INVERTING HVAC FOR ENERGY EFFCIENT THERMAL COMFORT IN POPULOUS EMERGING COUNTRIES

SIMULATOR AND MOBILE APPLICATION FOR HAWADAAR

Senior Project
2020

Department of Computer Science
Syed Baber Ali School of Science & Engineering
Lahore University of Management Sciences (LUMS)

Muhammad Ahmed Bappi

Certificate

I certify that the senior project titled “TBA” was completed under my supervision by the following students:

____

____

____

and the project deliverables meet the requirements of the program.

------------------------------------- **Advisor (Signature)**	Date:
------------------------------------- **Co-advisor (if any)**	Date:

CONTENTS

Introduction

One of the biggest problems faced by world's populous emerging countries today is the excessive power wastage through the usage of room-level standalone thermal devices. In the majority of homes and offices, there is no central temperature control system like HVAC. Stand-alone thermal devices like air conditioners and heaters run for long continuous periods in their respective seasons, consuming huge amounts of power. Along with the power wastage, the thermal levels reached by the continuous usage of these devices are hazardous for individuals' health. In his paper [1], Dr Hamad Alizai presents an inverted HVAC approach to solve this problem. Using the existing infrastructure and thermal devices, a central temperature control system is implemented which is cheap to implement and is very effective in maintaining healthy thermal levels and saving considerable amounts of energy. This report presents a simulated environment to show the working of the system implemented in the paper. Along with that, the report also presents the working of a front-end mobile application developed (iOS platform) for the end users to control the central temperature control system.

Background

Emerging countries predominantly rely on room-level air conditioning units (window ACs, space heaters, ceiling fans) for thermal comfort. These distributed units have manual, decentralized control leading to suboptimal energy usage for two reasons: excessive setpoints by individuals, and inability to interleave different conditioning units for maximal energy savings. If we aim to have a centrally controlled cooling/heating system in buildings in these countries, what are the possible solutions? Well one solution is to purchase and install a HVAC system to centrally control factors like temperature and humidity. The problems with this solution are: 1) The equipment and its installation costs are very high. 2) It would be easier to install it in new construction sites, otherwise we have to break through the walls and electricity wiring. So, what to do for many houses and office buildings, in the emerging countries, that already have standalone temperature devices installed? How can we achieve centrally controlled thermal comfort system using existing infrastructure and devices and keep the installation cost of the new system affordable?

Before elaborating on the answer to the question given above, it is essential to understand why is it important to have a central environment control system in the first place. Is it necessary or is it just another luxury for personal comfort? In developing countries like Pakistan, with exponential population growth, every year there is a large increase in the number of houses and buildings. As the country has extreme climate seasons, both hot and cold, throughout the year, people use standalone temperature devices in buildings on a per room basis. Each room has its own air conditioner or heater. The issue with these standalone devices is that they run continuously for hours without anyone caring to turn them off. As a result, a lot of energy resources are wasted. We are in an era of global warming, with most studies indicating excessive use of energy being its leading cause. By most accounts, the greatest contribution to global energy expenditure and the ensuing green house gasses will come from emerging Asian countries [7, 8], such as China, India, Pakistan, and Bangladesh that in combination account for nearly half of the world population [9]. With their economic upsurge, the energy demands of these countries are rapidly rising; thermal comfort of built spaces making a significant proportion. It is projected that there will be a tenfold increase in the world consumption of energy for cooling by 2050. Thus, for example, China alone is expected to surpass the USA by 2020 as the world's biggest consumer of electricity for air conditioning estimated at a trillion kWh. Another issue with these standalone devices is that the thermal comfort level become unbearable with these devices having inappropriate sizing for the room and then continuously running. For example, in the mid-June summer season when the outdoor temperature is really hot, between 40^{o} C to 50^{o} C, people inside the houses wear sweaters and use blankets to cover them while the air conditioner is running. This has a detrimental effect on the individual's health. To save energy consumption and to achieve the appropriate comfort levels, it is important to have a central control system to manage the thermal levels in the buildings.

The paper, "Inverting HVAC for Energy Efficient Thermal Comfort in Populous Emerging Countries" [1], proposes a unique solution to provide a centrally controlled thermal system using the already installed standalone thermal devices in the buildings. It presents a novel inverted HVAC approach: cheaply retrofiting these distributed units with "on-off" control and providing centralized control augmented with room and environmental sensors. The binary control approach exploits an understanding of device consumption characteristics at on/off and factors this into the control algorithms to minimize consumption. Each device is plugged into a "smart switch" that can be controlled by a wireless z-wave signal. Two temperature reading devices are used for inputs, one inside the room to read indoor temperature and one outside the room to get the effects of heat loss and room insulation. The control algorithms are implemented on an inexpensive arbitrary device that has the ability to implement the algorithms, get input from the temperature reading devices, send signals to the smart switches and communicate with the mobile app. There are two types of thermal comfort modes a user can choose using the mobile application that are implemented in the central thermostat. One is the 'set point' mode in which the user chooses a specific temperature value to be maintained in the room. The other is 'pmv' mode, in which thermal levels are maintained in the room automatically according to ASHRAE's PMV standards, using the temperature and humidity readings as well as users' metabolic activity and clothing insulation. The control algorithms are intelligently designed to keep track of the devices' hard constraints, for example, generally the air conditioner manufacturers suggest to turn on the device after at least 3 minutes after it has been switched off. There are two kinds of hard constraints, the control algorithms take care of: **1) transient power** – when the device turns on, it consumes extra energy than when it is running steadily. Frequent switching of the device will cause extra energy consumption which of course is opposite of the desired goal. **2) short cycling** – actuating the device faster than the specified rate; aggressive actuation of the device will damage it. For each device, out of the two hard constraints, the one which requires the longer time is chosen and provided to the control algorithms.

