Final Year Project Report

Electricity Conservation and Speculation

BESE (Batch: 2012-13)

Project Advisor

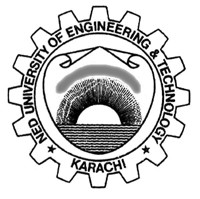
Dr Najmi Haider

Chairman of CS and IT Department  
NEDUET

Submitted by

M. Aamir Ali SE-067

Nida Masood SE-044



DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

NED University of Engineering and Technology, Karachi.

Preface

The project aims to conserve electricity by informing the electricity consumer about the way he/she uses it, because the main reason people get over budget electrical bills is that they don't know how they're using it, and hence, the project analyzes the trend in your electrical consumption and builds a formula-like model for it. In Pakistan people are unaware of what makes their electricity meters tick, from large industries to small factories, people are facing difficulties with their electricity budget; hence if we are able to see something coming, we can very much prepare for it if we receive an early warning of something rather than to suffer from unforeseen consequences.

An industry has to have precise reports, and a close eye regarding expenses therefore using prediction models for electrical consumptions will change the way we look at our expenses, the way we estimate or forecast current liabilities, because the better we understand so, the better we can utilize something.

ACknowledgment

First of all, we thank Almighty Allah who gives us the strength and ability to think, work and deliver what we are assigned to do. Secondly, we must be grateful to our internal supervisor Dr. Najmi Haider who guided us in this project. We also acknowledge our teachers who guided, taught and helped us during our study period. We would also like to thank all departmental staff and university staff, who had assisted us during our stay at the university.

group Members



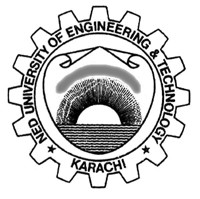
Back-end developer, database designer, systems-interconnector, Hardware designer and developer, and system tester.

Achievements: Developed Android Open Source Project OS, ITEC Python Experts Manager (2015), developed EMOS with Amjad Ali (2015)

Muhammad Aamir Ali SE-067  
Arduino programmer, Python developer and hardware designer   
Immediate Contact: (+923452108041, aamir00@hotmail.com)



Nida Masood Se-044  
Web User Interface designer and developer   
Immediate Contact: (+923333666019, nidaakhan35@gmail.com)



# *NED University of Engineering and Technology, Karachi*

# DEPARTMENT OF COMPUTER SCIENCE &

**INFORMATION TECHNOLOGY**

CERTIFICATE OF COMPLETION

This is to certify that the following students

M Aamir Ali SE-067

Nida Masood SE-044

have successfully completed their final year project titled

Electricity conservation and speculation

in the partial fulfillment for the requirements of the Degree of Bachelor of Computer Science & Information Technology/Bachelor of Software Engineering during the academic session 2012-2013.

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Advisor Name of Chairman

Abstract

This report aims to explain the working of the final year project Electricity conservation and speculation.

The introduction section of the report describes the understanding of systems and subsystems used in this project, so that the reader may grasp the point of view of the writer, as it contains the information on the hardware and software used and why was it used. Moreover, it describes why was the project needed, where can it be used and it features the type of inputs the project needs in order to fulfill it's requirements along with the outputs generated

The second section of the report, which is chapter 3 and 4, will lead to and show the usage scenario of the project; about where and how does the project exactly fit, such as what tasks does it perform. The further we go in this section, details of what the system is composed of as a big picture followed by the decomposition of its components and why were they needed, data handling, data flow , inner working and mechanism of overall architecture.

The third section of report, consisting of chapters 5 through 8, focus on the outputs of the system in detail, it's performance, and why was prototyping necessary, what challenges and system limitations compelled us to build it.

Table of contents

[1 Introduction 10](#_Toc468181663)

[.....1.1 Goals and objectives 10](#_Toc468181664)

[.....1.2 System statement of scope 10](#_Toc468181665)

[.....1.3 System context 11](#_Toc468181666)

[.....1.4 Theoretical Background 12](#_Toc468181667)

[.....1.5 Technology & Tools/hardware components (used in the Project) 13](#_Toc468181668)

[.....1.5.1 1.5.1 Raspberry-pi 14](#_Toc468181669)

[.....1.5.2 1.5.2 Python 16](#_Toc468181670)

[.....1.5.3 1.5.3 Java Script 17](#_Toc468181671)

[.....1.5.4 1.5.4 Mongo Database 18](#_Toc468181672)

[.....1.5.5 1.5.5 Arduino 19](#_Toc468181673)

[2 Usage Scenario / User Interaction 20](#_Toc468181674)

[.....2.1 User profiles 20](#_Toc468181675)

[.....2.2 Use-cases 21](#_Toc468181676)

[.....2.3 Special usage considerations 22](#_Toc468181677)

[3 Functional and Data Description 23](#_Toc468181678)

[.....3.1 System Architecture 23](#_Toc468181679)

[.....3.1.1 Architecture model 25](#_Toc468181680)

[.....3.1.2 Subsystem/modules overview 32](#_Toc468181681)

[.....3.2 Data Description 33](#_Toc468181682)

[.....3.2.1 Major data objects 34](#_Toc468181683)

[.....3.2.2 System level data model 35](#_Toc468181684)

[.....3.3 System Interface Description 38](#_Toc468181685)

[.....3.3.1 External machine interfaces 38](#_Toc468181686)

[.....3.3.2 External system interfaces 38](#_Toc468181687)

[.....3.3.3 System Limitations 38](#_Toc468181688)

[4 Subsystem/module Description 39](#_Toc468181689)

[.....4.1 Web UI 40](#_Toc468181690)

[.....4.1.1 Subsystem scope 40](#_Toc468181691)

[.....4.1.2 Web UI flow diagram 41](#_Toc468181692)

[.....4.1.3 Data Visualization 42](#_Toc468181693)

[.....4.1.4 Web Server 43](#_Toc468181694)

[.....4.2 Forecast and Prediction from Historical Data 45](#_Toc468181695)

[.....4.3 Arduino Data Generation 47](#_Toc468181696)

[.....4.3.1 Arduino Data Description 48](#_Toc468181697)

[.....4.3.2 Arduino Scope 49](#_Toc468181698)

[.....4.4 Python Time Series 51](#_Toc468181699)

[.....4.4.1 Time Series 51](#_Toc468181700)

[.....4.4.2 Scope of Python Time Series 53](#_Toc468181701)

[5 Behavioral Model and Description 54](#_Toc468181702)

[.....5.1 Description for system behavior 54](#_Toc468181703)

[.....5.1.1 Events/interrupts 54](#_Toc468181704)

[.....5.1.2 States 54](#_Toc468181705)

[.....5.2 State Transition Diagram 55](#_Toc468181706)

[.....5.3 Control specification 56](#_Toc468181707)

[6 System Estimates and Actual Outcome 57](#_Toc468181708)

[.....6.1 Historical data used for estimates 57](#_Toc468181709)

[.....6.2 Estimation techniques applied and results 58](#_Toc468181710)

