FORECASTING AND WEB APPLICATION FOR HOME AUTOMATION

A Project Report
Submitted for the Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

IN

Electrical Engineering
Submitted by

Guttikonda Haritha (15EE01007)

Under guidance of

Dr. P. Balakrishna

School of Electrical Sciences

Indian Institute of Technology Bhubaneswar

April 20

Table of Contents

List of figures

Figure **Name** **Page**

1. Abstract

The Home Automation infrastructure features the automatic control of various household appliances in the advanced metering infrastructure and sophisticated communication technologies which enables the connection of individual smart home systems to a smart grid. In a smart city frame work, price forecasting is required by producers and consumers. Both producers and consumers use day-ahead price forecasts to generate and utilize electricity efficiently. Therefore, accurate price estimates are crucial for producers to maximize their profits and for consumers to maximize their utilities. Forecasting electricity prices is difficult because unlike demand series, price series present such characteristics as non-constant mean and variance and significant outliers. We also assume that the home contains grid connected solar unit. Thus, we also forecast solar energy or irradiance for next day to switch between sources.

This project proposes a smart device that can forecast day-ahead values of Grid Price and Solar Generation and reschedule home appliances to maximize usage of solar energy and minimize power bills. This helps shift the heavy energy load from peak hours to nonpeak hours. Here, we implement a technique to forecast day-ahead electricity prices based on ARIMA models. The historical and usually ill-behaved price series is decomposed using the wavelet transform in a set of better-behaved constitutive series. Then, the future values of these constitutive series are forecast using properly fitted ARIMA models. Once the consumer is aware of the day-ahead grid prices and solar energy, a rescheduling algorithm is used to reschedule heavy and non-critical loads to non-peak hours. In addition to rescheduling, a web application is developed to control loads remotely. Main objective of this project is to establish a smart home environment using open source technologies.

2. Introduction

There are a variety of benefits provided in the emerging smart home infrastructure such as better customer experience and improved energy usage efficiency. For a single user, the smart home technology enables the control of household appliances with the computer and information technologies. Recent technological advances have witnessed the increasing popularity of smart devices, ranging from smart phones, tablets, smart light bulbs, smart refrigerator to smart laundry machines. Multiple devices and household appliances in a home are connected through a home area network such as WiFi. This network also connects to a smart meter which is linked to the power distribution system within the so called advanced metering infrastructure (AMI). The prevailing smart meter is based on advanced system-on chip technology which allows the complex computation to be conducted in an on-chip fashion. Specially, it can be designed to have a *smart controller*. This smart controller can obtain the electricity price from the utility and automatically schedule the household appliances. Main targets for automatic energy scheduling include the monetary cost reduction from users and the energy.

The academic research mostly focuses on more efficient and effective methodologies to schedule household appliances for various purposes. Major targets include the reduction of the electricity bill of customers and peak to average ratio of the power distribution system. This is also known as demand side management from the perspective of power system. A single day can be partitioned into multiple time intervals and the electricity price at one time interval can be different from others. Since a smart controller computes the schedule of the household appliances in advance, it needs to know the future electricity pricing. There are various Forecasting techniques available to predict day-ahead prices of electricity. Smart Home environment also facilitates remote accessing of loads(home appliances). A commercial Home Automation system uses tools like zigbee or wifi to control loads. As we use only open source technologies in this project, various technologies and softwares used to implement this system are discussed in the following chapters.

3. Project Overview

This project is mainly divided into 4 parts

i. *Forecasting Grid Price and Solar Irradiance.*

   In order to reschedule loads, we need to have day-ahead predicted values of solar generation and grid price. The data set used to train and test the algorithm is obtained from 'National Solar Radiation Data Base'

ii. *Load rescheduling.*

   By comparing the costs of running various appliances at different times we sort the usage of non-critical loads into hours where the electricity cost would be minimum.

iii. *Creating User Interface and dynamically operating by taking user input.*

   By creating a web application, user can switch ON or OFF loads remotely. User Interface also alerts user about usage of non-critical loads with the help of pop-up boxes. Any change in load profile will lead to intra-day rescheduling of loads.

iv. *Load control by Arduino interfaced with RStudio*

Every hour load status is sent to Arduino (load controller) via serial communication between RStudio and Arduino.

In this project, we however only discuss Forecasting and User Interface blocks which are explained in following sections in greater depth.

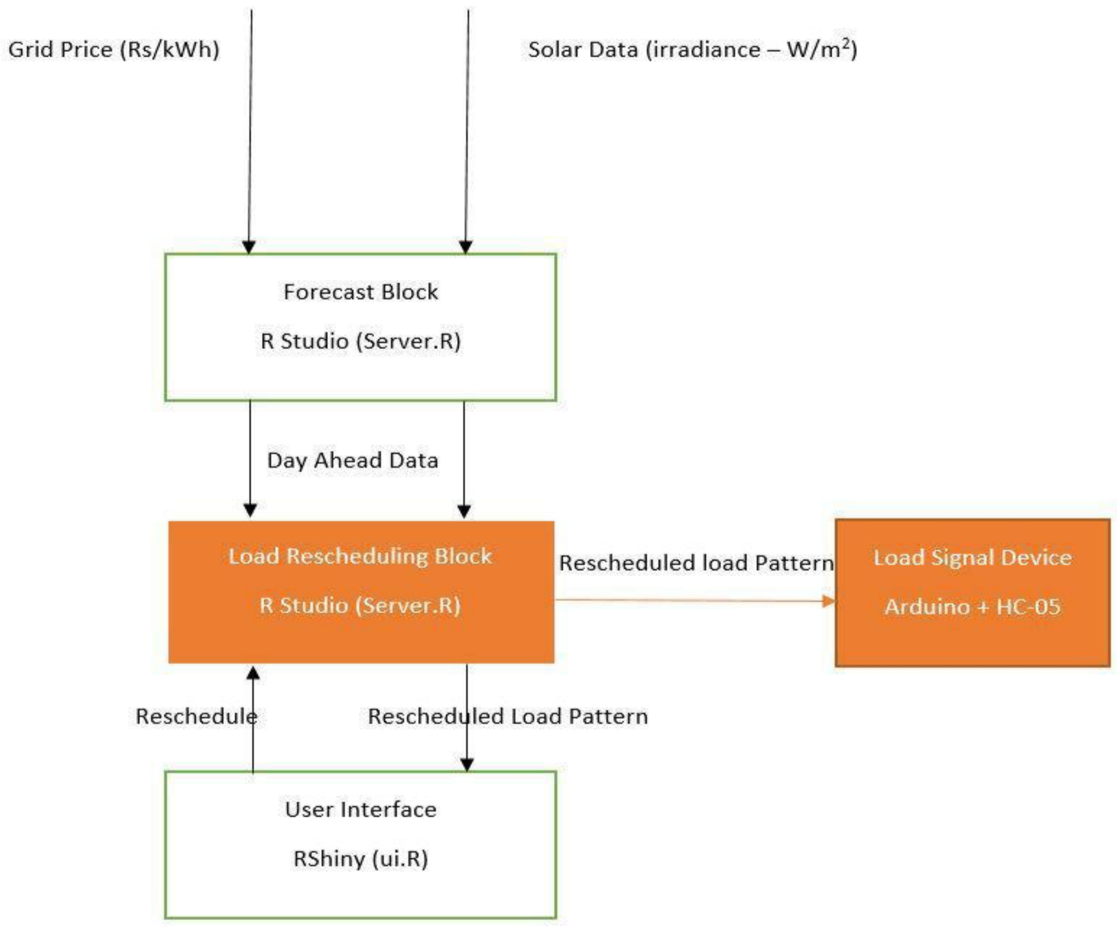


Fig.3.1. *System Block Diagram.*

Above block diagram gives the overview of the project.