Design

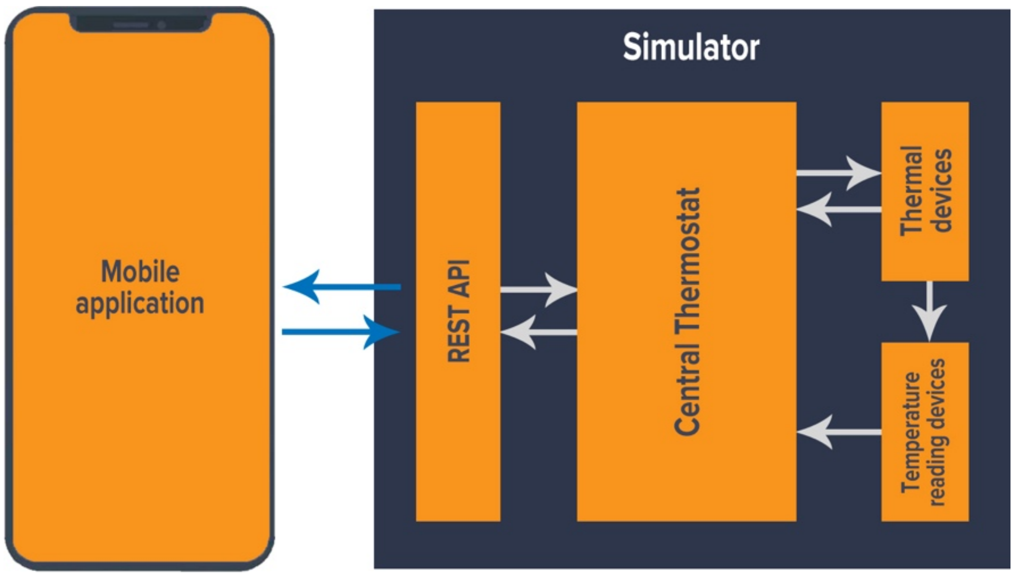
The simulation is designed to mimic the real-world scenario, i.e. a room consisting of a set of thermal devices that can be switched on or off by the central thermostat, two temperature reading devices, one for indoor temperature and one for outdoor temperature, the central thermostat with the control logics.

The simulation models a room of 12ft x 10ft x 10ft consisting of a window air conditioner of 1 ton with 3504 watts cooling capacity [12, 13], a 60 watts fan with 36 inch wing span, an electric space heater with 3 halogen elements of 400 watts each, 3 CFL bulbs of 30 watts each, with 10% efficiency, 2 persons who dissipate heat at an average rate of 97.22 j/s [14] each.

In the real-world system the temperature and humidity values are taken from temperature sensing devices. In the absence of these devices it was hard to simulate these readings. For the outdoor temperature and humidity, a live weather API [16] was used. For the indoor temperature a linear approximated model was used, using outdoor temperature, time of the day (different for day and night) and room's insulation. The indoor temperature varies with the type of thermal device currently running. For example, if the air conditioner is switched on, the indoor temperature will start to decrease while using factors like room and ac sizing, room insulation, heat transfer coefficient and heat capacity and the heat dissipated through bulb, people in room, fan motor etc to calculate the rate of this decrease.

The central thermostat contains the control algorithms of both thermal comfort modes, 'set point' and 'pmv'. They are the same as used in the paper [1]. To connect to the mobile application, a REST API was written which has four simple connection interfaces i.e. init, start, stop and change parameters.

The mobile application is designed with the goal to provide a simple remote like interface to control the central thermostat. The application's interface is simple and self-explanatory and it provides all the fundamental functionality to send and receive signals to and from the central thermostat. The main features include: power on/off, switch between the thermal modes, get live updates on the thermal levels in the room and change the modes' parameters like tolerance for the set point mode, metabolic rate and clothing insulation for the pmv mode etc.



Implementation

The implementation is two phased, first the working of the simulator is shown then the working of the front-end mobile application is shown.

Simulator

The simulator consists of four main parts, **room model** which has features of the room like sizing, insulation and thermal devices, **central thermostat** which implement the control algorithms for the two thermal comfort modes i.e. 'set point' and 'pmv', **thermal levels** reading model to read the indoor temperature, outdoor temperature, humidity and calculate the change in indoor temperature when the thermal devices are running, and a **REST API** to communicate with the mobile application.

The simulator is written in python3 using the FLASK framework for the REST API. The choice of the language and framework were based on their ability to provide all the functionality needed to build a robust simulator, ability to write maintainable and scalable code and ease of writing the code.