[.....6.3 System Resources (Required and Used) 59](#_Toc468181711)

[.....6.3.1 System Resources Required and Used 59](#_Toc468181712)

[7 Test Plan 60](#_Toc468181713)

[.....7.1 System Test and Procedure 60](#_Toc468181714)

[.....7.2 Testing strategy 60](#_Toc468181715)

[.....7.2.1 Unit testing 61](#_Toc468181716)

[.....7.2.2 Integration testing 69](#_Toc468181717)

[.....7.3 Testing resources and staffing 69](#_Toc468181718)

[.....7.4 Testing tools and environment 69](#_Toc468181719)

[.....7.5 Test record keeping and test log 70](#_Toc468181720)

[8 Future Enhancements and Recommendations 73](#_Toc468181721)

[.....8.1 Conclusion / Summary 73](#_Toc468181722)

**Table of Figures**

Figure 1 Raspberry-Pi GPIO 14

Figure 2 Business Process Model 21

Figure 3 System Architecture 23

Figure 4 Architectural Model 25

Figure 5 Micro-controller (ATMEGA16A) 26

Figure 6 Arduino Mega (Mega 2560) 29

Figure 7 Raspberry-Pi (model B+) 30

Figure 8 Simple Data flow 35

Figure 9 Overall Data Flow 37

Figure 10 System Interconnections 39

Figure 11 UML flow 41

Figure 12 Actual vs Predicted Load 42

Figure 13 Module Roles in Runtime 44

Figure 14 Data Generation 50

Figure 15 Data Collection with reference to Database 55

Figure 16 Gantt Chart 78

## Introduction

### Goals and objectives

The objective of this project is to conserve electrical consumption by providing a means of review and control on consumption by forecasting the electricity consumption based on its history. This project uses the Python Time Series and constructs a usage model using historical data and then simulating the data on that model, and to also generate future bill for the subjected electrical appliances.

### System statement of scope

The project monitors the load of electrical devices that run on mains' supply by taking voltage and current in real-time using Hall Effect sensors for alternating current and AC-AC adapter for voltage readings. The readings are then stored into a database so that it can be analyzed and visualized.

Arduino handles the sensors such that the sensors provide with analog values and they have to be digital so that it can be analyzed, hence it converts analog data into digital stream of bits; it acts as an intermediary between sensors and Raspberry-pi. The data is then elicited from the database and visualized using GUI, and so is the fore-casted data. Future bill of electricity consumption is then generated by interception of speculation data by entering a given date. The project can also compare the subjected devices' loads and suggesting what devices to run in order to tessellate the bill into given budget

### System context

The value of the project can be gaged by the fact that it can be used by various type of electricity consumers, which may include residential, commercial, industrial consumer.

The project is based on an idea that is neither popular nor new, however this kind of venture has never been accomplished before, nor hence we had to design and model a whole new system, and connect it through various underlying systems.

* The project had to go through a whole new development process, specially the initial steps; yielded a road map and drive the fulfillment during ramp-up.
* The identification and realization of system requirements; which are monitoring of electrical measures (the hardware part)
* The project, with help of different scripts, that are a system of their own, were combined using a Web UI, since the Web UI can run scripts and handle all of the operations in a single instance.
* The web UI allows visualization of the “Speculation” part of the project and it also carries it out.
* The feasibility of the project had to be explored before even beginning of the intent of the project because we had to complete it in our own budget, since we had no sponsors.
* The project is more of a product engineering instead of business process because of designing, development and assembly. The project flies off the tangent of Business process because the idea of ‘Linking tasks’ is far-fetched due to minimal interaction of users, namely a single user only to print and deliver reports to the higher ups such as management and owners.

The system is placed in a business or product line context. Strategic issues relevant to context are discussed. The intent is for the reader to understand the "big picture”. In this section discuss what are the prospects of system in market? In which domain it is best suited.

### Theoretical Background

Over the years several projects aimed at electricity conservation, have provided mostly an overview of the crisis of energy, but the main idea in this project was to predict the consumption so that we can plan ahead and avoid overuse of electricity.

One way to forecast the consumption was to use Python Time Series Modeling. (Jain, 2016)

Among the less known skills in the analytics space, Time Series (TS) is considered to be one of it, which is why it should be deeply understood. Our focus would be on the Python and no other resources of Time Series in R, due to the fact that we have used Python in the project.

Specialty of Time Series:

The collection at constant time intervals, of data points is defined as time series. In order to generate forecast and other forms of analysis, the data points are analyzed. Two factors differ Time Series form others. That are:

1) Its dependability upon time. So, fundamentally, a linear regression model is dependent upon the observation.

2) Seasonality trends are observed in most of the Time Series, along with the increment and decrement in trends. Seasonality is defined as 'variations specific to a particular time frame'. As an example, if sales of heavy clothing is observed over the years, higher sales will be observed in the season of winter.

### Technology & Tools/hardware components (used in the Project)

The project is capable of using almost any of the Linux or Windows based system, but for detailed testing and in-depth analysis of data flow/generation and accumulation, we have used the Raspberry-Pi. The development of the project required precise hands-on approach on serial data manipulation such that the data acquired from physical world (analog data), can be done using a standalone microprocessor or Arduino as analog to digital converter. The reason for using specific processor is discussed in detail in following sub-headings.

#### 1.5.1 Raspberry-pi

The raspberry-pi is a standard system-on-chip, with a decent processor and RAM on a chipboard only size of a credit card. It is system that runs using a Linux operating system called Raspbian-Jessie, capable of delivering exactly as any other personal computer running Linux with some exceptions:

The Raspberry-Pi is not capable of multi-threading specially the variation used in the project

* Does not have its own power supply
* Does not have a big processor, that is, only 1 GHz
* It is not capable of generating high level graphics

Exceptions aside, it has a unique feature, and that feature is why we used it.

**General Purpose Input Output Pins (GPIO)**

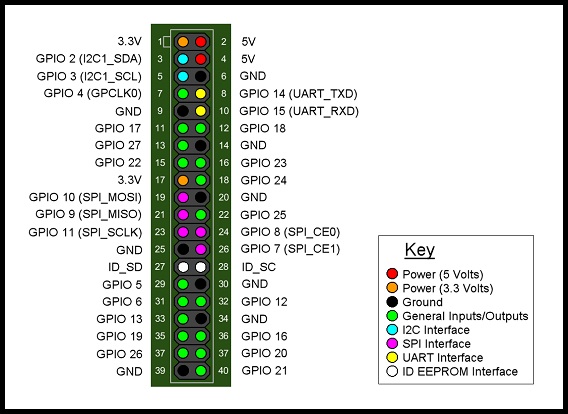
The raspberry-pi has input/output pins on its board that are its link to the outside or physical world. They function as an interface that could be either a switch at its simplest or a communication link between other hardware such as relays, cameras, sensors of numerous types and even other processors.