4. Forecasting

As we already established why we need forecasting, we shall now see what is forecasting and how we do it. The input data is a time series data of a year-long solar irradiation and grid power cost. Hence we need to do time series forecasting. Time series forecasting is basically using a model to predict future values based on previously observed values. It is widely used for stationery data like weather, stock price and retail sales etc,. Though there are many time series models we use ARIMA model because it only needs past values to predict future values and does not require any feature data.

4.1 ARIMA

ARIMA model is extension of ARMA model. ARMA models are mainly used in time series modelling. It is a class of model that captures a suite of different standard temporal structures in time series data. In ARMA model, AR stands for auto-regression and MA stands for moving average and I stands for Integrated. This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

- **AR**: *Autoregression*. A model that uses the dependent relationship between an observation and some number of lagged observations.
- **I**: *Integrated*. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- **MA**: *Moving Average*. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

4.1.a. Auto-Regressive Time Series Model

Let's understanding AR models using the case below:
The current GDP of a country say x(t) is dependent on the last year's GDP i.e. x(t – 1). The hypothesis being that the total cost of production of products & services in a country in a fiscal year (known as GDP) is dependent on the set up of manufacturing plants / services in the previous year and the newly set up industries / plants / services in the current year. But the primary component of the GDP is the former one. Hence, we can formally write the equation of GDP as:

$$\mathbf{x(t) = alpha * x(t - 1) + error\ (t)}$$

This equation is known as *AR(1) formulation*. The numeral one (1) denotes that the next instance is solely dependent on the previous instance.

4.1.b. Moving Average Time Series Model

Let's take another case to understand Moving average time series model. A manufacturer produces a certain type of bag, which was readily available in the market. Being a competitive market, the sale of the bag stood at zero for many days. So, one day he did some experiment with the design and produced a different type of bag. This type of bag was not available anywhere in the market. Thus, he was able to sell the entire stock of 1000 bags (lets call this as x(t)). The demand got so high that the bag ran out of stock. As a result, some 100 odd customers couldn't purchase this bag. Lets call this gap as the error at that time point. With time, the bag had lost its woo factor. But still few customers were left who went empty handed the previous day. Following is a simple formulation to depict the scenario.

$$\mathbf{x(t) = beta * error(t-1) + error\ (t)}$$

The primary difference between an AR and MA model is based on the correlation between time series objects at different time points. The correlation between x(t) and x(t-n) for n > order of MA is always zero. This directly flows from the fact that covariance between x(t) and x(t-n) is zero for MA models (something which we refer from the example taken in the previous section). However, the correlation of x(t) and x(t-n) gradually declines with n becoming larger in the AR model. This difference gets exploited irrespective of having the AR model or MA model. The correlation plot can give us the order of MA model. Therefore, we plot ACF and PACF to determine p and q values for the ARIMA.

4.1.c. Implementation of ARIMA Model

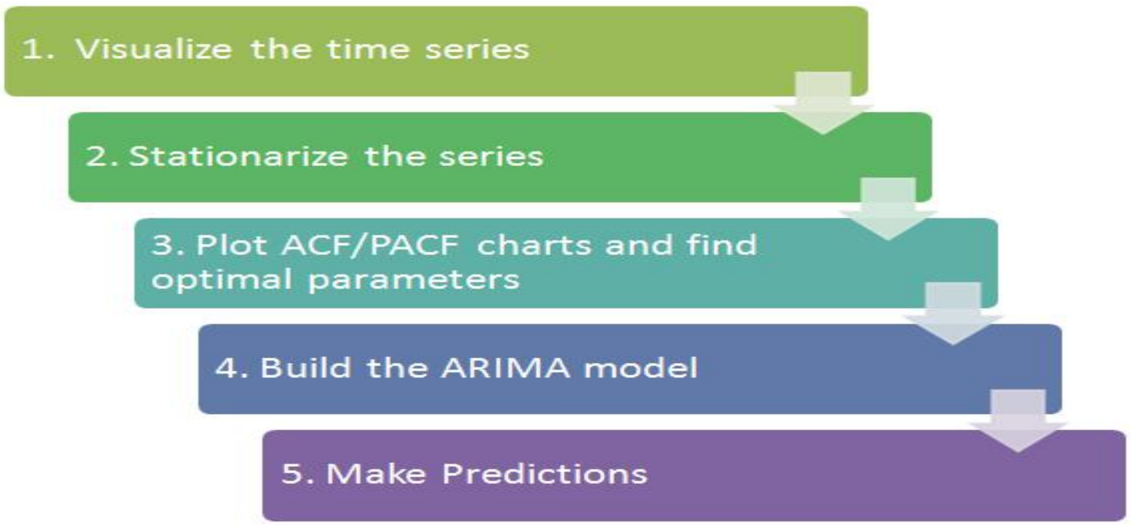

1. Visualize the time series
2. Stationarize the series
3. Plot ACF/PACF charts and find optimal parameters
4. Build the ARIMA model
5. Make Predictions

Fig.4.1. *flow chart of working of ARIMA by Analytics Vidhya, 2015*

For better understanding and observation, we shall consider a data set containing hourly temperature of 4 days which has around 150 data points. We shall divide this into two sets of 126 and 24 each. The first dataset is used to model the ARIMA called training set and the second set is used to test the model called testing set. Testing set is used to check results and calculate errors.

a. Visualise and stationarize data

We have to plot the data initially in order to check whether the data is stationary or not. Because only stationary data can be used for time series analysis. If the data contains any time dependent variance or covariance or mean, we use log function or difference function to make it stationary.
From above plot we can say that the data is fairly stationary. But we still use logarithmic function to eliminate and time dependent covariance and difference the data to minimise variable mean.

