Building Management System : Scheduler and a web service to log data and control devices

Student Name: Divay Prakash, Amogh Vithalkar
Roll Number: 2014039, 2014134

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science and Engineering/
B.Tech. in Electronics and Communication Engineering
on November 17, 2016

BTP Track: Engineering

BTP Advisor
Prof. Hemant Kumar

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

We hereby declare that the work presented in the report entitled **Building Management System : Scheduler and a web service to log data and control devices** submitted by us for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* and *Bachelor of Technology* in *Electronics and Communication Engineering* respectively at Indraprastha Institute of Information Technology, Delhi, is an authentic record of our work carried out under guidance of **Prof. Hemant Kumar**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

.... **Place & Date: ....**
Divay Prakash, Amogh Vithalkar

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

.... **Place & Date: ....**
Prof. Hemant Kumar

Abstract

The building management HVAC(heating, ventilation and air conditioning) system for Phase II of IIIT-Delhi campus is designed to take care (and advantage) of diversity of use. Instead of using large AHUs(Air Handling Units), we will have individual units in faculty rooms, labs, and other spaces. This would allow us to condition air of the spaces that are occupied and the system would be able to maintain desired temperature more closely. The disadvantage of this approach is higher capital cost and a larger I/O points for BMS but the running cost will be saved. In this project we are developing a scheduler hosted on a web server to control the ACs valves connected through Raspberry Pi and Arduino.

Keywords: building management system, scheduler, web server

Acknowledgments

We take this opportunity to express our deepest gratitude and appreciation to all those who have helped us directly or indirectly towards the successful completion of this project.

We would like to express our sincere gratitude to our advisor Prof. Hemant Kumar for providing his invaluable guidance, comments and suggestions throughout the course of the project.

Work Distribution

The distribution of work done by the team members over the course of this project is as follows -

- Divay Prakash
  - Designed and implemented the frontend for the central server
  - Documented Python code written for the sub-server
  - Documented Arduino code written for the microcontroller
  - Ported sub-server from `BaseHTTPServer` to Django server
- Amogh Vithalkar
  - Implemented the backend for the central server
  - Built a scheduler for the central server

Contents

List of Figures

List of Abbreviations

AHU	Air Handling Unit
API	Application Program Interface
BMS	Building Management System
CSS	Cascading Style Sheets
CSU	Ceiling Suspended Unit
DBMS	Database Management System
FCU	Fan Coil Unit
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HVAC	Heating Ventilation and Air Conditioning
I/O	Input/Output
LAN	Local Area Network
MAC	Media Access Control
OS	Operating System
REST	Representational State Transfer
SD	Secure Digital
SoC	System on Chip
VFD	Variable Frequency Drive
WiFi	Wireless Fidelity

Chapter 1

Introduction

1.1 BMS for phase I

HVAC for the Phase I buildings (Academic, Lecture halls and library building) of IIIT-Delhi campus used large floor mounted AHU's to condition large spaces except lecture halls which were served by CSU's (ceiling suspended units). Library building has one AHU per floor feeding all the labs and rooms. The academic building has 6 AHUs (three serving A wing and the other three wing B). Hostels have one FCU per room. The large AHUs of the academic block control volume of cold air by reducing the fan speed using VFD drives. As the area per AHU is large and the need very diverse (labs with varying number of occupants and equipment vs faculty rooms), the HVAC system is not very energy efficient and uniform temperature can't be maintained. The HVAC system has a very basic centralized control that can set temperature to be achieved in each one of the AHUs or CSUs. It also allows switching on/ off, monitoring parameters etc centrally. Other than AHUs/CSUs it can display status and parameters of chillers, cooling towers and hot water generator.

In BMS/ HVAC terminology each point that is controlled or read is an I/O point and an I/O summary is prepared for any BMS installation to estimate the cost. The I/O points are of 4 types - digital input, digital output, analog input and analog output. In phase I, we had chosen to only include I/O that either needed to be controlled or the I/O that were to be sensed and used for controlling to contain cost. The system was provided by Trane.

Later a parallel system for monitoring and data collection was implemented by a research group led by Dr Amarjeet Singh. This group installed wireless temperature sensors (13) and Ethernet based power meters (500) all over the campus to optimize HVAC and energy use based on data collected.

1.2 BMS for phase II

The HVAC for Phase II of IIIT-Delhi campus is designed to take care (and advantage) of diversity of use. So instead of large AHUs, we will have individual units in faculty rooms, labs, and other spaces. This would allow us to condition air of the spaces that are occupied and the system would be able to maintain desired temperature more closely. The disadvantage of this approach is higher capital cost and a larger I/O points for BMS but the running cost will be saved.

Chapter 2

Problem Statement

Given the large number of units to monitor and control, the cost of BMS was estimated to be very high. In addition, for phase II, the traditional system would be close-ended and proprietary.

Most of the cost (about 60%) would have been in the wiring the large number of units to controllers. This cost can be saved if individual WiFi-based controllers (500-600) were to be deployed. These wireless devices can use the existing institute LAN infrastructure to further save costs.