Figure Raspberry-Pi GPIO

These pins are bidirectional, hence they can be used as both, input and output (not both simultaneously). They can be programmed using one of several programming language including C and Python, by opening their interfaces using the python configuration file, known as the Raspberry-Pi's BIOS.

#### 1.5.2 Python

Python is an interpreted programming language that is dynamic in nature and high-level. The idea behind its design emphasizes in code readability, and the syntax allows us to use fewer lines of code for than we would have in other languages, furthermore it's native functions allows us to produce small scripts or modules, each for a designated task, to constitute a larger system.

Python was used in this project for almost all of the back-end coding that doesn't concern the user, for example, the acquiring of data from Arduino, storing data into database.

#### 1.5.3 Java Script

Java Script is also a high-level interpreted programming language that works dynamically. It is one of the core programming languages that make up most of the web pages, with the exception of storage, graphics, networking and Input/output techniques.

Java Script was used in this project for data visualization and data fetching from database and for invoking modules of Python using the web pages built for User Interface.

#### 1.5.4 Mongo Database

Mongo DB is an open source database that uses a report situated information display.

Mongo DB is one of a few database sorts to emerge in the mid-2000s under the “NoSQL” standard. Rather than utilizing tables and columns as a part of social databases, Mongo DB is based on a design of accumulations and reports. Archives include sets of key-esteem matches and are the essential unit of information in Mongo DB. Accumulations contain sets of archives and capacity as what might as well be called social database tables.

Mongo DB was used in the project as a flexible non-relational database that can import and export data as JSON.

#### 1.5.5 Arduino

Arduino is an open-source extend that made micro-controller-based units for building computerized gadgets and intelligent articles that can detect and control physical devices.

This venture depends on micro-controller board plans, delivered by a few consignors, utilizing different micro-controllers. These frameworks give sets of advanced and simple info/yield (I/O) sticks that can interface to different development sheets (named shields) and different circuits. The sheets highlight serial correspondence interfaces, including Universal Serial Bus (USB) on a few models, for stacking programs from PCs. For programming the micro-controllers, the Arduino extend gives a coordinated advancement environment (IDE) in view of a programming dialect named Processing, which additionally bolsters the dialects C and C++.

## Usage Scenario / User Interaction

### User profiles

The project is an isolated system that is not capable of modifying the input data from the user end, hence there is not much need for user access control (UAC) and therefore no user or user based scenario was included.

The only role of user is so comprehensive that people with no knowledge of electricity can operate it and actually understand the outputs, it may even eliminate the role of budget incorporation of electrical consumption. The only thing a user need to do (after installation) is to provide the system with an estimate of the amount of budget subjected electrical appliances and the system will generate a report for what devices need what duty cycles to befit that budget.

As for the forecast, it is generated and stored into database, with exception of historical predictions and forecast, the user may only provide system with a date of interception (end date for upper limit of forecast) and the process will begin.

### Use-cases

Figure Business Process Model

The above diagram shows how a simple user can input the budget and the system passes objects of data to sub-systems and what modules perform the given tasks. The end user only receives the generated reports so that it can delivered to higher-ups and organization's accountants, for incorporation into their finances.

### Special usage considerations

The project requires installation into either a dedicated panel for that device or group of devices, connection to devices must be in parallel for the voltage sensor and in series for the current sensor.

The Arduino (prototype) and the Micro-controller only requires 5v 1000 mA of power supply to operate and sense values accurately.

The raspberry-pi also requires 5v of supply but with 2000 mA requiring no additional hardware other than keyboard, mouse and a display. The project can also function on any other computer running Linux, having the standard python libraries but then micro-controllers cannot be used as a typical computer does not have General Input Output Pins.

## Functional and Data Description

### System Architecture



Figure System Architecture

The diagram above represents the overall architecture of the system.

The main components of the system as shown here are:

1. Raspberry-pi

It provides the actual platform for the sub-systems to interact with each other and with the processing power of a typical computer. It also handles all the data that is poured into the system, from the user end till the each output of every module the project is constituent of.

1. Web UI

A simple means of representing data to the user after it is processed. It is no more than a user interface that plays an extensive role, by taking user inputs, feeding it into the back-end code and invokes requested modules. It is the abstract point where all the subsystems are housed.

1. Sensing

Sensory equipment that inputs all the data from subjected devices. It can be called the monitoring part of the system, where analog data is injected into the system.

1. Speculation

It may be the most sophisticated part of the project as it implements Auto Regressive Moving Average (ARMA) on the historical data to forecast future data with respect to time.

#### Architecture model

The following diagram is the representation of roles of sub-systems with respect to physical and abstract parts of the system.



Figure Architectural Model

The system, as shown above has two parts, hardware and software, each counterpart has different setups, such as raspberry-pi requires being plugged into a Analog to Digital converter so that data from sensors can be parsed as data objects need recognition in order to be processed, hence sensors require to be connected to Analog to Digital Converter (ADC) and ADC itself to the main processor here that is raspberry-pi.

##### Micro-controller

A micro-controller is an even smaller version of a computer, the only exception being that it is on a single integrated circuit. The integrated circuit contains all the necessary components such as processor core, memory, programmable input and outputs for it's peripherals, RAM, and flash. Micro-controllers are significantly small in a way such that their original purpose is use in embedded systems.

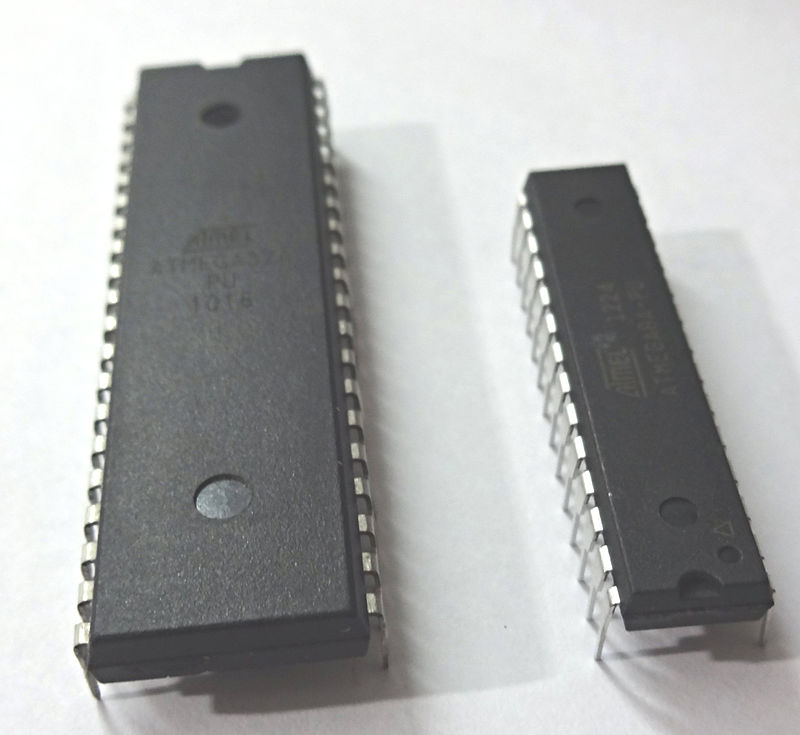
They are used in automatically controlled or smart appliances such as washing machines and smart refrigerators.

