

$$\boldsymbol{M\frac{dT}{dt} = -U(T - T_w) - U_\infty(T - T_\infty)} \tag{1}$$

where T is the measured pipe surface temperature, T_∞ is the measured ambient air temperature, and t is time. T_w is the internal water temperature of the cold inlet pipe assumed to equal T for no draw events, and assumed constant at 15°C for draw events. M represents the thermal mass, defined as the multiplication of material density, ρ, specific heat, C_P, and cross-sectional area, A, divided by axial unit length, *dx*. These properties were taken for a ¾ inch copper pipe at room temperature (300K). U and U_∞ represent the overall heat transfer coefficients for the internal and external thermal resistances [51], respectively, shown in Figure 2.1.

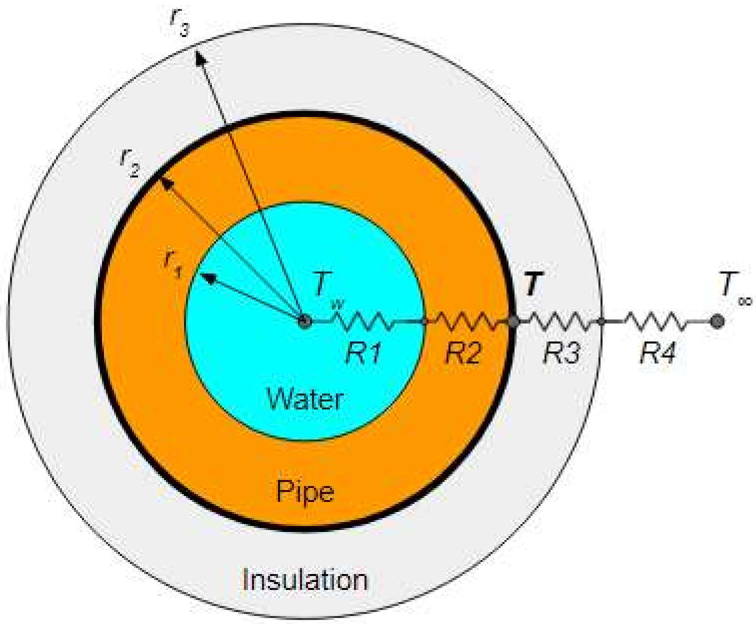


Figure 2.1: Representation of the thermal resistances, R1, R2, R3, and R4, involved within a water heater pipe heat transfer. The bolded outer surface of the pipe is where temperature, T, is to be measured, effectively splitting the resistances between internal resistance (R1 & R2) and external resistance (R3 & R4).

For water draw events, *R2* can be assumed negligible compared to *R1*. With the order of magnitudes of convection heat transfer coefficient, h_w, at 100 W/m^2·K, pipe thermal conductivity, k_{pipe}, at 100 W/m·K (assuming copper piping), and pipe radii, r_1 and r_2 from Figure 2.1, approximately being 20 mm and 25 mm respectively, Eq (2) validates the assumption through scale analysis.

$$\frac{\ln(r_2/r_1)}{k_{pipe}} = R2 \ll R1 = \frac{1}{h_w 2\pi r_1} \qquad (2)$$

Therefore, the internal overall heat transfer coefficient, U, can be simplified into Eq (3).

$$U = 1/(R1 + R2) \sim 1/R1 = h_w 2\pi r_1 \qquad (3)$$

When water is not being drawn, standby heat loss dominates energy transfer. In this no-draw natural cooling case, the first term on the right hand side of Eq (1) is negligible as the internal water, T_w, is assumed to be equivalent to the measured pipe surface temperature, T. Integrating Eq (1) yields an expression of U_∞ for this non-draw case:

$$\boldsymbol{U_\infty = -\frac{M(T_2 - T_1)}{\int_{t_1}^{t_2}(T - T_\infty)dt}} \tag{4}$$

where the denominator is numerically determined using the trapezoidal rule on each discrete measured data point over the course of the natural cooling time period, t_1 to t_2. Integration methods are chosen over derivative methods in attempt to mitigate errors stemming from small variations in temperature measurements. Once determined, U_∞ is assumed to be constant for all water heater events.

During each detected hot water draw event, the last unknown, U, in Eq (1) is solved by integrating over the discrete water draw data set, between t_1 and t_2. Separating variables, knowing measured pipe temperature, T, is the only variable changing over time, yields Eq (5).

$$\boldsymbol{T_2 - T_1 = \left(\frac{U}{M} + \frac{U_\infty}{M}\right)\int_{t_1}^{t_2} T\,dt + \left(\frac{U}{M}T_w + \frac{U_\infty}{M}T_\infty\right)(t_2 - t_1)} \tag{5}$$

By measuring temperature over a water draw period, U can be determined from Eq (5). The convection heat transfer coefficient, h_w, can then be derived from U, using

Eq (3), and is related to the flow rate. The Dittus-Boelter equation for cooling [52], represented in Eq (6), relates h_w to the Reynolds number.

$$\boldsymbol{Nu_w = \frac{h_w D}{k_w} = 0.023\, Re^{4/5}\, Pr^{0.3}} \tag{6}$$

where Nu_w is the Nusselt number, D is the internal pipe diameter, k_w is the thermal conductivity of the water, Pr is the Prandtl number of the water, and Re is the Reynolds number of the flow which contains the desired flow rate term, $\dot{m}$, in Eq (7). The variable μ represents water viscosity.

$$\boldsymbol{\dot{m} = Re \frac{\pi D \mu}{4}} \tag{7}$$

After solving for U in Eq (5), the desired value of flow rate for a water draw event is calculated using equations (4), (6), and (7). This process allows a water draw temperature data set to be related to flow rate through a storage water heater cold inlet pipe.

2.2 Automated Detection Approach

Determining appropriate water draw data sets is automated by evaluating measured temperature slopes. The detection method performed assumes isolated draw events with sufficient reheating time (about 25 minutes, determined imperially) after cooling due to water draw. This post processing event detection utilizes the unique water heater trait of an assumed heated cold inlet pipe (approximately 45°C) facing rapid cooling from forced internal convection of cold inlet water (approximately 15°C) upon water draw.

Data extraction for such draw events is initiated when measured inlet pipe temperature data suddenly decreases at a rate of 1°C per second or greater. This starting criterion was determined imperially. To achieve automation, extraction persists until a point of increasing slope is detected. This ending criterion represents the idea that the flow has stopped, and the internal heat from the water tank has propagated back up the pipe through free convection (assuming not all the hot water has been replaced during the draw event). Each set of extracted draw event data is then processed using flow rate calculations mentioned previously in 2.1 .

After draw events are detected, remaining no-draw events are split into two categories of natural heating and natural cooling. Natural heating occurs after a water draw, where the decreased pipe temperature naturally recovers to a heated state based on internal tank temperature (increasing slope). Then, natural cooling occurs as the heated pipes after natural heating respond to the tank's standby heat loss (decreasing slope). These no-draw events are much less extreme compared to draw events and occur over a longer time period (typically greater than 3 minutes). Spline fitting of every 10 data points (or less if the data set between draw events was sparse) was used to differentiate increasing and decreasing slope to avoid temperature sensor precision error. The 10-point scope was determined imperially to avoid notable error. Discussed event detection code is found in Appendix A.

2.3 Circuit Design for Data Collection and Control

Pipe temperature data collection was accomplished through implementation of a deployable package consisting of temperature sensors, a wireless microprocessor, and water heater activation control. This package was made available for both gas and electric storage water heaters, but required development of two circuit board designs for respective heater control. Both circuit boards contain three temperature sensors connected to a wireless microprocessor. The electric board design (shown in Figure 2.2) intercepts the electricity going towards electric water heaters, which then powers the board. Raw circuit designs can be found in Appendix B along with a parts list. To control heater activation, it utilizes relay switches controlled by the microprocessor to connect or disconnect intercepted electricity

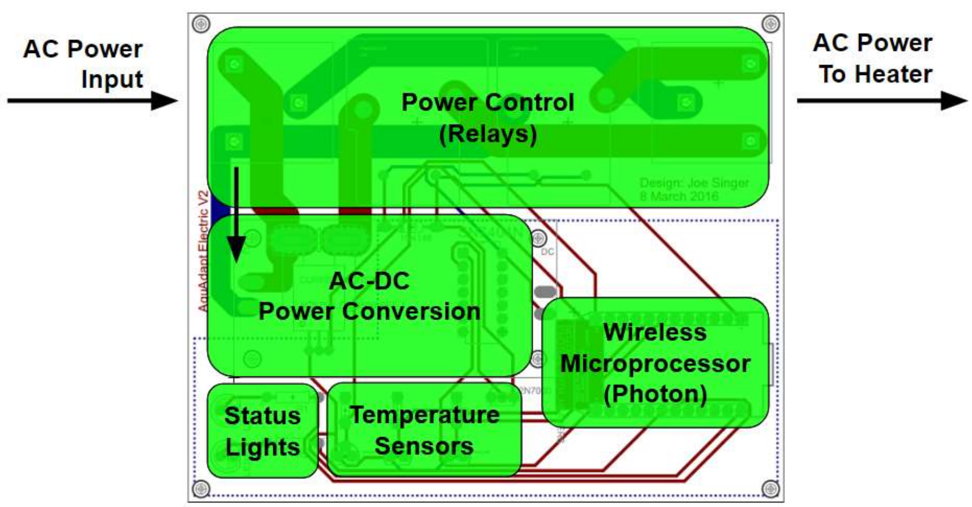


Figure 2.2: Circuit design of storage water heater controller for electric powered systems.

The gas board design exploits small voltage control which opens and closes existing gas solenoid valves fed to the tank. Powered by a wall outlet, the microprocessor intercepts voltage readings of existing controllers, and sends the activating signal accordingly depending on activation schedules. The gas board is shown below in Figure 2.3.

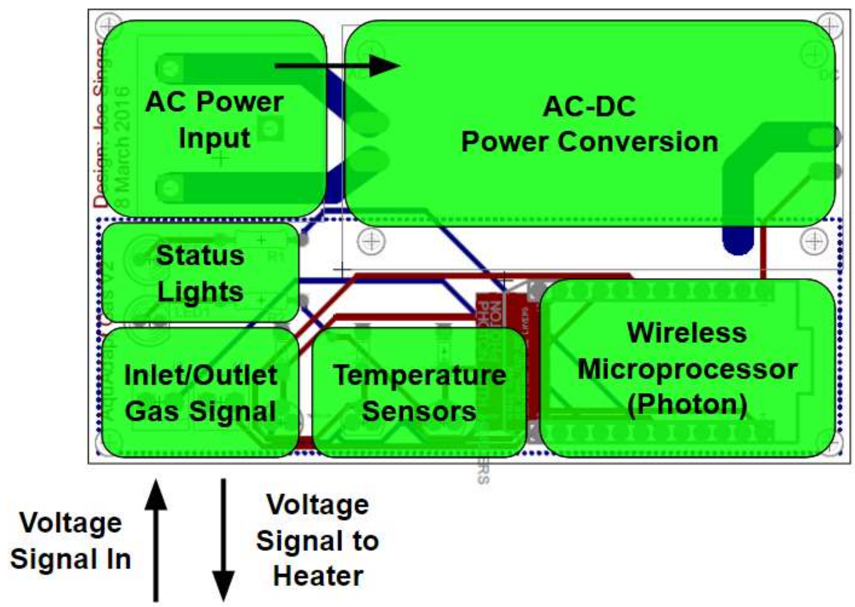


Figure 2.3: Circuit design of storage water heater controller for gas powered systems.

For each design, three TMP36 temperature sensors were used to monitor the temperature change of the cold inlet and hot outlet pipes as well as the ambient air. These sensors have an accuracy of ±2°C, precision of ±0.5°C, and a temperature specification of -40°C to 125°C [53]. Application of TMP36 sensors was suitable for experimentation due to a wide temperature range, high precision for quantitative measurements, and low cost for practical deployment.

To easily and non-invasively measure pipe temperatures, two of the TMP36 sensors were incorporated into 3D printed clamps. These clamps were sized for water heater pipes and placed on the cold inlet and hot outlet pipes approximately 6 inches from the water heater. This location was determined to both reduce the effect of water tank temperature during its heat generation process, and to ensure heated pipes after prolonged no-draw events. Sensor measurements were taken by a Particle Photon [54] microprocessor, which sent measured data over a wireless internet connection to an off-site computer for the more intensive data processing. Readings were taken, sent, and stored at a time interval of 5 seconds throughout the day. Figure 2.4 shows the deployed electric circuit board with two clamps for water heater pipes.

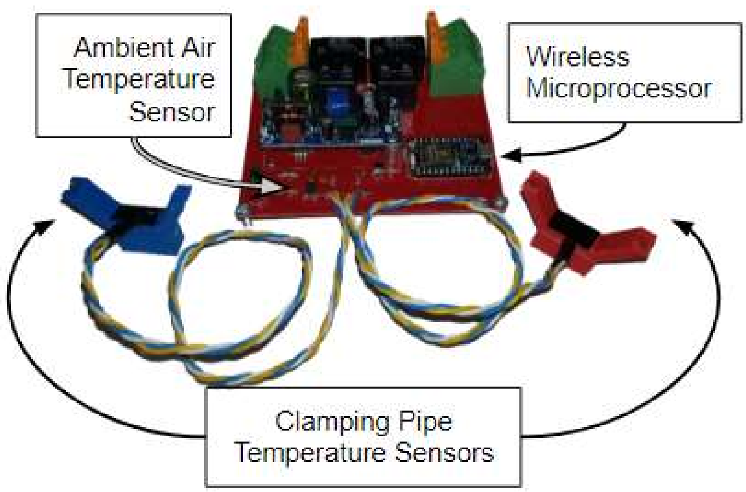


Figure 2.4: Deployable package in the form of a printed circuit board purposed to collect and wirelessly send temperature data. Lower right and left clamps are used to enhance thermal contact of temperature sensors.

2.4 Experimental Setup

An experiment was developed to validate automated event detection and flow rate analysis. The test bed used for analysis consisted of a 10 gallon electric water heater connected to a sink used to draw hot water from the water heater, shown in Figure 2.5. The described deployable package measured water heater pipe temperature 6 inches away from the tank. Collected data was sent to an off-site computer for post processing analysis.

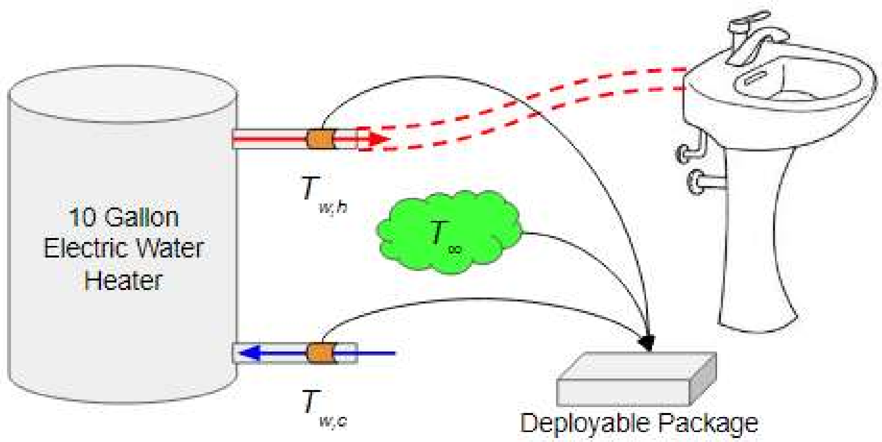


Figure 2.5: Experimental setup measuring cold inlet and hot outlet pipe temperature and ambient air temperature. A controlled amount of hot water was drawn at a regular schedule through a sink connected to tank. The deployable package wirelessly sent the measured data to an off-site server.

To simulate home usage, hot water was periodically drawn. A beaker was used to measure the actual amount of water drawn over the duration of the draw events. Volume of water collected over the duration of water draw produced actual average flow rate of water, acting as a ground truth for experimental calculations. Water draws were performed and recorded over the course of several days. There was at least 25 min between each draw to allow for the stored hot water to reheat the pipe and internal water. Draw durations ranged from 5 to 90 seconds and draw intensities ranged from 3 to 13 L/min due to limitations of the faucet.

2.5 Results & Discussion

Discrete temperature data was collected using the deployable package. Post-processing event detection methodology was able to identify 100% of isolated water heater events within the experimental flow range of 3 to 13 L/min. Figure 2.6 shows resulting event detection classification for two water draw events. Sufficient time allocated between all draw events, allowing the cold inlet pipe to reheat from a no-draw natural heating event, may cause limitations to the evaluation methods.