Room model

A room of dimensions 12ft x 10ft x 10ft is used with walls that are made of a brick layer of thickness 230mm in between two cement layers of thickness 15mm. The room has 3 thermal devices: a window air conditioner of 1 ton with a thermal capacity of 3504 watts that is kept on a minimum temperature value of 18°C with a relative air velocity of 0.5 m/s, a 60 watts fan with 36 inches wing span set to give a relative air velocity of 1.4 m/s, and an electric space heater with 3 halogen elements of 400 watts each with all the halogen element running to maintain a maximum temperature of 26°C in the room. The room also contain 3 CFL bulbs with heat dissipation of 27 watts heat and 2 people who dissipate heat at an average rate of 97.22 j/s each. The room's insulation and the effective heat dissipated by all the devices are used to calculate the change in the indoor temperature levels at different instances.

The running status of the devices is represented as a dictionary. The device status given below shows that the ac is powered on while the fan and heater are powered off.

deviceStatus = {"ac": 1, "fan": 0, "heater": 0}

Table 1 shows the density, heat capacity and heat transfer coefficient (k) values [3, 10, 11] of the materials used in the model to calculate the room's insulation and hence the thermal levels in the room. The 'V' in the calculation of the heat transfer coefficient of air is the relative air velocity.

Material	Density (kg/m^3)_	Heat capacity (J/(kg.K))	Heat transfer coefficient (W/m^2K)
Sintered Brick	1526	523	-
Cement Plaster	1406	1050	-

Air at 300K to 350 K	1.225	1005	$(10.45 – V + 10V^{0.5}) / 0.85984$
Wall (230 mm brick, 15mm cement x 2)_	Calculated using individual materials	Calculated using indivisual materials	1.787

***Table 1**: density, heat capacity, heat transfer coefficient of materials used in the model*

Central Thermostat

The central thermostat takes input from the REST API to start/ stop/ change the thermal mode. The control algorithms are implemented in the central thermostat. The algorithms for both thermal modes are presented below.

Algorithm 1
Setpoint based adaptive two-position control. Input: desrired setpoint (Ts), tolerance

```
1 while True do
2     Tin ← read_sensor()
3     Ton ← Ts + tolerance / 2
4     Toff ← Ts − tolerance / 2
5     if Tin > Ton then
6         switch_on(AC)
7     else if Tin ≤ Toff then
8         predicted_temperature ← $T_{toff+tsafe}$
9         if predicted_temperature ≤ Ton then
10            switch_on(AC)
11            update(prediction_parameters)
12            reset(tolerance)
13        else
14            extend tolerance(tolerance)
```

Setpoint algorithm. To begin, the algorithm needs two inputs, the required setpoint Ts and comfort bounds (soft constraint); the latter defines two control positions T_{off} and T_{on} symmetrically around Ts . Thus, with a cooling device, in order to achieve Ts as the average temperature, we must satisfy $T_{toff+tsafe} \leq T_{on}$, i.e. once the device is turned off, the temperature after the minimum safe off duration t_{safe} (hard constraint) will not exceed T_{on}. This may, depending upon the thermal load, require the algorithm to symmetrically extend the tolerance range on both sides of Ts, as described in Algorithm 1. For this, the algorithm needs to predict at time t_{off} (cf. Secton 4.3) the temperature at time $t_{off} + t_{safe}$ in order to decide whether to turn off the device or extend the tolerance range to meet the hard constraint. With a minor adjustment, i.e. by swapping T_{off} with T_{on}, this algorithm is also used to actuate a heating device.

Predicting the room's heat transfer rate

The room's heat transfer depends upon both external and internal thermal loads. The external thermal loads result in heat transfer through the building envelope from external elements such as the sun, the earth, and the outside environment. While, internal thermal loads come from heat generated within the room by people, lighting, equipment etc. As discussed above, in order to satisfy the hard constraint, the CT needs to predict the temperature $T_{toff+tsafe}$. The prediction model should thus account for both external and internal thermal loads whilst being simple and self calibrating. We note that the maximum duration of this prediction, in our setup, is just three minutes imposed by the AC. This, by the way, also provides a maximum theoretical duration of discomfort. Since we dynamically update our prediction parameters, as discussed below, our system can repeatedly fix prediction errors.

Instead of developing a complex thermal model for the room, we use measurements to glean the heat transfer coeffcient using Netwon's law of cooling as a frst approximation. A similar strategy has also been employed in a personalized comfort system [2]. According to the law, "the rate of heat loss of a body is proportional to the difference in temperatures between the body and its surroundings". Thus, given an outside temperature T_{out} and room temperature T_{in}, the rate of thermal energy loss for the room is proportional to the temperature difference:

$$\frac{dT}{dt} = -k(Tin - Tout)$$

We set $T = T_{toff + tsafe}$, $T_{in} = T_{off}$ and $t = t_{safe}$ and solve the above equation to estimate a room's heat transfer coefficient (k):

$$k = (\ln((Toff, tsafe - Tout) \,/\, (Toff - Tout)))\,/tsafe$$

However, we repeatedly update k just before T_{on} (i.e., when all the devices are off), aiming for an aggregate account of both external and internal thermal loads using a single heat transfer coefficient (k). The sensed $T_{toff + tsafe}$ in round n is used to calculate the k value to predict $T_{toff + tsafe}$ in round n + 1 as follows:

$$Ttoff + tsafe = Tout + (Toff - Tout)e^{-ktsafe}$$

Algorithm 2
PMV based two-position control. Input: comfort range, set of on devices (ON)

```
1 while True do
2     calculate(PMV)
3     if PMV > upper bound then
4         if ON= ∅ then
5             v ← AIR_VELOCITYfan
6             calculate(PMV)
7             if PMV ∈ comfort_range then
8                 ON ← ON UNION fan // switch-on fan
9             else
10                v ← AIR VELOCITYAC
11                ON ← ON UNION AC // switch-on AC
12        else if fan ∈ ON then
13            ON ← ON \ fan // switch-off fan
14            v ← AIR VELOCITYAC
15            ON ← ON UNION AC
16    else if PMV ≤ lower bound then
17        O N ← ∅ // switch-off all devices
```

PMV algorithm. The same baseline algorithm is used for delivering personalized comfort with the following two extensions: First, the thermal constraint is not the heat index but PMV described as follows:

$$PMV = f\,(M, Ta, Tr, v, Pa, Icl\,)$$

Where, **M** is the metabolic rate of the occupant; **Ta** is the air temperature; **Tr** is the mean radiant temperature (set equal to Ta); **v** is the relative air velocity in m/s^{-1} ; **Pa** is the relative humidity; and **Icl** is the clothing insulation factor of the occupant. In our work, we assume some of these parameters to calculate PMV and realize that in practice these are difficult to accurately ascertain; however, this does not affect the fidelity of the results. Second, while the heat index based setpoint only utilizes the heater and AC, additional factors in equation 2, such as air movement, also allow us to use the ceiling fan for maintaining a desired PMV level. Thus, as described in Algorithm 2, we program the CT to prioritize the use of lowenergy fan, and only turn on the AC when air circulation alone cannot keep the PMV within the required comfort bounds.

Thermal levels reading Model

The thermal level readings model was difficult to build because in the real-world scenario, the temperature and humidity readings and the heat transfer coefficients are measured using real sensors. This was not possible for the simulated environment and hence we have to calculate these reading using linear approximated model and the heat coefficients of standard materials taken from the internet. The outdoor temperature and humidity levels are taken from a live weather API. The thermal reading model is divided into two parts: one part to calculate the expected indoor temperature when room is left idle using the outdoor temperature, time of the day and room insulation, and the other part to calculate the change in indoor temperature when specific devices are running using the heat dissipated inside the room and room's heat transfer coefficient.

Expected indoor temperature

A linear model is build using the high and low temperatures in a day, taken from the weather API, and the time of the day. The algorithm takes the high and low outdoor temperature and estimate the indoor temperature values at these two outdoor temperatures by using the average difference between the indoor and outdoor temperatures given the room insulation at the hottest and coldest hours of the day. Using these 2 points i.e. (Outdoor_high, indoor_high) and (outdoor_low, indoor_low) an approximate linear model is generated which then takes the current outdoor temperature and the hour of the day to calculate the current expected indoor temperature. The temperature reading is taken at 60s intervals and after each interval the whole process written above is repeated. So, for each iteration fresh temperature values are taken from the weather API and a fresh linear is generated to output the latest readings. Expected indoor temperature is a replacement for the indoor temperature sensor taking readings when room is idle bringing into account the temperature difference at different hours of the day.

Indoor temperature change when devices run

The input temperature change (rise or fall), when specific thermal devices are running. For example, when the AC is turned on, the temperature starts to decrease and when the AC is turned off the temperature starts to rise and given enough off time, it reaches to a constant maximum value. The constant maximum value is evaluated from the expected temperature. According to Newton's law, "the rate of heat loss of a body is proportional to the difference in temperatures between the body and its surroundings". The new indoor temperature value is calculated using this law. The formulas used for the calculation are described below:

First, we need to calculate the total heat dissipated inside the room. To calculate this, we use the formula:

$$totalHeatDispitated = heatDisspated\_thermal\_devices + heatDissipated\_bulbs + heatDissipated\_people$$

Note: if the AC is turned on the heatDissipated_thermal_devices will be negative because it is cooling the temperature.

The new indoor temperature is calculated after every 60s. To calculate the indoor temperature of the next iteration, the following formula is used [2]:

$$T_{in}(s + 1) = T_{in}(s) + \ (totalHeatDissipated - k(T_{in} - T_{out}) \ ) \ / \ C$$

'K' is the heat transfer coefficient of the desired materials through which the heat is transferred and 'C' is the respected heat capacity. These values are selected according to Table 1. The difference in the old and new temperature is calculated in two phases, first the change in the air temperature by the thermal devices inside the room is calculated, then the temperature difference by heat loss through the room's walls is calculated. Both these values are added to get the total temperature difference in the indoor temperature's current and next value. This difference value is then added to the current value to get the new temperature value. If the new value is going to be less than the previous value, the difference will automatically be calculated to be negative.