Figure Micro-controller (ATMEGA16A)

The role of micro-controller in this project is to deliver analog data to main processor in real-time while converting it into digital at the same time.

##### Arduino

Arduino as an open-source project, created micro-controllers' kit that are used for construction of digital devices and devices that are interactive, such as RF-ID keys, or smart solar switches.

These micro-controllers are embedded into yet another credit card sized chip board with simple digital and analog input output pins; while the difference being that these pins have a threshold of up to 5 volts, each pin having from 0 to 1023 bandwidth, meaning if we make a pin do something, we have 1024 values to choose from for each of pulse width off of pulse width modulation or PWM pins (digital). The analog part of these pins is to handle physical data, whether they are output or input, each signal will have a width of 1024, it is safe to assume that they have an accuracy of 0.0048 which makes them fairly good for sensing current and voltage, the way we have in this project.

The Arduino variation used in this project has a micro-controller based on ATmega 2560, it has a 16 MHz crystal oscillator and the circuitry to make it work, including ICSP header, reset button, power input jack, and USB connector. The set of pins provided in it are a total of 54, 15 of them are PWM capable, 16 can be used as analog input and 4 UART (Universal Asynchronous receiver/transmitter).

The purpose of Arduino in this project is convert analog data, by taking analog input from sensors as a value (out of those 1024 mentioned above) and convert them into digital signals (serial data) that can be made use of by the Raspberry-Pi, and is used only in the prototype version of the project.



Figure Arduino Mega (Mega 2560)

##### Raspberry-Pi

The raspberry-pi is a standard system-on-chip, with a decent processor and RAM on a chipboard only size of a credit card. It is system that runs using a Linux operating system called Raspbian-Jessie, capable of delivering exactly as any other personal computer running Linux with some exceptions:

The Raspberry-Pi is not capable of multi-threading specially the variation used in the project

Does not have its own power supply

Does not have a big processor, that is, only 1 GHz

It is not capable of generating high level graphics

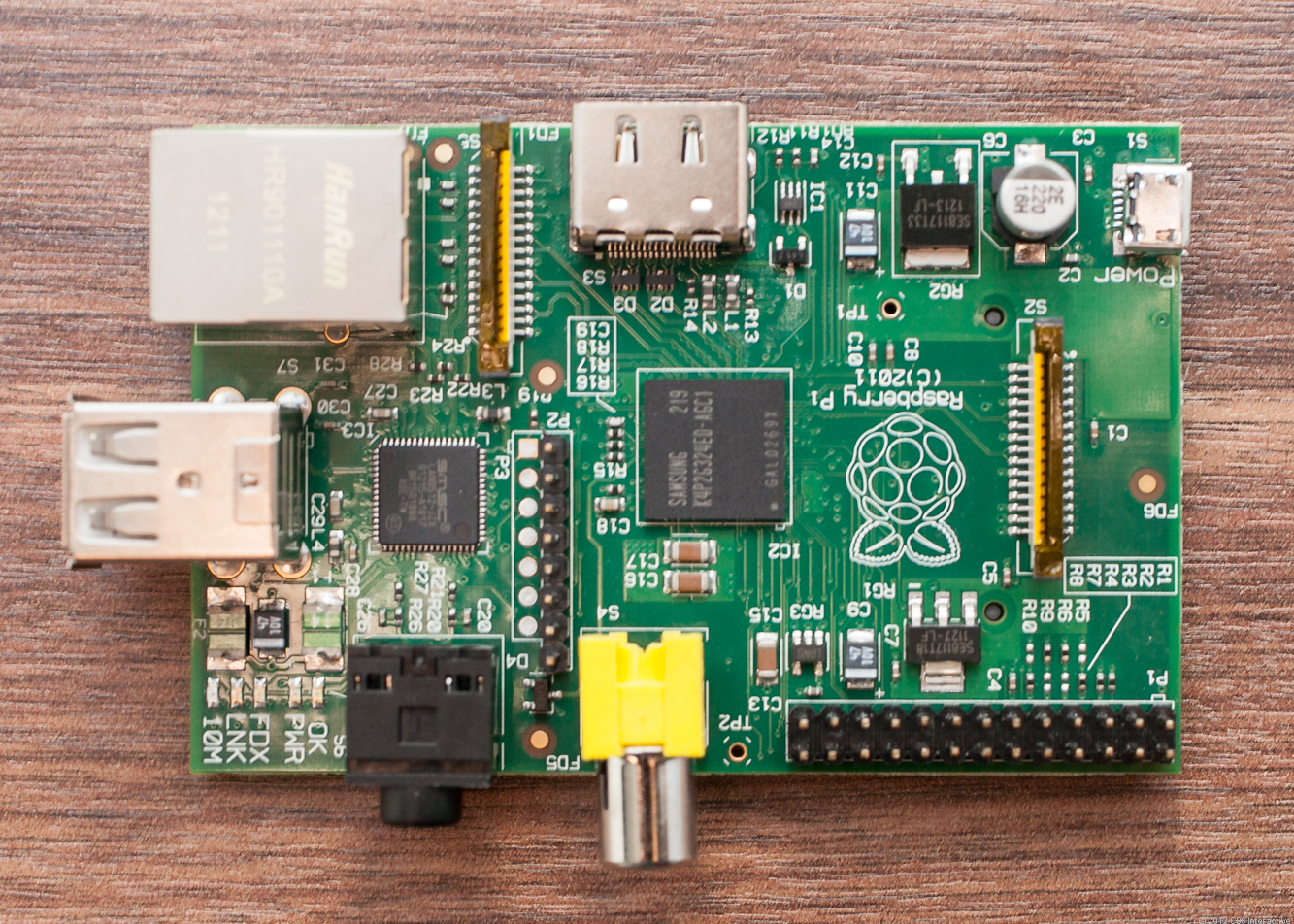
Exceptions aside, it has a unique feature, and that feature is why we used it.

Figure Raspberry-Pi (model B+)

General Purpose Input Output Pins (GPIO)

The raspberry-pi has input/output pins on its board that are it's link to the outside or physical world. They function as an interface that could be either a switch at its simplest or a communication link between other hardware such as relays, cameras, sensors of numerous types and even other processors.

These pins are bidirectional, hence they can be used as both, input and output (not both simultaneously). They can be programmed using one of several programming language including C and Python, by opening their interfaces using the python configuration file, known as the raspberry-pi's BIOS.

#### Subsystem/modules overview

This system of system is composed of Python modules designed and developed for dedicated tasks, there is only one sub system that is the interface that has Java Script modules, combined into a subsystem of its own.