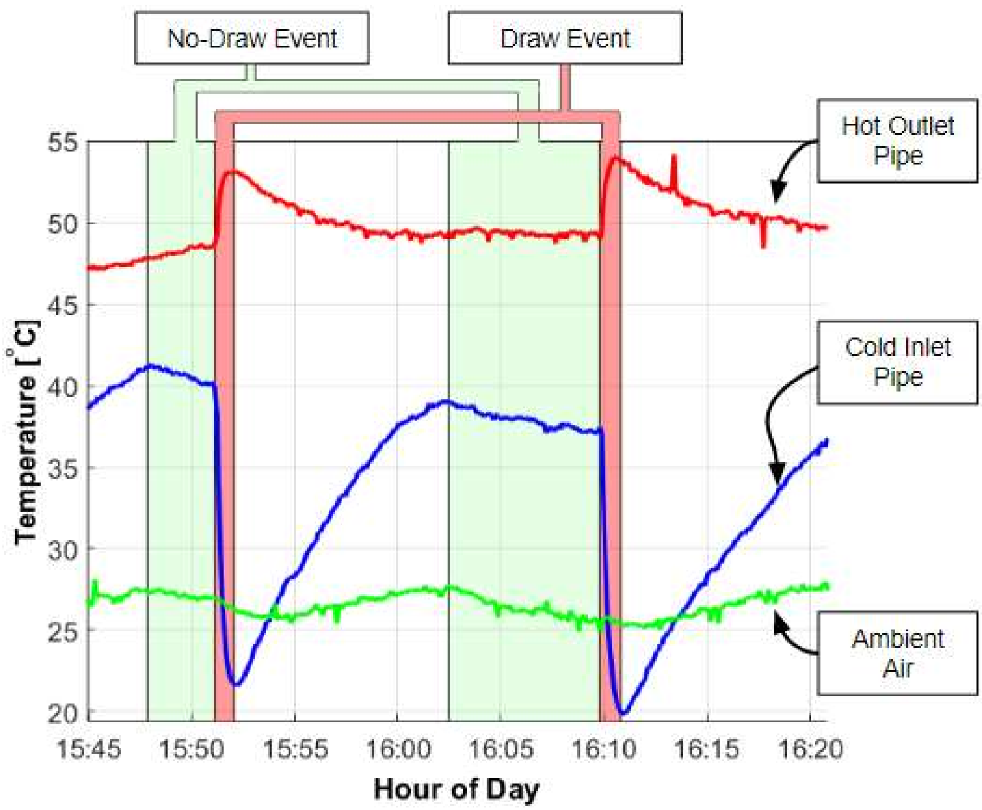


Figure 2.6: Classification of 3 different water heater events. Cold inlet and hot outlet pipe temperatures respectively drop and rise as water is being drawn from the water heater (water draws shown in red shading). Vice versa, the same pipe temperatures respectively raise and drop as natural heating events occur (white shading). Non-labeled events (green shading) represent times where the water heater and pipes are assumed to be naturally cooling.

A single draw event is shown in Figure 2.7, showing temperature data was accurately extracted for a water draw event as intended. All detected start times matched the actual start times within a resolution of ±5 seconds (the time interval of data collection). Draw event data sets such as the one shown below are automatically filtered through the energy conservation equations for flow rate correlations.

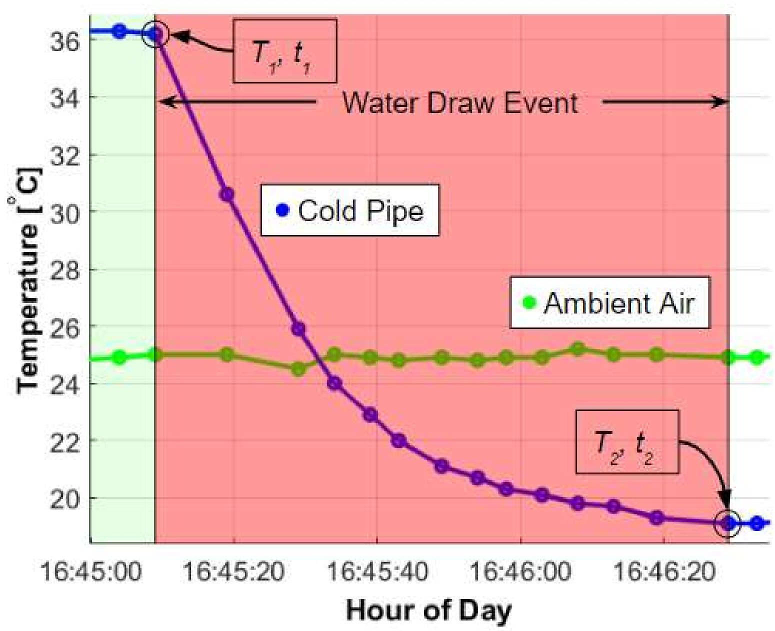


Figure 2.7: Demonstration of extracted water draw event data used to calculate flow rate, where cold inlet pipe temperature rapidly decreases. Actual draw duration is 60 seconds (from $\boldsymbol{t_1}$ to $\boldsymbol{t_2}$) and draw intensity is 8 L/min.

For flow intensity correlations, an analysis was first performed to determine how draw duration affects accuracy of the flow calculations. Multiple draw event durations at identical 12 L/min draw intensities were compiled. Comparing draw durations, Figure 2.8 presents greater consistency after about 40 seconds of water draw. This time dependent result is due to integration process of the flow derivation. Integration can be sensitive to ending time, t_2, and actual draw duration will not be accurately reflected in the ending time determined in event detection analysis. Natural convection of cold water residually cools the cold inlet pipe after the water has stopped flowing internally causing this duration uncertainty. As water draw durations increase, these residual effects will be less influential on flow calculations.

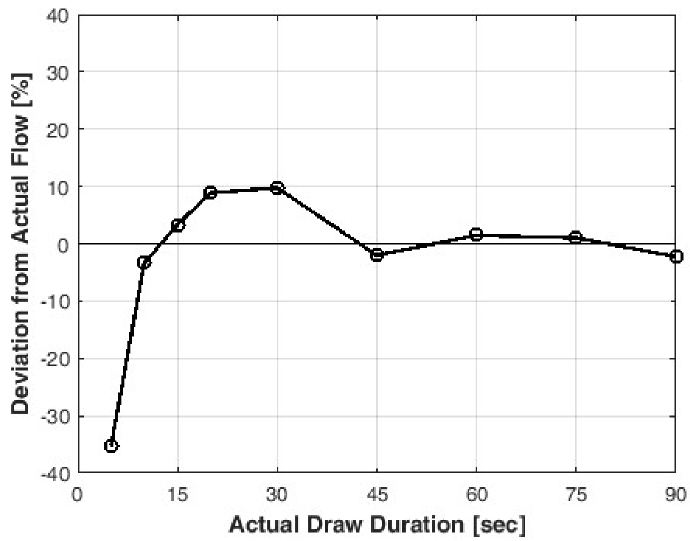


Figure 2.8: Water was drawn at constant flow rates for various durations. A longer draw time (greater than about 40 seconds) will result in less flow rate deviation.

Short duration of water draws (<40 seconds) will not significantly contribute to actual hot water usage for two reasons. First, it takes time for actual drawn hot water to travel from the tank to the destination (assuming water in intermediary pipes start as cold). If hot water does not exit at the draw location, it is not utilized and need not be considered for desired hot water flow detection. Second, most significant energy usage in a water heater is dominated by larger draw events such as a shower or washing machine, making shorter draws negligible for energy savings.

Another analysis was performed to quantifiably differentiate flow rate intensities based on pipe temperature change, shown in Figure 2.9. Each draw lasted 60 seconds to eliminate short draw duration error as previously discussed. Based on conventional statistical analysis, a correlation factor was found to be 0.67 based on the number of samples. Results indicate the chance of having better linear correlation is 2%. As such, correlation between the actual and calculated flow rate is significant at a 95% confidence level. Discrepancies in direct flow accuracy are presumably due to assumptions made throughout the derivation process. High contributing assumptions include, but not limited to:

1. Negligible weather conditions
2. Constant value T_∞ during draw events
3. Overall constant values of T_w, M, and U_∞

Weather and exterior conditions can alter temperature profiles over the course of a day. Incoming water, T_w, can be affected by these conditions, causing propagated error during calculations. M and U_∞ are calculated assuming material properties at 300K. Depending on temperature, these properties are also subjected to change.

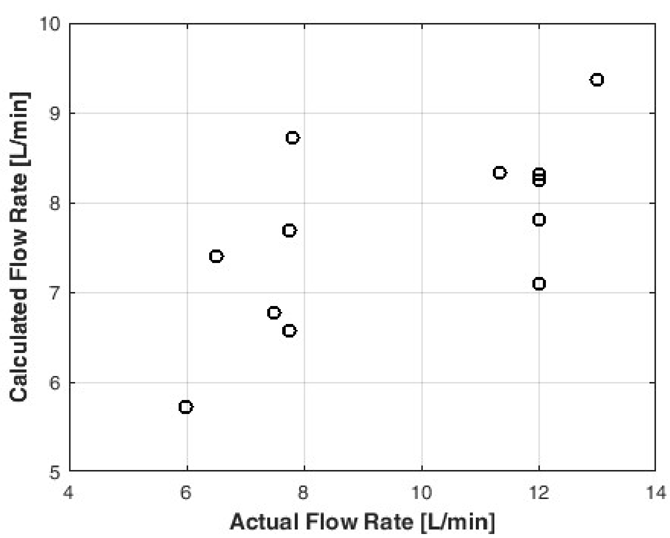


Figure 2.9: Comparison of calculated and actual flow rates for various draw intensities. Each draw was one minute in duration.

This section discussed a low cost, non-invasive, deployable package which collects and wirelessly sends temperature measurements for CPS related analysis. Automated algorithms were developed to detect when an occupant used hot water. Energy conservation calculations were then used with draw event data to relate pipe temperature change to a relative flow rate used by the occupant. Based on these hot water usage patterns, water heater activation was controlled to achieve 33% energy savings. Such CPS can be easily incorporated into existing homes to help reduce home energy consumption.

Chapter 3: EnergyPlus Integration into UCEF

Building CPS development requires interdisciplinary knowledge to accurately relate inherently complex and interconnected physical building attributes. For this reason, current CPS often rely on occupant remote controllability as opposed to automated control. The EnergyPlus building simulation software can consider complex physical building interactions, replacing physical real-time measurements in CPS testing and validation processes. NIST's UCEF is a platform allowing data transfer between simulation tools. The open-source EnergyPlus building calculations can be used to co-simulate in a UCEF environment for simple CPS development. An existing EnergyPlus interface is used to exchange data between the HLA/RTI environment.

3.1 Approach

EnergyPlus currently has an existing co-simulation interface through the Functional Mock-up Interface (FMI) standard created by Modelisar [55]. The standard accomplishes interoperability by connecting simulation platforms to an external model by use of a zip file (with extension *.*fmu*) known as a Functional Mock-up Unit (FMU). The zip file contains three elements: an Extensible Markup Language (XML) file, compiled C code binaries, and optional documentation for data exchange. The XML file establishes interfacing data, the C code manages data exchange, and the documentation can define and specify operation.

FMI and HLA are not currently compatible. The two standards have different notions of time management, and UCEF does not support data exchange using FMI. To bridge the two standards, we create an FMU with capabilities for bi-directional communication between EnergyPlus and a UCEF Java federate. The Java federate will be customized to wrap EnergyPlus for data exchange to an RTI federation. This data communication, represented in Figure 3.1, is done through TCP/IP socket communication between our FMU and Java federate.

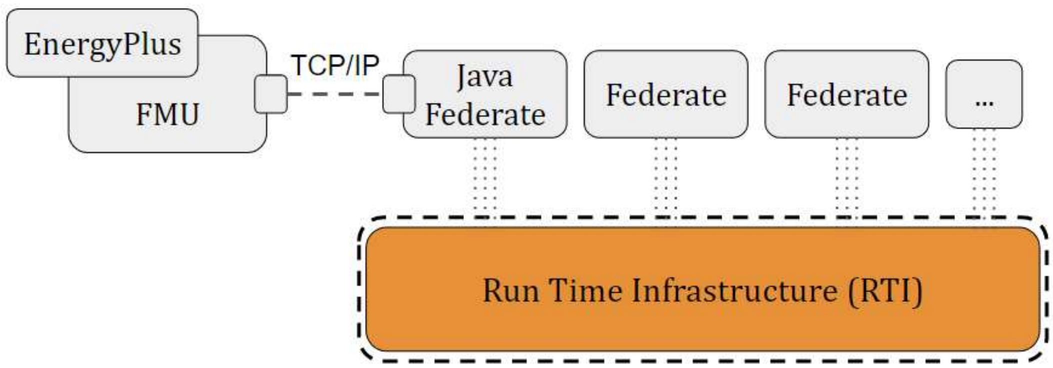


Figure 3.1: EnergyPlus has capability to interface with an FMU. Using TCP/IP socket communication inside a simple FMU allows for connectivity to a UCEF Java federate for HLA/RTI data exchange.

Connecting EnergyPlus to an FMU involves specific modification of an EnergyPlus input data file (IDF). An IDF defines parameters to perform building energy simulations, such as building materials, components, and equipment. Using an IDF component called *FunctionalMockupUnitImport*, co-simulation is linked between EnergyPlus and the FMU. This component initializes the FMI master and slave architecture where slaves are coordinated and executed by the master program. EnergyPlus acts as the master in this configuration, which initializes the FMU as an executable slave instance. EnergyPlus version 8.7 was used.

Upon simulation start, EnergyPlus locates and unpacks the linked FMU zip file to begin processes represented in Figure 3.2. Execution of the FMU's customized C binaries is controlled by EnergyPlus to run select FMI functions [56] that have been modified and implemented to exchange data with the HLA RTI. EnergyPlus first calls the *fmiInstantiateSlave* function to parse through the unpacked XML file, properly allocating memory for the interface data. Next, the *fmiInitializeSlave* function uses TCP/IP sockets to establish connection to a server hosted in the UCEF Java federate. After TCP/IP connection is verified, EnergyPlus time step calculations begin.

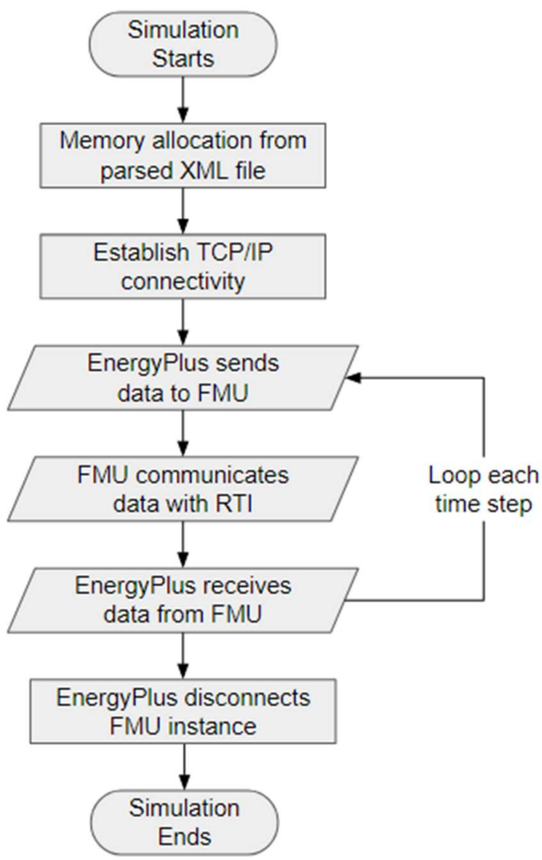


Figure 3.2: EnergyPlus as a master program for FMI calls select functions throughout simulation to perform specific tasks. At each time step, three tasks are called to transfer EnergyPlus data.

At each time step, EnergyPlus sends data to the FMU as a *real* data type using the function *fmiSetReal*. The FMU will then utilize socket connection in the *fmiDoStep* function to send the EnergyPlus data (as a concatenated string) to the Java federate. The format of this string is standardized and represented as follows:

HEADER\r\nTIMESTAMP\r\nNAME\r\nVALUE\r\n….
NAME\r\nVALUE\r\n\r\n

The "HEADER" defines handling procedures of the string. Data sent from FMU to the Java federate will either contain the header "UPDATE" or "TERMINATE". An "UPDATE" header is used at each EnergyPlus time step to signify incoming information to the Java federate. A "TERMINATE" header informs the Java federate that EnergyPlus simulation has ended. Data received by the FMU from the Java federate will either contain "SET" or "NOUPDATE" headers. "SET" indicates federation interactions will change EnergyPlus variables for the following time step, and "NOUPDATE" indicates no variables will change. After the header, the "TIMESTAMP" communicates simulation time (in seconds) for logical time management. Next, for each "UPDATE" and "SET" header, "NAME" and "VALUE" respectively represent each variable name and corresponding value of interfacing data defined through the XML file. Each piece of information is separated by a carriage return followed by a line feed ("\r\n"). Two consecutive cartridge returns and line feeds at the end signify the end of the string.

EnergyPlus will remain in the *fmiDoStep* function until the Java federate responds with a concatenated string. After a string is returned, the FMU will parse through the returned string in *fmiDoStep*. The *fmiGetReal* function passes received information back into the EnergyPlus model as a *real* data type. The described data exchange pipeline is represented in Figure 3.3. The master EnergyPlus program will exchange data with the Java federate at each time step. After the final time step, the FMU slave instance is disconnected, and the simulation ends. The described functions written in C is found in Appendix C.

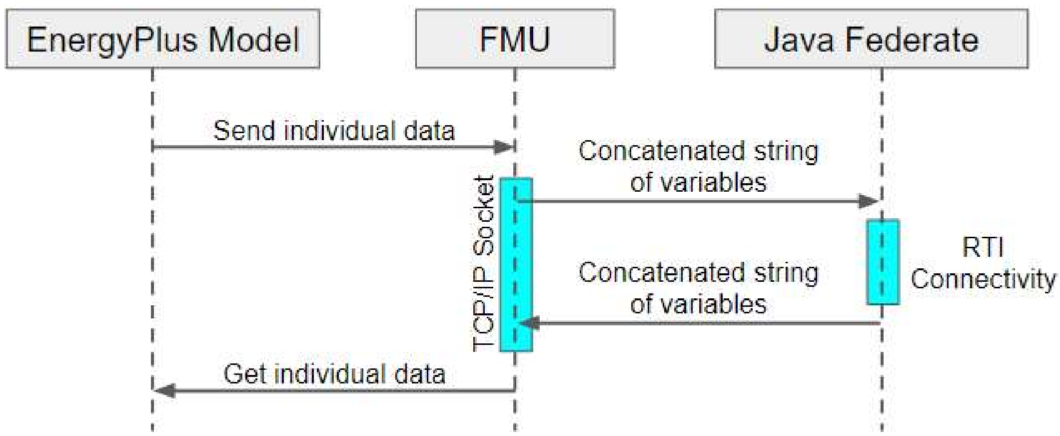


Figure 3.3: UML diagram representing data communication between the master EnergyPlus program and a UCEF Java federate via FMU slave instance.

The Java federate developed in UCEF communicates information between the FMU slave instance and the RTI. This federate begins by hosting a TCP/IP server for the FMU client connection. During simulation, the federate parses each received string from the FMU and passes its information to the RTI federation. The federate then waits for messages from the RTI that should be sent to EnergyPlus. A concatenated string containing the content of these messages, is then returned to the FMU client.