Chapter 3

Architecture

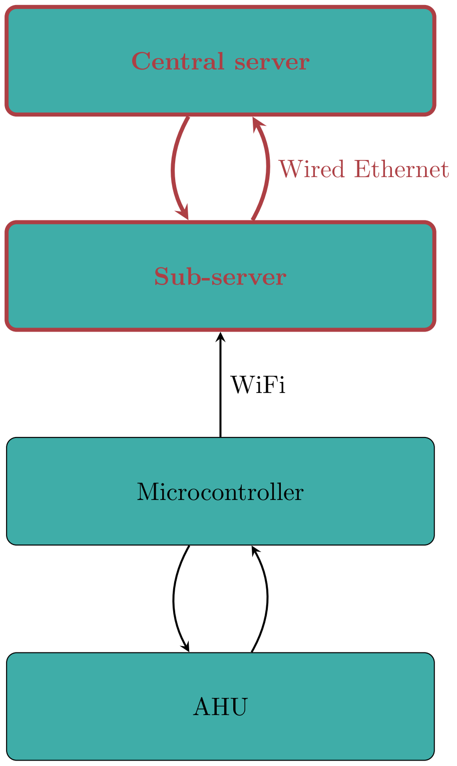


Figure 3.1: Block diagram of the building management system project

The BMS/ HVAC system, is structured in the manner described by figure 3.1. At the lowest level is the AHU (Air Handling Unit), which performs the actual cooling/heating functionality of the system. It is controlled by the microcontroller. The microcontroller monitors various system parameters and accordingly runs the AHU. It is in turn controlled by a sub-server unit, which is responsible for logging data and passing control instructions to the microcontroller unit according to policies set by the central server. The highlighted sections in figure 3.1 are the modules that were worked upon over the course of this project. In addition, code documentation for the entire stack was also written.

Chapter 4

Design

4.1 Overview

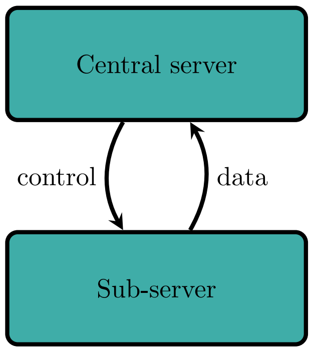


Figure 4.1: Higher level control modules of the building management system

The higher level control of the building management system is carried out by the central server in conjunction with various sub-server units managing their own group of microcontroller devices. The details of the central server and the sub-servers are given in the following sections.

4.2 Central server

The central server of the system is a software implementation only. There are no specific hardware requirements for the same. The server is implemented in such a manner so as to be able to provide interfaces to both the sub-server devices using a REST API and also to system administrators by way of a GUI. If required, the central server could be on the cloud. The server can be broken down into two main modules, the frontend and the backend. Both of these modules were built over the course of this project.

4.2.1 Design specifications

Frontend

The frontend of the central server consists of a hierarchy of web pages created using HTML/ CSS/JavaScript/jQuery and the Bootstrap framework. This provides the GUI which can be used by system administrators for overall analysis/control of the system.

Backend

The backend of the central server consists of a web application created using the Python web framework Django. The system used MySQL as the database management system.

4.2.2 Design choices

There are various design choices that had to be made in order to create a cohesive and well-structured central server for the BMS project. The central server is implemented as a web application. The choice of a web-application was made keeping in mind the advantages of web applications over traditional desktop applications -

- Easier deployment - The application does not need to be individual deployed to any client machine. The client machine only requires a functioning web browser to access the system.
- Maintenance - System maintenance is vastly easier as all updates and bug fixes need to be introduced at the server end only. In the same manner, regular updates are also easy.
- Platform independent - This was the major reason in favour of using a web application. The server implemented as a web application provides complete functionality to clients running any OS.
- Faster development process - By using a web application, users access the system by way of a web browser. This creates a uniform environment across platforms. While user interaction with the application needs to be thoroughly tested on different web browsers, the application itself needs only be developed for a single operating system. Theres no need to develop and test it on all possible operating system versions and configurations. This makes development and troubleshooting much easier.
- Scalability - Being independent of hardware configurations, web applications are easily scalable. It is possible to scale the system with growing I/O points or number of client instances.

4.3 Sub-server

4.3.1 Design specifications

The sub-servers are individual Raspberry Pi units running a Linux OS distribution. These devices are connected to the central server using a wired Ethernet connection and each unit manages up to 20 AVR microcontroller units over a WiFi interface. The Raspberry Pi devices host a web server which provides various functionalities. Over the course of this project, the web server was ported from a `BaseHTTPServer` to a Django-based server, both coded in Python.

4.3.2 Design choices

The choice of using a Raspberry Pi device for the sub-server was made considering the following points -

- Ease of deployment - Due to the small size of the device, it is easy to deploy unobtrusively in extremely small spaces such as wiring closets and channels, where a power supply and Ethernet cables are available.
- SoC - The Raspberry Pi provides us a complete system on chip ie. it integrates all the components of a computer on one single chip. This micro-computer does not require any additional chips for its functionality, with built-in Ethernet and WiFi interfaces.

Chapter 5

Implementation

5.1 Central server

5.1.1 Description

The central server is a Django web application which uses a MySQL DBMS. The frontend part of the web application is done using HTML/CSS and bootstrap.

5.1.2 Functionality

The web application takes inputs from the user to set the time and temperature range for each micro-controller for which the AHU's should work and logs the data in the MySQL database on the server according to the microcontroller's MAC address. Central server communicates to sub-servers using the client-server model.

5.2 Sub-server

5.2.1 Description

The Raspberry Pi units serving as sub-servers run a Django based web application which serves a REST API. This is utilised by both the microcontrollers and the central server which communicate with the device using HTTP messages. In addition, the sub-server device also functions as a client in some cases. This is further explained in the next section.

5.2.2 Functionality

For microcontroller

All communication taking place between sub-server devices and the microcontrollers follows the client-server model. However, the server (Rapberry Pi device) cannot initiate a message send to the client (microcontroller) without a prior request from the client. Due to memory and threading constraints at the microcontroller end, interrupts have not been used. Thus all communication is initiated by the client device.

