Hector E. Perez
Eric Burger
CE 290-2 Energy Systems and Control
Spring 2014 – Project Report

Cloud Enabled Model Predictive Control of a Home Heating System

Abstract

This work creates a smart cloud-enabled thermostat for an electric home heating system. The objective is to optimize the operation of heaters to minimize the cost of electricity and maintain a comfortable temperature based on time-varying weather forecasts and electricity price data. This involves modeling and controlling the home through sensor deployment, Internet-based control system design, and smartphone application development. The goals of this project are to: a) monitor the real-time thermal conditions in an apartment, b) reduce peak loads on the power grid, c) make use of weather forecast data to predict heating requirements, and d) enable smart home technology development.

Introduction and Motivation

According to the U.S. Department of Energy's "Buildings Energy Data Book" and based on data from 2010, commercial and residential buildings account for 41% of U.S. primary energy consumption [1]. This is more than either the transportation sector or the industrial sector, which consume 29% and 30% of primary energy, respectively. Heating, ventilation, and air-conditioning (HVAC) compose 43% of commercial and 54% of residential building site energy end-use. As shown in Figure 1 below, space heating is the most significant site energy end-use type for both residential and commercial buildings. Accordingly, to reduce peak loads on the power grid, this work focuses on predicting and controlling the space heating demand in buildings.

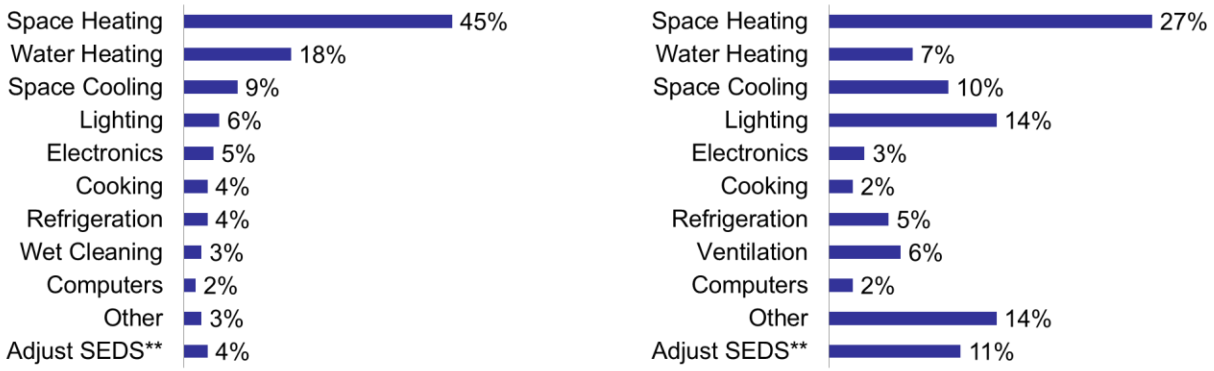

Figure 1. Residential (Left) and Commercial (Right) Site Energy Consumption by End-Use Type [1]
****Adjustment to State Energy Data Systems**

A common method for residential space heating is the utilization of electric resistance heaters. Such systems are cheap to produce but costly to operate due to poor primary to end-use energy efficiency (about 30%) and the varying cost of electricity [2]. However, because homeowners

often pay time-of-use electricity rates, it is possible to know the price of electricity a priori and to use the price as a proxy for grid demand. By forecasting the space heating demand and incorporating the electricity price schedule, residential space heating can be controlled in a way that reduces electricity costs and peak loads on the power grid.

This project is also motivated by existing research in the modeling and control of thermal zones. The work in [6] and [7] propose frameworks for online estimation of states and unknown parameters of buildings using Extended and Unscented Kalman filters. Both papers employ RC-circuit representations of conductive and convective heat transfer. [6] chooses to model the radiative heat absorption of each wall and to represent the radiative heat transfer to ambient using the Stefan-Boltzmann law (in addition to a time-varying disturbance). By contrast, [7] lumps all non-RC terms along with time-varying disturbances. A model predictive controller is implemented in [8] by optimizing the HVAC system to minimize total energy consumption, peak power consumption, and total comfort violations. This project focuses on identifying a simple thermal model of an apartment through the use of modern sensing technologies, and implementing a model predictive control system in the cloud.

To research the potential to reduce electricity costs and peak loads on the power grid, this project a) deployed a wireless network of temperature sensors in an apartment with an electric heating system, b) recorded the indoor temperature, outdoor temperature, heater state (On/Off), and weather conditions for 3 days, c) identified and validated the thermal parameters of the space, d) implemented an Internet-based control system with a discrete-time thermal model of the apartment and a model predictive control algorithm that minimizes the cost of heating, e) developed a smartphone application to monitor the behavior of the system, and f) simulated the control system for a period of 3 days.

Sensor Deployment Study

The project employed six sensor nodes that were assembled and deployed in a home environment (represented by the red dots in Figure 2 below) to analyze the spatial temperature heterogeneity. The electric baseboard heaters are represented by the blue rectangles in Figure 2.

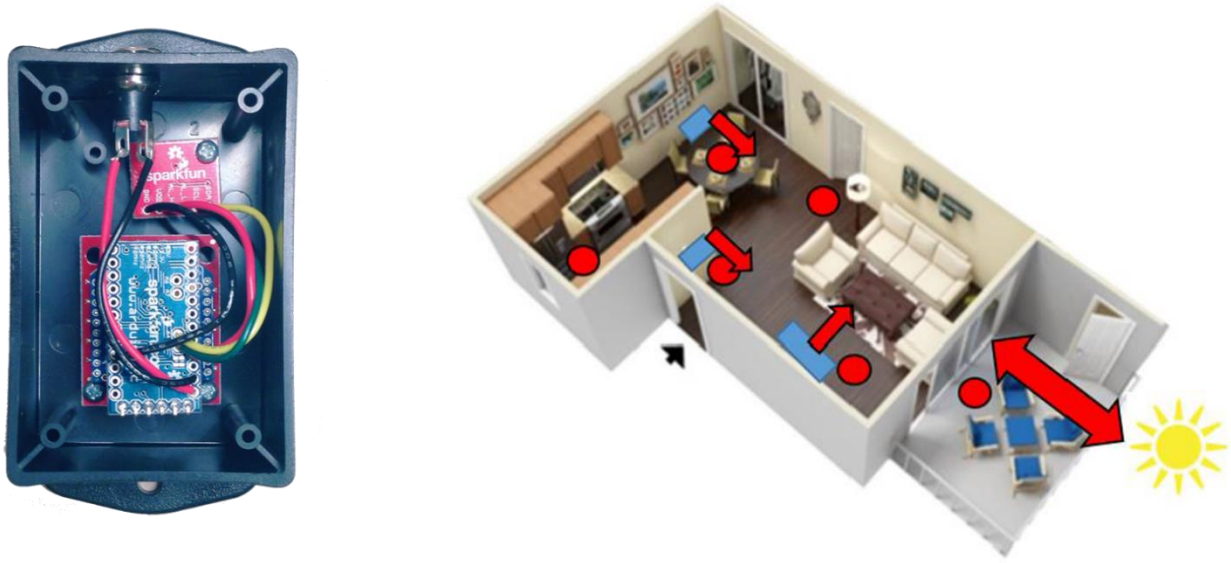

Figure 2. Sensor Node (Left) and Node locations in Home Layout (Right)

Each sensor node consisted of a temperature sensor, microcontroller, radio regulator board, and wireless radio, as shown in Figure 3 below. The components were soldered together by the team to form the sensor node shown in Figure 2 above.