3.2 Experimental Validation

A series of simulations were executed to validate EnergyPlus communication with an HLA RTI federation. A simple three-room house model, shown in Figure 3.4, was created in an EnergyPlus IDF, which can be referenced in Appendix D. The home was located in San Francisco, CA, USA using weather information from June 2017. The home was equipped with a dual set-point HVAC system operating at a temperature range between 21℃ and 23℃. The first simulation executed the simple EnergyPlus model without the implemented FMU external interface. Environmental temperature, zone temperature, and HVAC energy usage information were recorded at each time step. Resulting HVAC energy consumption using these “naive” set-points was intended to resemble non-energy conscious behaviors.

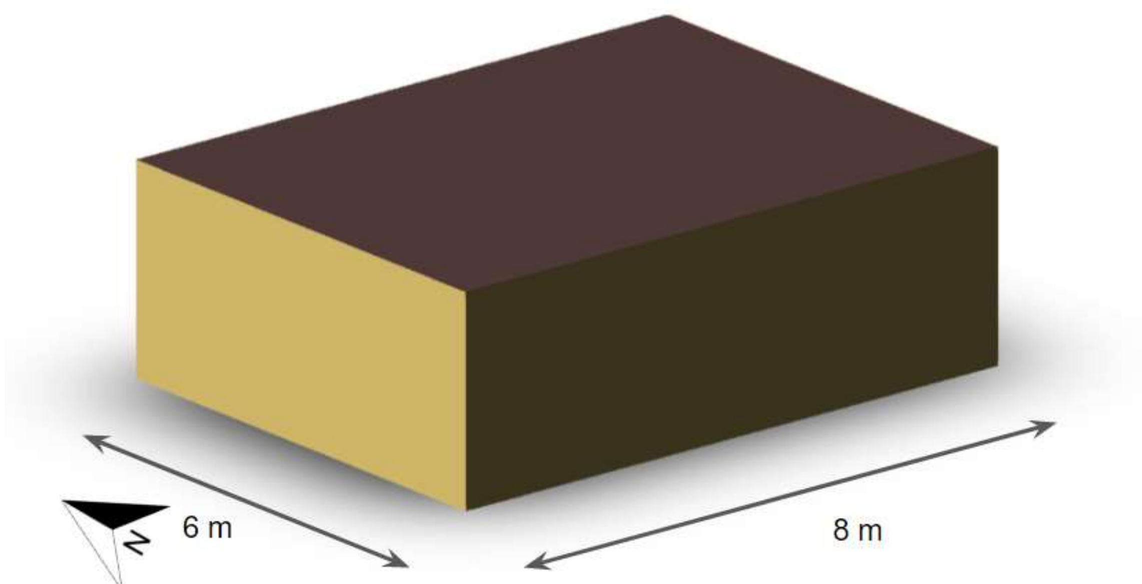


Figure 3.4: A simple EnergyPlus house model consisting of a single room home located in San Francisco, CA.

The second simulation directly ran environment temperature data from simulation one through a thermostat controller algorithm. The algorithm (written in Java) adjusts heating and cooling temperature set-points based on user comfort and environment temperature. Assuming occupant comfort ranges between 20℃ and 25.5℃, HVAC operation dynamically changes to minimize work required to heat or cool a home. EnergyPlus and UCEF were not used in this second simulation. Rather, environment temperature recorded in the first simulation was directly fed through the thermostat controller to return dynamic dual set-points. The results of this second experiment act as a ground truth for EnergyPlus and UCEF connectivity.

The final simulation implemented the FMI external interface with the IDF described in the first simulation. Updated IDF is found in Appendix E. The *FunctionalMockupUnitImport* class enforced data exchange with the developed FMU, linking EnergyPlus to the modified Java federate. The federation was created using UCEF, binding EnergyPlus and the secondary thermostat controller algorithm through RTI. Shown in Figure 3.5, environment temperature from EnergyPlus was sent through RTI to the thermostat controller at each time step. Before time step progression, the controller returned a heating and cooling set-point to the HVAC system in EnergyPlus. HVAC set-points and zone temperature were recorded.

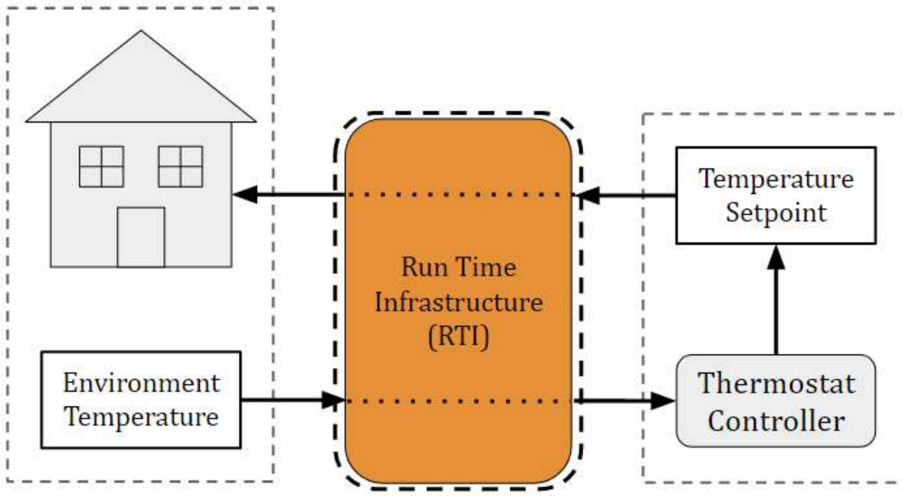


Figure 3.5: Representation of the data transfer using Run Time Infrastructure between the EnergyPlus Java federate (left) and the thermostat controller Java federate (right).

3.3 Results & Discussion

The first simulation recorded sub-hourly temperature and HVAC energy consumption data of a simple EnergyPlus model. Dual set-point of an HVAC system between 21℃ and 23℃ caused activation. Heating activated in the morning and evening, and cooling activated mid-day, shown in Figure 3.6. HVAC operation between the narrow temperature range represents excess consumed energy by a non-energy conscious occupant. The following simulations attempt to incorporate intelligent CPS to control the model HVAC system.

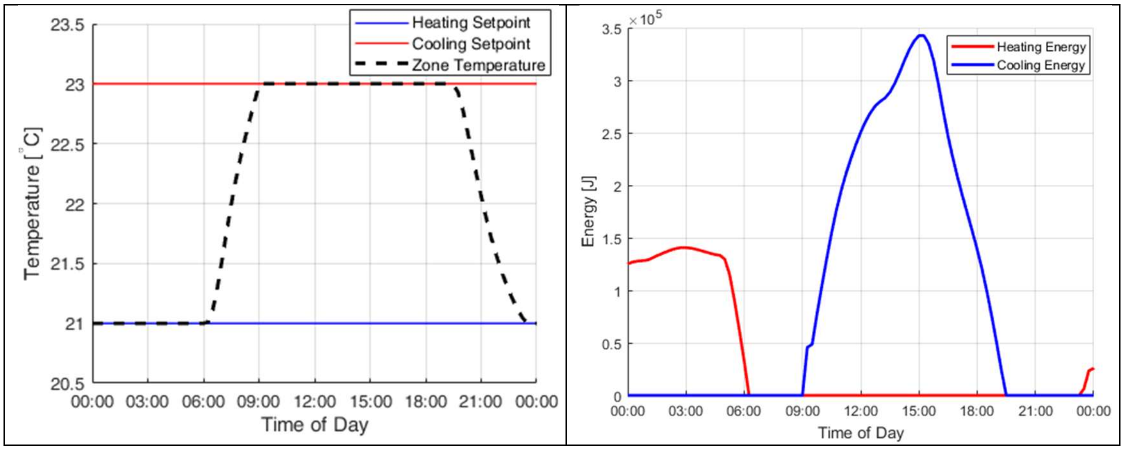


Figure 3.6: Naively created HVAC set-points and zone temperature (left) and corresponding heating and cooling power consumption (right).

A thermostat controller output dynamic set-points based on environment temperature and defined user comport (between 20°C and 25.5°C). Direct input of data from the second simulation and EnergyPlus/RTI input of the third simulation yielded identical results, shown in Figure 3.7. Matching outputs of the two simulations validates continuous and accurate EnergyPlus integration with UCEF.

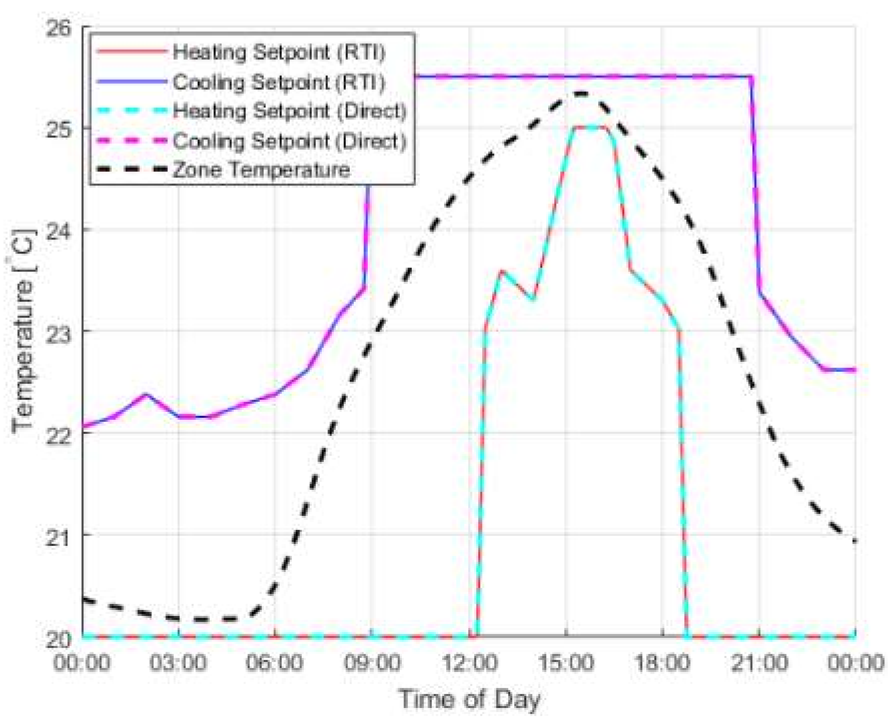


Figure 3.7: HVAC heating and cooling set-points based on an external thermostat controller. Direct connection and RTI connection yield consistent results.

Figure 3.7 also shows internal zone temperature of the EnergyPlus model. Dynamic thermostat controller outputs cause no HVAC activation for this simulation day. Compared to the naive set-points of the first simulation occupant, EnergyPlus co-simulation with the intelligent thermostat controller removed unnecessary energy consumption. Results verify UCEF integration does not impact simulated results.

This section discussed bi-directional communication between the EnergyPlus building simulation software and a co-simulation CPS environment called UCEF. Existing EnergyPlus interfacing capabilities were exploited to be connected to an HLA data exchange protocol for improved logical time step progression. As UCEF and EnergyPlus are each open-source software, development of building related CPS is made simply and inexpensively. Such capabilities allow for more availability for CPS development to improve home energy savings.

Chapter 4: Conclusion & Future Work

Two advancements in building CPS were accomplished to reduce residential energy consumption. First, a domestic water heater sensor was developed to provide occupant usage information to CPS. The conducted water heater experiment implemented a low cost, non-invasive, deployable package to collect and wirelessly send temperature measurements. Post processing methods were developed to relate temperature change rate of cold inlet and hot outlet pipes with hot water usage. Water draw events were effectively detected within a resolution of ±5 seconds. Flow rate correlations were significant at a 95% confidence level. Results suggest we can use these methods to detect patterns and qualitatively differentiate amount of flow through a water heater for CPS water heater control.

The second advancement was an open-source integration of a building simulation software with UCEF for the design and validation of CPS. By developing a simple FMU with a TCP/IP connection to a modified Java federate, calculated data at each time step was communicated between an EnergyPlus model and an HLA federation. This successful integration allows co-simulation between EnergyPlus models and CPS tools in the form of HLA federates. Simulated results validate UCEF-based federations can exchange data with EnergyPlus models without negative impact on results. More complex control algorithms and other simulation tools integrated into EnergyPlus creates an environment that can produce sophisticated CPS that reduce energy consumption in residential buildings.

Integration of EnergyPlus into UCEF as a new federate type enhances the platform's capabilities through added support of building simulations.

4.1 Future Work: Water Usage Sensor

Several additional concepts can be further investigated for more robust hot water sensing evaluations. First, event detection may need an added mechanism to take into account non-isolated draws. For example, it is possible for a household to have two overlapping draw events such as a concurrent shower and washing machine. Currently, two simultaneous events may not be able to be distinguished, but classification of two individual events can lead to further awareness.

Second, integration calculations can become more detailed with addition of more variables such as temperature adjusting material properties. Also, certain constants such as U_∞ and M can be self automated for seamless transitions in environment, such as weather conditions. Improvements may further develop by giving cold inlet and hot outlet pipe information a weight towards transient vs steady state cases. More accurate results can be achieved with these fine-tuned assumptions.

Lastly, incorporation of machine learning can lead to further optimizations. For example, as more diverse data becomes available, pattern recognition processes can be used to predict future hot water usage as well as possible flow irregularities, as in the case for leak detection. Additionally, as these predictions occur, water heaters can externally be controlled to activate and heat only when water is needed, saving up to ⅓ of home water heating energy [57].

4.2 Future Work: EnergyPlus Integration with UCEF

Additional concepts can be further investigated for more robust development. Modifications of FMU configuration files may be necessary for different simulation designs requiring different building model information. Currently, the IDF and the XML file need to be created manually based on the desired interface data. UCEF has support for the automatic generation of configuration files based on the content of fields in its graphical user interface. A user should be able to enter desired EnergyPlus variable information directly into the UCEF interface to automatically generate and update the IDF and XML file, rather than having to write the files themselves. Future work could address this usability feature through extensions to the UCEF graphical interface.

The presented approach using TCP/IP sockets could be further leveraged to integrate other FMI tools into UCEF. FMUs connected to other programs can utilize the TCP/IP concatenated string protocol to communicate with the Java federate in UCEF. Expanding co-simulation diversity to FMI connected tools can vastly improve UCEF simulator and emulator inventory. UCEF integration can increase development effectivity by allowing for improved logical timing control of these FMI tools.

References

[1] Lawrence Livermore National Laboratory, 2016, "Energy Flow Charts: Charting the Complex Relationships among Energy, Water, and Carbon," https://flowcharts.llnl.gov/. Access Date December 31, 2017.

[2] Morello, L., 2012, "Global CO2 Emissions from Fossil-Fuel Burning Rise into High-Risk Zone," https://www.scientificamerican.com/article/global-co2-emissions-from/. Access Date December 31, 2017.

[3] Nelson, A., Rakau, O., and Dorrenberg, P., 2010, "Green Buildings - A niche becomes mainstream," RREEF Research.

[4] NASA, 2017, "Global Climate Change: Vital Signs of the Planet," https://climate.nasa.gov/evidence/. Access Date December 31, 2017.

[5] Climate4you, 2017, "Greenhouse Gasses," http://www.climate4you.com/GreenhouseGasses.htm. Access Date December 31, 2017.

[6] Mckinsey & Company, 2009, "Pathways to a Low-Carbon Economy, Version 2 of the Global Green House Gas Abatement Cost Curve," p. 104.

[7] Ma, Z., Cooper, P., Daly, D., and Ledo, L., 2012, "Existing building retrofits: Methodology and state-of-the-art," Energy and Buildings, 55, pp. 889-902.

[8] Bureau of Transportation Statistics, 2017, "National Transportation Statistics," https://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national_transportation_statistics/html/table_04_23.html. Access Date December 31, 2017.

[9] CarChex, 2015, "How Often Do People Chan Cars?," https://www.carchex.com/research-center/auto-warranties/how-often-do-people-change-cars/. Access Date December 31, 2017.

[10] Weisbaum, H., 2006, "What's the life expectancy of my car?," http://www.nbcnews.com/id/12040753/ns/business-consumer_news/t/whats-life-expectancy-my-car/#.WjFinUqnGM8. Access Date December 31, 2017.

[11] Nguyen, T. A., and Aiello, M., 2013, "Energy intelligent buildings based on user activity: A survey," Energy and Buildings, 56, pp. 244-257.

[12] Hong, T., and Lin, H.-W., 2013, "Occupant Behavior: Impact on Energy Use of Private Offices," Conference: ASim 2012 - 1st Asia conference of International Building Performance Simulation Association Shanghai, China.

[13] Griffor, E. R., Greer, C., Wollman, D. A., and Burns, M. J., June 2017, "Framework for Cyber-Physical Systems: Volume 1, Overview," https://dx.doi.org/10.6028/NIST.SP.1500-201. Access Date December 31, 2017.

[14] Grispos, G., Gilsson, W. B., and Choo, K.-K. R., 2017, "Medical Cyber-Physical Systems Development: A Forensics-Driven Approach," IEEE, p. 108.

[15] Zhao, Y., and Ioannou, P., 2016, "A Traffic Light Signal Control System with Truck Priority1*This work has been supported in part by the National Science Foundation under the CPS program and in part by the University Transportation Center METRANS at University of Southern California," IFAC PapersOnLine, 49, pp. 377-382.

[16] Fantini, P., Tavola, G., Taisch, M., Barbosa, J., Leitao, P., Liu, Y., Sayed, M. S., and Lohse, N., 2016, "Exploring the integration of the human as a flexibility factor in CPS enabled manufacturing environments: Methodology and results," IEEE, p. 5711.

[17] Mohamed, N., Al-Jaroodi, J., Lazarova-Molnar, S., and Jawhar, I., 2017, "Middleware Challenges for Cyber-Physical Systems," Scalable Computing: Practice & Experience, 18(4), pp. 331-346.

[18] Distech Controls, 2017, "Innovative Controls for Greener Buildings," http://www.distech-controls.com/en/us/. Access Date December 31, 2017.