- Data logging - The sub-server device provides a REST API method for microcontroller devices to log data. The microcontroller devices makes an HTTP POST request to the Raspberry Pi sub-server, which processes the attached data and stores it in the database. This data is stored in a MySQL database using microcontroller MAC addresses and the HTTP message timestamp as keys. Thereafter, the sub-server sends a confirmation message back to the client.
- Fetching commands - The microcontroller devices also fetch commands from the sub-server units. For this purpose, the microcontroller makes an HTTP GET request to the sub-server. The sub-server extracts the microcontroller's MAC address from the request and queries its database for any pending commands to be sent to that device. If found, it returns the same in the HTTP response to the client, else an empty response is sent.

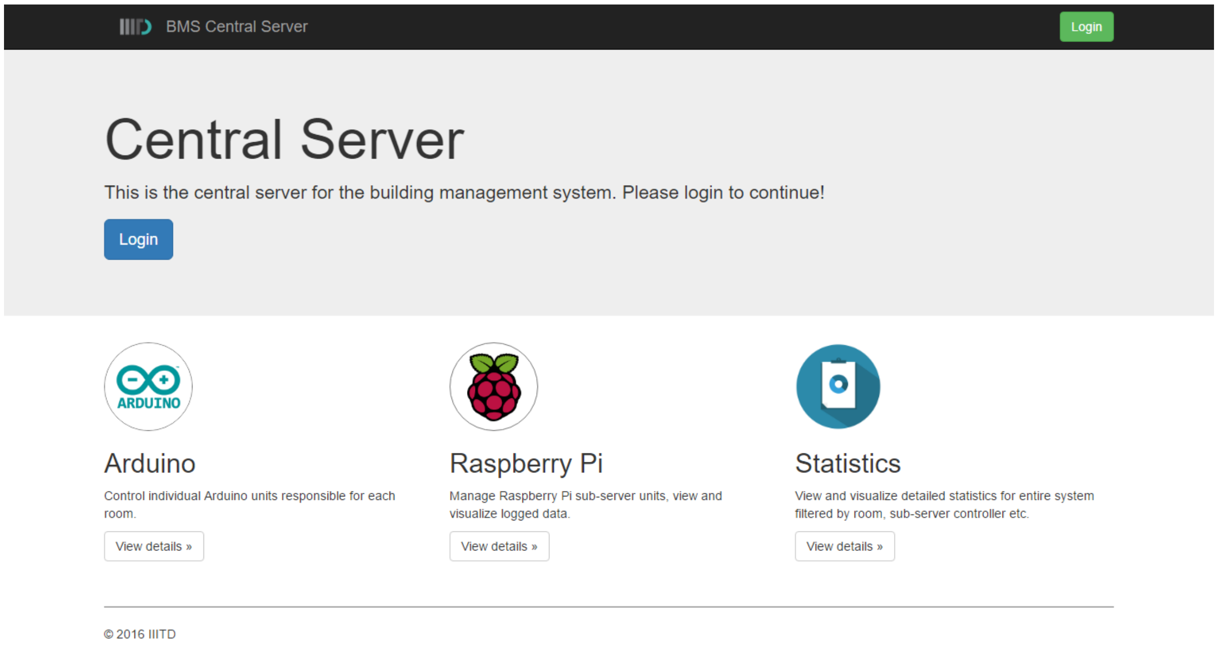
For central server

In case of the communication with the central server, the model followed is again client-server model. However, both the sub-server and central server can initiate message sending.

- Receive commands - The sub-server device acts as a client for this method, used by the central server to transmit commands to the sub-server using HTTP messages.
- Send data
  - With prior request - This method is followed if the central server requests data logs from the sub-server using an HTTP GET message as-and-when required.
  - With no prior request - The Raspberry Pi device uses an SD card to store data persistently. To ensure longevity of the system, it is essential to keep the number of read/write cycles on the card to a minimum. Thus data logs are stored in primary memory. While this solves the SD card issue, it creates another as main system memory is being taken up by static data. To resolve this issue, the Raspberry Pi device dumps the data logs to the central server using HTTP POST messages at a fixed time interval. This enables the deletion of the files at sub-server end, freeing up valuable memory resources.

Chapter 6

Specification


BMS Central Server

Login

Central Server

This is the central server for the building management system. Please login to continue!

Login

Arduino

Control individual Arduino units responsible for each room.

View details »

Raspberry Pi

Manage Raspberry Pi sub-server units, view and visualize logged data.

View details »

Statistics

View and visualize detailed statistics for entire system filtered by room, sub-server controller etc.

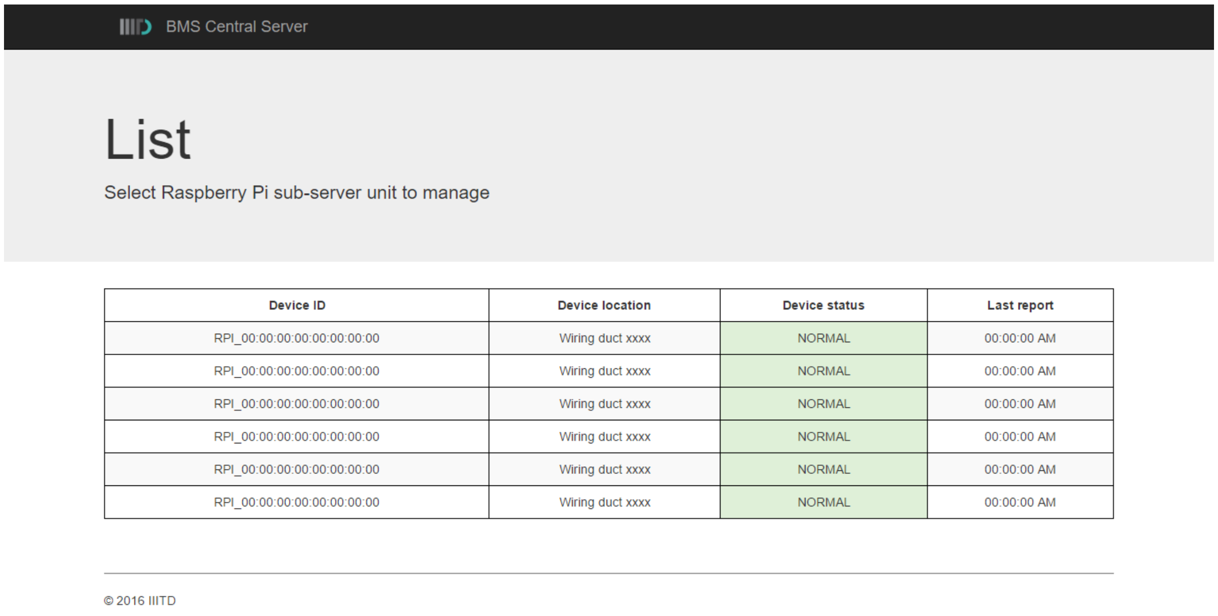
View details »