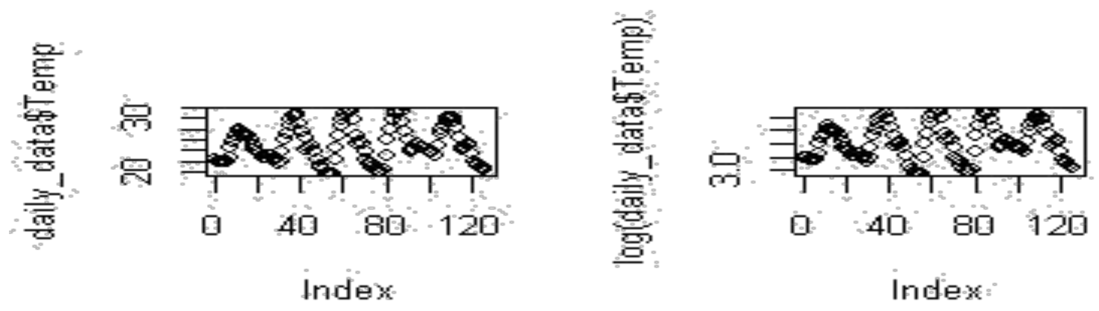


Fig.4.2. *Plot of input() and log(input())*

b. ACF and PACF plots

If the ACF plot is gradually decreasing and PACF plot cuts after n delay then we **should** consider q=n. The same way if PACF is gradually decreasing an ACF cuts off at lag n, then we should consider

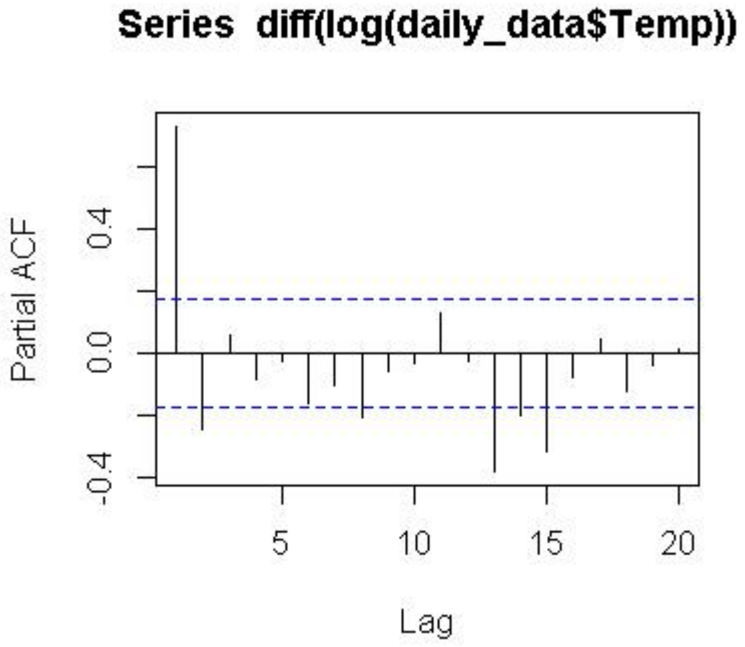


Fig.4.3. *Plot of Partial Auto correlation function*

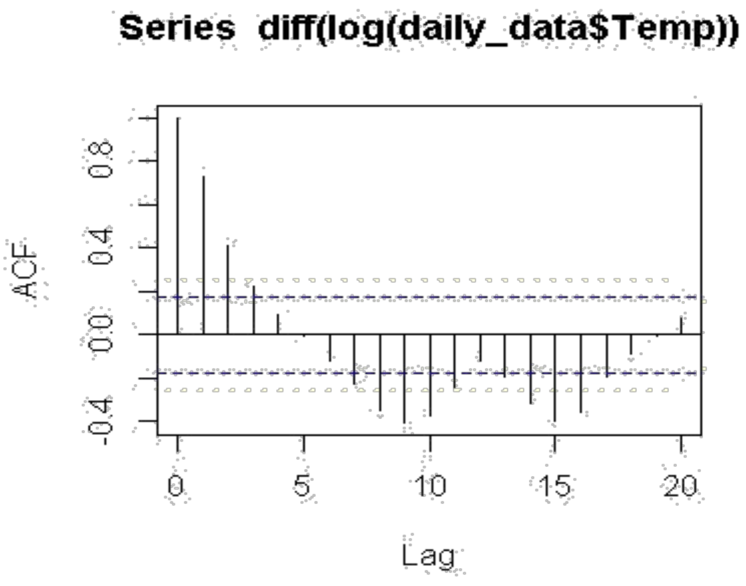


Fig.4.4. *Plot of Auto correlation function*

From the ACF and PACF plots we can see that ACF is not gradually reducing. Which means p=0 and PACF cuts off after lag 1. Hence q=1. As we also differenced the data once, the final order of the ARIMA is (0,1,1)

c. Build ARIMA model

With the parameters in hand, we can now try to build ARIMA model. The value found in the previous section might be an approximate estimate and we need to explore more (p,d,q) combinations.

d. Make predictions

Once we have the model, we can fit the data into our model and predict future values. In this example, we predicted temperature of next 24 hours.

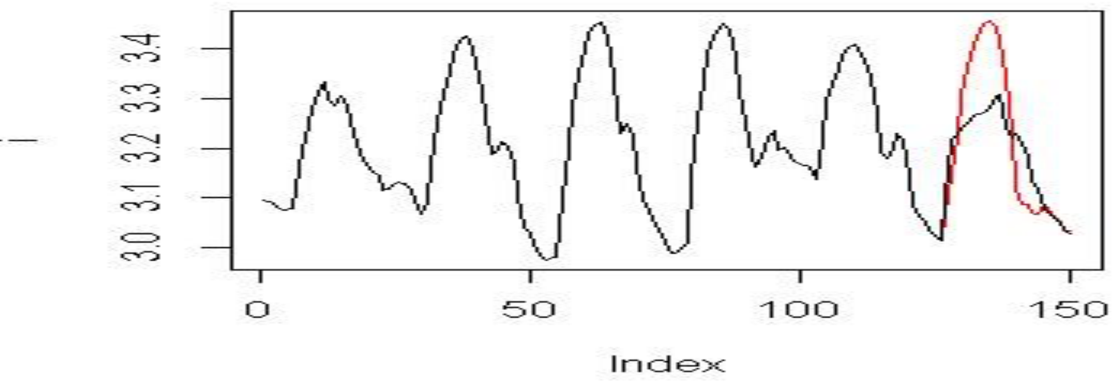


Fig.4.5. *overlap of real and predicted data*

Mean Square error is 9.21% for parameters (0,0,1)

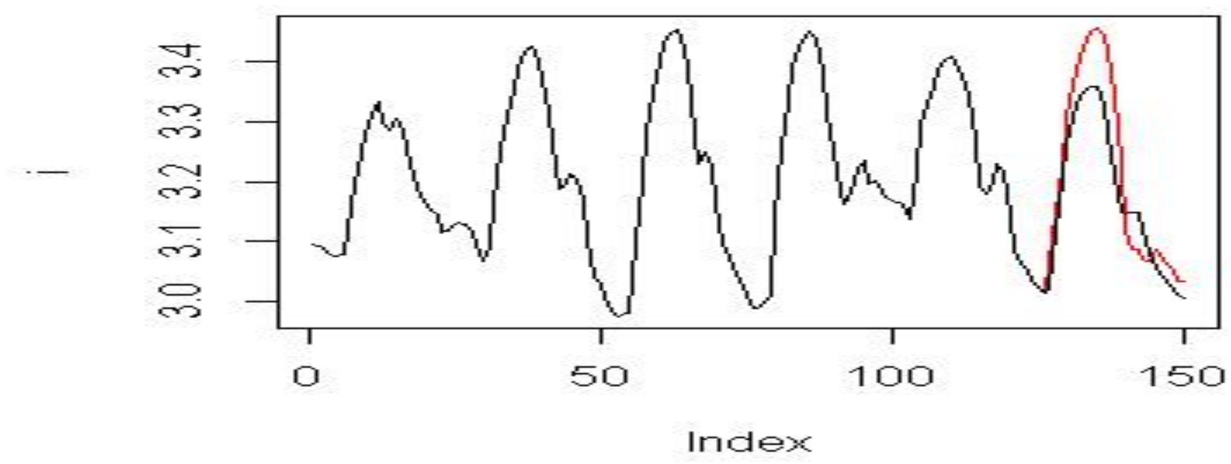


Fig.4.6. *overlap of real and predicted data*

Mean Square error is 4.57% for parameters (0,1,1) which we estimated from the plots.