[19] Terpening, E. D., and Littleton, A., 2017, "The State of Internet of Things in the Home: Part II: Opportunities and Challenges for Brands Selling IOT Products for the Home," Altimeter Group - Research Reports, p. 1.

[20] Nest Labs, 2017, "Nest Thermostat," https://nest.com/thermostats/nest-learning-thermostat/overview/. Access Date December 31, 2017.

[21] Osama, S., Alfonse, M., and Salem, A.-B. M., 2017, "Intelligent Techniques for Smart Home Energy Management Based on Internet of Things (IoT) Paradigm," Egyptian Computer Science Journal, 41(3), pp. 33-43.

[22] Shirazi, E., and Jadid, S., 2017, "Cost reduction and peak shaving through domestic load shifting and DERs," Energy, 124, pp. 146-159.

[23] Basman, M. H. A., and William, A., 2016, "Design and Simulation of a Smart Home managed by an Intelligent Self-Adaptive System," International Journal of Engineering Research and Applications, Vol 6, Iss 8, Pp 64-90 (2016)(8), p. 64.

[24] Liu, Y., Qiu, B., Fan, X., Zhu, H., and Han, B., 2016, "Review of Smart Home Energy Management Systems," Energy Procedia, 104, pp. 504-508.

[25] Alaa, M., Zaidan, A. A., Zaidan, B. B., Talal, M., and Kiah, M. L. M., 2017, "Review: A review of smart home applications based on Internet of Things," Journal of Network and Computer Applications, 97, pp. 48-65.

[26] US Department of Energy, 2017, "Heat and Cool," https://energy.gov/energysaver/heat-and-cool.

[27] Energy.gov, 2013, "New Infographic and Projects to Keep Your Energy Bills Out of Hot Water," https://energy.gov/downloads/energy-saver-101-water-heating-infographic. Access Date December 31, 2017.

[28] US Department of Energy, 2010, "EnergyStar Water Heater Market Profile."

[29] Alliance for Water Efficiency, "Indoor Water Use," https://www.home-water-works.org/indoor-use. Access Date December 31, 2017.

[30] US Department of Energy, "Heat Pump Water Heaters," https://energy.gov/energysaver/heat-pump-water-heaters. Access Date December 31, 2017.

[31] Zhaojing, Y., Yanbo, C., Dezhi, L., Huanan, L., and Dongmin, Y., 2017, "Optimal Scheduling Strategy for Domestic Electric Water Heaters Based on the Temperature State Priority List," Energies (19961073), 10(9), pp. 1-15.

[32] Energy.gov, 2017, "Savings Project: Insulate Your Water Heater," https://energy.gov/energysaver/projects/savings-project-insulate-your-water-heater-tank. Access Date December 31, 2017.

[33] Ji, H., Gao, X., Wang, B., Huang, Z., and Li, H., 2013, "A New Method for Flow Rate Measurement in Millimeter-Scale Pipes," Sensors, pp. 1563-1577.

[34] Munir, M. M., Surachman, A., Fathonah, I. W., Billah, M. A., Khairurrijal, Mahfudz, H., Rimawan, R., and Lestari, S., 2015, "Development of microcontroller based water flow measurement," AIP Conference Proceedings, 1656(1).

[35] Nguyen, N. T., 1997, "Micromachined flow sensors—a review," Flow Measurement and Instrumentation, 8(1), pp. 7-16.

[36] Zuzunaga, A., and Maron, B., 2013 "A Survey of Non-Invasive and Semi-Invasive Flow Meters for Mining Applications: Understanding and Selecting the Right Technology for the Application," International Meeting on Mining Plan Maintenance (MAPLA)BI0497.

[37] Froehlich, J. E., Larson, E., Campbell, T., Haggerty, C., Fogarty, J., and Patel, S. N., 2009, "HydroSense," Proceedings of the 11th International Conference: Ubiquitous Computing, pp. 235-244.

[38] Stephens, A., 2004, "Flow Rate Measurements Using Flow-Induced Pipe Vibration," Journal of Fluids Engineering.

[39] Kim, J. H., and Lee, C. S., 2014, "Hydrodynamic Effects on Spectroscopic Water Detection in Gasoline Pipe Flow," Energies (19961073), pp. 3810-3822.

[40] Tawackolian, K., Büker, O., Hogendoorn, J., and Lederer, T., 2013, "Calibration of an ultrasonic flow meter for hot water," Flow Measurement and Instrumentation, 30, pp. 166-173.

[41] Tasaka, Y., Birkhofer, B., Furuichi, N., Kikura, H., Minagawa, H., Murai, Y., Murakawa, H., Motozawa, M., Nahar, S., Obayashi, H., Sawada, T., Shaik, A. K. J., Takeda, Y., Tezuka, K., Tsuji, Y., Yanagisawa, T., Wada, S., Wiklund, J., and Windhab, E. J., 2012, "Practical Applications," Springer Japan.

[42] Omega, 2017, "Transit-Time Ultrasonic Flowmeters for Clean Liquids," http://www.omega.com/subsection/Transit-time-ultrasonic-flow-meters.html. Access Date December 31, 2017.

[43] Poudel, S., Ni, Z., and Malla, N., 2017, "Real-time cyber physical system testbed for power system security and control," International Journal of Electrical Power and Energy Systems, 90, pp. 124-133.

[44] MathWorks, 2017, "Simulink," https://www.mathworks.com/products/simulink.html. Access Date December 31, 2017.

[45] Garraghan, P., McKee, D., Ouyang, X., Webster, D., and Xu, J., 2016, "SEED: A Scalable Approach for Cyber-Physical System Simulation," IEEE Transactions on Services Computing, p. 199.

[46] Magnusson, P. S., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hogberg, J., Larsson, F., Moestedt, A., and Werner, B., 2002, "Simics: A full system simulation platform," Computer(2), p. 50.

[47] IEEE, 2010, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-- Framework and Rules," IEEE Std 1516-2010, pp. 1-38.

[48] Roth , T., Song , E., Burns , M., Neema , H., Emfinger, W., and Sztipanovits, J., 2017, "Cyber-Physical System Development Environment for Energy Applications," ASME 2017 11th International Conference on Energy SustainabilityCharlotte, North Carolina, USA.

[49] Pokorny, T., and Fraser, M., 2017, "The Portico Project," http://tsigimpokorny.github.io/public/index.html. Access Date December 31, 2017.

[50] US Department of Energy's Building Technologies Office, and National Renewable Energy Laboratory (NREL), 2017, "EnergyPlus," https://energyplus.net/. Access Date December 31, 2017.

[51] Bergman, T. L., Dewit, D. P., and Lavine, A. S., 2011, "Fundamentals of Heat and Mass Transfer," John Wiley & Sons, New Jersey.

[52] Winterton, R. H. S., 1998, "Int. J Heat Mass Transfer." p. 41.

[53] Analog Devices, 2015, "Datasheet and Product Info," http://www.analog.com/en/products/analog-to-digital-converters/integrated-special-purpose-converters/digital-temperature-sensors/tmp36.html#product-overview. Access Date December 31, 2017.

[54] Particle, 2015, "Photon Datasheet," https://docs.particle.io/datasheets/photon-datasheet/. Access Date December 31, 2017.

[55] Junghanns, A., 2017, "FMI Functional Mock-up Interface," http://fmi-standard.org/. Access Date December 31, 2017.

[56] Modelisar, 2010, "Functional Mock-up Interface for Co-Simulation v1.0 (Documentation)."

[57] Jansen, S., Simmons, M., and Singer, J., 2016, "Smart Control for Home Water Heater Saving," Santa Clara University, Senior Design Thesis, Mechanical Engineering Dept.

Appendices

Appendix A Event Detection Code

Table of Contents

MAIN -- POST PROCESSING WATER HEATER DATA

```
clear; clc; close all;
```

Collects Files

add path to linked functions

```
addpath('Collecting Data', 'Evaluating Flow', 'Filtering Data', ...
    'Evaluating Tau', 'Evaluating Slope', 'Plotting Data')
pathToData = 'C:\Users\jsinger\Desktop\Collected Data\Olympus
\SingerHouse\';
% path = 'C:\Users\jsinger\Desktop\Collected Data\Olympus\SingerHouse
\';

searchStr = '*11_16_16*.xls*'; % specific day

listFilesInDir=dir(strcat(pathToData,searchStr));
    % dir(strcat(path,'*07_12_16_Hutch_Junipero1.xls*'));
numFiles=numel(listFilesInDir);
```

Defines Variables

```
global t  plotStyle plotNum  maxy  showPlots   FS

showPlots = 1; % if wanted to show plots or not
FS = 14;
plotNum = 1;
%used to correct the fact that the last row may have time of 00:00:01
corr = 1;
%------------------------------------------
```

Loops Through Each File

```
for k = 1 % set to 1 for specific day
```

```
    % create full screen window
    if showPlots
        figure('units','normalized','outerposition',[0 0 1 1])
    end
    fileName = listFilesInDir(k).name;
    fprintf('Working with %s \n', fileName)
    [num, txt, raw] = xlsread(strcat(pathToData,
listFilesInDir(k).name));

    %-----changes time stamps to datetime and
seconds--------------------
    Date = fileName(1:8);
    Date = strrep(Date, '_', '/');
    Time = txt(2:end-corr, 1); % used for machine learning array
    Dates = cell(length(Time), 1);
    Dates(:) = {[Date ' ']};
    TimesRaw = ...
        datetime(strcat(Dates, Time), 'InputFormat','MM/dd/yy
HH:mm:ss');
    timesRaw = datenum(txt(2:end-corr, 1), 'HH:MM:SS')*86400 -...
        datenum('00:00:00', 'HH:MM:SS')*86400;
     % for the specific range of time
    Times = TimesRaw;
    times = timesRaw;
%     Times = extractDataTimes(TimesRaw, TimesRaw, hourRange);
%     times = extractDataTimes(timesRaw, TimesRaw, hourRange);

  %----------------------------------------------------------------------

    %-----------Collects the appropriate temp
arrays---------------------
    [airCol, coldCol, hotCol, currentCol, flowCol] =
dataMiningCols(raw);

  %----------------------------------------------------------------------

    maxy= max(max(num(:, 1:3))); %max of temp data
    if any(airCol) % collects and plots air temperature readings
        plotStyle{plotNum} = 'g';
        airTempRaw = num(1:end-corr, airCol-1);
        airTemp = eliminateZeros(airTempRaw);
%         airTemp = extractDataTimes(airTemp, TimesRaw, hourRange);
        if showPlots
            plotSmoothedProfile(airTemp, Times);
            hold on
        end
        legendInfo{plotNum} = ['Air Temp']; %#ok<*SAGROW>
        plotNum = plotNum+1;
    end
    if any(hotCol) % collects and plots hot pipe readings
        plotStyle{plotNum} = 'r';
        temp_hRaw = num(1:end-corr, hotCol-1);
        temp_h = eliminateZeros(temp_hRaw);
%         temp_h = extractDataTimes(temp_h, TimesRaw, hourRange);
```

```
        if showPlots
            plotSmoothedProfile(temp_h, Times);
            hold on
        end
        legendInfo{plotNum} = ['Hot Outlet Temp']; %#ok<*SAGROW>
        plotNum = plotNum+1;
    end
    if any(coldCol) % collects and plots cold pipe readings
        plotStyle{plotNum} = 'b';
        temp_cRaw = num(1:end-corr, coldCol-1);
        temp_c = eliminateZeros(temp_cRaw);
%           temp_c = extractDataTimes(temp_c, TimesRaw, hourRange);
        if showPlots
            plotSmoothedProfile(temp_c, Times);
            hold on
        end
        legendInfo{plotNum} = ['Cold Inlet Temp'];
        plotNum = plotNum+1;
        temp = temp_c;
    end
    if showPlots % adds titles to plot
        legend(legendInfo, 'Location', 'northeast', 'FontSize', FS)
        title(strcat(strrep(listFilesInDir(k).name(1:8), '_', '/'), '
 Temp Data'))
    end
```

Extracts Draw Events

```
    coldDRAW = coldDrawExtract(temp_c, Times);
    coldHEAT = coldHeatExtract(temp_c, Times, coldDRAW);
    coldHEAT(:, 1) = coldHEAT(:, 1)-1; % matches better
    coldSS = coldSSExtract( coldDRAW, coldHEAT, temp_c, Times);
    coldDRAW = correctDRAW( coldDRAW, coldHEAT, coldSS );
    coldDRAW(:, 2) = coldDRAW(:, 2)-3;
```

Finds Tau Value

```
    lengthSS = seconds(Times(coldSS(:, 2))-Times(coldSS(:, 1)));
    longSSloc = find(lengthSS==max(lengthSS));
    l1 = coldSS(longSSloc, 1);
    l2 = coldSS(longSSloc, 2);
    tau = getTauNatural(temp_c(l1:l2), airTemp(l1:l2), times(l1:l2));
```

Calculates Flow Rate

```
    rhoAcCp = 2952; %approx
    Mc = rhoAcCp;
    Uinf = tau*Mc;
    Twc = 15; % assme cold temp

    flowOut = zeros(size(coldDRAW, 1), 1);
    R2F = 9.6197e-05;
```

```
    rip = 0.0209296; % [m] 0.824 inches
    Dip = 2*rip; % [m] inner diameter
    rop = 0.02667;% [m]1.05 inches
    Dop = 2*rop; % [m] outer diameter
    mu_w = 577e-6; % [Ns/m^2](@320K)
    Pr=3.77; % from scott code
        offset = 0;
        for i = 1:size(coldDRAW, 1)
            if ~isempty(coldDRAW(i, 1):coldDRAW(i, 2))
            extractingInds = coldDRAW(i, 1):coldDRAW(i, 2);
            t = times(extractingInds);
            t = t-t(1);
            T = temp_c(extractingInds);
            Tinf = mean(airTemp(extractingInds));
            A = T(end)-T(1);
            B = trapz(t, T);
            C = t(end)-t(1);
            Uc = (A*Mc+Uinf*B-Uinf*Tinf*C)/(-B+C*Twc);

            R1F = 1./Uc+R2F;
            hwf = 1./(R1F.*2.*pi.*rip);
            Nu = hwf.*Dip./k;
            n = 0.3; % for cooling (n = 0.04 for heating)
            Re=(Nu./(0.023.*(Pr^n))).^(5/4);
            flowOut(i-offset) = Re.*4.*Dop.*mu_w.*60;
%             fprintf('\nA = %.2f \nB = %.2f \nC = %.2f \nUc = %.2f
 \n', A, B, C, Uc);
            startTimes{i-offset} =
 datestr(Times(extractingInds(1)), 'HH:MM:SS PM');
            lengthDrawMeas(i-offset) = t(end);
            coldDRAW(i-offset, 1) = coldDRAW(i, 1);
            coldDRAW(i-offset, 2) = coldDRAW(i, 2);
            else
                offset = offset+1;
            end
            flowOut = flowOut(1:end-offset);
            startTimes = startTimes(1:end-offset);
            lengthDrawMeas = lengthDrawMeas(1:end-offset);
        end

            coldDRAW = coldDRAW(1:end-offset, :);
```

Plots Flow

```
    if showPlots
        ax = gca;
        for draw = 1:size(coldDRAW, 1)
            grab = coldDRAW(draw, 1):coldDRAW(draw, 2);
            p=patch(datenum([Times(grab(1)) Times(grab(end))
 Times(grab(end)) Times(grab(1))]),...
                [ax.YLim(1) ax.YLim(1) ax.YLim(2) ax.YLim(2)],'g');
            set(p,'FaceAlpha',0.3);
            hold on
```

```
        end

        for ss = 1:size(coldSS, 1)
            grab = coldSS(ss, 1):coldSS(ss, 2);
            p=patch(datenum([Times(grab(1)) Times(grab(end))
 Times(grab(end)) Times(grab(1))]),...
                [ax.YLim(1) ax.YLim(1) ax.YLim(2) ax.YLim(2)],'k');
            set(p,'FaceAlpha',0.1);
            hold on
        end

        for heat = 1:size(coldHEAT, 1)
            grab = coldHEAT(heat, 1):coldHEAT(heat, 2);
            p=patch(datenum([Times(grab(1)) Times(grab(end))
 Times(grab(end)) Times(grab(1))]),...
                [ax.YLim(1) ax.YLim(1) ax.YLim(2) ax.YLim(2)],'r');
            set(p,'FaceAlpha',0.3);
            hold on
        end
        plot(Times, temp_c, plotStyle{3}, 'LineWidth', 2.5)
        plot(Times, temp_h, plotStyle{2}, 'LineWidth', 2.5)
        plot(Times, airTemp, plotStyle{1}, 'LineWidth', 2.5)
    end

figure;
thisHour = 0;
oneMax = max(flowOut);
for j = .5:1:23.5
%     liters = 0;
    maxFlow = 0;
    for i = 1:length(flowOut)
        if hour(Times(coldDRAW(i, 1)))== thisHour
%             drawMin = minutes(Times(coldDRAW(i, 2))-
Times(coldDRAW(i, 1)));
%             liters = liters+flowOut(i)*drawMin;
            maxFlow = max([maxFlow, flowOut(i)]);
        end
    end
    bar(j, maxFlow, 'b')
    hold on
    thisHour = thisHour+1;
end
xlabel('Hour of Day','FontWeight','bold','FontSize',14)
ylabel('Max Flow Rate [L/min]','FontWeight','bold','FontSize',14)
grid on
xlim([0 24])
set(gca, 'XTick', [0:1:24],'FontSize',12)
set(gcf,'color','w');

end
```

Published with MATLAB® R2016a

```
function [airCol, hotCol, coldCol, currentCol, flowCol] =
 dataMiningCols(raw)
%dataMiningCols Uses expected column headings to extract column number
%   Collects the column names, then determines which column number the
 name
%   resides in based on pre-determined and expected strings.  The
 column
%   numbers are the output.
    headers = raw(1, :);

    airCol = getCol(headers, 'Air');
    hotCol = getCol(headers,'Hot');
    coldCol = getCol(headers,'Cold');
    pressCol = getCol(headers,'Pressure');
    upCol = getCol(headers,'Upper');
    contCol = getCol(headers,'Control');
    currentCol = getCol(headers,'Current');
    flowCol = getCol(headers,'Flow');
end
```