© 2016 IIITD

Figure 6.1: Home page

The web application starts with this page. The user has to login to the application to use its features. Once the user logs in to the application, they can control individual sub-servers and microcontrollers. The user can also see the current system statistics as well as the previous parameter settings.

If the user wants to manage a sub-server, the user can navigate through the Raspberry Pi section to reach the following page (figure 6.2).


BMS Central Server

List

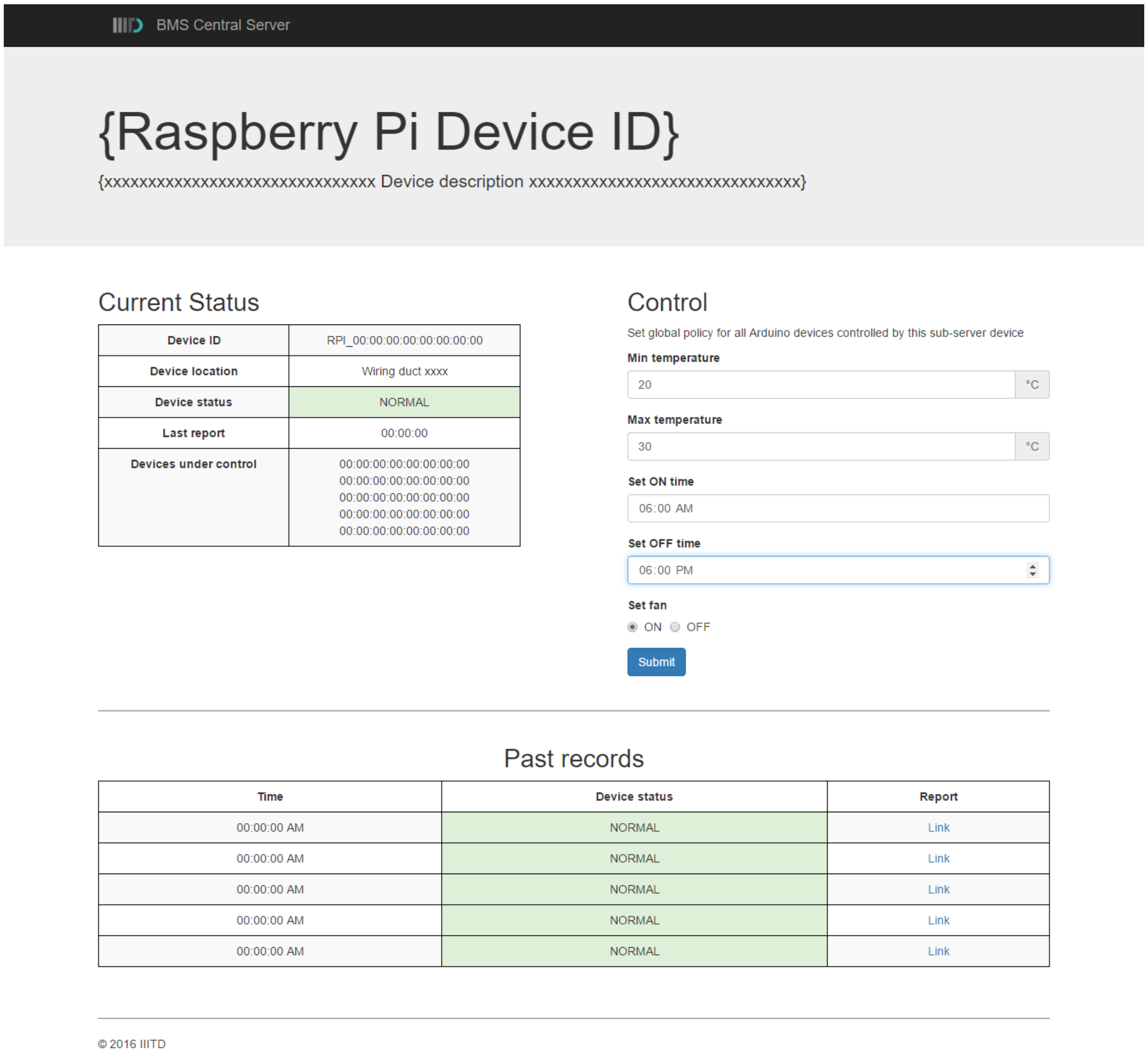
Select Raspberry Pi sub-server unit to manage

Device ID	Device location	Device status	Last report
RPI_00:00:00:00:00:00:00:00	Wiring duct xxxx	NORMAL	00:00:00 AM
RPI_00:00:00:00:00:00:00:00	Wiring duct xxxx	NORMAL	00:00:00 AM
RPI_00:00:00:00:00:00:00:00	Wiring duct xxxx	NORMAL	00:00:00 AM
RPI_00:00:00:00:00:00:00:00	Wiring duct xxxx	NORMAL	00:00:00 AM
RPI_00:00:00:00:00:00:00:00	Wiring duct xxxx	NORMAL	00:00:00 AM
RPI_00:00:00:00:00:00:00:00	Wiring duct xxxx	NORMAL	00:00:00 AM

© 2016 IIITD

Figure 6.2: List of available sub-servers

In this page the user selects the MAC address of the sub-server unit and is redirected to the following page (figure 6.3). This page also shows the location and status of the sub-server units.