5. Web Application

The power outlets in our homes have switches and sockets with wired connections. A person has to physically move and operate the switch either on or off and apply or control power to the home appliance. The person who is away from home cannot either control appliance or know the current status of the same and this might result in wastage of electrical energy. People may experience electrical shock in case the connections are exposed. An automated system is needed to eliminate bulky wired connections of switches and sockets in the power outlet. The design of automated system should be simple and easy to control the home appliances remotely and also able to monitor their status at the same time.

In this project, as one part of it is already in Rstudio, we use Rshiny Package to create a Graphic User Interface. R shiny package is used to create interactive web applications that can be stand-alone. It is used to build dashboards and can also be extended with CSS themes, HTML widgets or Java Script. Rshiny is easy to write and comes with lot of in-built packages. In this project we use Rshiny package to build a user dash board that can switch ON or OFF loads and display alert messages that take user inputs. Any Shiny web application has two files ui.R and server.R. ui.R contains all the elements that are to be displayed on the Web App whereas server.R contains all the code and renders the output to ui.R.

5.1. Ui.R

Shiny uses the function fluidpage to create a display that automatically adjusts to the dimensions of your user's browser window. You lay out the user interface of your app by placing elements in the fluidpage function.
The user interface can be broadly divided into three categories:

- **Title Panel:** The content in the title panel is displayed as metadata, as in top left corner of above image which generally provides name of the application and some other relevant information.

- **Sidebar Layout:** Sidebar layout takes input from the user in various forms like text input, checkbox input, radio button input, drop down input, etc. It is represented in dark background in left section of the above image.

- **Main Panel:** It is part of screen where the output(s) generated as a result of performing a set of operations on input(s) at the server.R is / are displayed.

These functions place content in either the sidebar or the main panels. The sidebar panel will appear on the left side of your app by default. titlePanel and sidebarPanel create a basic layout for your Shiny app, but you can also create more advanced layouts. You can use navbar to give your app a multi-page user interface that includes a navigation bar. Or you can use fluidRow and fluidColumn to build your layout up from a grid system. As we use shiny alert messages, we have to include useshinyalert() in the fluidpage of the ui.

We used shinydashboard package to create a dashboard theme web application. The syntax for using dashboard is little different. We use dashboard page instead of fluidpage. dashboardHeader, dashboardSidebar, dashboardBody are used instead of titlePanel and sidebarLayout. Unlike normal shiny ui, in dashboard we have to use useshinyalert() in the body of the dashboard. dashboardSidebar include action buttons that control the appliances. A load profile button is used to display complete load profile.

Packages used in shiny ui.R are 1. Shiny

2. Shinydashboard

3. shinyalert

Below figure shows the Dashboard interface of the web application

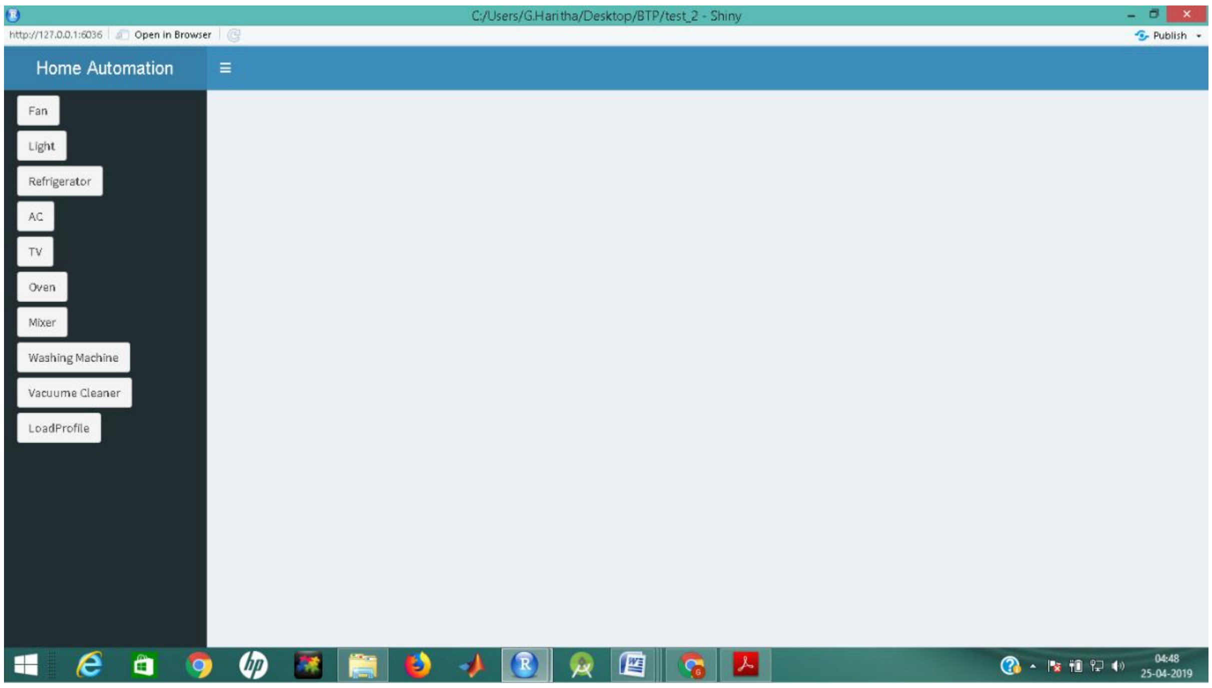


Fig.5.1. GUI

5.2. Server.R

Shiny Server is a back end program that makes a big difference. It builds a web server specifically designed to host Shiny apps. With Shiny Server you can host your apps in a controlled environment, like inside your organization, so your Shiny app (and whatever data it needs) will never leave your control. You can also use Shiny Server to make your apps available across the Internet when you choose. Shiny Server will host each app at its own web address and automatically start the app when a user visits the address. When the user leaves, Shiny Server will automatically stop the app.