1. Stream Encoder; resides in Arduino made using C language,
2. Stream Parser and storage; uses Python Script to analyze and store stream,
3. Web UI; houses almost all subsystems in one place, and invokes them, also provides data visualization,
4. Python Time Series; Analytics’ method for Electrical Usage and forecast.

### Data Description

Since this project's aim was to only predict electricity consumption, the only data in the database was electricity related, meaning voltages, currents and fore-casted consumption, however discussed and shown here is the data object that originates from the very low level, Arduino program, not much of it was required.

The user input of his budget is kept in the session, using Java Script, due to the fact that reports and intercepts keep on changing, the real-time nature of data entries make it infeasible for us to have stored the user's estimate into the database, and to keep the system efficient because if the user needs his historical data he can view it by using the Web UI, and the data that can be stored can range from the past year to the future month (of prediction) already.

In order to perform ARMA (Jaine,2016) modeling database needs indexing with respect to time, because the machine learning and modeling process requires a unified scripture, such as date-time in this case, to be able to utilize the data.

#### Major data objects

The only “Data Object” that is stored is generated by the Arduino (Micro- controller in final version) at the instant of when a reading is taken, which is every 10 seconds , contains only the reading of current and voltage of different devices.

At the time of data capture, there is only one data object available, but when a prediction model of historical data is generated, a future forecast of data is also written. The forecasted data is actually written in a buffer sort of database, because each time user requests a new and updated prediction model is generated and hence data of upcoming times.

#### System level data model

Data Flow Diagram

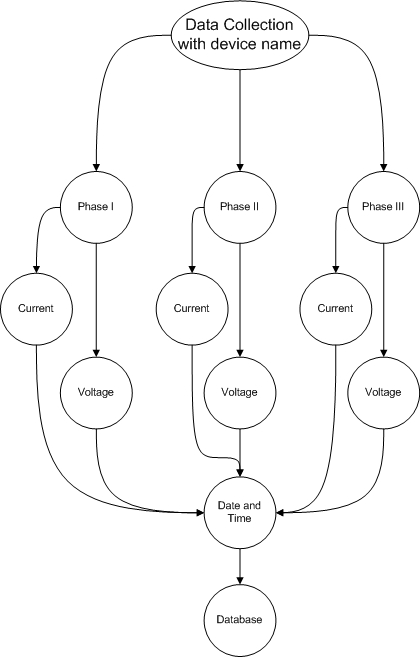


Figure Simple Data flow

As the diagram suggests that data is collected In accordance with the name of device that is being monitored. A data base entry is generated when a reading is taken, the reading can be from all three phases of the device at the point of data collection, it is when the Arduino (micro-controller in case of final version) reads from the sensors, converts the analog readings to digital, and envelopes it into a JSON stream, where the only data is; in it's raw from, a field name and a parameter (a data value), per device.

An example of data element before being enveloped:

{

“current1” : 2.25,

“volt1” : 235

}

An example of data element after envelop:

[

{

"cur3 ": 2.41,

“volt1”: 240,

"time": "2016=10-16 04:33:57"

}

]

Since the Arduino is not capable of keeping time, without additional hardware (time shield) the date-time parameter is added by the receiving end, or Raspberry-Pi in this case.

Shown below is simplified version of data flow

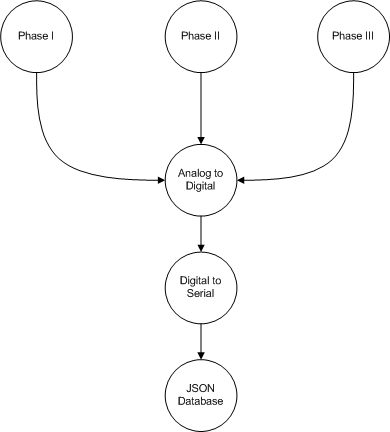


Figure Overall Data Flow

The JSON object, after being received is injected with date-time and inserted as a record with respect to the given time into Mongo DB.

The Mongo DB is not a relational database, but it is records can be encapsulated into JSON stream. This method was chosen to avoid errors that may occur while parsing the stream, such events can occur when the sending end has already begun sending but the receiving end is either not ready or stream begins at a different instant than the intended beginning of stream. If this method was not implemented, we would have needed synchronous data transmission, introducing yet another complexity in the hardware.

### System Interface Description

#### External machine interfaces

The Raspberry-Pi is our actual system, the part which the user needs to see or interact with at any point in the project’s life cycle

#### External system interfaces

The project draws its data from the outside world, data elements are formed from readings of other electrical appliances, and these appliances can be anything if they follow the sensor specified ratings.

#### System Limitations

The data collected by monitoring device for speculation purposes does not deviate more than 50% of its average, because then the speculation may be thrown off course and the targeted budget will not be properly accomplished by following suggestions of the system. The sensors be built, being able to sense at least 10 Amps of current.

The whole software part of the system must be able to run on minimal hardware requirements such as at least 256 MB of RAM and 1 GHz of processing power. The volume of software part is not to exceed 500 MB of storage because the system is meant to be portable and flexible i.e. run on different devices that aren’t running Microsoft Windows.

The device one which it runs has a browser capable of running Python Scripts and has essential Python libraries including matplotlib, numpy, the time series, stats models and tkinter.

## Subsystem/module Description

As mentioned earlier, each subsystem was combined into a single project, using the Web UI. Details of how this was done are provided under heading Subsystem Interconnections.



Figure System Interconnections

The diagram shows what subsystems are used and how are they connected.

### Web UI

The purpose of the web UI was to house each and every single python module, a module as a subsystem of its own.

#### Subsystem scope

Provide a user interface that which is light weight and easy to use. They interface is able to run on any platform/ operating system.

The features include:

* Report generation
* Bill Generation
* Generate graphs for data comparison against Forecasts
* Fetch data from database

#### Web UI flow diagram



Figure UML flow

#### Data Visualization

The data after being fetched calls a function “draw” and the results are as following