BMS Central Server

{Raspberry Pi Device ID}

{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Device description xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}

Current Status

Device ID	RPI_00:00:00:00:00:00:00:00
Device location	Wiring duct xxxx
Device status	NORMAL
Last report	00:00:00
Devices under control	00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00

Control

Set global policy for all Arduino devices controlled by this sub-server device

Min temperature: 20 °C

Max temperature: 30 °C

Set ON time: 06:00 AM

Set OFF time: 06:00 PM

Set fan: [✓ ON] OFF

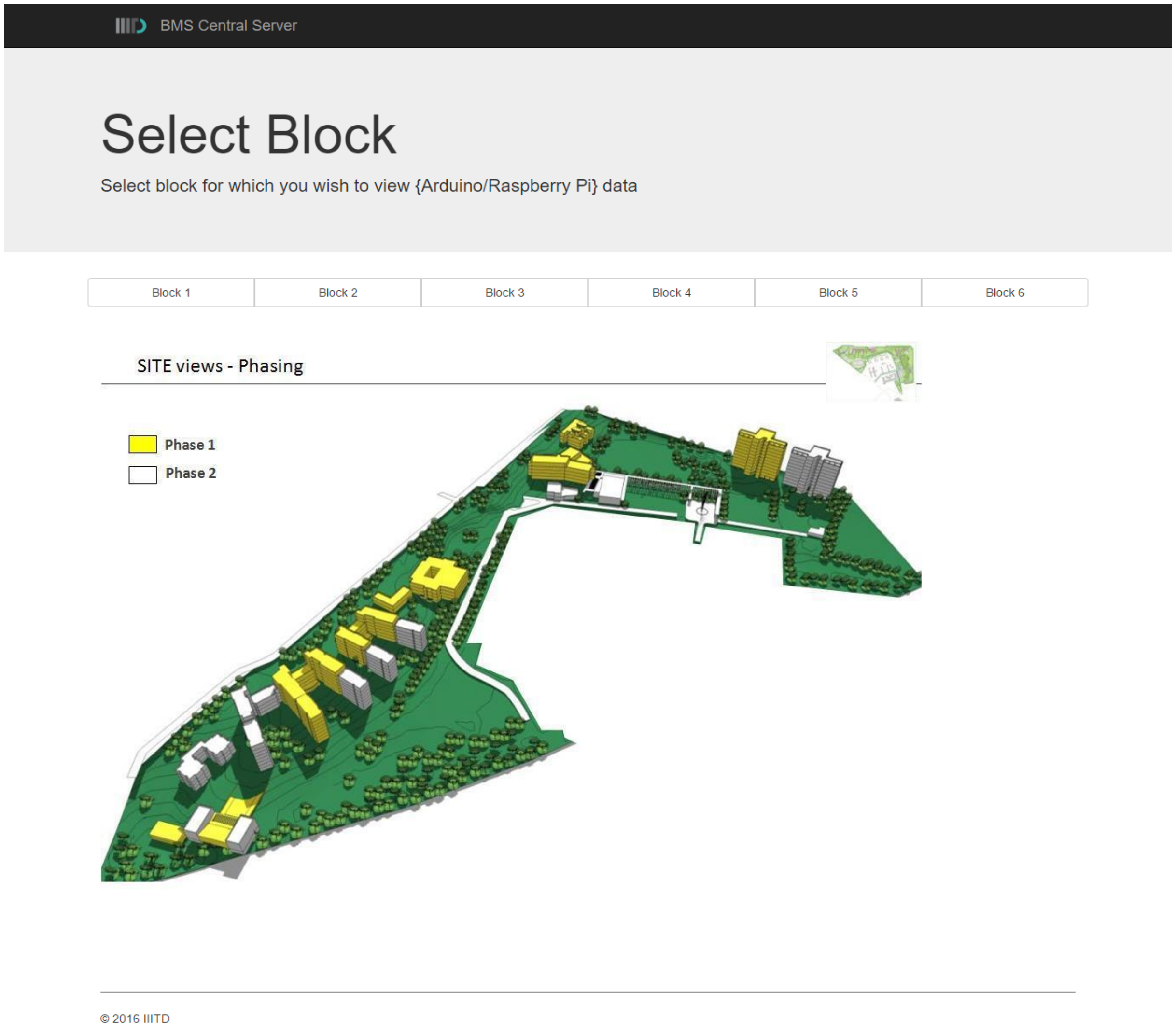
Past records

Time	Device status	Report
00:00:00 AM	NORMAL	Link
00:00:00 AM	NORMAL	Link
00:00:00 AM	NORMAL	Link
00:00:00 AM	NORMAL	Link
00:00:00 AM	NORMAL	Link

© 2016 IIITD

Figure 6.3: Managing a sub-server

This page displays the information of the sub-server. The user can set the global policy for all the microcontrollers connected to the sub-server. The global policy includes the temperature range and time range in which the AHUs should operate. The user can also view the past records of the sub-server. Once the user clicks submit, the data is entered in MySQL database. If the user wants to manage individual microcontroller, the user can navigate through the Arduino section in the home page to reach the following page (figure 6.4).