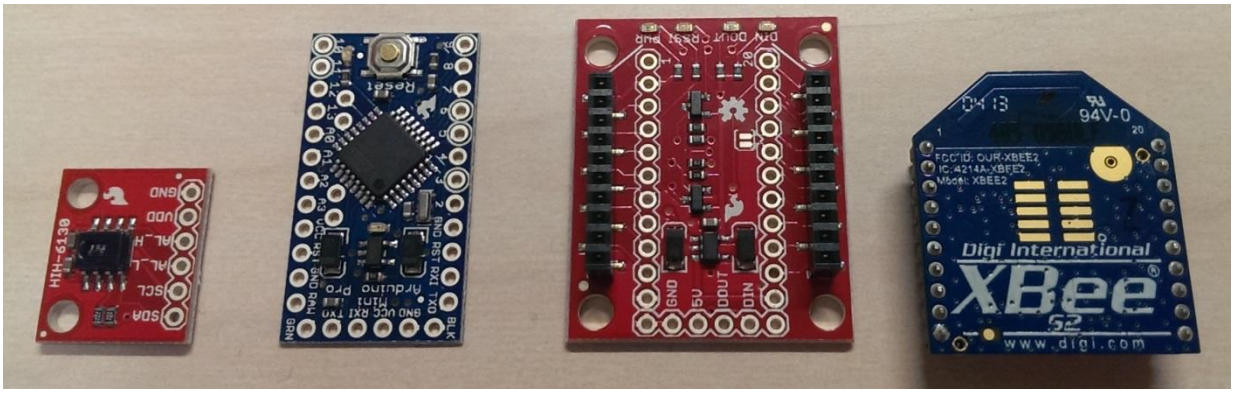

Figure 3. Honeywell Temperature Sensor (Left), Arduino Pro-Mini Microcontroller (Middle-Left), Radio Regulator Board (Middle-Right), XBee ZigBee Radio (Right)

The project employed Honeywell HIH-6130 humidity/temperature sensors with a resolution of 14 bits, accuracy of ±1ºC, and an operating range of 5 to 50ºC. The sensor produces a digital I2C output that is read by an Arduino Pro-Mini 3.3v microcontroller. The microcontroller is programmed to retrieve the temperature reading once per minute and prepare the data to be sent to the radio. The XBee S2 radio is connected to the Arduino microcontroller through the

regulator board. The XBee radio uses the ZigBee protocol to transmit the temperature data via a 2.4 GHz radio frequency to an Internet-connected computer.

To record the heater state (On/Off) during the study, the team built a switch consisting of a toggle switch, LED, Arduino Uno microcontroller, a radio regulator board, and an XBee radio, as shown in Figure 4 below. When the heater was turned on or off, the toggle switch was set accordingly. The Arduino microcontroller was programmed to detect the switch state and send the information via the XBee radio to the Internet-connected computer.

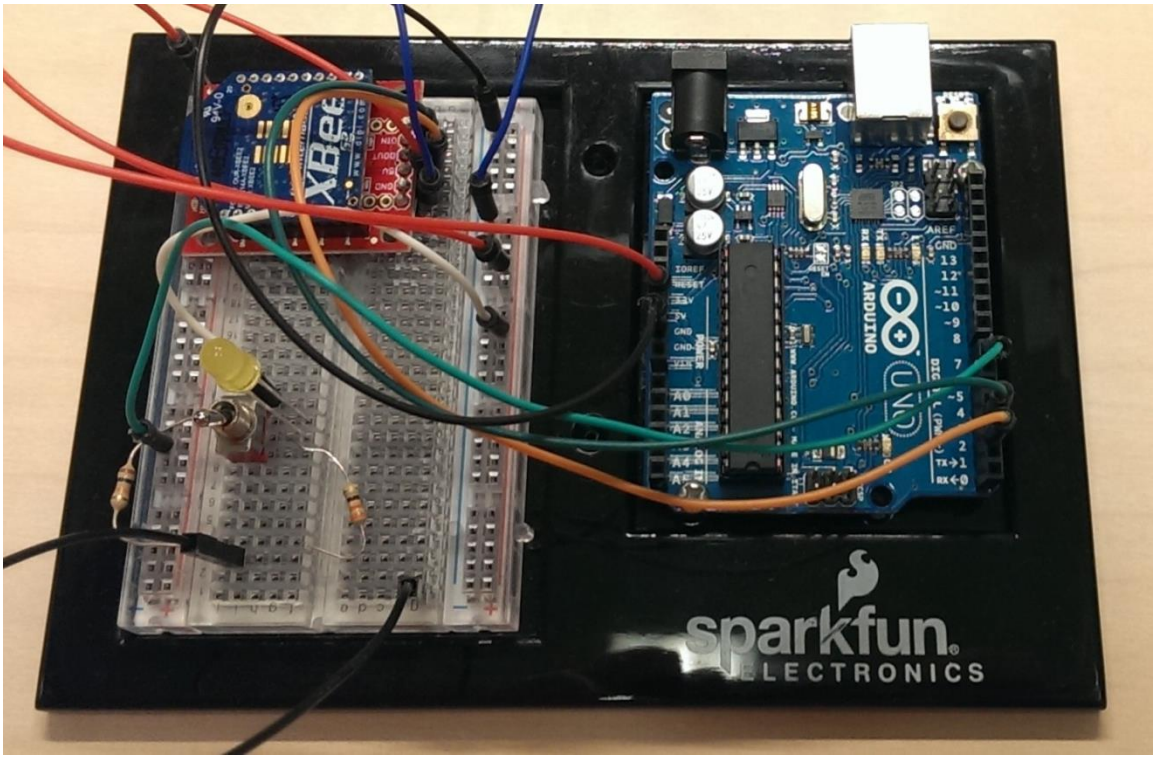

Figure 4. Switch for Recording the Heater State (On/Off)

The Internet-connected computer, as shown in Figure 5 below, consisted of a BeagleBone Black Linux ARM computer and an XBee S2 radio. The computer was programmed to collect the data being transmitted by the sensor nodes and the switch. The data was stored locally on the computer and sent to the Xively database service to be stored on the cloud. In our internet control system design, the Internet-connected computer would also act as the home's thermostat. However, in this sensor deployment study, the computer is only used to collect the data from the sensor nodes and the switch.