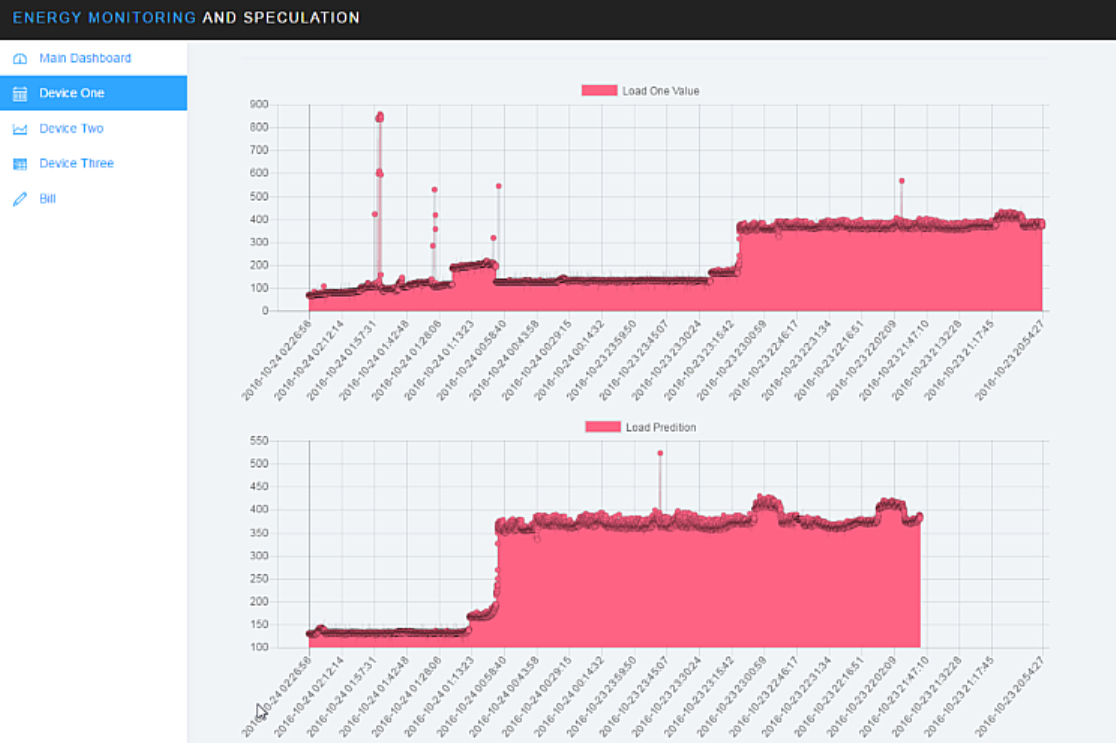


Figure Actual vs Predicted Load

#### Web Server

The web server plays a crucial role in development and design of project.

Pico server is a small, lightweight and portable web server, capable of executing python scripts. Its core functionality includes passing values to and fro python modules. The following are its features:

* Portable and small
* Lightweight and fast
* CGI-BIN Support
* Includes Auto-indexer for directories
* Provides access to error logging
* Forking or single-connection at command

Pico Server (pServ) is composed in convenient C (K&R style so it can assemble on more seasoned compilers as well) and games a few alternatives that by method for #define explanations can modify the conduct, the performance and the list of capabilities so to have the capacity to fit better the necessities.



Figure Module Roles in Runtime

As it can be seen that the Pico server is the data bridge that transfers data from just being data and makes it visible to the user

### Forecast and Prediction from Historical Data

ARIMA Modeling

ARIMA (p,d,q) anticipating condition: ARIMA models are, in principle, the most broad class of models for determining a period arrangement which can be made to be "stationary" by differencing (if vital), maybe in conjunction with nonlinear changes, for example, logging or flattening (if fundamental). An arbitrary variable that is a period arrangement is stationary if its factual properties are all consistent after some time. A stationary arrangement has no pattern, its varieties around its mean have a steady adequacy, and it squirms in a reliable manner, i.e., its transient irregular time designs dependably have a striking resemblance in a measurable sense. The last condition implies that its autocorrelations (relationships with its own particular earlier deviations from the mean) stay steady after some time, or identically, that its energy range stays consistent over the long run. An arbitrary variable of this frame can be seen (not surprisingly) as a blend of flag and commotion, and the flag (in the event that one is evident) could be an example of quick or moderate mean inversion, or sinusoidal wavering, or fast shift in sign, and it could likewise have a regular part. An ARIMA model can be seen as a "channel" that tries to isolate the flag from the clamor, and the flag is then extrapolated into the future to get gauges.

The ARIMA determining condition for a stationary time arrangement is a straight (i.e., relapse sort) condition in which the indicators comprise of slacks of the needy variable and additionally slacks of the estimate blunders. That is:

Anticipated estimation of Y = a steady and additionally a weighted aggregate of one or later estimations of Y as well as a weighted total of one or later estimations of the mistakes.

On the off chance that the indicators comprise just of slacked estimations of Y, it is an immaculate autoregressive ("self-relapsed") show, which is only an uncommon instance of a relapse model and which could be fitted with standard relapse programming. For instance, a first-arrange autoregressive ("AR(1)") demonstrate for Y is a straightforward relapse display in which the free factor is just Y slacked by one period (LAG(Y,1) in Statgraphics or Y\_LAG1 in RegressIt). On the off chance that a portion of the indicators are slacks of the blunders, an ARIMA display it is NOT a direct relapse show, on the grounds that there is no real way to determine "last period's mistake" as an autonomous variable: the mistakes must be figured on a period-to-period premise when the model is fitted to the information. From a specialized viewpoint, the issue with utilizing slacked blunders as indicators is that the model's forecasts are not straight elements of the coefficients, despite the fact that they are direct elements of the past information. In this way, coefficients in ARIMA models that incorporate slacked blunders must be assessed by nonlinear enhancement techniques ("slope climbing") as opposed to by simply comprehending an arrangement of conditions.

The acronym ARIMA remains for Auto-Regressive Integrated Moving Average. Slacks of the stationarized arrangement in the determining condition are called "autoregressive" terms, slacks of the conjecture mistakes are called "moving normal" terms, and a period arrangement which should be differenced to be made stationary is said to be an "incorporated" rendition of a stationary arrangement. Irregular walk and arbitrary pattern models, autoregressive models, and exponential smoothing models are all unique instances of ARIMA models.

A nonseasonal ARIMA model is delegated an "ARIMA (p,d,q)" demonstrate, where:

p is the quantity of autoregressive terms,

d is the quantity of nonseasonal contrasts required for stationarity, and

q is the quantity of slacked conjecture mistakes in the forecast condition.

### Arduino Data Generation

The Arduino measures the basic electrical components namely the voltage, current, so that the basic parameters can easily be computed. This will help to calculate other important parameters like the time period and power.

#### Arduino Data Description

Current estimation is a vital part of the power stream examination of any physical framework, the measure of load associated with any framework is reflected in type of current, and in this manner its intermittent checking is vital for insurance and relentless stream of the framework.

We are utilizing microcontroller as our focal unit, which just peruses voltage as contribution through Analog or Digital Convertor (ADC) and chips away at little streams, subsequently the underlying assignment was to venture down high ebbs and flows, segregate high current hardware from microcontrollers and change over current into voltage identical wave.

One technique for accomplishing the undertaking is to utilize Current Transformers (CTs), yet they make the framework bulkier subsequently to encourage conservativeness and precision we have utilized Hall Effect current sensor which ventures down to a DC esteem bringing about a rectangular wave which likewise encourages in Power Factor (PF) estimation.

The resulting line of record is shown below

{

"cur2" : 4.21,

"cur3" : 2.80,

"cur1" : 0.38,

"volt2" : 219.88,

"volt3" : 242.88,

"volt1" : 236.88,

}

#### Arduino Scope

Electricity monitoring device is a microcontroller-controlled electronic device that has the capability to transmit data to any remote location via Ethernet instantly and accurately at a very fast rate.