BMS Central Server

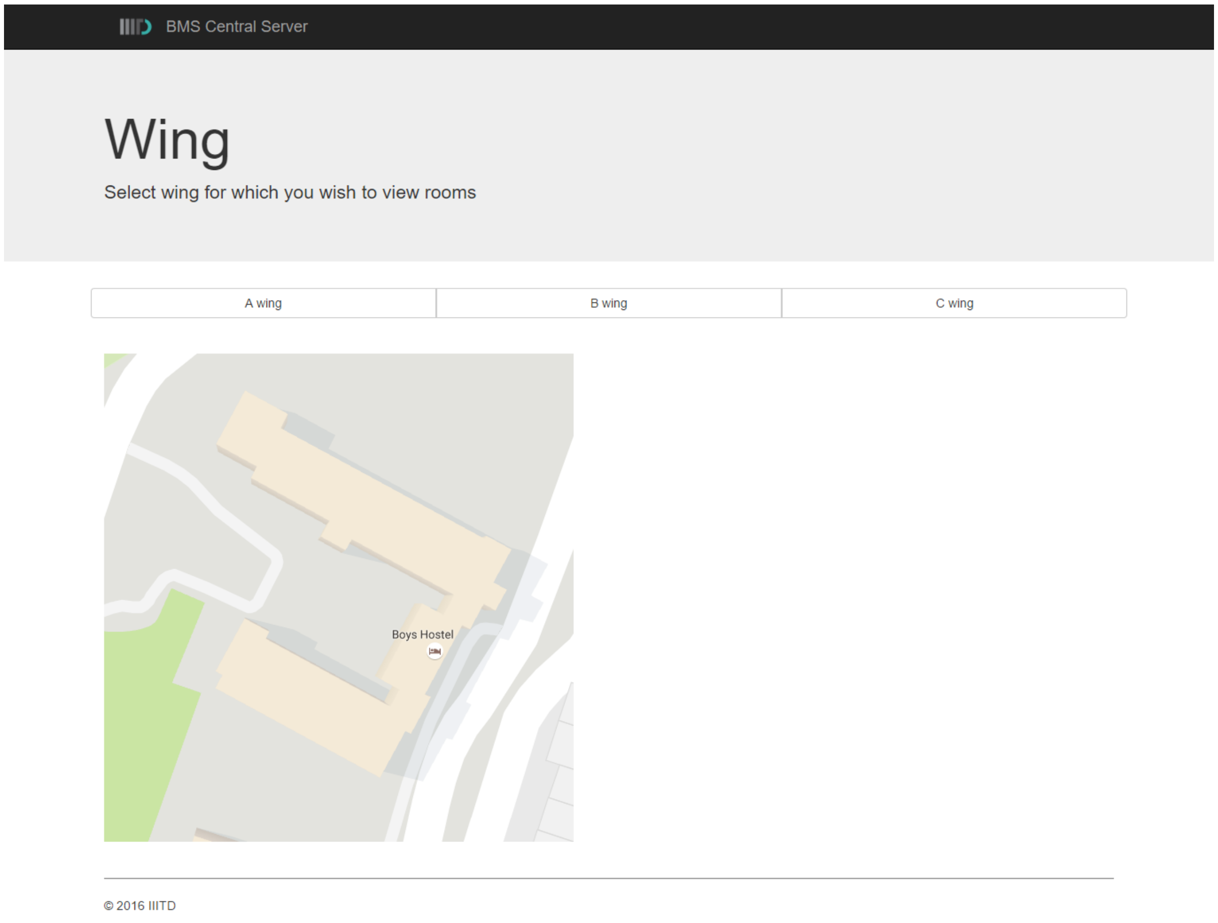
Select Block

Select block for which you wish to view {Arduino/Raspberry Pi} data

© 2016 IIITD

Figure 6.4: Blocks in Phase II

The sub-servers in phase II are distributed in various blocks. In this page the user has to select the block for which the microcontroller needs to be managed. On selecting the block, the user will be directed to the following page (figure 6.5).



Wing

Select wing for which you wish to view rooms

© 2016 IIITD

Figure 6.5: Wings in a block

Depending on the block, the sub-servers are then divided into wings. In this page the user has to select the wing for which the user wants to manage the device. The user will then be redirected to the following page (figure 6.6).