REST API

The REST API takes is used to transfer signals back and forth between the mobile application and the simulator. It provides 4 simple connection interfaces:

1) base_Url/init:
   This is used to send the mobile application the current state of the central thermostat when the application is launched. The thermostat stores its state in a json file in the following format:

```
dictionary = {
        "city",
        "mode",
        "outdoor_temperature",
        "indoor_temperature",
        "humidity",
        "set_point_value",
        "set_point_tolerence",
        "pmv_comfort_range"
        "pmv_metabolic rate",
```

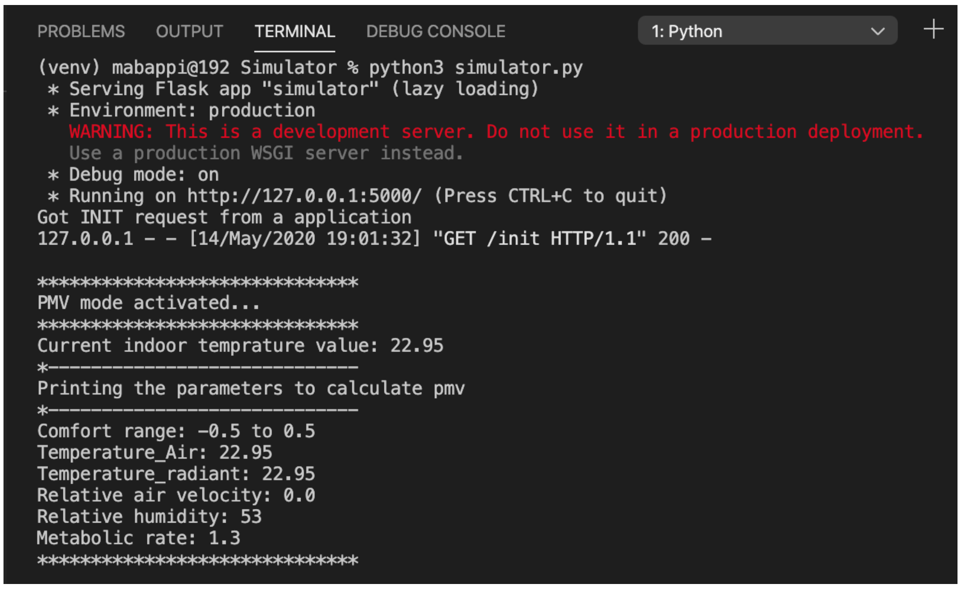
"pmv_clothing_insulation"}
This helps the mobile application to set its user interface according to the current state of the thermostat. This includes whether the thermostat is currently running or not, the temperature and humidity levels in the room and if the thermostat is running, the current mode.

2) base_url/start:
This is used to start the central thermostat. The user chooses a mode (set point / pmv) and sends the start signal to the simulator. The application gets the mode selected through the UI and the set point temperature if the mode is setpoint, and sends this information via a POST request to the simulator in the start request. The simulator parses the start request and start the system with respect to the values sent by the application.

3) base_url/stop:
This is used to stop the central thermostat. The user sends the stop signal when desired. The thermostat parses the stop request and stops the system.

4) Base_url/changeParams:
This is used to change the thermal modes parameter values from the mobile application. The example parameter values are tolerence for 'set point' mode, metabolic rate, clothing insulation, comfort range for the 'pmv' mode.

Extensive error handling was done while writing the simulator to make it robust and error free. The simulator is independent of the application and can be used with any other application using its REST API.



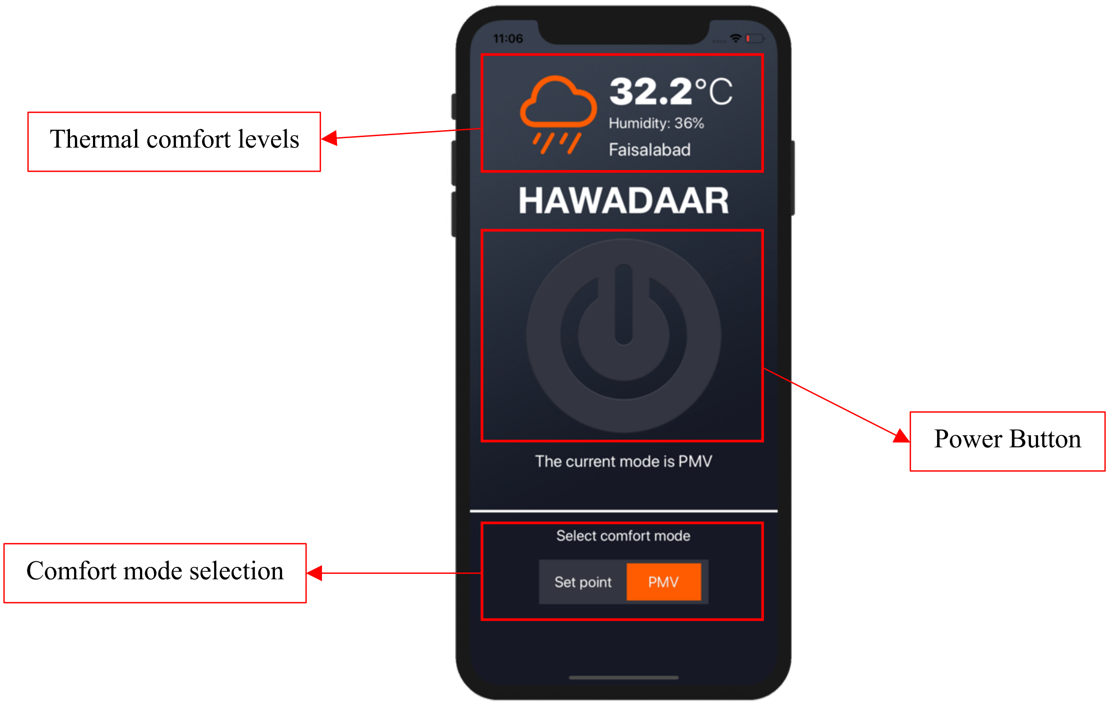
```
(venv) mabappi@192 Simulator % python3 simulator.py
 * Serving Flask app "simulator" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Got INIT request from a application
127.0.0.1 - - [14/May/2020 19:01:32] "GET /init HTTP/1.1" 200 -

******************************
PMV mode activated...
******************************
Current indoor temprature value: 22.95
*-----------------------------
Printing the parameters to calculate pmv
*-----------------------------
Comfort range: -0.5 to 0.5
Temperature_Air: 22.95
Temperature_radiant: 22.95
Relative air velocity: 0.0
Relative humidity: 53
Metabolic rate: 1.3
******************************
```