Published with MATLAB® R2016a

```
function [col] = getCol(headers, str)
%getCol Determines which heading containes the input string.
%   takes cell array of headers and a desired string as input.  The
%   function will determine which cell contains the input string.  Be
 aware
%   for if more than one header in a cell contains the input string.

    headers(cellfun(@(x) any(isnan(x)),headers)) = [];
    colArr = strfind(headers,str);
    col = find( ~cellfun('isempty', colArr) , 1);

end
```

Published with MATLAB® R2016a

```
function [data] = eliminateZeros(data)
%eliminateZeros Converts any low readings to average of surrounding #s
%   With data as an input, this is the decided method to account for
%   unexpected readings.  Outputs slightly filtered data where values
 less
%   than zero turn to an average of the numbers on either side.
 (previous
%   version just changed zero values to "nan"). NOTE: if many zero
 values in a row,
%   it will flat line until a non-zero number occurs.

%perc = .80; %percent off
%aveData = mean(data);

zeroLogic = data<=0;
zeroLocs = find(zeroLogic);
dataL = length(data);
for i = 1:length(zeroLocs)
    j = 0;
    while zeroLocs(i)+j<dataL
        j = j+1;
        if zeroLocs(i)+j>dataL
            data(zeroLocs(i):end) = data(zeroLocs(i)-1);
        elseif data(zeroLocs(i)+j)>0

            data(zeroLocs(i):zeroLocs(i)+j-1) = ...
                mean([data(zeroLocs(i)-1),data(zeroLocs(i)+j)]);
            break;
        end
    end

end

% figure
% plot(Times, temp_cRaw, '*-')
% xlabel('Times')
% ylabel('temp')
% hold on
% % plot(Times(theseBigTempDiff), temp_cRaw(theseBigTempDiff), 'o')
% these = temp_cRaw~=temp_cRawNew;
% plot(Times(these), temp_cRawNew(these), 'o')
% grid on


end
```

Published with MATLAB® R2016a

```
function [] = plotSmoothedProfile(temp, times)
%plotSmoothedProfile Plots temperature profile
%   With the temp readings and times this function will plot a
 smoothed
%   data profile.
%   May want FONT_SIZE to be global variable to be changed uniformly
 though
%   all plots in all scripts.
%   plotStyle is color, plotNum tracks which color to choose, maxy
 adjusts
%   the y axis bounds for each graph, showPlots is true or false to
%   determine if want to plot
    global plotStyle plotNum maxy FS
    FONT_SIZE = FS;
    %plot(times, temp, plotStyle{plotNum}) %original profile


    hold on
    plot(times, temp, plotStyle{plotNum}, 'LineWidth', 2.5) %smoothed
 profile

    %-----plot Formatting--------
    xlabel('Hour of
 Day','FontSize',FONT_SIZE,'FontWeight','bold','Color','k')
    ylabel('Temperature [{^
\circ}C]','FontSize',FONT_SIZE,'FontWeight','bold','Color','k')
    set(gca,'fontsize',FONT_SIZE)
    set(gcf,'color','w')
    set(gca,'YLim',[15 maxy])
%     set(gca,'YTick',[15:2:60])
    grid on
end
```

Published with MATLAB® R2016a

```
function [ newDRAW ] = coldDrawExtract( temp_c, Times )
%COLDDRAWEXTRACT finds the locations where cold pipe temperature drops
%significantly
%   Assumes that the cooling is very fast, corresponding to a fast
 flow
%   rate
% global showPlots
% Fixed temp difference (for filtering other non-draws)
choiceTempDrop = 3;
% Finds the tempDifference of all points
%reduced one in length
tempDiff = diff(temp_c);
% assures consistant length as original temp_c
tempDiff = [tempDiff;0];
% finds logic array where temps are decreasing
tempDrops = tempDiff<0;
% finds where slope direction change
% reduces one in length
switching = diff(tempDrops);
% assures consistant length as original temp_c
switching = [switching;0];
%identifies where it switches to decreasing slope
switchS = switching==1;
% identifies where it switches to increasing slope
switchE = switching==-1;

%array to get indicies of starting (decreasing) and ending
 (increasing)
%slope
indS = find(switchS)+1;
indE = find(switchE)+1;
% initializes
rc = 1;
for i = 1:length(indS)
    % takes new index of new decreasing slope, and sees where it first
 ends
    thisIndS = indS(i);
    % first index of increasing slope after decreasing slope
    thisIndE = indE(find(indE>thisIndS, 1));
    % finds temp at each of the indices
    tempS = temp_c(thisIndS);
    tempE = temp_c(thisIndE);
    % adds to output array only if the total temp diff is larger than
    % desired number
    if tempS-tempE>=choiceTempDrop
        newDRAW(rc, 1:2) = [thisIndS, thisIndE];
        rc = rc+1; %increments counter
%         if showPlots
%             plot(Times(thisIndS:thisIndE),
 temp_c(thisIndS:thisIndE), 'b-', 'linewidth', 2)
%         end
    end
```

Table of Contents


```
function [ newHEAT ] = coldHeatExtract( temp_c, Times, coldDRAW )
```

```
%COLDHEATEXTRACT finds the locations where cold pipe temperature drops
%signigicantly
%   Detailed explanation goes here
```

splines data

```
[r, c] = size(coldDRAW);
timeOld = 0;
timeSlot = [1 1];
% initializes
rc = 1;
```

Goes Through Data

```
for i = 1:r+1
    if i==1
        earlierTimeInd = 1;
        laterTimeInd = coldDRAW(i, 1);
    elseif i==r+1
        earlierTimeInd = coldDRAW(i-1, 2);
        laterTimeInd = length(Times);
    else
        earlierTimeInd = coldDRAW(i-1, 2);
        laterTimeInd = coldDRAW(i, 1);
    end

    timeSlot = [earlierTimeInd, laterTimeInd];
    lengthTimeSlot = length(timeSlot(1):timeSlot(2));
    if lengthTimeSlot>2

        adjust = timeSlot(1);
        desiredPts = 20;
        pts = min([desiredPts, ceil(lengthTimeSlot/desiredPts)]);
        newTemp = temp_c(timeSlot(1):timeSlot(2));
        x = timeSlot(1):pts:timeSlot(2);
        y = temp_c(x);
        xq = timeSlot(1):timeSlot(2);

% %         figure(2)
% %         plot(timeSlot(1):timeSlot(2), newTemp, 'ko-')
% %         plot(x, y, 'b*-')
```

```
        yNew = spline(x, y, xq);
        yNewp = pchip(x, y, xq);
% %
% %         plot(xq, yNew, 'rx-')
% %         plot(xq, yNewp, 'yx-')
% %         plot(x, y, 'b*-')
```

get large heatings

```
        TEMP = yNew';
% %         figure(3)
% %         if timeSlot(2)==length(temp_c)
% %             plot(Times(timeSlot(1):timeSlot(2)), yNew, 'y-')
% %         else
% %             plot(Times(timeSlot(1):timeSlot(2)), yNew, 'y-')
% %         end
%         global showPlots
        % Fixed temp difference (for filtering other non-draws)
        choiceTempDrop = 2;
        % Finds the tempDifference of all points
        %reduced one in length
        tempDiff = diff(TEMP);
        % assures consistant length as original temp_c
        tempDiff = [tempDiff;0];
        % finds logic array where temps are increasing
        tempDrops = tempDiff>0;
        % finds where slope direction change
        % reduces one in length
        switching = diff(tempDrops);
        %checks to see it it starts increasing
        if tempDrops(1)+tempDrops(2)==2
            switching(1) = 1;
        end
        % assures consistant length as original temp_c
        switching = [switching;0];
        %identifies where it switches to increasing slope
        switchS = switching==1;
        % identifies where it switches to decreasing slope
        switchE = switching==-1;

        %array to get indicies of starting (decreasing) and ending
 (increasing)
        %slope
        indS = find(switchS)+1;
        indE = find(switchE)+1;
        for j = 1:length(indS)
            % takes new index of new decreasing slope, and sees where
 it first ends
            thisIndS = indS(j);
            % first index of increasing slope after decreasing slope
            thisIndE = indE(find(indE>thisIndS, 1));
            % finds temp at each of the indices
            tempS = TEMP(thisIndS);
```

```
            tempE = TEMP(thisIndE);
            % adds to output array only if the total temp diff is
 larger than
            % desired number
            if tempE-tempS>=choiceTempDrop
                newHEAT(rc, 1:2) = [thisIndS+timeSlot(1), thisIndE
+timeSlot(1)];
                rc = rc+1; %increments counter
%                 if showPlots
%                     plot(Times([thisIndS:thisIndE]+timeSlot(1)),
 TEMP(thisIndS:thisIndE), 'r-', 'linewidth', 2)
%                 end
            end
        end

    end
end

%solves error where it may add index more than the length of draw
newHEAT(newHEAT>length(temp_c))=length(temp_c);

end
```

Published with MATLAB® R2016a

```
function [ newSS ] = coldSSExtract( coldDRAW, coldHEAT, temp_c, Times)
%UNTITLED15 Summary of this function goes here
%   Detailed explanation goes here
% global showPlots
indStart = 1;
indEnd = 0;
rDRAW = 1;
rHEAT = 1;
rSS = 1;
newSS = [0, 0];
loopCount = 1;
while indEnd<length(temp_c)
%     fprintf('LoopCount = %d -------------------', loopCount)
    loopCount = loopCount+1;
    rDRAW = find(coldDRAW(:, 1)>indStart, 1);
    rHEAT = find(coldHEAT(:, 1)>indStart, 1);
    if numel(rDRAW)==0 && numel(rHEAT)~=0
        m = [inf, coldHEAT(rHEAT, 1)];
    elseif numel(rHEAT)==0 && numel(rDRAW)~=0
        m = [coldDRAW(rDRAW, 1), inf];
    elseif numel(rDRAW)+numel(rHEAT)==0;
        m = length(temp_c);
    else
        m = [coldDRAW(rDRAW, 1), coldHEAT(rHEAT, 1)];
    end
    [M, I] = min(m);
    indEnd = M;
    if indEnd-indStart > 2
        newSS(rSS, 1:2) = [indStart, indEnd];
        rSS = rSS+1;
    end

    if I==1 % means draw was next
        indStart = coldDRAW(rDRAW, 2);
    elseif I==2 %means heat was next
        indStart = coldHEAT(rHEAT, 2);
    else
        warning('Something went wrong')
    end
%     if showPlots
%         plot(Times(newSS(rSS-1, 1):newSS(rSS-1, 2)),
 temp_c(newSS(rSS-1, 1):newSS(rSS-1, 2)), 'y-', 'linewidth', 2)
%     end
end
end
```

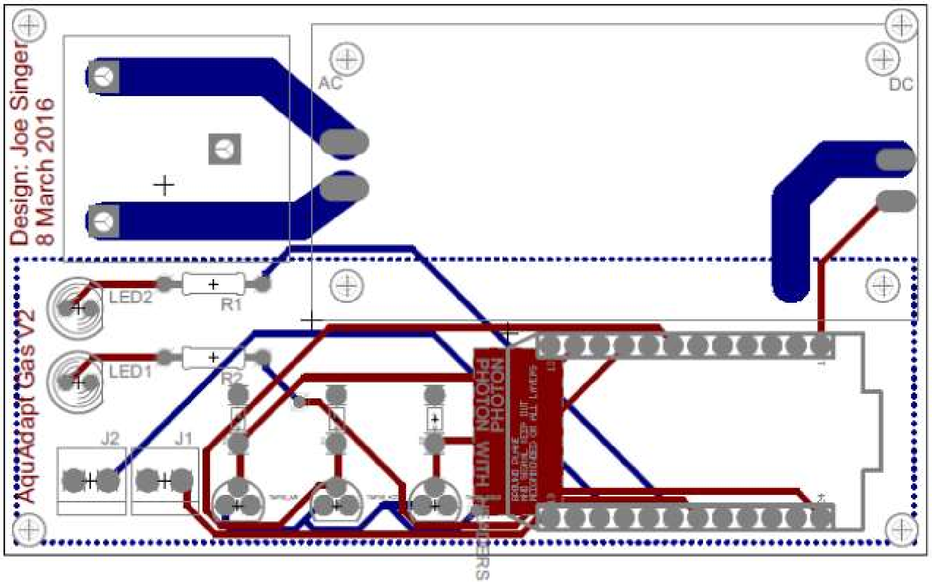
Published with MATLAB® R2016a

```
function [ tau ] = getTauNatural( Tpipe, Tinf, time)
%getTauNatural Calculates area under temp curves to get tau
%   integral of T2-T1=integral(T-Tamb)
time = time-time(1);
tau = -(Tpipe(end)-Tpipe(1))/...
    (trapz(time, Tpipe-Tinf));
end
```

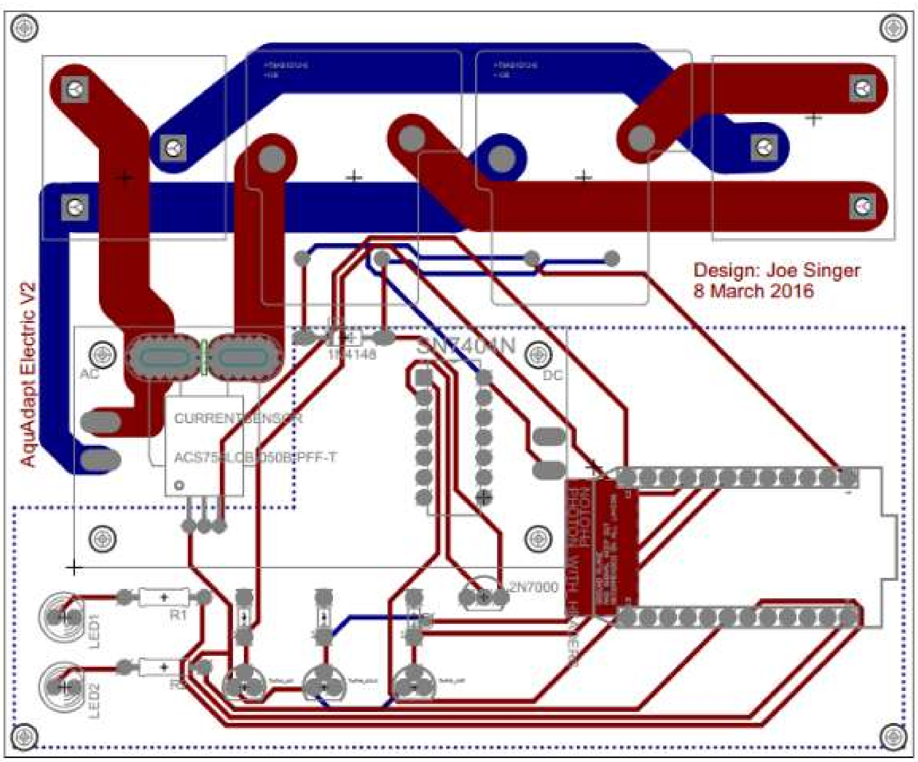
Published with MATLAB® R2016a

Appendix B Circuit Board Schematics and Parts

Gas Water Heater Design



Electric Water Heater Design



Parts List

Parts per Board	**Description/Part Number**	**Quantity (Gas)**	**Quantity (Electric)**
Printed Circuit Board	Custom	1	1
Spacers	1”	6	6
Acrylic	Clear	1	1
Large Terminal Block	3POS 7.5MM 30DEG	1	2
Microprocessor	Particle Photon	1	1
Temperature Sensors	TMP36	3	3
Capacitors	0.1μF	3	3
Power Converter	110/220V AC to 5V 2A 10W	1	1
Small Terminal Block	2POS 2.54MM PCB	2	0
Power Relays	T9AS1D12-5	0	2
Current Sensor	ACS758KCB	0	1
MOSFET	2N7000	0	1
Inverter	SN7404N	0	1
Diodes	1N4148TA	0	1