The device is also very cheap, manufactured from the equipment’s easily available in the market. The device can also inform supervisors about the status of the circuit breakers connected at different locations of grid station. EMD has industrial feasibility due to the fact that these quantities with microcontrollers and sensors that can take analog quantities as input and can display digital output to any desired level. The quantities measured by EMD include:

• Voltage

• Current



Figure Data Generation

The above data flow is a cycle, hence it never ends.

### Python Time Series

As the name proposes, Time Series is a gathering of information focuses gathered at consistent time interims. These are examined to decide the long haul drift to conjecture the future or play out some other type of investigation.

#### Time Series

A period arrangement is a progression of information focuses recorded (or recorded or charted) in time arrange. Most ordinarily, a period arrangement is a grouping taken at progressive similarly dispersed focuses in time. In this manner it is a succession of discrete-time information. Cases of time arrangement are statures of sea tides, tallies of sunspots, and the everyday shutting estimation of the Dow Jones Industrial Average.

Time arrangement are often plotted through line graphs. Time arrangement are utilized as a part of insights, flag preparing, design acknowledgment, econometrics, scientific back, climate anticipating, wise transport and direction determining, tremor forecast, electroencephalography, control building, stargazing, correspondences designing, and to a great extent in any area of connected science and building which includes worldly estimations.

Time arrangement examination includes strategies for breaking down time arrangement information so as to concentrate important measurements and different attributes of the information. Time arrangement gauging is the utilization of a model to foresee future qualities in light of already watched values. While relapse examination is frequently utilized so as to test speculations that the present estimations of at least one autonomous time arrangement influence the present estimation of some other time arrangement, this sort of investigation of time arrangement is not called "time arrangement examination", which concentrates on looking at estimations of a solitary time arrangement or numerous needy time arrangement at various focuses in time.

Time arrangement information have a characteristic transient requesting. This sets aside a few minutes arrangement investigation unmistakable from cross-sectional studies, in which there is no regular requesting of the perceptions (e.g. clarifying individuals' wages by reference to their separate training levels, where the people's information could be entered in any request). Time arrangement investigation is additionally particular from spatial information examination where the perceptions regularly identify with geological areas (e.g. representing house costs by the area and additionally the natural qualities of the houses). A stochastic model for a period arrangement will for the most part mirror the way that perceptions near one another in time will be more firmly related than perceptions promote separated. Furthermore, time arrangement models will frequently make utilization of the characteristic one-path requesting of time so that qualities for a given period will be communicated as getting somehow from past qualities, instead of from future qualities (see time reversibility.)

Time arrangement investigation can be connected to genuine esteemed, ceaseless information, discrete numeric information, or discrete typical information (i.e. arrangements of characters, for example, letters and words in the English dialect).

#### Scope of Python Time Series

The python time series is an analytics’ solution by means for ARMA or ARIMA modeling, hence it is used for load prediction and forecast. The time series has a few prerequisites:

* Subject data must have time stamps
* Data must be stationary
* Time stamps must be parse-able i.e. date time data type
* Must pass the Dickey-Fuller Test

In order to correctly predict data, the historical data must be in order, hence the following operations are performed in order to achieve stationary data

* Aggression - taking normal for an era like month to month/week by week midpoints
* Smoothing – taking moving midpoints
* Polynomial Fitting – fit a relapse demonstrate

As a result, non-fluctious data acts as an input to the forecast algorithm, hence data is predicted, for the time ahead

## Behavioral Model and Description

When the user opens up the web UI, he has to choose between devices he has to view consumption of, when the default view of consumption is given, the user has to enter the beginning and end date of the data he’s trying to fetch, so that the records shown are limited and the system can easily do that, without delay.

When the user clicks on predictions’ tab on the UI, he will be asked for future dates, as target dates for module, then apply the historical data into the model, and pass a single date for target prediction.

### Description for system behavior

The source events are based upon the user, as most methods in this project are automated.

#### Events/interrupts

* User enters estimate into the webpage
* User views the graphs, i.e. graph generates
* Sensor plugged in Raspberry-Pi
* Arduino takes reading; automatically each 10 seconds

#### States

Inserts records; the real-time record has been added

Viewing; the bill and reports has been viewed

Receiving; Raspberry-Pi listens to plugged sensors

Predict; Page load invokes ARMA methods

### State Transition Diagram



Figure Data Collection with reference to Database

### Control specification

Consider the framework out of gear state. To begin from the extremely fundamental when the framework is to be installed, a circuit repairman and an establishment work force introduce and arrange the power observing sensors and raspberry-pi at client's premises. As the client expends electricity, the gadget takes the perusing and sends it to raspberry-pi. The framework then sends these information entities to the Mongo DB stored and maintained on the raspberry-pi storage itself.

The framework demonstrates the measurable examination of the power utilization of the client on the website. Other than factual investigation, the framework likewise produces charging reports, indicates cost utilization and apparatus level planning

The framework additionally upgrades the levy in the database of the site in light of the occasion that the power supply organization changes its charging duty for its customers. This undertaking is performed by the chairman of the site.

Finally, the client of the framework can redesign his create account settings and additionally his proffered edge settings for his record.

## System Estimates and Actual Outcome

This section provides cost estimates for the project

### Historical data used for estimates

There is a need to authentic information on any type of an altered gauge. The issue with the authentic information is that the information accumulation requires exertion, time and cash. What's more, it is regularly utilized as the need to spend assets to produce, gather or purchase verifiable information bugaboo of imperviousness to information accumulation and as an apparatus to maintain a strategic distance from the utilization of parametric estimation procedures or recorded.

Authentic information can be as basic as scrum group accumulate speed or the profitability of each race and used to ascertain the normal for arranging and assessing or as mind boggling as an arrangement of information that groups utilizing gratefulness fringe gathering, which incorporates bed more intense than the information, including the venture's endeavors, and the size, length, the limit of the group The setting of the venture. In both cases, the information must be gathered for the technique that you utilize and the level of detail that you will gauge or plan. For instance, on the off chance that you assess at the venture level you require information at the venture level. In the event that you appraise the level of the employment that you have to gather verifiable information up to the errand.

### Estimation techniques applied and results

Programming undertakings are infamous for going past their due date, for example, going over the spending constantly. The issue lies in the estimation of the measure of exertion required for the improvement of a venture. The cost estimation is typically subject to the measure of the venture, and for this situation, cost of equipment, and cost of equipment get together i.e. wires, clock precious stones, bread loads up, vero-loads up, and welding costs. There are a few distinct systems for performing programming cost estimation, including master judgment and algorithmic models.

The COCOMO 81 model is a static cost estimation strategy, with three forking branches

* Basic Model; It gives a harsh gauge of exertion in view of size and method of programming improvement
* Intermediate Model; It refines the essential model by utilization of 15 cost drivers. These are subjective characteristics. The effect of cost drivers is to be considered at the venture level. The exertion is further changed in accordance with mull over time subordinate imperatives
* Detailed Model; This model further refines the estimation by considering the stage insightful effect of cost-drivers, and the calculations in this are for different sub-frameworks and modules.