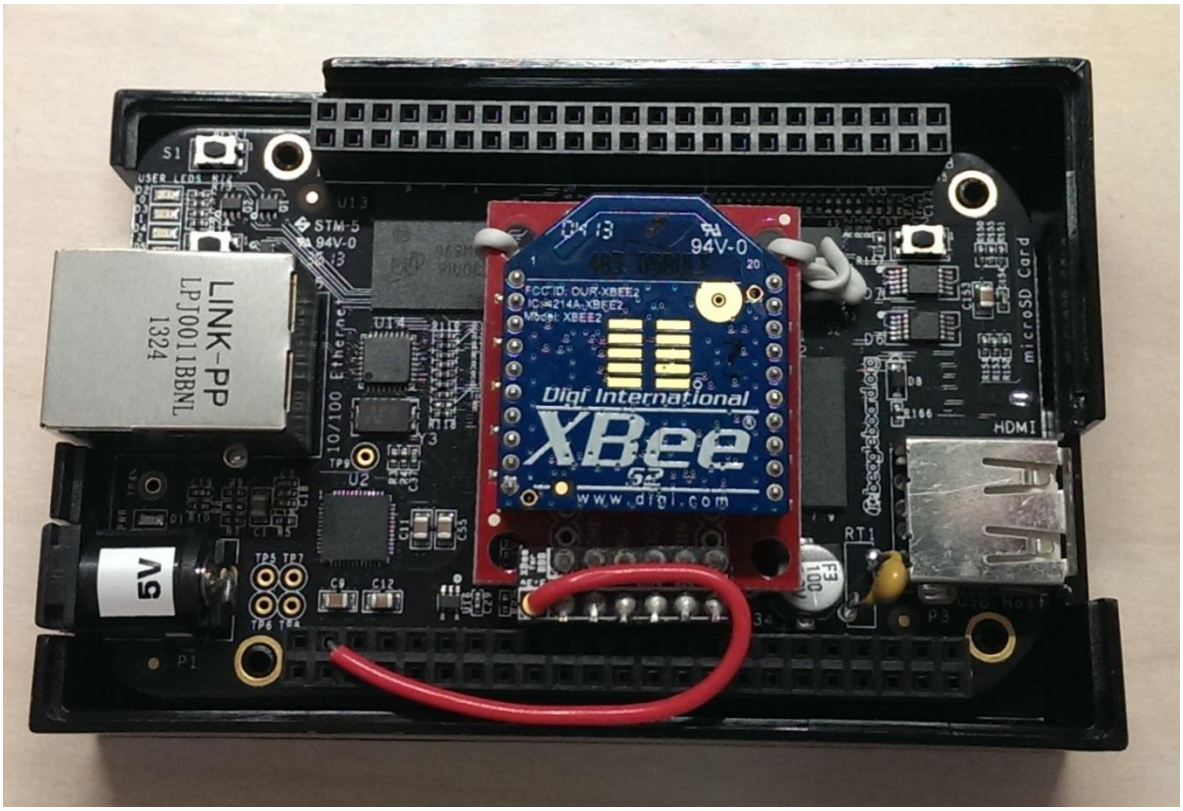

Figure 5. Internet-Connected Computer (XBee radio, Regulator Board, and BeagleBone Black ARM Computer)

For this study we focused on the temperature in the living room, kitchen, and common area. We also took a measurement of the outside temperature for comparison to the weather forecast data extracted from the Wunderground API. Note that the pictured furniture layout is just shown for representative purposes, the actual layout is slightly different. Each sensor node has a unique channel ID number, are shown in the table below.

Location	Outside	Front	Middle	Side	Back	Kitchen
Channel ID	40A8BEDE	40AA17DC	40A8C307	40A8BEB2	40A8BEE3	40A8C291

Table 1. Sensor ID

The data was taken for approximately 3 days and is presented in the Figure 6 below.

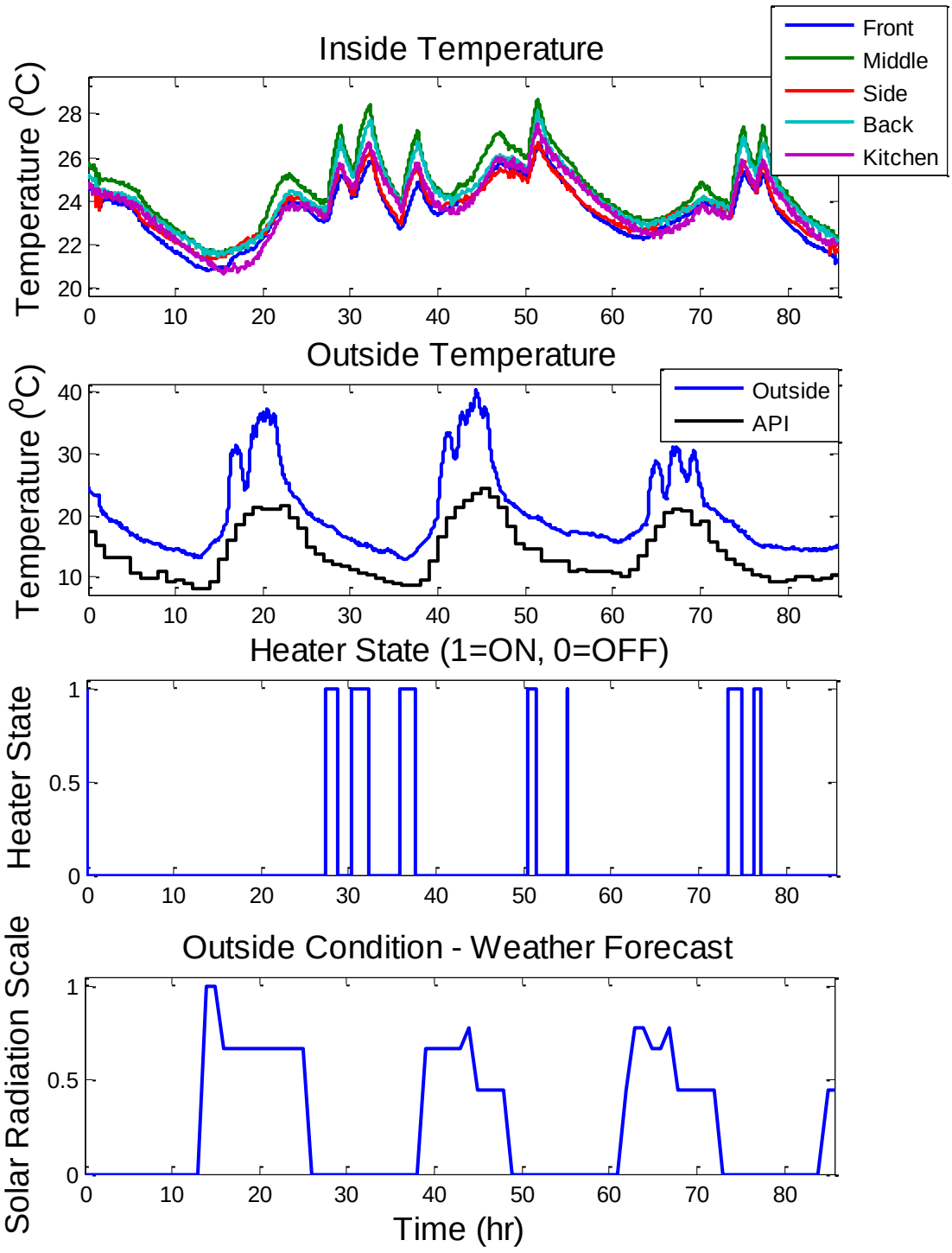

Figure 6. Sensor Deployment Data:
Inside (Top), Outside (Middle-Top), Heater State (Middle-Bottom), and Solar Radiation (Bottom)

We note from the top plot of the inside measurement data that the temperatures are within 2.5 degrees Celsius. The highest temperature comes from the sensor nearest to the light labeled as MIDDLE, followed by the sensor in the back close to the rear heater labeled BACK. They are probably highest due to their distance from a window or a door. The lowest temperatures come

from the sensors labeled FRONT, KITCHEN, and SIDE. They are probably lowest due to their proximity to a window or a door. Our observations are that the temperature in the apartment is colder closer to a window or a door, and the temperature gets hotter as one moves away from the window or door (eg. Interior). We will use the FRONT temperature data for identification and validation purposes since that is assumed to be the location where the occupant spends most of their time when present in the home.