Mobile application's design

The mobile application was built for iOS users using the native swift programming language. The main goal of the app's design was to work as a remote-control interface to control the central thermostat. A simple looking single screened design is built which is self-explanatory and easy to use while it works perfectly fine for the purpose it was intended for. Modern design patterns like MVC and delegate design pattern are used to write the code to make it maintainable and scalable. Extensive error handling was done for the application, to make it robust and error free and hence the application is not depended on the simulator, rather it is depended on the REST API provided by the simulator. Any other simulator or real-world system that provides the same API can be used with the application.

Main features of the application:



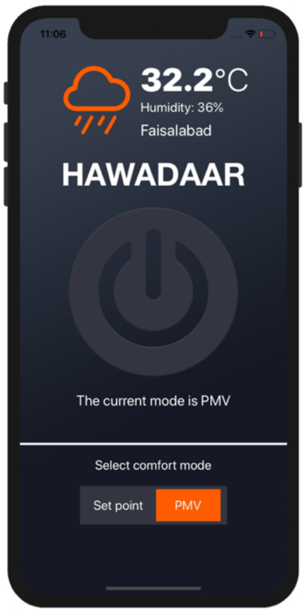
1) Thermal levels in the room – The thermal levels show the temperature and humidity levels in the room. The data is fetched from the simulator when the application starts.
2) Power button – The power button starts or stops the central thermostat
3) Comfort mode selection – to select the comfort mode from the two options i.e. set point and pmv.

Mobile application use cases

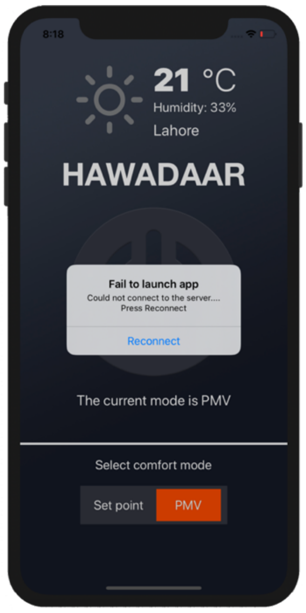
Launch application

When the application is launched, it makes an attempt to connect to the simulator.

If the connection is established the simulator sends room's thermal levels and the current state of the simulator i.e. whether the central thermostat is running or not and if it is running what mode is currently active. The state of the simulator and thermal levels are then updated in the application's user interface. In the image below, when the app is launched, the central thermostat is not running, so the app shows the power button in off state and the default mode PMV.



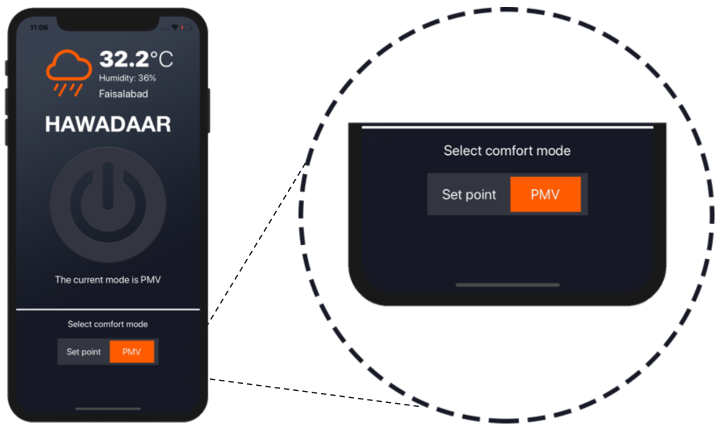
If the connection is not established, the application shows an error message and asks the user to reconnect. Unless a connection is made, the application will not launch.