Appendix C Functions (in C) Called by the FMU


```c
// Basic built in files
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
// Project linker files
#include "FMU_Header_Files\Joe_ep_fmu.h" // Need to be in same folder as fmi header files to link properly
#include "Parser_Files\xml_parser.h"  // Put in folder for organization

ModelDescription* md; //creates md from xml file. Requires Parse Files

#include<winsock2.h>
WSADATA wsa;
SOCKET s;
int isWarmupFlag = 1;
int nextStringIndex = 0;
char nextString[2048];

char* getNextString(char myMsg[2048])
{
    //printf("\n===== In getNextString Function =====\n");
    int currentIndex = nextStringIndex;
    memset(nextString, '\0', 2048); // resets memory to null
    while (!(myMsg[currentIndex] == '\r' && myMsg[currentIndex + 1] == '\n'))
    {
        //printf("currentInd: %d\n", currentInd);
        nextString[currentIndex - nextStringIndex] = myMsg[currentIndex];
        currentIndex = currentIndex + 1;
    }
    //printf("diffInd %d\n", currentInd - nextStringIndex);
    if ((currentIndex - nextStringIndex) == 0)
    {
        nextStringIndex = 0;
        return '\0';
    }
    nextString[currentIndex - nextStringIndex] = '\0';
    //printf("nextString = %s\n", nextString1);
    //printf("<===== End of getNextString Function\n");
    nextStringIndex = currentIndex + 2;

    return nextString;
    //return currentInd;
}

int myLineReader(SOCKET s, char server_reply[2048], char buffer[2048])
{
    int currentBufferLocation = 0;
    int recievedReplyLength;
    while ((recievedReplyLength = recv(s, server_reply, 2048, 0)) > 0)
    {
        server_reply[recievedReplyLength] = '\0';
```

```
        if (currentBufferLocation == 0)
        {
            sprintf(buffer, "%s", server_reply); // or else something happens
        }
        else
        {
            sprintf(buffer, "%s%s", buffer, server_reply);
        }
        currentBufferLocation = currentBufferLocation + recievedReplyLength;
        if (currentBufferLocation >= 2048)
        {
            fprintf(myOutputLog, "bufferSize too large. Truncated incoming data.\n");
            break;
        }
        if (server_reply[recievedReplyLength - 2] == '\r' && server_reply[recievedReplyLength - 1] == '\n')// && bufferSize >= 2
        {
            //fprintf(myOutputLog, "Recieved end packet. \n");
            break;
        }
    }
    return 1;
}

/*---------------------------------------------------------------------------------------------------*/
// FMI FUNCTIONS: Platform, Version, Logging ...
/*---------------------------------------------------------------------------------------------------*/
const char* fmiGetTypesPlatform()
{
    return fmiPlatform;
}

const char* fmiGetVersion()
{
    return fmiVersion;
}

fmiStatus fmiSetDebugLogging(fmiComponent c, fmiBoolean loggingOn)
{
    return fmiOK;
}

/*---------------------------------------------------------------------------------------------------*/
// FMI DATA EXCHANGE FUNCTIONS: fmiGet__()
/
```

```c
*-----------------------------------------------------------------------------------------------*/
fmiStatus fmiGetReal(fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiReal value[])
{
    fprintf(myOutputLog, "\n========================= fmiGetReal =========================\n");
    fprintf(myOutputLog, "MASTER (E+) GETTING VARS FROM SLAVE (fmu)\n");

    unsigned int i;
    for (i = 0; i < nvr; ++i)
    {
        ScalarVariable* myInst = getVariable(md, vr[i], elm_Real); // pulls specific var from md
        const char* thisVarName = getName(myInst); // gets name of var from md

        fprintf(myOutputLog, "SENT var to E+\n");
        fprintf(myOutputLog, "  Name: ------------- %s \n", thisVarName);
        value[i] = my_values[vr[i]]; // replaces E+ value from FMI's stored values
        fprintf(myOutputLog, "  Value: ------------ %.2f \n", (float)value[i]);
        fprintf(myOutputLog, "  Value Reference --- '%d' \n", vr[i]);
    }
    fflush(myOutputLog);
    return fmiOK;
}

/*----------------------ONLY ABOVE USED ^^^---------------------------------------------------*/
fmiStatus fmiGetInteger(fmiComponent c, const fmiValueReference vr[],
    size_t nvr, fmiInteger value[])
{
    return fmiError;
}

fmiStatus fmiGetBoolean(fmiComponent c, const fmiValueReference vr[],
    size_t nvr, fmiBoolean value[])
{
    return fmiError;
}

fmiStatus fmiGetString(fmiComponent c, const fmiValueReference vr[],
    size_t nvr, fmiString  value[])
{
    return fmiError;
}

/*-----------------------------------------------------------------------------------------------*/
// FMI DATA EXCHANGE FUNCTIONS: fmiSet__()
/
```

```c
  *---------------------------------------------------------------------------------
  -----------------*/
fmiStatus fmiSetReal(fmiComponent c, const fmiValueReference vr[],
    size_t nvr, const fmiReal    value[])
{
    fprintf(myOutputLog, "\n======================== fmiSetReal ========================\n");
    fprintf(myOutputLog, "MASTER (E+) SENDING VARS TO SLAVE (fmu)\n");

    unsigned int i;
    for (i = 0; i < nvr; ++i)
    {
        ScalarVariable* myInst = getVariable(md, vr[i], elm_Real); // pulls specific var from md
        const char* thisVarName = getName(myInst); // gets name of var from md

        fprintf(myOutputLog, "RECIEVED var from E+\n");
        fprintf(myOutputLog, "  Name: ------------ %s\n", thisVarName);
        fprintf(myOutputLog, "  Value: ----------- %.2f\n", value[i]);
        fprintf(myOutputLog, "  Value Reference: -- '%d'\n", vr[i]);
        my_values[vr[i]] = (float)value[i]; // copys E+ values into the FMI's stored values
    }
    fflush(myOutputLog);
    return fmiOK;
}

/*----------------------ONLY ABOVE USED
  ^^^--------------------------------------------------*/
fmiStatus fmiSetInteger(fmiComponent c, const fmiValueReference vr[],
    size_t nvr, const fmiInteger value[])
{
    return fmiError;
}

fmiStatus fmiSetBoolean(fmiComponent c, const fmiValueReference vr[],
    size_t nvr, const fmiBoolean value[])
{
    return fmiError;
}

fmiStatus fmiSetString(fmiComponent c, const fmiValueReference vr[],
    size_t nvr, const fmiString  value[])
{
    return fmiError;
}

/
  *---------------------------------------------------------------------------------
  -----------------*/
// Creation and destruction of slave instances and setting debug status
/
```

```
    *-----------------------------------------------------------------------------------------------*/
fmiComponent fmiInstantiateSlave(fmiString  instanceName, fmiString  fmuGUID,
    fmiString  fmuLocation, fmiString  mimeType, fmiReal timeout, fmiBoolean visible,
    fmiBoolean interactive, fmiCallbackFunctions functions, fmiBoolean loggingOn)
{
    myOutputLog = fopen("../myOutputLog.log", "w"); // creates log file to write
    fprintf(myOutputLog, "\n======================== fmiInstantiateSlave =========================\n");
    fprintf(myOutputLog, "FMU LOCATION: %s\n", fmuLocation);
    fflush(myOutputLog);

    size_t sizeOfFmuLocation = strlen(fmuLocation); // finds size of location string
    char * fmuFilePath = (char *)malloc((sizeOfFmuLocation + 1) * sizeof(char)); //allocates memory for file location string
    for (size_t i = 0; i < sizeOfFmuLocation; i++) // switches direction of backslashes to work properly
    {
        if (fmuLocation[i] == '\\') {
            fmuFilePath[i] = '/';
        }
        else {
            fmuFilePath[i] = fmuLocation[i];
        }
    }
    if (fmuFilePath[sizeOfFmuLocation - 1] == '/')  // removes final slash if there is one
    {
        fmuFilePath[sizeOfFmuLocation - 1] = '\0';
    }
    fmuFilePath[sizeOfFmuLocation] = '\0'; // assures there is an ending null character (OK to have two)

    char * xmlFileName = "modelDescription.xml"; // name of xml in fmu
     //allocates memory for full file location
    char * xmlFilePath = (char *)malloc(sizeof(char)*(strlen(fmuFilePath) + 1 + strlen(xmlFileName) + 1));
    sprintf(xmlFilePath, "%s/%s", fmuFilePath, xmlFileName); // copys fmupath and xml file name for full xml path
    fprintf(myOutputLog, "XML LOCATION: %s\n", xmlFilePath);

    md = parse(xmlFilePath); // parses through xml to populate model description, md

    int myNumStates = getNumberOfStates(md);
    int myNumEventIndicators = getNumberOfEventIndicators(md);
    int myNumVars = md->n;
    fprintf(myOutputLog, "myNumStates: ----------- %d\n", myNumStates);
    fprintf(myOutputLog, "myNumEventIndicators: -- %d\n", myNumEventIndicators);
    fprintf(myOutputLog, "myNumVars: ------------- %d\n", myNumVars);
```

```c
    fflush(myOutputLog);
    /* CODE TO PRINT CONTENTS OF XML FOR SANITY PURPOSE
    FILE * xmlFile = fopen(xmlFilePath, "r");
    if (xmlFile == NULL)
    {
    fprintf(myOutputLog, "ERROR: No XML file found in FMU\n");
    // logs error if no xml file in fmu
    }

    char nextCharacter;
    fprintf(myOutputLog, "\n========== XML Content ==========\n");
    while ((nextCharacter = getc(xmlFile)) != EOF) {
    fprintf(myOutputLog, "%c", nextCharacter);
    }
    fprintf(myOutputLog, "\n");
    fclose(xmlFile);
    /*END CODE TO PRINT CONTENTS OF XML*/

    // removes vars from memory
    free(xmlFilePath);
    free(fmuFilePath);
    fflush(myOutputLog);

    //TODO: restructure to find # vars based on modelDescription, md
    my_values = malloc(sizeof(float) * 10); // 10 represents # of variables
      transfered (should change appropriately)


    // TODO: HARD CODED -- WANT SOME WAY AROUND THIS
    my_values[4] = (float)25; //starting cooling Start
    my_values[3] = (float)23; //starting heating Start
    fprintf(myOutputLog, "NOTE:\n");
    fprintf(myOutputLog, "  - EnergyPlus cannot be one day duration (for warmup
      distinction).\n");
    fprintf(myOutputLog, "  - Should look into finding total \"input\" and
      \"output\" causalities.\n");
    fprintf(myOutputLog, "      -- Information will change allocation for
      my_values array.\n");
    fprintf(myOutputLog, "  - Inaccuracies exist when turning recieving string
      value to float.\n");
    fprintf(myOutputLog, "  - Initial conditions (first iteration) hardcoded
      here. \n");
    fprintf(myOutputLog, "      -- Will be replaced by socket before E+ reads it.
      \n");
    fflush(myOutputLog);



    return my_values; //Seems need to return allocated memory
} // End fmiInstantiateSlave()

fmiStatus fmiInitializeSlave(fmiComponent c, fmiReal tStart, fmiBoolean
```

```c
    StopTimeDefined, fmiReal tStop)
{
    fprintf(myOutputLog, "\n======================== fmiInitializeSlave ========================\n");

    if (tStop - tStart > 86400)
    {
        isWarmupFlag = 0;
    }

    if (isWarmupFlag == 0)
    {
        fprintf(myOutputLog, "------ STARTING SOCKET CONNECTION ------\n");
        // THIS IS SOCKET STUFF
        struct sockaddr_in server;

        fprintf(myOutputLog, "Initializing Winsock...\n");
        if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
        {
            fprintf(myOutputLog, "Failed. Error Code : %d", WSAGetLastError());
            //return 1;
        }
        fprintf(myOutputLog, "...Initialized.\n");

        //Create a socket
        if ((s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET)
        {
            fprintf(myOutputLog, "Could not create socket : %d", WSAGetLastError());
        }
        fprintf(myOutputLog, "...Socket created.\n");

        //IP information
        server.sin_addr.s_addr = inet_addr("192.168.56.101"); //ip address
        server.sin_family = AF_INET;
        server.sin_port = htons(6789); //port #

                                        //Connect to remote server
        if (connect(s, (struct sockaddr *)&server, sizeof(server)) < 0)
        {
            fprintf(myOutputLog, "connect error\n");
            //return 1;
        }
        fprintf(myOutputLog, "...Connected\n");
    }
    return fmiOK;
} // End fmiInitializeSlave()

fmiStatus fmiTerminateSlave(fmiComponent c)
{
    return fmiOK;
}
```

```c

fmiStatus fmiResetSlave(fmiComponent c)
{
    return fmiOK;
}

/*-------------------------------- fmiFreeSlaveInstance()
  ---------------------------------------*/
// Master FMU is done with the Slave.
/
  *---------------------------------------------------------------------------
  -----------------*/
void fmiFreeSlaveInstance(fmiComponent c)
{
    fprintf(myOutputLog, "\n========================= fmiFreeSlaveInstance
      =========================\n");
    // frees remaining vars from memory
    free(my_values);
    freeElement(md); // frees modelDescription, md

    if (isWarmupFlag == 0)
    {
        printf("--- CLOSING CONNECTION ---\n");
        fprintf(myOutputLog, "--- CLOSING CONNECTION ---\n");

        char message[1024];
        sprintf(message, "TERMINATE\r\n\r\n");
        if (send(s, message, strlen(message), 0) < 0)
        {
            fprintf(myOutputLog, "Send failed\n");
            //return 1;
        }
        fprintf(myOutputLog, "Data Sent: %s\n", message);
        closesocket(s); // frees socket
        WSACleanup(); // cleans socket
    }

    fclose(myOutputLog);    //closes output file
}
/*-----------------------------------------------------------------*/
// DERIVIATIVES
/
  *---------------------------------------------------------------------------
  -----------------*/
fmiStatus fmiSetRealInputDerivatives(fmiComponent c, const  fmiValueReference vr
  [],
    size_t nvr, const  fmiInteger order[], const  fmiReal value[])
{
    return fmiError;
}

fmiStatus fmiGetRealOutputDerivatives(fmiComponent c, const   fmiValueReference
```

```c
    vr[],
    size_t  nvr, const   fmiInteger order[], fmiReal value[])
{
    return fmiError;
}

/*-----------------------------------------------------------------------
fmiCancelStep() -----*/
// STEPS
/*--------------------------------------------------------------------------------------------
-----------------*/
fmiStatus fmiCancelStep(fmiComponent c)
{
    return fmiError;
}

fmiStatus fmiDoStep(fmiComponent c, fmiReal currentCommunicationPoint,
    fmiReal communicationStepSize, fmiBoolean newStep)
{

    fprintf(myOutputLog, "\n========================= fmiDoStep (%.0f sec) =========================\n", currentCommunicationPoint);
    unsigned int i;
    for (i = 1; i<5; ++i)
    {
        ScalarVariable* myInst = getVariable(md, i, elm_Real); // pulls specific var from md
        const char* thisVarName = getName(myInst); // gets name of var from md

        fprintf(myOutputLog, "%d: %f ---------- %s\n", i, my_values[i], thisVarName);
        fflush(myOutputLog);
    }
    if (isWarmupFlag == 0)
    {
        // SOCKET CONNECTION VARS
        char mySendMsg[2048], server_reply[2048];
        int recv_size;
        char buffer[2048];

        // ========== Sending Data ==========
        fprintf(myOutputLog, "<---------- Sending Data ---------->\n");
        printf("<---------- Sending Data ---------->\n");

        // Loops through all output vars into string to send
        sprintf(mySendMsg, "UPDATE\r\n%.0f\r\n", currentCommunicationPoint); // sets steart of outgoing message with timestamp
        for (i = 1; i < 3; ++i) // TODO loop through all "input" causalities
        {
            ScalarVariable* myInst = getVariable(md, i, elm_Real); // pulls specific var from md
```

```
            const char* thisVarName = getName(myInst); // gets name of var from md
            fprintf(myOutputLog, "  Preparing to send %s as %f\n", thisVarName, my_values[i]);
            fflush(myOutputLog);
            printf("Preparing to send %s as %f\n", thisVarName, my_values[i]);
            sprintf(mySendMsg, "%s%s\r\n%f\r\n", mySendMsg, thisVarName, my_values[i]);
        }
        sprintf(mySendMsg, "%s\r\n", mySendMsg); // adds final \r\n to outgoing message
        // Send through socket command
        if (send(s, mySendMsg, strlen(mySendMsg), 0) < 0)
        {
            fprintf(myOutputLog, "  Send failed\n");
            printf("Send failed\n");
            return fmiError;
        }
        fprintf(myOutputLog, "  Data Sent!\n");
        printf("Data Sent!\n");

        // ---------- Recieving Data ----------
        fprintf(myOutputLog, "----------> Recieving Data <----------\n");
        printf("----------> Recieving Data <----------\n");

        if (myLineReader(s, server_reply, buffer) < 0)
        {
            fprintf(myOutputLog, "  Could not read line properly\n");
            fflush(myOutputLog);
            return fmiError;
        }
        fprintf(myOutputLog, "  Data Recieved! \n");
        fflush(myOutputLog);
        printf("Data Recieved\n");

        // ----- Parsing recieved data string -----
        char* thisHeading;
        char* thisTimeString;
        char* thisVarName;
        char* thisValue;

        nextStringIndex = 0;

        //finds header -- first expected value
        thisHeading = getNextString(buffer);
        fprintf(myOutputLog, "  Recieved Header as %s\n", thisHeading);
        fflush(myOutputLog);
        printf("Recieved Header as %s\n", thisHeading);

        if (!strcmp(thisHeading, "SET"))
        {
            //finds time value -- second expected value
```

```c
            thisTimeString = getNextString(buffer);
            int thisRecievedTime = atoi(thisTimeString); //converted string to int
            fprintf(myOutputLog, "  Recieved Time as %d\n", thisRecievedTime);
            fflush(myOutputLog);
            printf("Recieved Time as %d\n", thisRecievedTime);

            while ( (thisVarName = getNextString(buffer)) != '\0') // until nothing is between \r\n\r\n
            {
                // Expected Variable Name
                // **update for thisVarName happens when testing while loop condition
                fprintf(myOutputLog, "  Recieved %s ", thisVarName);
                fflush(myOutputLog);
                printf("Recieved %s ", thisVarName);

                // gets value reference
                ScalarVariable* myInst = getVariableByName(md, thisHeading);
                int myValRef = getValueReference(myInst);

                // Expected Value
                thisValue = getNextString(buffer);
                float thisRecievedValue = atof(thisValue); //converted string to float
                fprintf(myOutputLog, "as %f\n", thisRecievedValue);
                fflush(myOutputLog);
                printf("as %f\n", thisRecievedValue);

                // Replacing values
                my_values[myValRef] = thisRecievedValue;
                fprintf(myOutputLog, "   --New my_values[%d] = %f\n", myValRef, my_values[myValRef]);
                fflush(myOutputLog);
                //have condition if no matching vr
            }
        }
        else if (thisHeading == "NOUPDATE\0")
        {
            fprintf(myOutputLog, "NOUPDATE\n");
            printf("NOUPDATE\n");;
            fflush(myOutputLog);
        }
        else
        {
            fprintf(myOutputLog, "No expected headers found\n");;
            fflush(myOutputLog);
        }
    }
    return fmiOK;
} // End fmiDoStep()
```

```c

  /
   *-----------------------------------------------------------------------------
   -------------------*/
  // FMI STATUS FUNCTIONS
  /
   *-----------------------------------------------------------------------------
   -------------------*/
fmiStatus fmiGetStatus(fmiComponent c, const fmiStatusKind s, fmiStatus* value)
{
    return fmiError;
}

fmiStatus fmiGetRealStatus(fmiComponent c, const fmiStatusKind s, fmiReal*
  value)
{
    return fmiError;
}

fmiStatus fmiGetIntegerStatus(fmiComponent c, const fmiStatusKind s, fmiInteger*
  value)
{
    return fmiError;
}

fmiStatus fmiGetBooleanStatus(fmiComponent c, const fmiStatusKind s, fmiBoolean*
  value)
{
    return fmiError;
}

fmiStatus fmiGetStringStatus(fmiComponent c, const fmiStatusKind s, fmiString*
  value)
{
    return fmiError;
}
```