As shown in the outside temperature plot in Figure 6 above, the sensor labeled OUTSIDE reported temperature readings significantly higher than the Wunderground Weather API. Additionally, we observe rapid changes in the OUTSIDE reading that do not appear in the Wunderground data. For these reasons, we believe that the OUTSIDE data is heavily polluted by solar gain and have chosen not to use the data from the OUTSIDE sensor for system identification or control. Instead, we use the outside temperature measurement along with the weather conditions from the Wunderground Weather API for identification and validation purposes.

Thermal Model Identification

Thermal Model

The dynamics of an apartment can be modeled by the heat transfer between the interior and the environment, solar radiation from the sun to the interior of the home, as well as the heat added by the electric baseboard heaters. Radiative heat transfer from the ambient will not be considered. Therefore, the change in temperature within the apartment can be represented by the state equation

$$\dot{T}(t) = -\frac{1}{RC}T(t) + \frac{1}{RC}T_O(t) + \frac{1}{C}P_{Heat}s(t) + \frac{1}{C}\tau_W A_W Q_{Solar} d(t)$$

where $T(t), s(t), T_O(t), d(t)$ are the indoor temperature (state), heater state (input), outdoor temperature (disturbance input), and solar radiation scale (disturbance input), respectively. The parameters used in this model are $R(^oC/kW),\ C(kJ/^oC), P_{Heat}\ (kW), \tau_W,\ A_W(m^2),\ Q_{Solar}\ (W/m^2)$ which represent the thermal resistance, thermal capacitance, baseboard heater power, transmittance factor of the glass window, window area, and maximum solar radiation, respectively.

Parametric Modeling

We reformulate the temperature dynamics equation into a linear in the parameters form for identification purposes

The linear in the parameters model is derived as followed

$$\dot{T}(t) = -\frac{1}{RC}T(t) + \frac{1}{RC}T_O(t) + \frac{1}{C}P_{Heat}s(t) + \frac{1}{C}\tau_W A_W Q_{Solar} d(t)$$

$$\dot{T}(t) = \frac{1}{RC}\left(T_O(t) - T(t)\right) + \frac{1}{C}\left(P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t)\right)$$

Let

$$z(t) = \dot{T}(t)\,, \theta = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = \begin{bmatrix}\frac{1}{RC} \\ \frac{1}{C}\end{bmatrix}, \emptyset = \begin{bmatrix}\emptyset_1 \\ \emptyset_2\end{bmatrix} = \begin{bmatrix}T_O(t) - T(t) \\ P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t)\end{bmatrix}$$

Where

$$\theta_1 = \frac{1}{RC}\,, \theta_2 = \frac{1}{C}\,, \emptyset_1 = T_O(t) - T(t)\,, \emptyset_2 = P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t)$$

The dynamic equation becomes

$$z(t) = \theta_1 * \emptyset_1 + \theta_2 * \emptyset_2$$

$$z(t) = [\theta_1 \quad \theta_2]\begin{bmatrix}\emptyset_1 \\ \emptyset_2\end{bmatrix}$$

Which is

$$z(t) = \theta^T \emptyset$$

Input Parameters
Throughout this work we use the following parameters

$$P_{Heat} = 1.5kW$$
$$\tau_W A_W Q_{Solar} = 0.46kW$$

When the sun is out, the input $d(t)$ is equal to a normalized scaled number based on the weather condition. When the sun is down (eg. at Night), $d(t) = 0$. The values of $d(t)$ are summarized in the table below.

Weather Condition		Scale	d
Day	Clear	4.5	1
	Scattered Clouds	3.5	0.778
	Partly Cloudy	3	0.667
	Mostly Cloudy	2	0.444
	Overcast	0.5	0.111
Night	Night	0	0.000

Table 2. Solar Radiation Scale

Identification Algorithm
A normalized recursive gradient update law is given by

$$\frac{d}{dt}\hat{\theta}(t) = \Gamma\epsilon(t)\phi(t), \qquad \hat{\theta}(0) = \hat{\theta}_0,$$

$$\epsilon(t) = \frac{z(t) - \hat{\theta}^T(t)\phi(t)}{m^2(t)},$$

$$m^2(t) = 1 + \phi^T(t)\phi(t)$$

where $\Gamma = \Gamma^T > 0$ is a symmetric positive definite matrix, $\epsilon(t)$ is the normalized prediction error, and $m^2(t)$ is the normalization signal (which better conditions the signal).

Implementation

We implement the identification algorithm in MATLAB and the dataset used for identification is shown in Figure 7 below.

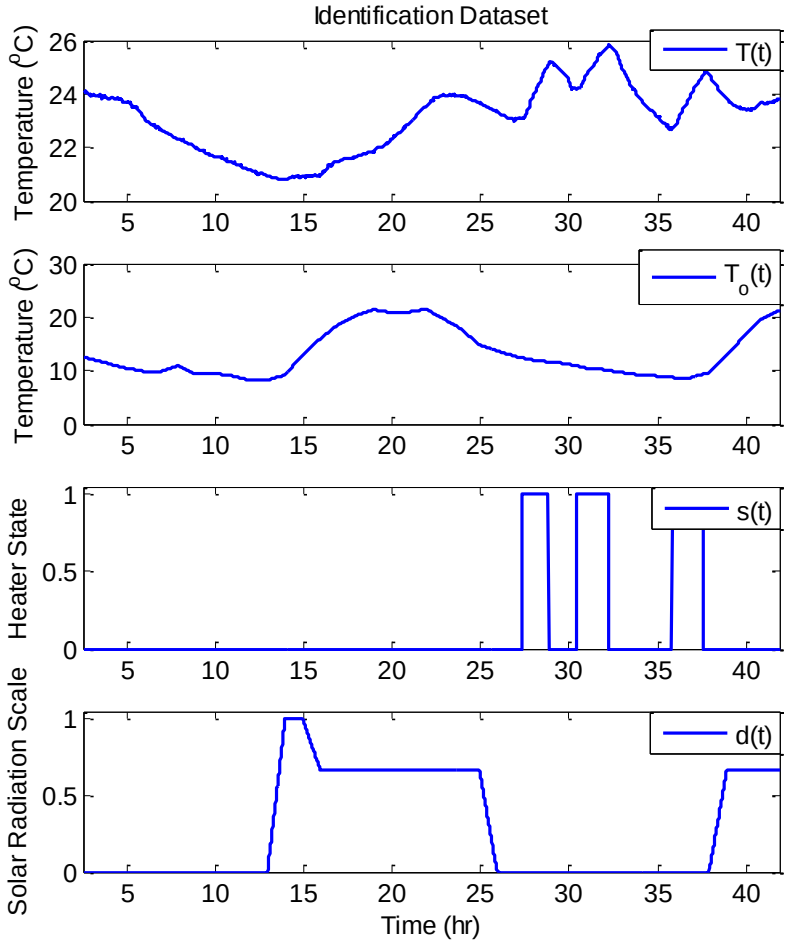

Figure 7. Identification Dataset:
Inside (Top), Outside (Middle Top), Heater State (Middle Bottom), and Solar Radiation (Bottom)

Let $\Gamma = 10^{-4.1} * I$, and the initial parameter guess $\hat{\theta}_1(0) = 0.014, \hat{\theta}_2(0) = 0.0003$. The resulting parameter estimates are shown in Figure 8 below