In our project all the prediction and rescheduling blocks are inside server file. As shiny code is compiled and output is obtained it is rendered to user interface via renderText, renderPlot or renderDatatable functions. The inputs taken in ui.R file are accessed using $ operator (input$InputName). The outputs are also referred using the $ operator (output$OutputName).We used many libraries inside server file like readxl, ggplot2, forecast, tseries, shiny, shinyalert. Shiny code includes observeEvent functions that react to the action buttons present in the user interface. Any action that has to be taken at the click of action button must be specified in its own observeEvent function

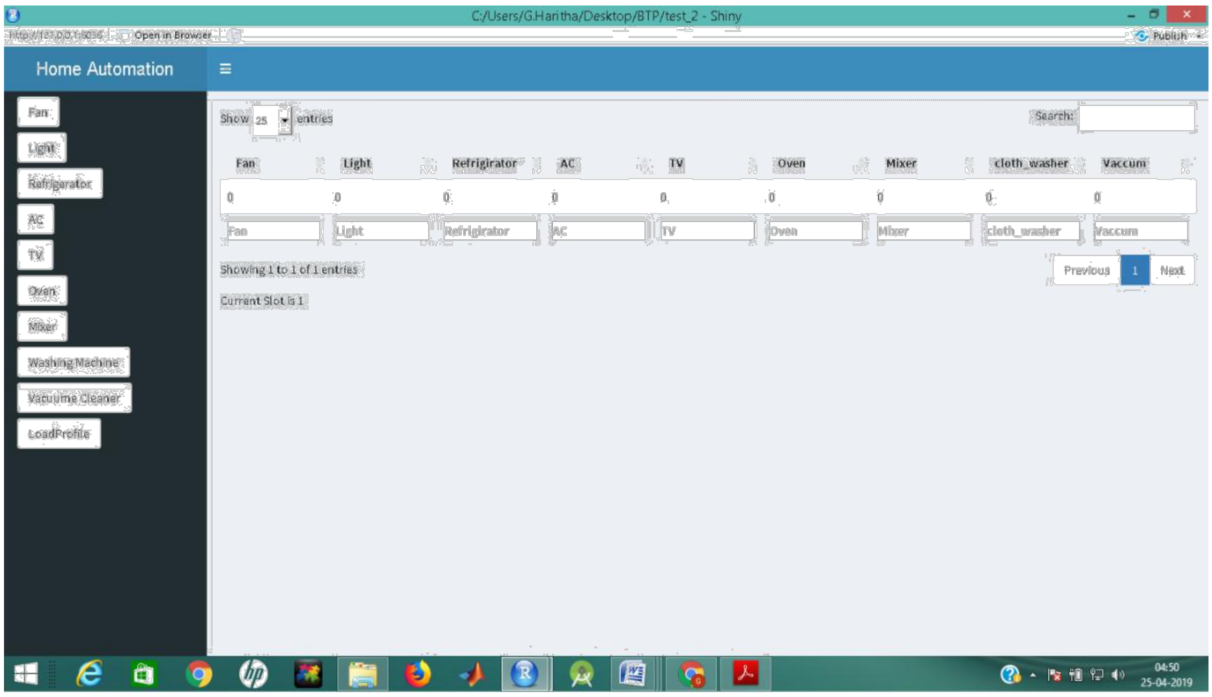
5.3. Working

Rshiny is extremely easy to work with. Once we create our webapp with both ui.R and server.R, we can run the app. As we run app, a user interface opens and all the ui elements show on the screen while server code is running. Once the server code is compiled and output is obtained, the output is sent to ui using render function and displayed on user interface. Inputs and outputs are accessed by using $ symbol in ui and server. Our system has nine load buttons and one loadprofile button. Three out of nine buttons are considered critical loads and the other non critical loads. Critical loads are Fan, Light, Refrigerator. Non-critical loads are Oven, TV,AC, Mixer, Washing Machine, Vacuum Cleaner.

Code in the ui for output will be like “dataTableOutput("load")”

Code in the ui for output will be like “output$load <- renderDataTable({ status_table})”

Below is figure of how GUI looks after server is compiled and output is rendered to ui.



Show 25 entries

Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
0	0	0	0	0	0	0	0	0

Showing 1 to 1 of 1 entries

Current Slot is 1

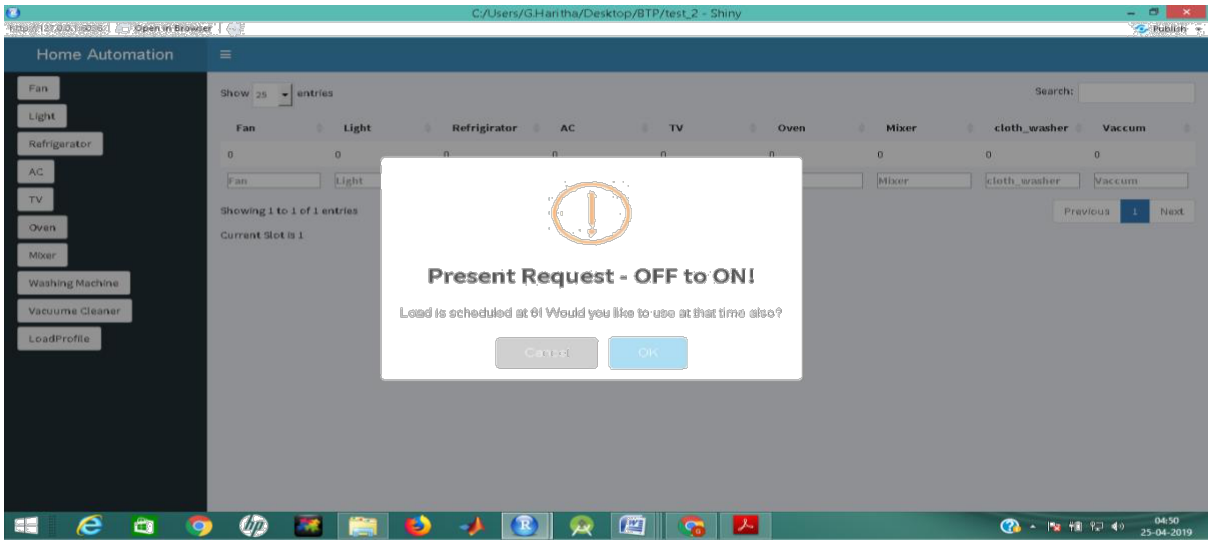
Fig.5.2. GUI after output

5.3.a. Switch OFF to ON

When a user wishes to turn ON a critical load, it directly switches ON as we are not rescheduling critical loads. But when the user tries to switch ON a load that is non-critical, a alert message appears on screen asking user if he/she would like to use the appliance later also. If user clicks OK, load status changes in current slot and appliance load does not change in later hours. But in case if the user does not wish to use the appliance later and want to use it now, then the load status of the appliance is changed to ON in the current slot and OFF in the later slot. As there is change in the load profile, we reschedule all loads from the next hour to accommodate changes.

For this mechanism to work the clicking of the load button event must be observed in the server side. In critical loads status just changes from 0 to 1. Whereas in non-critical loads, a shiny alert message pops and ask user to confirm their usage in later hours. For this a

shinyalert() function is used and the output is accessed by using a callback function.



Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
0	0	0	0	0	0	0	0	0

Showing 1 to 1 of 1 entries

Current Slot is 1

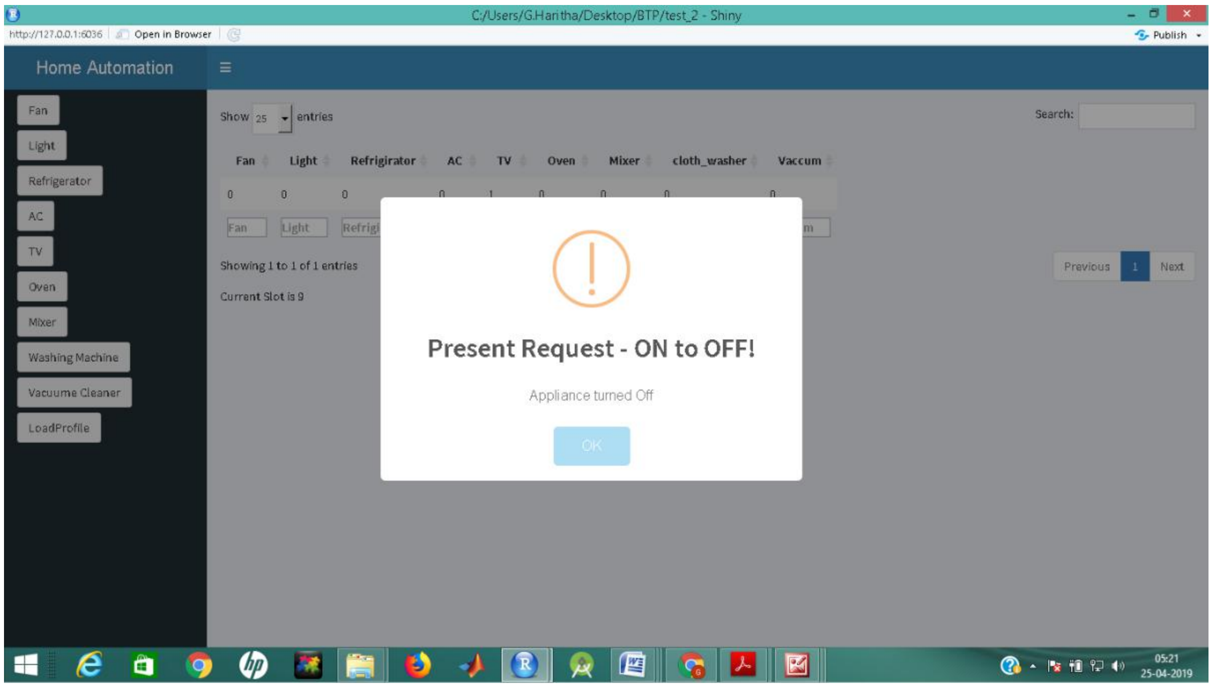
Present Request - OFF to ON!

Load is scheduled at 6! Would you like to use at that time also?

Fig.5.3. GUI Notification OFF to ON

5.3.b. Switch ON to OFF

When user wishes to switch OFF any load, if the load is critical it is switched OFF immediately. Whereas for non-critical loads, a alert message shows up to get user confirmation to switch OFF.



Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
0	0	0	0	1	0	0	0	0

Showing 1 to 1 of 1 entries

Current Slot is 9

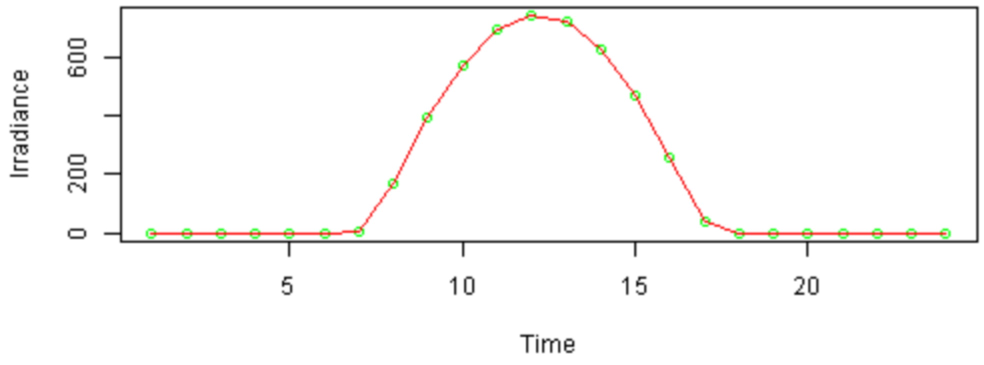
Present Request - ON to OFF!

Appliance turned Off

Fig.5.4. GUI Notification ON to OFF

6. Result

Forecasting output of Solar Grid Price and Solar Irradiance



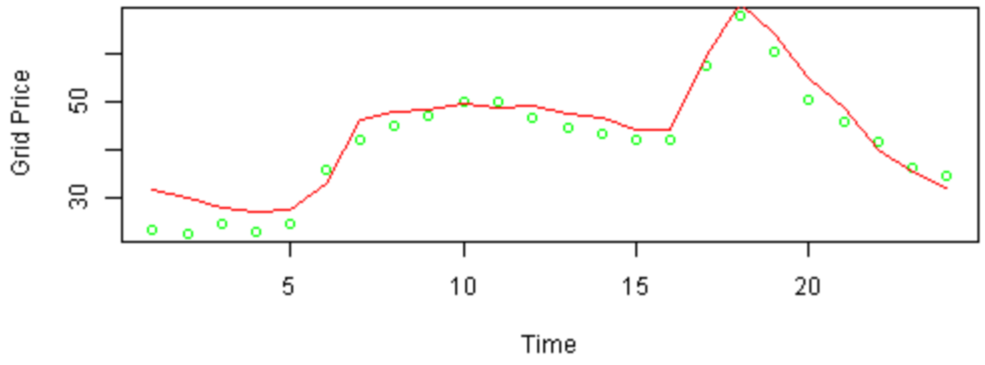
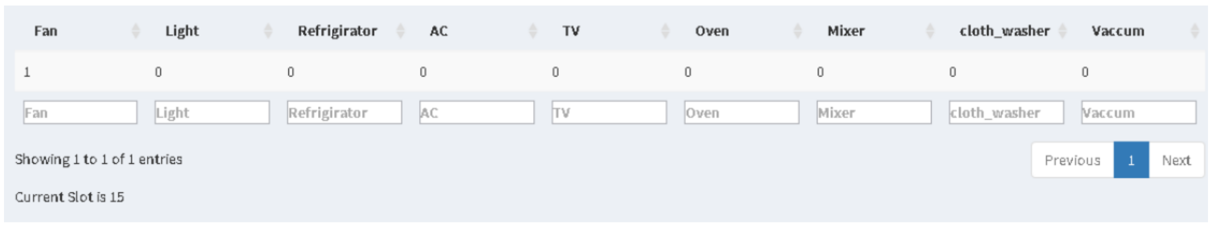


Fig.6.1. Overlap of predicted and real data

Let us consider few test cases to observe results.