Appendix D EnergyPlus Simple IDF Model

```
                           BasicHeatingSetpoint.idf
!-Generator IDFEditor 1.49
!-Option OriginalOrderTop UseSpecialFormat

!-NOTE: All comments with '!-' are ignored by the IDFEditor and are generated
automatically.
!-      Use '!' comments if they need to be retained when using the IDFEditor.

ZoneControl:Thermostat,
    myZ1Thermo,              !- Name
    ZONE ONE,                !- Zone or ZoneList Name
    ALWAYS 4,                !- Control Type Schedule Name
    ThermostatSetpoint:DualSetpoint,  !- Control 1 Object Type
    myDualSetpoint;          !- Control 1 Name

ThermostatSetpoint:DualSetpoint,
    myDualSetpoint,          !- Name
    myStartHeating,          !- Heating Setpoint Temperature Schedule Name
    myStartCooling;          !- Cooling Setpoint Temperature Schedule Name

RunPeriod,
    myRunPeriod,             !- Name
    6,                       !- Begin Month
    1,                       !- Begin Day of Month
    6,                       !- End Month
    2,                       !- End Day of Month
    UseWeatherFile,          !- Day of Week for Start Day
    Yes,                     !- Use Weather File Holidays and Special Days
    Yes,                     !- Use Weather File Daylight Saving Period
    No,                      !- Apply Weekend Holiday Rule
    Yes,                     !- Use Weather File Rain Indicators
    Yes,                     !- Use Weather File Snow Indicators
    1,                       !- Number of Times Runperiod to be Repeated
    Yes;                     !- Increment Day of Week on repeat

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    ZONE ONE,                !- Output:Variable Index Key Name
    Zone Air Temperature ,   !- Output:Variable Name
    Joe_ep_fmu.fmu,          !- FMU File Name
    Joe_ep_fmu,              !- FMU Instance Name
    epSendZoneMeanAirTemp;   !- FMU Variable Name

ExternalInterface,
    FunctionalMockupUnitImport;  !- Name of External Interface

ExternalInterface:FunctionalMockupUnitImport,
    Joe_ep_fmu.fmu,          !- FMU File Name
    0,                       !- FMU Timeout {ms}
    1;                       !- FMU LoggingOn
                                  Page 1
```

```
ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    Environment,             !- Output:Variable Index Key Name
    Site Outdoor Air Drybulb Temperature ,  !- Output:Variable Name
    Joe_ep_fmu.fmu,          !- FMU File Name
    Joe_ep_fmu,              !- FMU Instance Name
    epSendOutdoorAirTemp;    !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
    myStartHeating,          !- Name
    myTemps,                 !- Schedule Type Limits Names
    Joe_ep_fmu.fmu,          !- FMU File Name
    Joe_ep_fmu,              !- FMU Instance Name
    epGetStartHeating,       !- FMU Variable Name
    17;                      !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
    myStartCooling,          !- Name
    myTemps,                 !- Schedule Type Limits Names
    Joe_ep_fmu.fmu,          !- FMU File Name
    Joe_ep_fmu,              !- FMU Instance Name
    epGetStartCooling,       !- FMU Variable Name
    30;                      !- Initial Value

ScheduleTypeLimits,
    myTemps,                 !- Name
    ,                        !- Lower Limit Value
    ,                        !- Upper Limit Value
    Continuous,              !- Numeric Type
    Temperature;             !- Unit Type

! Introduction to EnergyPlus - Exercise 1A
!
! Building: Fictional 1 zone building with lightweight walls and 2 windows.
!           8m x 6m x 2.7m high, long side facing N and S
!           20C heating, 24C cooling
! Internal: None.
! System:   Purchased Air.
! Plant:    None.
! Environment:  Chicago, IL, USA, Summer and Winter design days
!
!
Version,8.7;

Building,
    Exercise 1A,             !- Name
    0.0,                     !- North Axis {deg}
    Country,                 !- Terrain
```

```
    0.04,                    !- Loads Convergence Tolerance Value
    0.4,                     !- Temperature Convergence Tolerance Value {deltaC}
    FullInteriorAndExterior, !- Solar Distribution
    ,                        !- Maximum Number of Warmup Days
    6;                       !- Minimum Number of Warmup Days

Timestep,2;
SurfaceConvectionAlgorithm:Inside,TARP;
SurfaceConvectionAlgorithm:Outside,TARP;
HeatBalanceAlgorithm,ConductionTransferFunction;

ShadowCalculation,
    AverageOverDaysInFrequency,  !- Calculation Method
    20;                      !- Calculation Frequency

SimulationControl,
    No,                      !- Do Zone Sizing Calculation
    No,                      !- Do System Sizing Calculation
    No,                      !- Do Plant Sizing Calculation
    Yes,                     !- Run Simulation for Sizing Periods
    Yes;                     !- Run Simulation for Weather File Run Periods

Site:Location,
    CHICAGO_IL_USA TMY2-94846,  !- Name
    41.78000,                !- Latitude {deg}
    -87.75000,               !- Longitude {deg}
    -6.000000,               !- Time Zone {hr}
    190.0000;                !- Elevation {m}

! CHICAGO_IL_USA Heating 99.6%, MaxDB= -21.20 Wind Speed=   4.60 Wind Dir= 270.00
SizingPeriod:DesignDay,
    CHICAGO_IL_USA Heating 99.6% Conditions,  !- Name
    1,                       !- Month
    21,                      !- Day of Month
    WinterDesignDay,         !- Day Type
    -21.20000,               !- Maximum Dry-Bulb Temperature {C}
    0.0,                     !- Daily Dry-Bulb Temperature Range {deltaC}
    ,                        !- Dry-Bulb Temperature Range Modifier Type
    ,                        !- Dry-Bulb Temperature Range Modifier Day Schedule
Name
    Wetbulb,                 !- Humidity Condition Type
    -21.20000,               !- Wetbulb or DewPoint at Maximum Dry-Bulb {C}
    ,                        !- Humidity Condition Day Schedule Name
    ,                        !- Humidity Ratio at Maximum Dry-Bulb
{kgWater/kgDryAir}
    ,                        !- Enthalpy at Maximum Dry-Bulb {J/kg}
    ,                        !- Daily Wet-Bulb Temperature Range {deltaC}
    99063.21,                !- Barometric Pressure {Pa}
```

```
    4.600000,                !- Wind Speed {m/s}
    270.0000,                !- Wind Direction {deg}
    No,                      !- Rain Indicator
    No,                      !- Snow Indicator
    No,                      !- Daylight Saving Time Indicator
    ASHRAEClearSky,          !- Solar Model Indicator
    ,                        !- Beam Solar Day Schedule Name
    ,                        !- Diffuse Solar Day Schedule Name
    ,                        !- ASHRAE Clear Sky Optical Depth for Beam Irradiance
(taub) {dimensionless}
    ,                        !- ASHRAE Clear Sky Optical Depth for Diffuse
Irradiance (taud) {dimensionless}
    0.0;                     !- Sky Clearness

Site:GroundTemperature:BuildingSurface,18.3,18.2,18.3,18.4,20.1,22.0,22.3,22.5,22.5,
20.7,18.9,18.5;

Material,
    PLASTERBOARD-1,          !- Name
    MediumSmooth,            !- Roughness
    0.01200,                 !- Thickness {m}
    0.16000,                 !- Conductivity {W/m-K}
    950.000,                 !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    FIBERGLASS QUILT-1,      !- Name
    Rough,                   !- Roughness
    0.066,                   !- Thickness {m}
    0.040,                   !- Conductivity {W/m-K}
    12.000,                  !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    WOOD SIDING-1,           !- Name
    Rough,                   !- Roughness
    0.00900,                 !- Thickness {m}
    0.14000,                 !- Conductivity {W/m-K}
    530.000,                 !- Density {kg/m3}
    900.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
```

```
    0.600000;                !- Visible Absorptance

Material,
    PLASTERBOARD-2,          !- Name
    Rough,                   !- Roughness
    0.01000,                 !- Thickness {m}
    0.16000,                 !- Conductivity {W/m-K}
    950.000,                 !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    FIBERGLASS QUILT-2,      !- Name
    Rough,                   !- Roughness
    0.1118,                  !- Thickness {m}
    0.040,                   !- Conductivity {W/m-K}
    12.000,                  !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    ROOF DECK,               !- Name
    Rough,                   !- Roughness
    0.01900,                 !- Thickness {m}
    0.14000,                 !- Conductivity {W/m-K}
    530.000,                 !- Density {kg/m3}
    900.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    HF-C5,                   !- Name
    MediumRough,             !- Roughness
    0.1015000,               !- Thickness {m}
    1.729600,                !- Conductivity {W/m-K}
    2243.000,                !- Density {kg/m3}
    837.0000,                !- Specific Heat {J/kg-K}
    0.9000000,               !- Thermal Absorptance
    0.6500000,               !- Solar Absorptance
    0.6500000;               !- Visible Absorptance

Construction,
    LTWALL,                  !- Name
```

```
    WOOD SIDING-1,           !- Outside Layer
    FIBERGLASS QUILT-1,      !- Layer 2
    PLASTERBOARD-1;          !- Layer 3

Construction,
    LTFLOOR,                 !- Name
    HF-C5;                   !- Outside Layer

Construction,
    LTROOF,                  !- Name
    ROOF DECK,               !- Outside Layer
    FIBERGLASS QUILT-2,      !- Layer 2
    PLASTERBOARD-2;          !- Layer 3

Zone,
    ZONE ONE,                !- Name
    0,                       !- Direction of Relative North {deg}
    0, 0, 0,                             !- X,Y,Z  {m}
    1,                       !- Type
    1,                       !- Multiplier
    2.7000,                  !- Ceiling Height {m}
    129.6;                   !- Volume {m3}

GlobalGeometryRules,
    UpperLeftCorner,         !- Starting Vertex Position
    Counterclockwise,        !- Vertex Entry Direction
    WorldCoordinateSystem;   !- Coordinate System

BuildingSurface:Detailed,
    SURFACE NORTH,           !- Name
    Wall,                    !- Surface Type
    LTWALL,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0.50,                    !- View Factor to Ground
    4,                       !- Number of Vertices
    8.00, 6.00, 2.70,                    !- X,Y,Z  1 {m}
    8.00, 6.00, 0,                       !- X,Y,Z  2 {m}
    0, 6.00, 0,                          !- X,Y,Z  3 {m}
    0, 6.00, 2.70;                       !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE EAST,       !- Name
    Wall,                    !- Surface Type
    LTWALL,                  !- Construction Name
```

```
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0.50,                    !- View Factor to Ground
    4,                       !- Number of Vertices
    8.00, 0, 2.70,                      !- X,Y,Z  1 {m}
    8.00, 0, 0,                         !- X,Y,Z  2 {m}
    8.00, 6.00, 0,                      !- X,Y,Z  3 {m}
    8.00, 6.00, 2.70;                   !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE SOUTH,      !- Name
    Wall,                    !- Surface Type
    LTWALL,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0.50,                    !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 0, 2.70,                         !- X,Y,Z  1 {m}
    0, 0, 0,                            !- X,Y,Z  2 {m}
    8.00, 0, 0,                         !- X,Y,Z  3 {m}
    8.00, 0, 2.70;                      !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE WEST,       !- Name
    Wall,                    !- Surface Type
    LTWALL,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0.50,                    !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 6.00, 2.70,                      !- X,Y,Z  1 {m}
    0, 6.00, 0,                         !- X,Y,Z  2 {m}
    0, 0, 0,                            !- X,Y,Z  3 {m}
    0, 0, 2.70;                         !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE FLOOR,      !- Name
    Floor,                   !- Surface Type
    LTFLOOR,                 !- Construction Name
```

```
    ZONE ONE,                !- Zone Name
    Ground,                  !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    NoSun,                   !- Sun Exposure
    NoWind,                  !- Wind Exposure
    0,                       !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 0, 0,                            !- X,Y,Z  1 {m}
    0, 6.00, 0,                         !- X,Y,Z  2 {m}
    8.00, 6.00, 0,                      !- X,Y,Z  3 {m}
    8.00, 0, 0;                         !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE ROOF,       !- Name
    Roof,                    !- Surface Type
    LTROOF,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0,                       !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 6.00, 2.70,                      !- X,Y,Z  1 {m}
    0, 0, 2.70,                         !- X,Y,Z  2 {m}
    8.00, 0, 2.70,                      !- X,Y,Z  3 {m}
    8.00, 6.00, 2.70;                   !- X,Y,Z  4 {m}

ScheduleTypeLimits,
    Any Number;              !- Name

Schedule:Compact,
    ALWAYS 4,                !- Name
    Any Number,              !- Schedule Type Limits Name
    Through: 12/31,          !- Field 1
    For: AllDays,            !- Field 2
    Until: 24:00, 4;         !- Field 4

Schedule:Compact,
    ALWAYS 20,               !- Name
    Any Number,              !- Schedule Type Limits Name
    Through: 12/31,          !- Field 1
    For: AllDays,            !- Field 2
    Until: 24:00, 20;        !- Field 4

Schedule:Compact,
    ALWAYS 24,               !- Name
    Any Number,              !- Schedule Type Limits Name
```

```
    Through: 12/31,          !- Field 1
    For: AllDays,            !- Field 2
    Until: 24:00, 24;        !- Field 4

ZoneHVAC:EquipmentConnections,
    ZONE ONE,                !- Zone Name
    ZONE ONE Equipment,      !- Zone Conditioning Equipment List Name
    ZONE ONE Supply Inlet,   !- Zone Air Inlet Node or NodeList Name
    ,                        !- Zone Air Exhaust Node or NodeList Name
    ZONE ONE Zone Air Node,  !- Zone Air Node Name
    ZONE ONE Return Outlet;  !- Zone Return Air Node Name

ZoneHVAC:EquipmentList,
    ZONE ONE Equipment,      !- Name
    ZoneHVAC:IdealLoadsAirSystem,  !- Zone Equipment 1 Object Type
    ZONE ONE Purchased Air,  !- Zone Equipment 1 Name
    1,                       !- Zone Equipment 1 Cooling Sequence
    1;                       !- Zone Equipment 1 Heating or No-Load Sequence

ZoneHVAC:IdealLoadsAirSystem,
    ZONE ONE Purchased Air,  !- Name
    ,                        !- Availability Schedule Name
    ZONE ONE Supply Inlet,   !- Zone Supply Air Node Name
    ,                        !- Zone Exhaust Air Node Name
    50,                      !- Maximum Heating Supply Air Temperature {C}
    13,                      !- Minimum Cooling Supply Air Temperature {C}
    0.015,                   !- Maximum Heating Supply Air Humidity Ratio
{kgWater/kgDryAir}
    0.01,                    !- Minimum Cooling Supply Air Humidity Ratio
{kgWater/kgDryAir}
    NoLimit,                 !- Heating Limit
    ,                        !- Maximum Heating Air Flow Rate {m3/s}
    ,                        !- Maximum Sensible Heating Capacity {W}
    NoLimit,                 !- Cooling Limit
    ,                        !- Maximum Cooling Air Flow Rate {m3/s}
    ,                        !- Maximum Total Cooling Capacity {W}
    ,                        !- Heating Availability Schedule Name
    ,                        !- Cooling Availability Schedule Name
    ConstantSupplyHumidityRatio,  !- Dehumidification Control Type
    ,                        !- Cooling Sensible Heat Ratio {dimensionless}
    ConstantSupplyHumidityRatio,  !- Humidification Control Type
    ,                        !- Design Specification Outdoor Air Object Name
    ,                        !- Outdoor Air Inlet Node Name
    ,                        !- Demand Controlled Ventilation Type
    ,                        !- Outdoor Air Economizer Type
    ,                        !- Heat Recovery Type
    ,                        !- Sensible Heat Recovery Effectiveness {dimensionless}
    ;                        !- Latent Heat Recovery Effectiveness {dimensionless}
```

```
ThermostatSetpoint:DualSetpoint,
    Office Thermostat Dual SP Control,  !- Name
    ALWAYS 20,              !- Heating Setpoint Temperature Schedule Name
    ALWAYS 24;              !- Cooling Setpoint Temperature Schedule Name

Output:Variable,*,Site Outdoor Air Drybulb Temperature,Timestep;
Output:Variable,*,Zone Air Temperature,Timestep;
Output:Surfaces:Drawing,DXF;
Output:Constructions,Constructions;
Output:VariableDictionary,Regular;
```

Appendix E EnergyPlus FMU Incorporated IDF Model

epToHLA.idf

```
!-Generator IDFEditor 1.49
!-Option OriginalOrderTop UseSpecialFormat

!-NOTE: All comments with '!-' are ignored by the IDFEditor and are generated
automatically.
!-      Use '!' comments if they need to be retained when using the IDFEditor.

Output:Variable,myStartCooling,Schedule Value ,Timestep;
Output:Variable,myStartHeating,Schedule Value,Timestep;
Output:Meter,Heating:EnergyTransfer:Zone:ZONE ONE ,Timestep;
Output:Meter,Cooling:EnergyTransfer:Zone:ZONE ONE ,Timestep;

ZoneControl:Thermostat,
    myZ1Thermo,              !- Name
    ZONE ONE,                !- Zone or ZoneList Name
    ALWAYS 4,                !- Control Type Schedule Name
    ThermostatSetpoint:DualSetpoint,  !- Control 1 Object Type
    myDualSetpoint;          !- Control 1 Name

ThermostatSetpoint:DualSetpoint,
    myDualSetpoint,          !- Name
    myStartHeating,          !- Heating Setpoint Temperature Schedule Name
    myStartCooling;          !- Cooling Setpoint Temperature Schedule Name

RunPeriod,
    myRunPeriod,             !- Name
    6,                       !- Begin Month
    1,                       !- Begin Day of Month
    6,                       !- End Month
    2,                       !- End Day of Month
    UseWeatherFile,          !- Day of Week for Start Day
    Yes,                     !- Use Weather File Holidays and Special Days
    Yes,                     !- Use Weather File Daylight Saving Period
    No,                      !- Apply Weekend Holiday Rule
    Yes,                     !- Use Weather File Rain Indicators
    Yes,                     !- Use Weather File Snow Indicators
    1,                       !- Number of Times Runperiod to be Repeated
    Yes;                     !- Increment Day of Week on repeat

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    ZONE ONE,                !- Output:Variable Index Key Name
    Zone Air Temperature ,   !- Output:Variable Name
    Joe_ep_fmu.fmu,          !- FMU File Name
    Joe_ep_fmu,              !- FMU Instance Name
    epSendZoneMeanAirTemp;   !- FMU Variable Name

ExternalInterface,
    FunctionalMockupUnitImport;  !- Name of External Interface
```