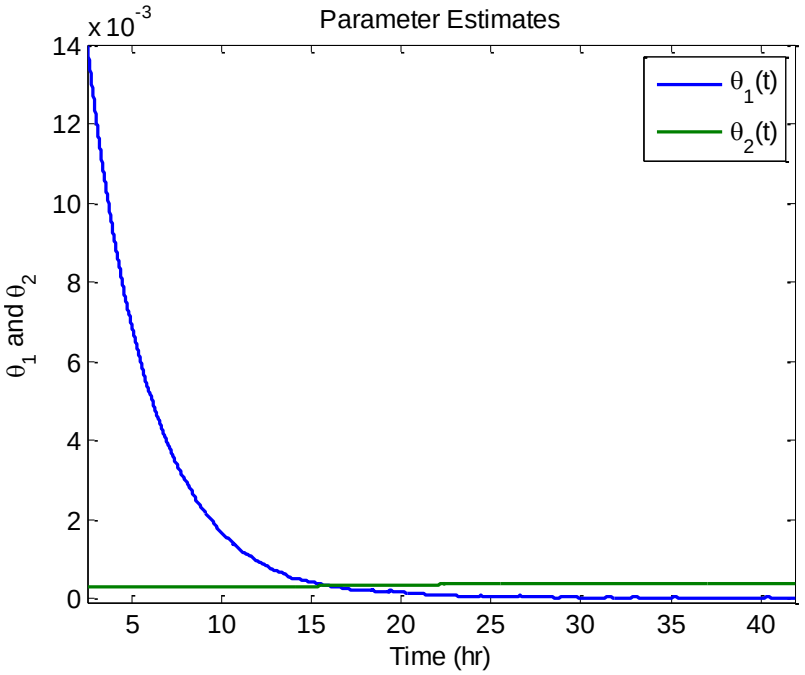

Figure 8. Parameter Estimation

The last parameters will be used for the model, and are

$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{bmatrix} = \begin{bmatrix} 9.6414 * 10^{-6} \\ 3.5590 * 10^{-4} \end{bmatrix}$$

Model Validation
Let's recall the following equation from earlier

$$\dot{T}(t) = \frac{1}{RC}\big(T_O(t) - T(t)\big) + \frac{1}{C}(P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t))$$

Which turns into

$$\dot{T}(t) = \theta_1\big(T_O(t) - T(t)\big) + \theta_2(P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t))$$

Reorganizing this in state space form we obtain the following

$$\dot{T}(t) = -\theta_1 T(t) + [\theta_1 \quad \theta_2] \begin{bmatrix} T_O(t) \\ P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

Where

$$A = -\theta_1$$

$$B = [\theta_1 \quad \theta_2]$$

$$u(t) = \begin{bmatrix} T_O(t) \\ P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

Since we are using the estimates obtained previously, this state space model can be rewritten as

$$\dot{T}(t) = -\hat{\theta}_1 T(t) + [\hat{\theta}_1 \quad \hat{\theta}_2] \begin{bmatrix} T_O(t) \\ P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

Where

$$A = -\hat{\theta}_1$$

$$B = [\hat{\theta}_1 \quad \hat{\theta}_2]$$

$$u(t) = \begin{bmatrix} T_O(t) \\ P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

Plugging in our estimates

$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{bmatrix} = \begin{bmatrix} 9.6414 * 10^{-6} \\ 3.5590 * 10^{-4} \end{bmatrix}$$

We obtain the following

$$\dot{T}(t) = -9.6414 * 10^{-6} T(t) + [9.6414 * 10^{-6} \quad 3.5590 * 10^{-4}] \begin{bmatrix} T_O(t) \\ P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

Where

$$A = -\hat{\theta}_1 = -9.6414 * 10^{-6}$$

$$B = [\hat{\theta}_1 \quad \hat{\theta}_2] = [9.6414 * 10^{-6} \quad 3.5590 * 10^{-4}]$$

$$u(t) = \begin{bmatrix} T_O(t) \\ P_{Heat}s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

Using these parameters in the model, it is now validated against a different dataset shown in Figure 9 below.

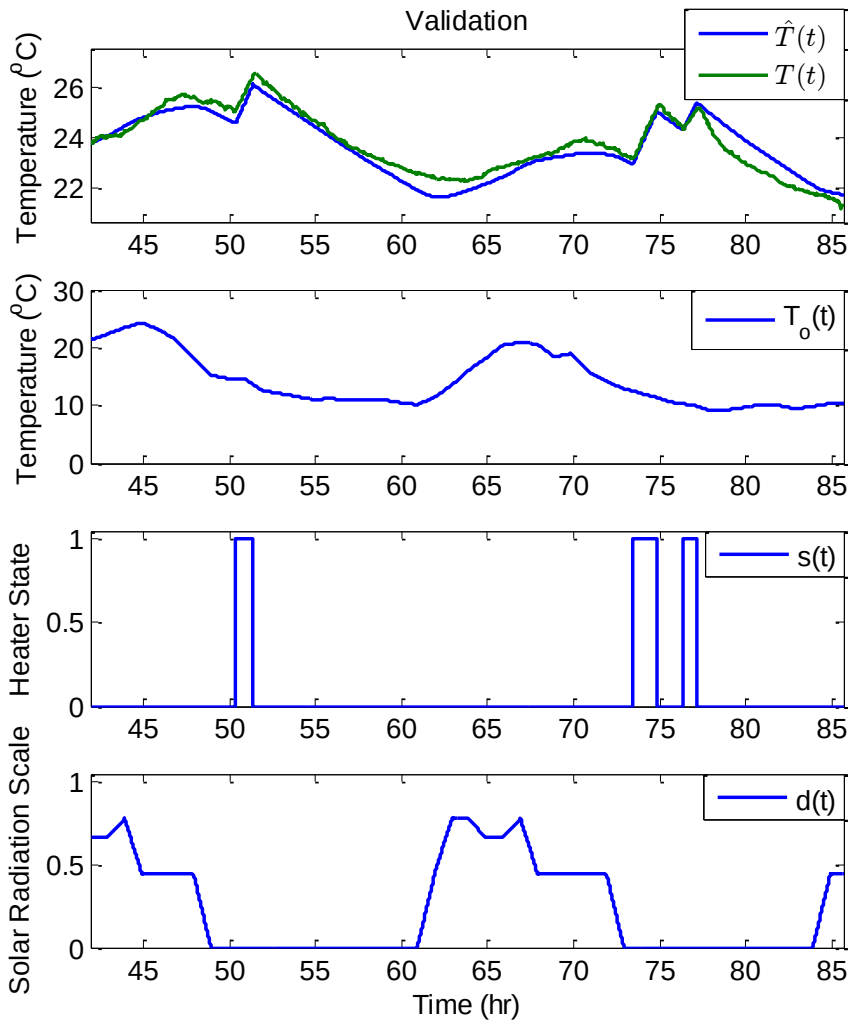

Figure 9. Validation Dataset:
Inside (Top), Outside (Middle Top), Heater State (Middle Bottom), and Solar Radiation (Bottom)

Model Discretization

Since the model we identified above is in the continuous time domain, we now transform the model into the discrete time domain for use in the optimization program.