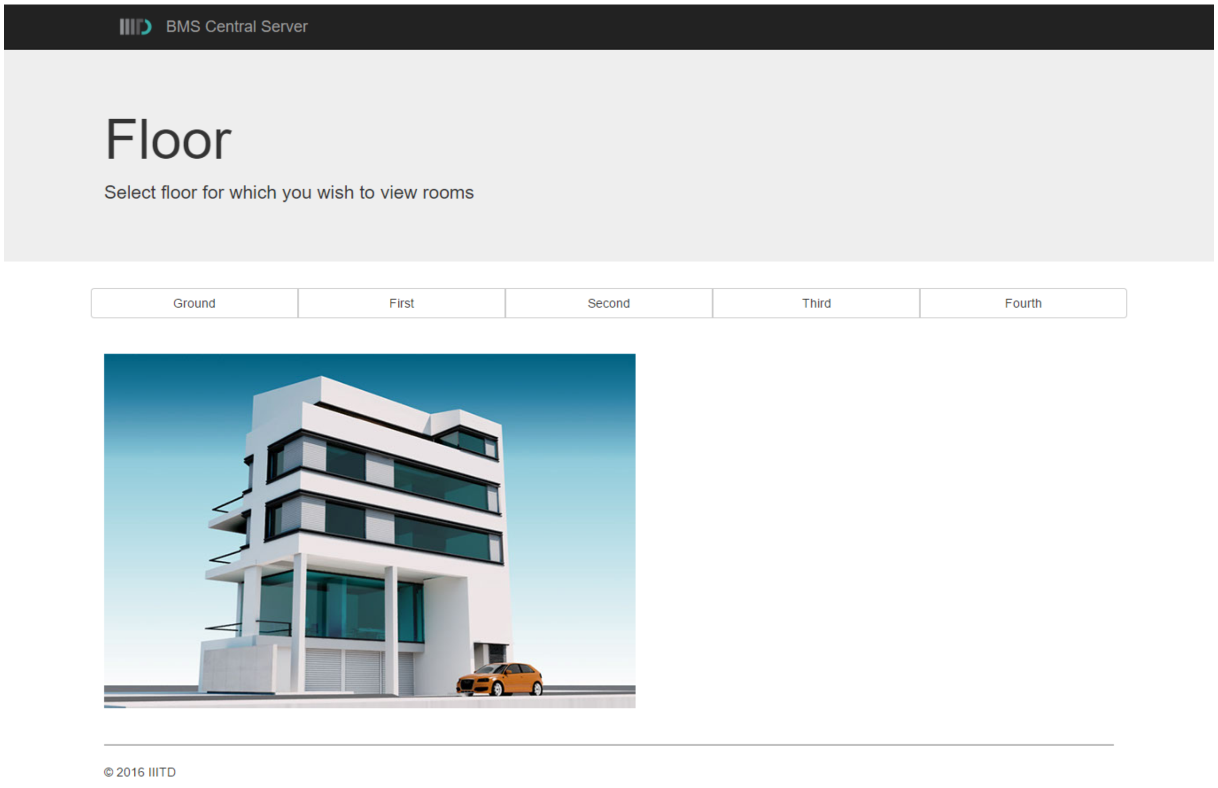
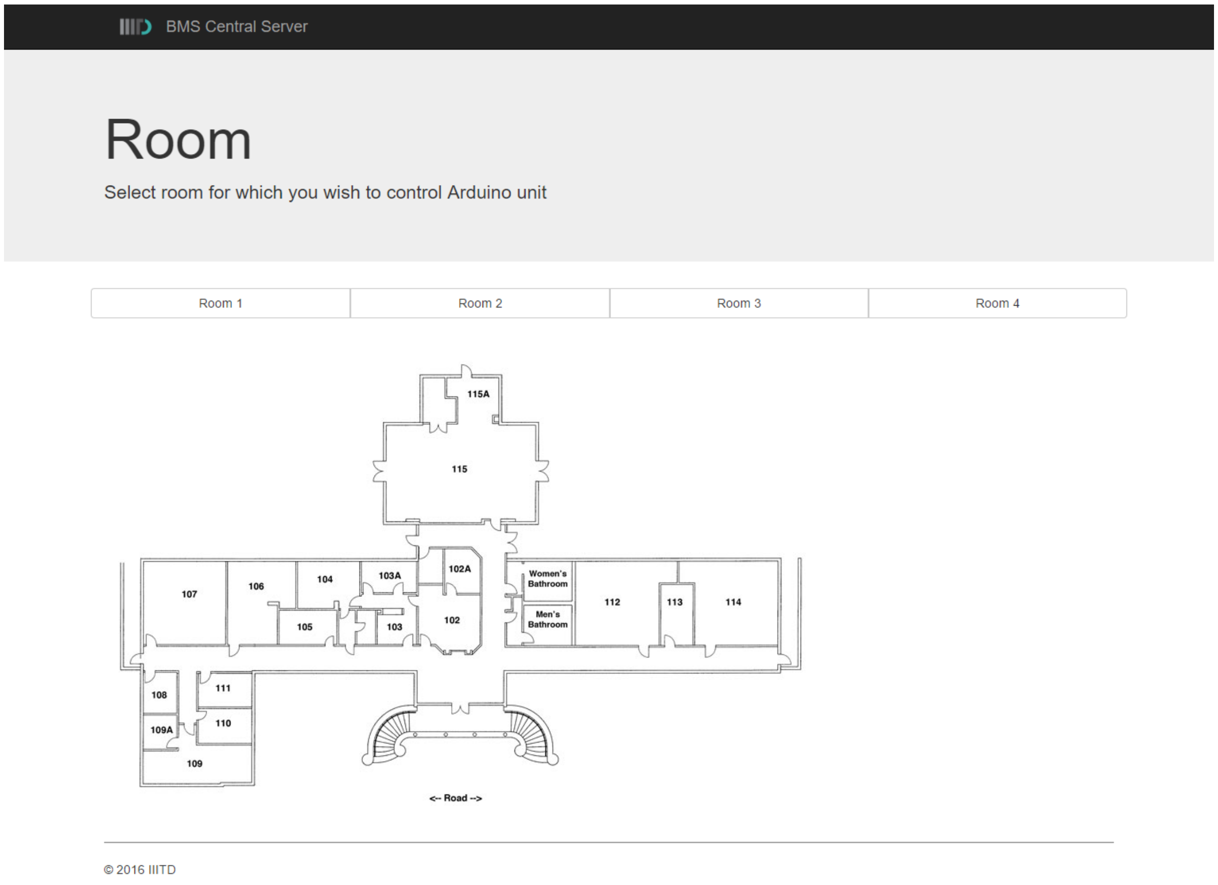


Figure 6.6: Floors in a wing

Depending on the wing, the sub-servers are further divided into different floors. In this page the user has to select the floor for which the microcontroller needs to be managed. On selecting the floor, the user will be directed to the following page (figure 6.7).