Page 1

```
ExternalInterface:FunctionalMockupUnitImport,
    Joe_ep_fmu.fmu,           !- FMU File Name
    0,                        !- FMU Timeout {ms}
    1;                        !- FMU LoggingOn

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    Environment,              !- Output:Variable Index Key Name
    Site Outdoor Air Drybulb Temperature ,  !- Output:Variable Name
    Joe_ep_fmu.fmu,           !- FMU File Name
    Joe_ep_fmu,               !- FMU Instance Name
    epSendOutdoorAirTemp;     !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
    myStartHeating,           !- Name
    myTemps,                  !- Schedule Type Limits Names
    Joe_ep_fmu.fmu,           !- FMU File Name
    Joe_ep_fmu,               !- FMU Instance Name
    epGetStartHeating,        !- FMU Variable Name
    17;                       !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
    myStartCooling,           !- Name
    myTemps,                  !- Schedule Type Limits Names
    Joe_ep_fmu.fmu,           !- FMU File Name
    Joe_ep_fmu,               !- FMU Instance Name
    epGetStartCooling,        !- FMU Variable Name
    30;                       !- Initial Value

ScheduleTypeLimits,
    myTemps,                  !- Name
    ,                         !- Lower Limit Value
    ,                         !- Upper Limit Value
    Continuous,               !- Numeric Type
    Temperature;              !- Unit Type

! Introduction to EnergyPlus - Exercise 1A
!
! Building: Fictional 1 zone building with lightweight walls and 2 windows.
!           8m x 6m x 2.7m high, long side facing N and S
!           20C heating, 24C cooling
! Internal: None.
! System:   Purchased Air.
! Plant:    None.
! Environment:  Chicago, IL, USA, Summer and Winter design days
!
!
Version,8.7;
```

```
Building,
    Exercise 1A,             !- Name
    0.0,                     !- North Axis {deg}
    Country,                 !- Terrain
    0.04,                    !- Loads Convergence Tolerance Value
    0.4,                     !- Temperature Convergence Tolerance Value {deltaC}
    FullInteriorAndExterior, !- Solar Distribution
    ,                        !- Maximum Number of Warmup Days
    6;                       !- Minimum Number of Warmup Days

Timestep,4;
SurfaceConvectionAlgorithm:Inside,TARP;
SurfaceConvectionAlgorithm:Outside,TARP;
HeatBalanceAlgorithm,ConductionTransferFunction;

ShadowCalculation,
    AverageOverDaysInFrequency,  !- Calculation Method
    20;                      !- Calculation Frequency

SimulationControl,
    No,                      !- Do Zone Sizing Calculation
    No,                      !- Do System Sizing Calculation
    No,                      !- Do Plant Sizing Calculation
    Yes,                     !- Run Simulation for Sizing Periods
    Yes;                     !- Run Simulation for Weather File Run Periods

Site:Location,
    CHICAGO_IL_USA TMY2-94846,  !- Name
    41.78000,                !- Latitude {deg}
    -87.75000,               !- Longitude {deg}
    -6.000000,               !- Time Zone {hr}
    190.0000;                !- Elevation {m}

! CHICAGO_IL_USA Heating 99.6%, MaxDB= -21.20 Wind Speed=   4.60 Wind Dir= 270.00
SizingPeriod:DesignDay,
    CHICAGO_IL_USA Heating 99.6% Conditions,  !- Name
    1,                       !- Month
    21,                      !- Day of Month
    WinterDesignDay,         !- Day Type
    -21.20000,               !- Maximum Dry-Bulb Temperature {C}
    0.0,                     !- Daily Dry-Bulb Temperature Range {deltaC}
    ,                        !- Dry-Bulb Temperature Range Modifier Type
    ,                        !- Dry-Bulb Temperature Range Modifier Day Schedule
Name
    Wetbulb,                 !- Humidity Condition Type
    -21.20000,               !- Wetbulb or DewPoint at Maximum Dry-Bulb {C}
    ,                        !- Humidity Condition Day Schedule Name
```

```
    ,                        !- Humidity Ratio at Maximum Dry-Bulb
{kgWater/kgDryAir}
    ,                        !- Enthalpy at Maximum Dry-Bulb {J/kg}
    ,                        !- Daily Wet-Bulb Temperature Range {deltaC}
    99063.21,                !- Barometric Pressure {Pa}
    4.600000,                !- Wind Speed {m/s}
    270.0000,                !- Wind Direction {deg}
    No,                      !- Rain Indicator
    No,                      !- Snow Indicator
    No,                      !- Daylight Saving Time Indicator
    ASHRAEClearSky,          !- Solar Model Indicator
    ,                        !- Beam Solar Day Schedule Name
    ,                        !- Diffuse Solar Day Schedule Name
    ,                        !- ASHRAE Clear Sky Optical Depth for Beam Irradiance
(taub) {dimensionless}
    ,                        !- ASHRAE Clear Sky Optical Depth for Diffuse
Irradiance (taud) {dimensionless}
    0.0;                     !- Sky Clearness

Site:GroundTemperature:BuildingSurface,18.3,18.2,18.3,18.4,20.1,22.0,22.3,22.5,22.5,
20.7,18.9,18.5;

Material,
    PLASTERBOARD-1,          !- Name
    MediumSmooth,            !- Roughness
    0.01200,                 !- Thickness {m}
    0.16000,                 !- Conductivity {W/m-K}
    950.000,                 !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    FIBERGLASS QUILT-1,      !- Name
    Rough,                   !- Roughness
    0.066,                   !- Thickness {m}
    0.040,                   !- Conductivity {W/m-K}
    12.000,                  !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    WOOD SIDING-1,           !- Name
    Rough,                   !- Roughness
    0.00900,                 !- Thickness {m}
```

```
    0.14000,                 !- Conductivity {W/m-K}
    530.000,                 !- Density {kg/m3}
    900.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    PLASTERBOARD-2,          !- Name
    Rough,                   !- Roughness
    0.01000,                 !- Thickness {m}
    0.16000,                 !- Conductivity {W/m-K}
    950.000,                 !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    FIBERGLASS QUILT-2,      !- Name
    Rough,                   !- Roughness
    0.1118,                  !- Thickness {m}
    0.040,                   !- Conductivity {W/m-K}
    12.000,                  !- Density {kg/m3}
    840.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    ROOF DECK,               !- Name
    Rough,                   !- Roughness
    0.01900,                 !- Thickness {m}
    0.14000,                 !- Conductivity {W/m-K}
    530.000,                 !- Density {kg/m3}
    900.00,                  !- Specific Heat {J/kg-K}
    0.900000,                !- Thermal Absorptance
    0.600000,                !- Solar Absorptance
    0.600000;                !- Visible Absorptance

Material,
    HF-C5,                   !- Name
    MediumRough,             !- Roughness
    0.1015000,               !- Thickness {m}
    1.729600,                !- Conductivity {W/m-K}
    2243.000,                !- Density {kg/m3}
    837.0000,                !- Specific Heat {J/kg-K}
    0.9000000,               !- Thermal Absorptance
```

```
    0.6500000,              !- Solar Absorptance
    0.6500000;              !- Visible Absorptance

Construction,
    LTWALL,                 !- Name
    WOOD SIDING-1,          !- Outside Layer
    FIBERGLASS QUILT-1,     !- Layer 2
    PLASTERBOARD-1;         !- Layer 3

Construction,
    LTFLOOR,                !- Name
    HF-C5;                  !- Outside Layer

Construction,
    LTROOF,                 !- Name
    ROOF DECK,              !- Outside Layer
    FIBERGLASS QUILT-2,     !- Layer 2
    PLASTERBOARD-2;         !- Layer 3

Zone,
    ZONE ONE,               !- Name
    0,                      !- Direction of Relative North {deg}
    0, 0, 0,                            !- X,Y,Z  {m}
    1,                      !- Type
    1,                      !- Multiplier
    2.7000,                 !- Ceiling Height {m}
    129.6;                  !- Volume {m3}

GlobalGeometryRules,
    UpperLeftCorner,        !- Starting Vertex Position
    Counterclockwise,       !- Vertex Entry Direction
    WorldCoordinateSystem;  !- Coordinate System

BuildingSurface:Detailed,
    SURFACE NORTH,          !- Name
    Wall,                   !- Surface Type
    LTWALL,                 !- Construction Name
    ZONE ONE,               !- Zone Name
    Outdoors,               !- Outside Boundary Condition
    ,                       !- Outside Boundary Condition Object
    SunExposed,             !- Sun Exposure
    WindExposed,            !- Wind Exposure
    0.50,                   !- View Factor to Ground
    4,                      !- Number of Vertices
    8.00, 6.00, 2.70,                   !- X,Y,Z  1 {m}
    8.00, 6.00, 0,                      !- X,Y,Z  2 {m}
    0, 6.00, 0,                         !- X,Y,Z  3 {m}
    0, 6.00, 2.70;                      !- X,Y,Z  4 {m}
```

```
BuildingSurface:Detailed,
    ZONE SURFACE EAST,       !- Name
    Wall,                    !- Surface Type
    LTWALL,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0.50,                    !- View Factor to Ground
    4,                       !- Number of Vertices
    8.00, 0, 2.70,                  !- X,Y,Z  1 {m}
    8.00, 0, 0,                     !- X,Y,Z  2 {m}
    8.00, 6.00, 0,                  !- X,Y,Z  3 {m}
    8.00, 6.00, 2.70;               !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE SOUTH,      !- Name
    Wall,                    !- Surface Type
    LTWALL,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0.50,                    !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 0, 2.70,                     !- X,Y,Z  1 {m}
    0, 0, 0,                        !- X,Y,Z  2 {m}
    8.00, 0, 0,                     !- X,Y,Z  3 {m}
    8.00, 0, 2.70;                  !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE WEST,       !- Name
    Wall,                    !- Surface Type
    LTWALL,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0.50,                    !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 6.00, 2.70,                  !- X,Y,Z  1 {m}
    0, 6.00, 0,                     !- X,Y,Z  2 {m}
    0, 0, 0,                        !- X,Y,Z  3 {m}
    0, 0, 2.70;                     !- X,Y,Z  4 {m}
```

```
BuildingSurface:Detailed,
    ZONE SURFACE FLOOR,      !- Name
    Floor,                   !- Surface Type
    LTFLOOR,                 !- Construction Name
    ZONE ONE,                !- Zone Name
    Ground,                  !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    NoSun,                   !- Sun Exposure
    NoWind,                  !- Wind Exposure
    0,                       !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 0, 0,                            !- X,Y,Z  1 {m}
    0, 6.00, 0,                         !- X,Y,Z  2 {m}
    8.00, 6.00, 0,                      !- X,Y,Z  3 {m}
    8.00, 0, 0;                         !- X,Y,Z  4 {m}

BuildingSurface:Detailed,
    ZONE SURFACE ROOF,       !- Name
    Roof,                    !- Surface Type
    LTROOF,                  !- Construction Name
    ZONE ONE,                !- Zone Name
    Outdoors,                !- Outside Boundary Condition
    ,                        !- Outside Boundary Condition Object
    SunExposed,              !- Sun Exposure
    WindExposed,             !- Wind Exposure
    0,                       !- View Factor to Ground
    4,                       !- Number of Vertices
    0, 6.00, 2.70,                      !- X,Y,Z  1 {m}
    0, 0, 2.70,                         !- X,Y,Z  2 {m}
    8.00, 0, 2.70,                      !- X,Y,Z  3 {m}
    8.00, 6.00, 2.70;                   !- X,Y,Z  4 {m}

ScheduleTypeLimits,
    Any Number;              !- Name

Schedule:Compact,
    ALWAYS 4,                !- Name
    Any Number,              !- Schedule Type Limits Name
    Through: 12/31,          !- Field 1
    For: AllDays,            !- Field 2
    Until: 24:00, 4;         !- Field 4

Schedule:Compact,
    ALWAYS 20,               !- Name
    Any Number,              !- Schedule Type Limits Name
    Through: 12/31,          !- Field 1
    For: AllDays,            !- Field 2
```

```
    Until: 24:00, 20;        !- Field 4

Schedule:Compact,
    ALWAYS 24,               !- Name
    Any Number,              !- Schedule Type Limits Name
    Through: 12/31,          !- Field 1
    For: AllDays,            !- Field 2
    Until: 24:00, 24;        !- Field 4

ZoneHVAC:EquipmentConnections,
    ZONE ONE,                !- Zone Name
    ZONE ONE Equipment,      !- Zone Conditioning Equipment List Name
    ZONE ONE Supply Inlet,   !- Zone Air Inlet Node or NodeList Name
    ,                        !- Zone Air Exhaust Node or NodeList Name
    ZONE ONE Zone Air Node,  !- Zone Air Node Name
    ZONE ONE Return Outlet;  !- Zone Return Air Node Name

ZoneHVAC:EquipmentList,
    ZONE ONE Equipment,      !- Name
    ZoneHVAC:IdealLoadsAirSystem,  !- Zone Equipment 1 Object Type
    ZONE ONE Purchased Air,  !- Zone Equipment 1 Name
    1,                       !- Zone Equipment 1 Cooling Sequence
    1;                       !- Zone Equipment 1 Heating or No-Load Sequence

ZoneHVAC:IdealLoadsAirSystem,
    ZONE ONE Purchased Air,  !- Name
    ,                        !- Availability Schedule Name
    ZONE ONE Supply Inlet,   !- Zone Supply Air Node Name
    ,                        !- Zone Exhaust Air Node Name
    50,                      !- Maximum Heating Supply Air Temperature {C}
    13,                      !- Minimum Cooling Supply Air Temperature {C}
    0.015,                   !- Maximum Heating Supply Air Humidity Ratio
{kgWater/kgDryAir}
    0.01,                    !- Minimum Cooling Supply Air Humidity Ratio
{kgWater/kgDryAir}
    NoLimit,                 !- Heating Limit
    ,                        !- Maximum Heating Air Flow Rate {m3/s}
    ,                        !- Maximum Sensible Heating Capacity {W}
    NoLimit,                 !- Cooling Limit
    ,                        !- Maximum Cooling Air Flow Rate {m3/s}
    ,                        !- Maximum Total Cooling Capacity {W}
    ,                        !- Heating Availability Schedule Name
    ,                        !- Cooling Availability Schedule Name
    ConstantSupplyHumidityRatio,  !- Dehumidification Control Type
    ,                        !- Cooling Sensible Heat Ratio {dimensionless}
    ConstantSupplyHumidityRatio,  !- Humidification Control Type
    ,                        !- Design Specification Outdoor Air Object Name
    ,                        !- Outdoor Air Inlet Node Name
```

```
    ,                        !- Demand Controlled Ventilation Type
    ,                        !- Outdoor Air Economizer Type
    ,                        !- Heat Recovery Type
    ,                        !- Sensible Heat Recovery Effectiveness {dimensionless}
    ;                        !- Latent Heat Recovery Effectiveness {dimensionless}

ThermostatSetpoint:DualSetpoint,
    Office Thermostat Dual SP Control,  !- Name
    ALWAYS 20,               !- Heating Setpoint Temperature Schedule Name
    ALWAYS 24;               !- Cooling Setpoint Temperature Schedule Name

Output:Variable,*,Site Outdoor Air Drybulb Temperature,Timestep;
Output:Variable,*,Zone Air Temperature,Timestep;
Output:Surfaces:Drawing,DXF;
Output:Constructions,Constructions;
Output:VariableDictionary,Regular;
```

Appendix F FMU XML File

```xml
<?xml version="1.0"?>
<fmiModelDescription
    fmiVersion="1.0"
    modelName="Joe ep fmu"
    modelIdentifier="Joe_ep_fmu"
    guid="{818642F1-D7D4-4DC7-8549-554862454199}"
    variableNamingConvention="structured"
    numberOfContinuousStates="0"
    numberOfEventIndicators="0">
    <!-- Have to be same as the name in C code-->

  <ModelVariables>
    <ScalarVariable
        name="epSendZoneMeanAirTemp"
        valueReference="1"
        causality="input">
        <!-- name has to match E+ "FMU Variable Name" -->
        <!-- valueReference must be unique -->
        <!-- input/output to/from the slave -->
      <Real />
    </ScalarVariable>
    <ScalarVariable
        name="epSendOutdoorAirTemp"
        valueReference="2"
        causality="input">
      <Real />
    </ScalarVariable>
    <ScalarVariable
        name="epGetStartHeating"
        valueReference="3"
        causality="output">
      <Real />
    </ScalarVariable>
    <ScalarVariable
        name="epGetStartCooling"
        valueReference="4"
        causality="output">
      <Real />
    </ScalarVariable>
  </ModelVariables>

  <Implementation>
    <CoSimulation_StandAlone>
      <Capabilities canBeInstantiatedOnlyOncePerProcess="true"
      canNotUseMemoryManagementFunctions="true" />
    </CoSimulation_StandAlone>
  </Implementation>
</fmiModelDescription>
```