Recall the continuous time domain model

$$\dot{T}(t) = -9.6414 * 10^{-6} T(t) + [9.6414 * 10^{-6} \quad 3.5590 * 10^{-4}] \begin{bmatrix} T_O(t) \\ P_{Heat} s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

Where

$$A = -9.6414 * 10^{-6}$$

$$B = [9.6414 * 10^{-6} \quad 3.5590 * 10^{-4}]$$

$$u(t) = \begin{bmatrix} T_O(t) \\ P_{Heat} s(t) + \tau_W A_W Q_{Solar} d(t) \end{bmatrix}$$

By using the following transformations, we obtain the discrete state matrices

$$A_d = e^{A\Delta t}$$

$$B_d = \left(\int_0^{\Delta t} e^{A\tau} d\tau\right) B$$

Since the A matrix is non-singular, we can find B_d as

$$B_d = A^{-1}(A_d - I)B$$

We solve for these discrete matrices with $\Delta t = 15\ min = 900\ sec$

$$A_d = 0.99136$$

$$B_d = [0.00864 \quad 0.31892]$$

The discrete time model (with $\Delta t = 15\ min$) is summarized as followed

$$T(k+1) = A_d T(k) + B_d u(k)$$

$$A_d = 0.99136$$

$$B_d = [0.00864 \quad 0.31892]$$

$$u(k) = \begin{bmatrix} T_O(k) \\ P_{Heat} s(k) + \tau_W A_W Q_{Solar} d(k) \end{bmatrix}$$

Where

$$P_{Heat} = 1.5kW$$
$$\tau_W A_W Q_{Solar} = 0.46kW$$

We reorganize this model in the following form

$$T(k+1) = A_d T(k) + B_d(2) P_{Heat} s(k) + B_d(1) T_O(k) + B_d(2) \tau_W A_W Q_{Solar} d(k)$$

We redefine the system matrices as follows

$$T(k+1) = A_d T(k) + B_s s(k) + B_O T_O(k) + B_w d(k)$$

Where

$$A_d = 0.99136$$
$$B_s = B_d(2) P_{Heat} = 0.31892 * 1.5 = 0.47838$$
$$B_O = B_d(1) = 0.00864$$
$$B_w = B_d(2) \tau_W A_W Q_{Solar} = 0.31892 * 0.46 = 0.1467$$

Optimization Formulation

The optimization program is formulated as follows:

$$\min_{s(k)} \sum_{k=0}^{N-1} c(k) P_{Heat} s(k) \frac{\Delta t}{3600}$$

Subject To

$$T(k+1) = A_d T(k) + B_s s(k) + B_O T_O(k) + B_w d(k) \quad for\ k = 0, \dots, N-1$$
$$T(k+1) \geq T_{min} \quad for\ k = 0, \dots, N-1$$
$$T(0) = T_O$$
$$s(k) = \{0, 1\} \quad for\ k = 0, \dots, N-1$$

The Mixed Integer Linear Program (MILP) minimizes the cost of electricity while keeping the indoor temperature above a minimum temperature. Since the home has an electric-resistance heating system, the optimal decision variable $s(k)^*$ will be binary (1 or 0) representing an ON or OFF state, respectively. The electricity prices, $c(k)$, are shown in the table below. The off, partial, and peak costs are based on PG&E's weekday summer rates [3]. The morning, and evening peak rates were added to make the problem more interesting as done in [9].

	Time	Cost ($/kWh)
Off Peak	12:00AM to 7:00AM and 11:00PM to 12:00AM	0.09841
Morning Peak	7:00AM to 9:00AM	0.25913
Partial Peak	9:00AM to 2:00PM and 9:00PM to 11:00PM	0.20808
Peak	2:00PM to 4:00PM and 6:00PM to 9:00PM	0.38119
Evening Peak	4:00PM to 6:00PM	0.44012

Table 3. Time of Use Cost of Electricity Schedule

As formulated, it is possible for the MILP to return an infeasible solution error. For example, if $T(0)$ is low and $T(k+1)$ cannot be raised above T_{min} in one time step or if the losses to ambient exceed the energy delivered by the heater for several time steps, then the minimum temperature constraint will be violated and the solver will return no solution. This limitation could be resolved by penalizing the MILP for allowing $T(k+1)$ to fall below T_{min} rather applying T_{min} as a lower bound constraint.

Model Predictive Control

The model predictive control (MPC) algorithm is executed as follows:

1. Set the current temperature measurement as the initial state, $T(0) = T_0$
2. Solve the MILP for the optimal open loop input sequence $s(0)^*, s(1)^*, .., s(N-1)^*$
3. Implement the first input $s(0)^*$ to advance the system one time step
4. Repeat the algorithm at the next time step

On the Cloud Implementation

The overall system diagram is shown in Figure 10 below.

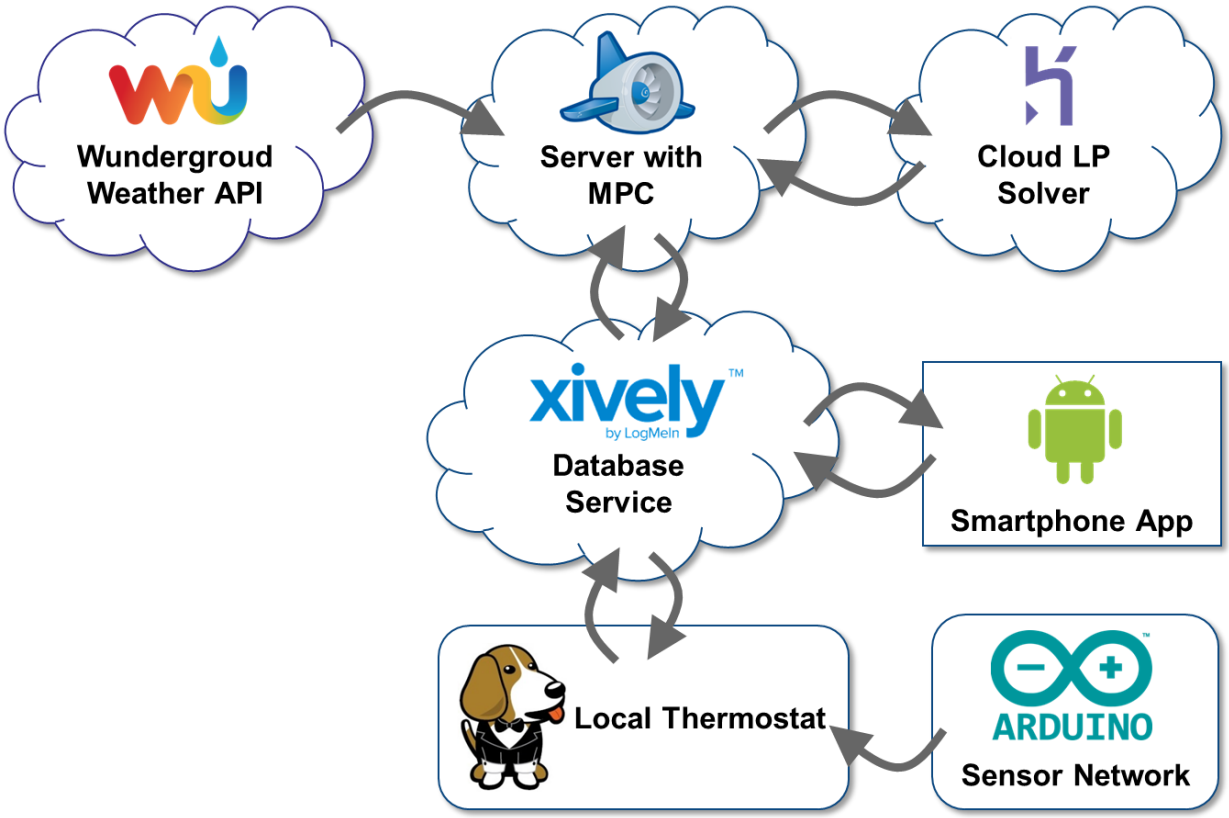

Figure 10. Internet Based Control System

The system is executed as follows:

- *Sensor Network*: The sensor network collects the temperature measurements once per minute and stores them in the thermostat and in the online database.
- *Weather Conditions/Forecast*: The server retrieves and parses the weather conditions and temperature forecast every 15 minutes, and sends them to the online database.
- *Mixed Integer Linear Program*: The mixed integer linear program matrices are constructed with the current sensor reading, weather forecast, electricity price schedule, and 6 hour time horizon (N=24). These matrices are sent to the LP Solver on the cloud.
- *Optimal Solution*: Once the problem is solved by the LP solver on the cloud, the optimal heater state (ON or OFF) is sent to the online database. An update is then pushed to the local thermostat for control actuation.
- *Smartphone App*: (Developed by the rest of the team in the CE271 course) The smartphone application retrieves and displays values from the online database at any time. It can send MPC Control State (ON or OFF) and can also send an override to the Heater State (ON or OFF) to the online database at any time.

The smartphone application, was developed to have both read and write capabilities, to monitor the behavior of the system. The user interface (UI) displays outdoor and indoor temperature readings in both °F and °C, as well as weather conditions from the Wunderground API. This information is read from the corresponding channels in the Xively web database. Additionally, the app writes to Xively through the two switches of the UI for heater and MPC control. When the switch is put in the "on" position, the app sends a 1 to the Xively channel; when the switch is put in the "off" position, the app sends a 0 to the Xively channel.

To update all readings including heater and MPC state, the UI provides a refresh button. This again reads from the channels in Xively and displays the latest information to the user.

This app implements an asynchronous model. Background operations are executed and results are published on the UI without having to manipulate threads and handlers. In other words, the UI remains responsive to additional requests such as button clicks while awaiting a response from the server for the initial request.

A snapshot of the user interface of the smartphone app is shown in Figure 11 below.

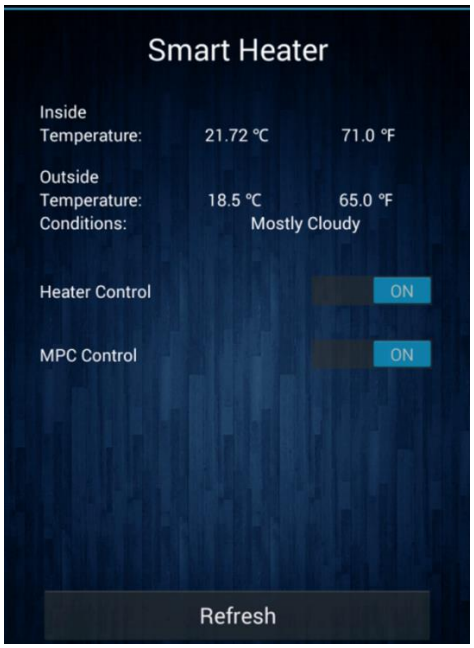

Figure 11. Smartphone App User Interface

Results and Conclusion

A real-time software in the loop simulation was conducted to compare the performance of a traditional deadband controller with our model predictive controller. The identified discrete thermal model with added noise was used to represent the evolution of the temperature inside the apartment. The simulation was run for approximately 3 days using the actual weather conditions/forecast. The outdoor conditions are presented in Figure 12, and the results are presented in Figures 13 and 14 below.