BMS Central Server

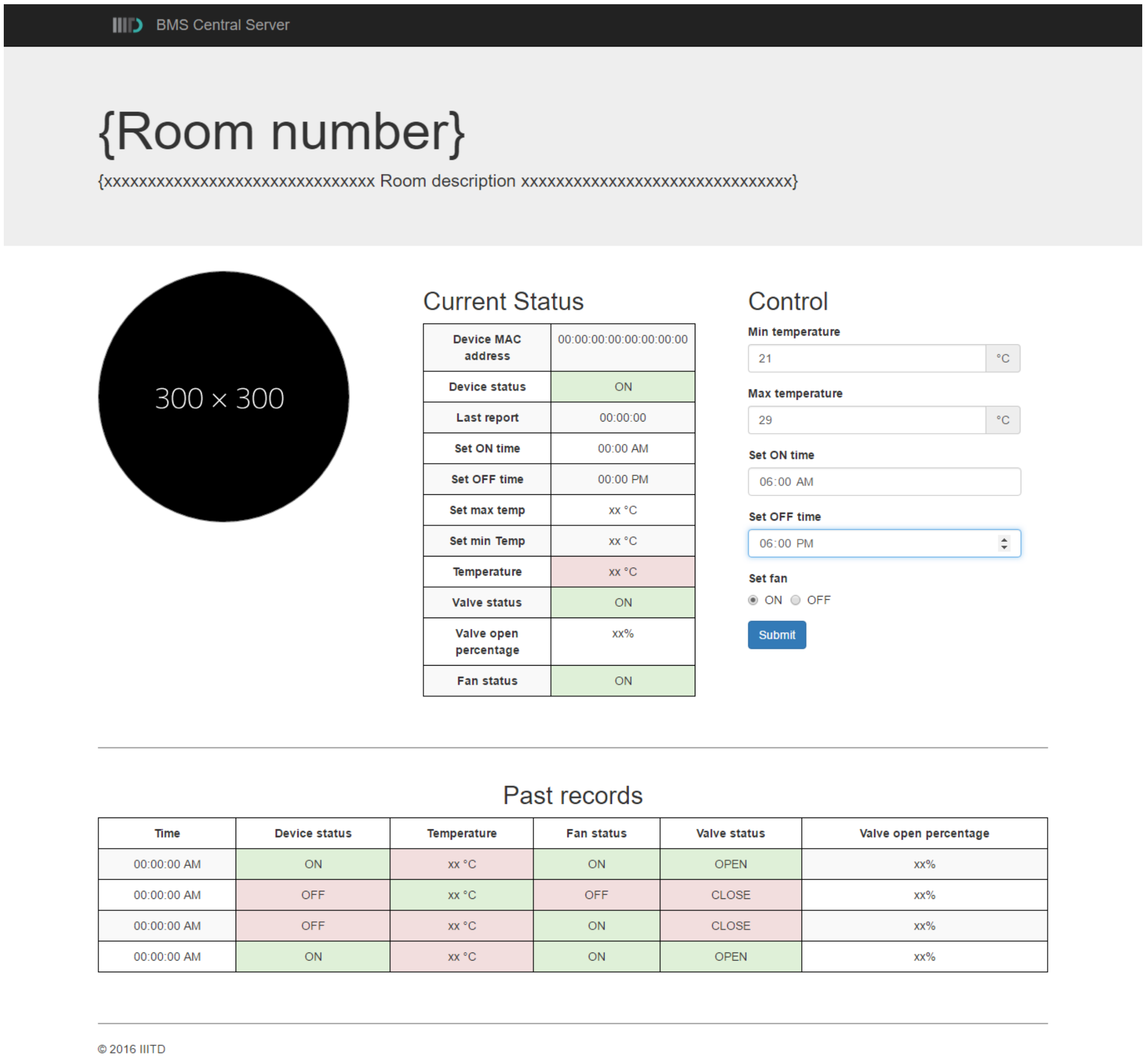
Room

Select room for which you wish to control Arduino unit

© 2016 IIITD

Figure 6.7: Rooms in a floor

In this page the user has to select the room for which the user wants to manage the microcontroller unit. The user will then be redirected to the following page (figure 6.8).



BMS Central Server

{Room number}

{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Room description xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}

300 × 300

Current Status

Device MAC address	00:00:00:00:00:00:00:00
Device status	ON
Last report	00:00:00
Set ON time	00:00 AM
Set OFF time	00:00 PM
Set max temp	xx °C
Set min Temp	xx °C
Temperature	xx °C
Valve status	ON
Valve open percentage	xx%
Fan status	ON

Control

Min temperature: 21 °C

Max temperature: 29 °C

Set ON time: 06:00 AM

Set OFF time: 06:00 PM

Set fan: [✓ ON] OFF

Submit

Past records

Time	Device status	Temperature	Fan status	Valve status	Valve open percentage
00:00:00 AM	ON	xx °C	ON	OPEN	xx%
00:00:00 AM	OFF	xx °C	OFF	CLOSE	xx%
00:00:00 AM	OFF	xx °C	ON	CLOSE	xx%
00:00:00 AM	ON	xx °C	ON	OPEN	xx%

© 2016 IIITD

Figure 6.8: Managing the microcontroller

This page displays the information of the microcontroller. The user can set the policy for the microcontroller. The policy includes the temperature range and time range in which the AHU should operate. The user can also view the past records of the microcontroller. Once the user clicks submit, the data is entered in MySQL database.

The central server then communicates with the python server to pass the command. A Python based task scheduler schedules the task for the sub-server and microcontroller according to the values in the database.

Bibliography

[1] Django. *Django Documentation*, 2014.

[2] Doxygen. *Documenting the code*. Doxygen, 2016.

[3] Get Bootstrap. *Getting started with Bootstrap*, 2016.

[4] Guides, G. *Mastering Markdown*. Github, 2014.

[5] Kumar, D. *Best Practices for building RESTful services*. Infosys, 2015.

[6] Python. *Base HTTP Server*, 2016.

[7] Reitz, K. *The Hitchhiker's Guide to Python*. O'Reily Books, 2016.