1. Critical Load ON->OFF

At slot 15 when light is switched OFF

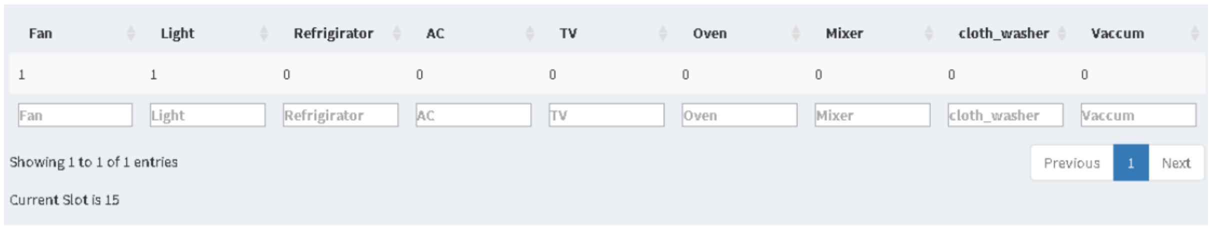


Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
1	0	0	0	0	0	0	0	0
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum

Showing 1 to 1 of 1 entries

Previous 1 Next

Current Slot is 15



Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
1	1	0	0	0	0	0	0	0
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum

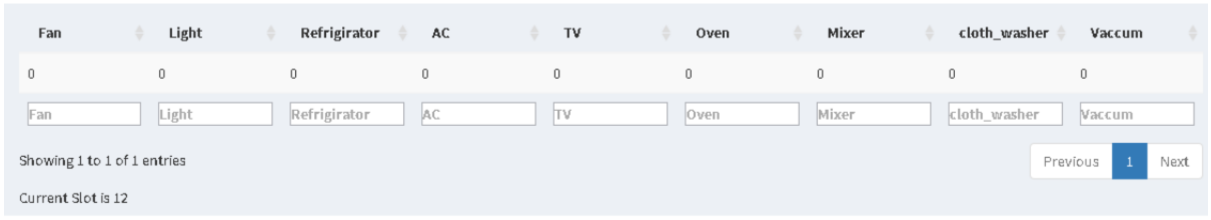
Showing 1 to 1 of 1 entries

Previous 1 Next

Current Slot is 15

2. Critical Load OFF->ON

At slot 12 when fan is switched ON

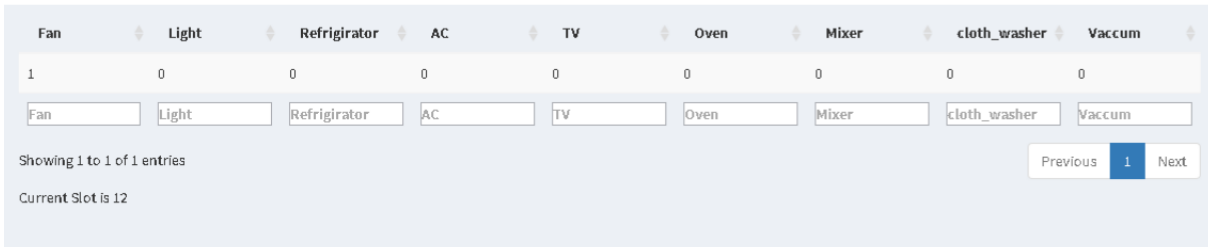


Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
0	0	0	0	0	0	0	0	0
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum

Showing 1 to 1 of 1 entries

Previous 1 Next

Current Slot is 12



Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
1	0	0	0	0	0	0	0	0
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum

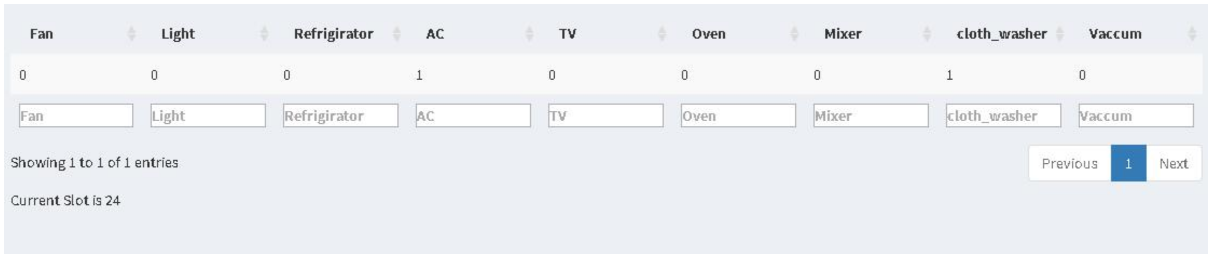
Showing 1 to 1 of 1 entries

Previous 1 Next

Current Slot is 12

3. Non-Critical Load ON->OFF

Washing Machine is switched OFF at slot 24

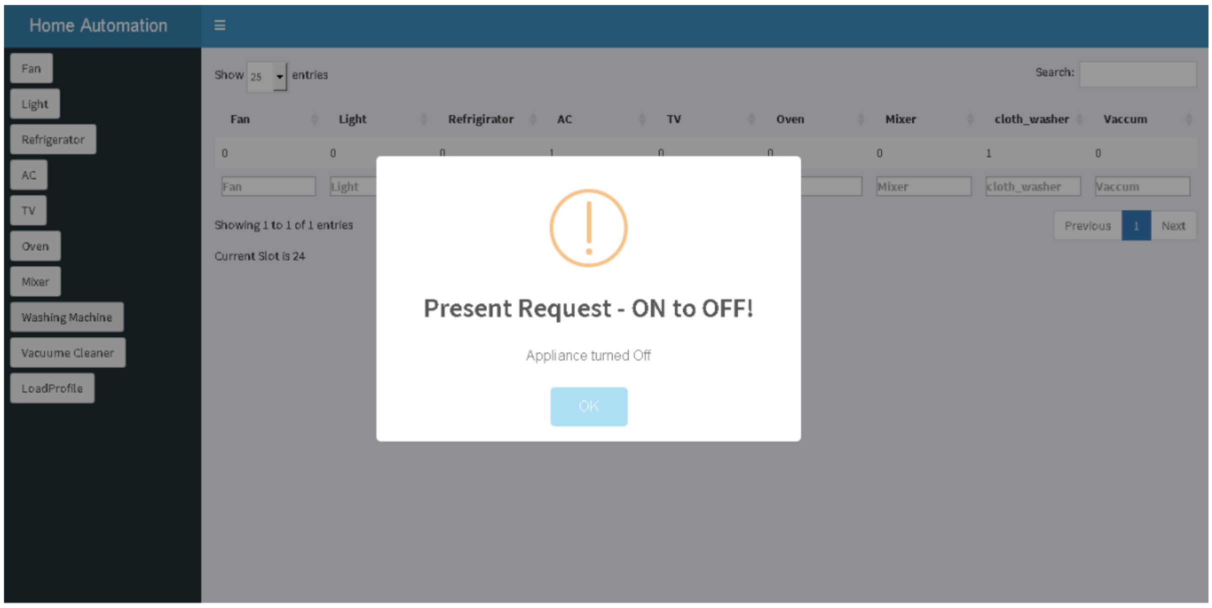


Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
0	0	0	1	0	0	0	1	0
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum

Showing 1 to 1 of 1 entries

Previous 1 Next

Current Slot is 24



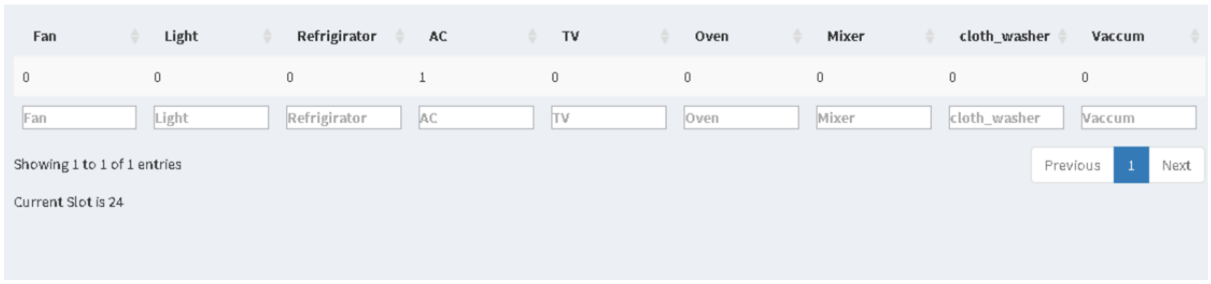
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
0	0	0	1	0	0	0	1	0

Showing 1 to 1 of 1 entries

Current Slot is 24

Present Request - ON to OFF!