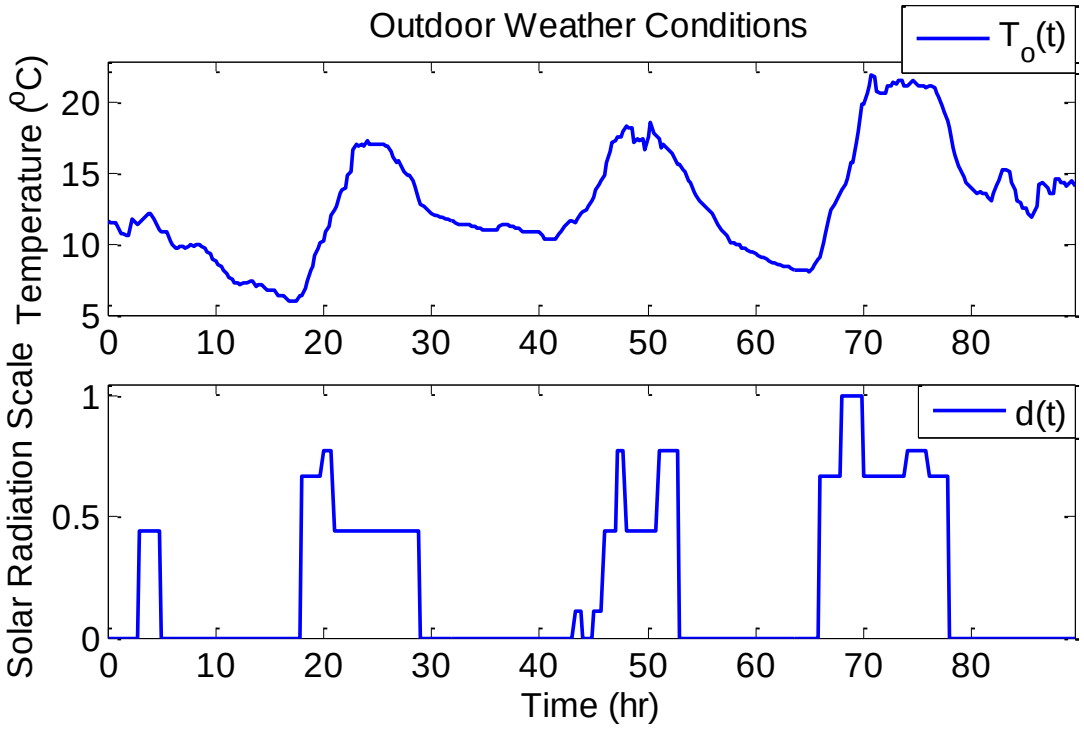

Figure 12. Outdoor Weather Data

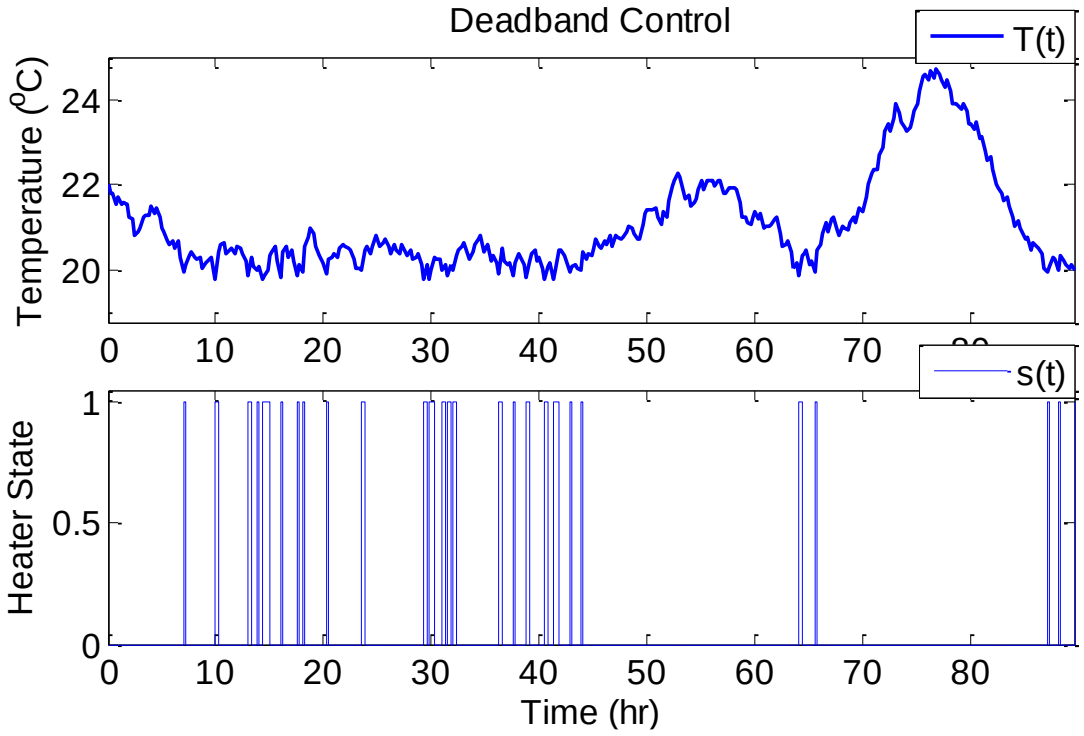

Figure 13. Deadband Control Results

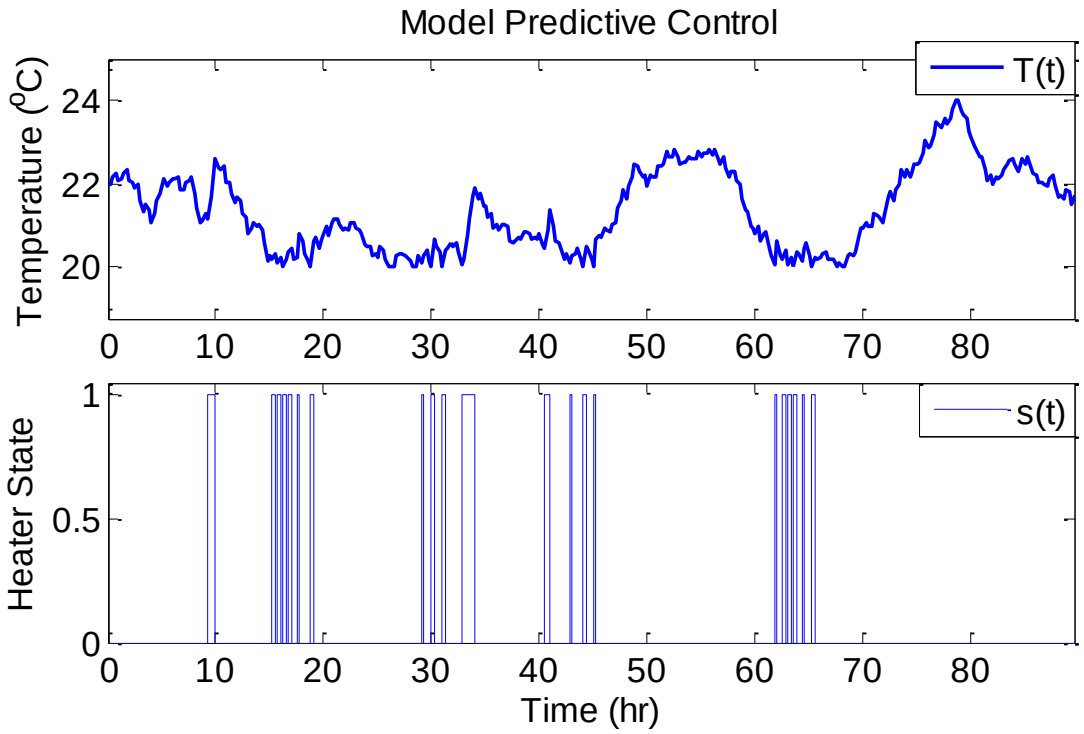

Figure 14. Model Predictive Control Results

As shown in the figures above, the deadband controller switches on whenever energy is needed to maintain the desired temperature in the apartment. In contrast, the MPC system avoids peak and partial-peak electricity prices, resulting in more concentrated periods of heating.

Based on the real time simulation, the MPC system showed a 31% reduction in heating cost compared to a traditional deadband controller. The results show that the system meets the objective of reducing electricity cost within the given constraints. Because the MPC system has achieved its objective by shifting the time of electricity use rather than decreasing the total energy demand, both the MPC and deadband controllers consume close to the same amount of energy in the simulation.

The results of the project show that the system is capable of a) monitoring the real-time thermal conditions in a space, b) reducing peak loads on the power grid by using electricity price as a proxy for peak demand, c) make use of weather forecast data to predict heating requirements according to the apartment's thermal dynamics, and d) enable smart home technology development with Internet-based systems and a smarphone application. Additionally, the MPC control system is capable of significantly reducing the consumer's electricity costs without compromising thermal comfort.

Acknowledgements

Many thanks to the Civil and Environmental Engineering Department and its Civil Systems program faculty, as well as the Energy, Controls, and Applications Lab (eCAL) for the resources and learning experiences required for this project. Thanks to Hoang Nguyen, Julien Nachef, and Michaella Chung for developing the Smartphone App.

References

[1] U.S Dept. of Energy. (2010). *Buildings Energy Data Book*. Available Online: http://buildingsdatabook.eren.doe.gov
[2] U.S Dept. of Energy. (2012). *Electric Resistance Heating*. Available Online: http://energy.gov/energysaver/articles/electric-resistance-heating
[3] Pacific Gas & Electric (PG&E) Electric Vehicle Cost of Electricity. Available Online: http://www.pge.com/tariffs/electric.shtml
[4] Wunderground API. Available Online: http://www.wunderground.com/weather/api/? MR=1
[5] Arens, E., Federspiel, C., Wang, D., & Huizenga, C. (2005). How ambient intelligence will improve energy efficiency in buildings. In W. Weber, J. Rabaey, & E. Aarts (Eds.), *Ambient Intelligence* (63-80). Berlin: Springer.
[6] Maasoumy, M., Moridian, B., Razmara, M., Shahbakhti, M, & Sangiovanni-Vincentelli, A. (2013, Oct.). Online Simultaneous State Estimation and Parameter Adaptation for Building Predictive Control. Paper presented at ASME 2013 Dynamic Systems and Control Conference, Palo Alto, California, USA.
[7] Radecki, Peter, & Hencey, Brandon (2012, June). Online Building Thermal Parameter Estimation via Unscented Kalman Filtering. Paper presented at 2012 American Control Conference, Montreal, Canada.
[8] Ma, Y., Kelman, A., Daly, A., & Borrelli, F. (2012, Feb.). Predictive Control for Energy Efficient Buildings with Thermal Storage: Modeling, Simulation, and Experiments. *Control Systems, IEEE*, 32 (1) 44-64.
[9] Perez, H., Burger, E., (2013, December). Cloud Enabled Smart Charging of Electric Vehicles. Poster session presented at: University of California, Berkeley. CE290i Course Project Presentation. Berkeley, California, USA.