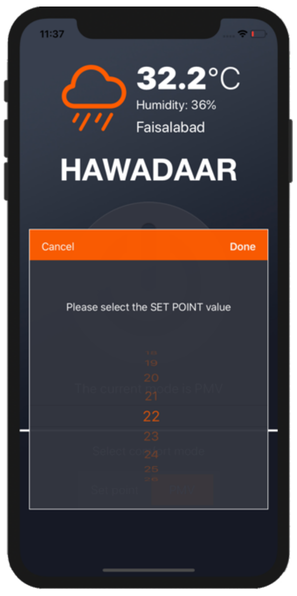
Fail to launch app
Could not connect to the server....
Press Reconnect
Reconnect

Select mode

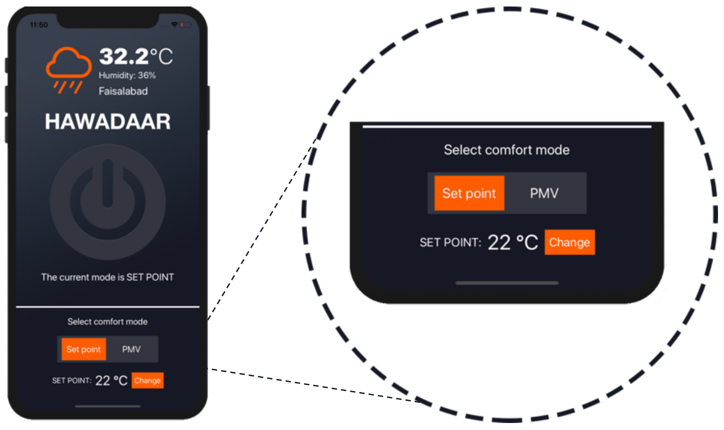
Choose the mode of thermal comfort from 'select mode' section.



If you select "set point" mode, a temperature values list is displayed to choose a value for the set point temperature.

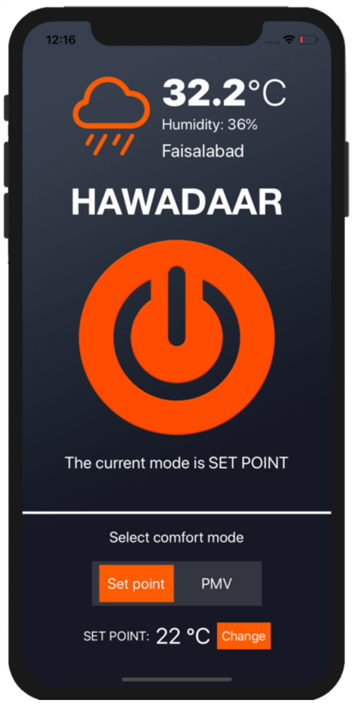


After you select a value from the list and click the "done" button at the top right corner, the set temperature value and a "change" value button are displayed in the "select mode" section.



Start the central thermostat

To start the central thermostat, after you choose the thermal comfort mode, press the power button. The application sends the start signal to the simulator. The simulator parses the start request and if everything is okay, it starts the desired comfort mode and sends back a confirmation message to the app. If the app receives the confirmation message it turns the color of the power button from grey to orange. If error occurs, the app displays the error message.



Stop the central thermostat

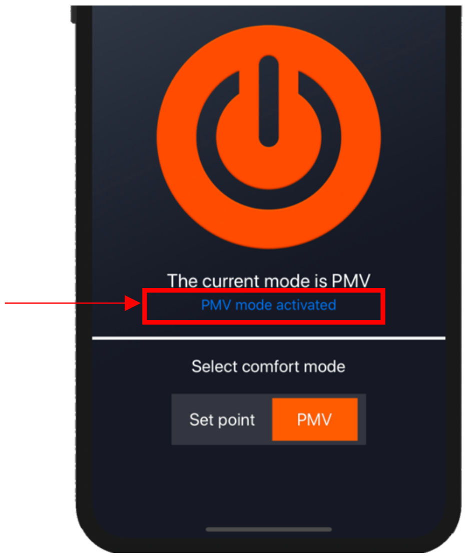
To stop the central thermostat, press the power button again. The simulator will parse the stop request and if everything is ok, it will send the app a confirmation message. If the app receives the confirmation message, it will turn the orange button back to gray. If error occurs, an error message will be displayed.

Change mode while the app is running

The modes can be switched while the app is running. If you change the mode while the app is running, the app will send a signal to the simulator to change the comfort mode. If everything is okay, the simulator will change the mode and send a confirmation message to the app. An error message will be displayed in case an error occurs. Furthermore if “set point” mode is running and you change the set temperature value, using the “change” button, the app will send the simulator a signal to change the set temperature value.

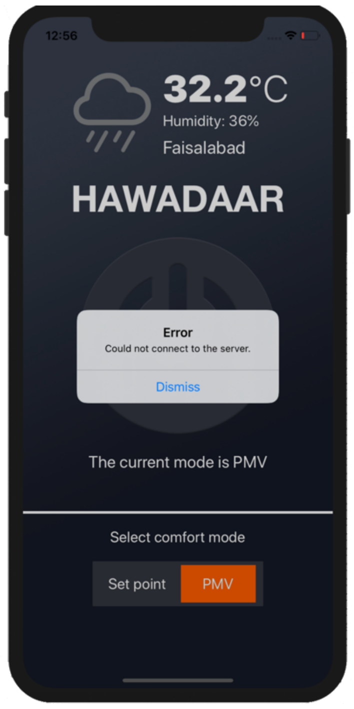
Displaying the confirmation messages from the server

When the app sends a signal to the simulator, in normal scenario, the simulator will reply back with a confirmation message. Specific messages for specific scenarios are displayed for a few second in the label shown in the image below. This feature is added for to make the user interface more expressive for the user, hence the user knows when things are working at the back end or which mode has been activated or deactivated.