Appliance turned Off



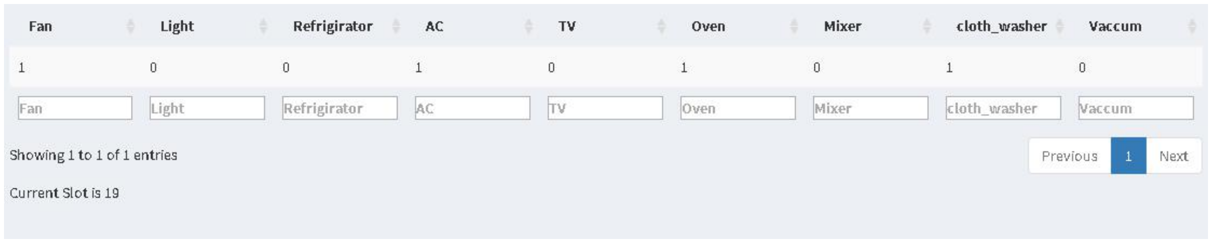
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
0	0	0	1	0	0	0	0	0

Showing 1 to 1 of 1 entries

Current Slot is 24

4. Non_Critical Load OFF->ON

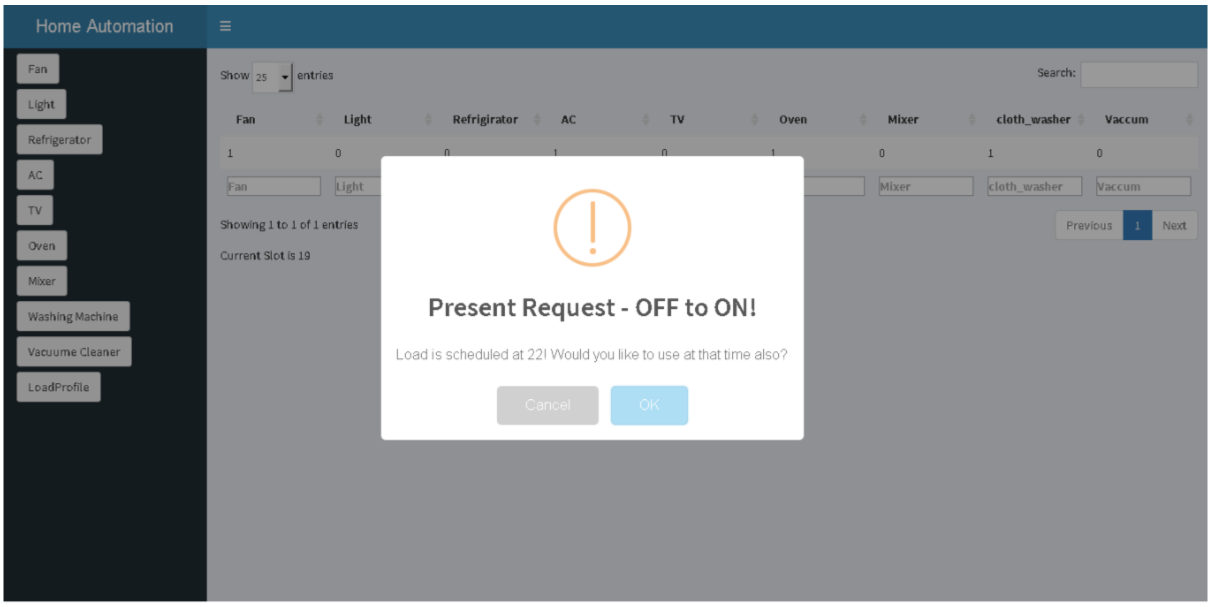
Mixer is switched ON at slot 19



Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
1	0	0	1	0	1	0	1	0

Showing 1 to 1 of 1 entries

Current Slot is 19



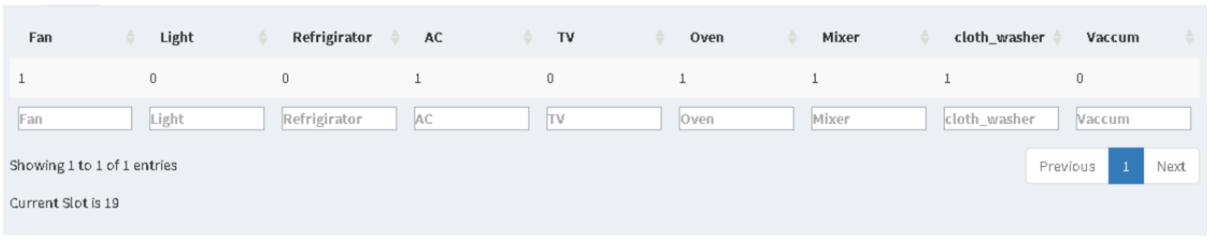
Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
1	0	0	1	0	1	0	1	0

Showing 1 to 1 of 1 entries

Current Slot is 19

Present Request - OFF to ON!

Load is scheduled at 22! Would you like to use at that time also?



Fan	Light	Refrigirator	AC	TV	Oven	Mixer	cloth_washer	Vaccum
1	0	0	1	0	1	1	1	0

Showing 1 to 1 of 1 entries

Current Slot is 19

7. Conclusion

A basic home automation model is implemented using open source technologies. Though it comes with lot of restrictions this model can be seen as a prototype for implementing bigger and commercial home automation devices. Few challenges faced during the project are

1. lack of enough documentation for many Rshiny functions

2. Rshiny when run as app does not show variable values, which makes it difficult to debug

3. limitation in user templates (without CSS)

8. References

[1] Day-Ahead Electricity Price Forecasting Using the Wavelet Transform and ARIMA Models Antonio J. Conejo, Fellow, IEEE, Miguel A. Plazas, Rosa Espínola, Student Member, IEEE, and Ana B. Molina

[2] Short-term hourly price forward curve prediction using Neural network and hybrid ARIMA-NN model Robert Skopal Department of Applied Mathematics VSB-TUO, Faculty of Electrical Engineering and Computer Science Ostrava, Czech Republic robert.skopal@vsb.cz

[3] ARTIFICIAL NEURAL NETWORK FOR LOAD FORECASTING IN SMART GRID HAO-TIAN ZHANG, FANG-YUAN XU, LONG ZHOU Energy System Group'City University London'Northampton Square'London'UK E-MAIL: abhb@city.ac.uk, abcx172@city.ac.uk, long.zhou.1@city.ac.uk