Error handling

Error handling is important an important aspect of any robust app. Extensive error handling was done while writing the application. An error alert box is displayed, if any kind of error occurs. A common error scenario is that when a user presses any button that triggers a signal to the simulator but the app fails to connect to the REST API, due to issues with the network or the simulator is not running in the first place.



Error
Could not connect to the server.

Evaluation and future work

The simulator provides a great virtual environment to see the working of the actual system. The central thermostat's working is approximately the same as the real device. The thermal levels control however is different. This is because a simple model for the room is implemented, which doesn't correspond to some of the features of a real room like heat absorbed by the furniture and stuff like books etc. in the room. In real scenario, obviously the temperature levels and heat transfer coefficient are measured using real sensors so they take into account the effects of these features. Never the less the model does take into account the room insulation from where the heat is transferred to outside the room and the heat dissipated by the thermal devices, hence making it dynamic. The thermal model decreases the temperature by 1.6 ^{o}C per minute if the air conditioner is running, when the outdoor temperature is at its peak and the temperature difference inside and outside the room is approximately 10 ^{o}C.

One benefit of using the simulator over the real scenario is that you can easily change the dimensions of the room as well as the sizing and thermal capacities of the thermal devices, which is difficult to achieve in real world. The simulator can be extended easily to make a more complex room and thermal model using the formulas already implemented.

The mobile application provides all the fundamental functionality, needed by the end user to operate the central thermostat. It is independent of the simulator and depends only on the REST API provided by the simulator. It can run with any other simulator or a real-world thermostat if they provide the same API. The provide have live temperature update feature as well as changing the parameters of the thermal modes and is extensively tested for different error handling situations. The app can be further extended to control the features of the room like room dimensions, insulation properties and the kind of thermal device being used. Obviously, these features will not be used by the end users very often so they were not implemented due to shortage of time. Furthermore, in the paper, one control thermostat device controls the thermal levels in a single room, hence the app was built with the assumption that there is currently only one room. The functionality for multiple rooms can be easily extended using the design patterns used in writing the app. Lastly the app is written for a single platform i.e. iOS mobile devices; it can be written for other platforms like android using the code of the current app.

The app is yet to be tested in a real environment, however, there was a lot of time spent to build the simulator, to simulate things as close to real scenario as possible so hopefully it will work fine in the real system too.

Appendix

The complete project including the simulator, iOS application and the installation guide is available on a GitHub repository
https://github.com/mAhmedBappi/Hawadaar

References

1) Khadija Hafeez, Ayesha Ali, Yasra chandio, Affan A.syed, AbuBakar, Muhammad Hamad Alizai, Tariq M.Jadoon: Inverting HVAC for Energy E□icient Thermal Comfort in Populous Emerging Countries

2) Peter Xiang Gao, S. Keshav: Optimal Personal Comfort Management Using SPOT+

3) Lili Zhang, Tao Luo, Xi Meng, Yating Wang, Chaoping Hou, Enshen Long: Effect of the thermal insulation layer location on wall dynamic thermal response rate under the air-conditioning intermittent operation

4) Jing Li, Xi Meng, Yanna Gao, Wei Mao, Tao Luo, Lili Zhang: Effect of the insulation materials filling on the thermal performance of sintered hollow bricks

5) Jennifer L. Nguyen, Joel Schwartz, Douglas W. Dockery: The relationship between indoor and outdoor temperature, apparent temperature, relative humidity, and absolute humidity

6) Lucas W. Davis and Paul J. Gertler. 2015. Contribution of Air Conditioning Adoption to Future Energy Use Under Global Warming. Proceedings of the National Academy of Sciences of the United States of America (2015).

7) Michael A. McNeil and Virginie E. Letschert. 2008. Future Air Conditioning Energy Consumption in Developing Countries and what can be done about it: The Potential of Efficiency in the Residential Sector. http://escholarship.org/uc/item/64f9r6wr

8) United Nations Population Devision. 2015. Revision of World Population Prospects. https://esa.un.org/unpd/wpp/.

9) Convective heat transfer: https://www.engineeringtoolbox.com/convective-heat-transfer-d_430.html

10) Specific heat capacities of air: https://www.ohio.edu/mechanical/thermo/property_tables/air/air_Cp_Cv.html

11) Air conditioning your home: https://www.nrcan.gc.ca/energy/publications/efficiency/residential/air-conditioning/6051

12) Air conditioners' cooling capacities: https://coolingpowercorp.com/news/everything-you-need-to-know-about-cooling-capacity/

13) Heat decimated by Humans: https://www.physlink.com/education/askexperts/ae420.cfm

14) Heat Transfer Coefficient: https://en.wikipedia.org/wiki/Heat_transfer_coefficient

15) Open weather map API: https://openweathermap.org/api