We have used the detailed model for cost estimation of the project.

### System Resources (Required and Used)

People, hardware, software, tools, and other resources proposed to build the software are noted here. On the bases of which cost estimations were performed

#### System Resources Required and Used

* Raspbian Jessie
* Stats models for python
* Pandas for python
* Raspberry-Pi Model B+
* Arduino mega 2560
* Custom made voltage sensors
* ACS 712 current sensor
* An electricity panel to wire in the sensors
* A local network to monitor values passed against values gathered
* External Power supply for hardware devices

## Test Plan

The fundamental technique for testing all through the outline of the venture was to set up that every individual class was executing not surprisingly. Keeping in mind the end goal to encourage this, every module actualizes its own primary technique and can be executed freely of whatever remains of the framework. This is to guarantee that every part works before entwining them. What's more, various print articulations have been set up at various indicates all through the program find disappointments when the program had been incorporated and the perplexing strings of execution were hard to take after.

Notwithstanding fundamental techniques being made, various test classes were made to guarantee adjust operation before the last accumulation of the code.

### System Test and Procedure

Since this project is a large collection of other sub-systems, and the feasibility and performance of each subsystem had to take into consideration. The system, was tested in isolated environments, that is, each subsystem was provided with dummy data at first and it was run without any interconnection among other subsystems.

### Testing strategy

Since the project is composed of various sub-systems, down to their elements, where each sub-system covers a different domain. Put simply, this project crosses at least 4 domains, 2 platforms, and consists of a non-relational database, which is Mongo DB, while also crossing over data barriers. Furthermore, for projects that cross over, skip over and jump platforms hardly cover a single testing strategy.

Every sub-system has a different test strategy, to cover as many scenarios as possible.

The electronics’ and hardware counter part of the project was tested with extreme values which are within specified constraints.

#### Unit testing

##### Arduino

Sensory pins consist of internal pull up and pull down resistors to dump unnecessary data which is also known as noise; however there is one more limitation at concern here, which is, what will happen if an input value goes over or under specified limits, thankfully during the design period, this matter was taken care of by using current limiters, voltage limiters and fuses.

Current Sensor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID:** | Sensor\_01 |  |  |  |  |
| **Test Priority:** | medium |  |  |  |  |
| **Module Name:** | ACS 712 |  |  |  |  |
| **Test description:** | Testing sensory current limits in Amperes alongside a digital Multi-meter | | | | |
|  |  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |  |
| 1 | Start module | module starts | module starts | Pass |  |
| 2 | Observe value | 2.075 | 2.075 | Pass |  |
| 3 | Verify value | 2.1 | 2.1 | Pass |  |

**Current sensor lower limits**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** | Sensor\_02 | |  |  |  |  |  |  |
| **Test Priority:** | medium |  |  |  |  |  |  |  |
| **Module Name:** | ACS 712 |  |  |  |  |  |  |  |
| **Test description:** | Testing lowest sensory current limits in Amperes alongside a digital Multi-meter | | | | | | | |
|  |  |  |  |  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |  |  |  |  |
| 1 | Start module | module starts | module starts | Pass |  |  |  |  |
| 2 | Turn off all loads | |  |  |  |  |  |  |
| 3 | Observe value | 0.03 | 0 | Fail |  |  |  |  |
| 4 | Verify value | 0 | 0 | Pass |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Current sensor Upper Limit**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** | Sensor\_03 | |  |  |  |  |
| **Test Priority:** | medium |  |  |  |  |  |
| **Module Name:** | ACS 712 |  |  |  |  |  |
| **Test description:** | Testing highest sensory current limits in Amperes alongside a digital Multi-meter | | | | | |
|  |  |  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |  |  |
| 1 | Start module | module starts | module starts | Pass |  |  |
| 2 | Turn on a few A/C | |  |  |  |  |
| 3 | Observe value | 29.53 | 30.00 | Pass |  |  |
| 4 | Verify value | 30.0 | 30.1 | Pass |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |

**Voltage Sensor Upper Limit**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID:** | Sensor\_04 | |  |  |  |
| **Test Priority:** | medium |  |  |  |  |
| **Module Name:** | Voltage Sensor | |  |  |  |
| **Test description:** | Testing lowest sensory Voltage limits in Amperes alongside a digital Multi-meter | | | | |
|  |  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |  |
| 1 | Start module | module starts | module starts | Pass |  |
| 2 | Observe value | 0 | 0 | Pass |  |
| 3 | Verify value | 0 | 0 | Pass |  |

**Voltage Sensor Upper limit**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID:** | Sensor\_05 | |  |  |  |
| **Test Priority:** | medium |  |  |  |  |
| **Module Name:** | Voltage Sensor | |  |  |  |
| **Test description:** | Testing highest sensory voltage limits in Amperes alongside a digital Multi-meter | | | | |
|  |  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |  |
| 1 | Start module | module starts | module starts | Pass |  |
| 2 | Observe value | 350 | 349 | Pass |  |
| 3 | Verify value | 350 | 348.9 | Pass |  |

Given that nothing burned up, or nothing exploded in the third trial

**Arduino JSON Envelop**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID:** | Arduino |  |  |  |  |
| **Test Priority:** | medium |  |  |  |  |
| **Module Name:** | JSON Enveloper | |  |  |  |
| **Test description:** | Testing for validation of JSON Object generated in Arduino after translating values into digital | | | | |
|  |  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |  |
| 1 | Start module | module starts | module starts | Pass |  |
| 2 | Observe value | |  |  |  |
| 3 | Run script for JSON test | | |  |  |
| 4 | Verify value | |  | Pass |  |

**Mongo DB Entry Test**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** | Data Collection | |  |  |  |  |  |
| **Test Priority:** | medium |  |  |  |  |  |  |
| **Module Name:** | Mongo Recorder | |  |  |  |  |  |
| **Test description:** | Testing for validation of JSON Object generated into Database records | | | | | | |
|  |  |  |  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |  |  |  |
| 1 | Start module | module starts | module starts | Pass |  |  |  |
| 2 | Pass Dummy data | |  |  |  |  |  |
| 3 | Run script for JSON test | | |  |  |  |  |
| 4 | Verify Records | |  | Pass |  |  |  |
|  |  |  |  |  |  |  |  |

#### Integration testing

The system Integration testing was carried out by passing uncertain and or abnormal values across sub systems.

### Testing resources and staffing

Since this project has a physical counterpart and an abstract one, there were digital Multi-meters involved in testing for parallel runs.

### Testing tools and environment

As said earlier, the device is useless without a load to monitor and predict, hence the following set of environmental elements were required

* Electrical wiring through Current sensor
* Electrical wiring parallel to Voltage sensor
* A load to monitor
* 5 volts DC supply to both, Arduino and the Raspberry-Pi

The user may proceed with caution here because

* There is a danger of getting electrocuted
* If the power supply to these units is interrupted or less than specified, readings will differ
* Only operate with proper wiring; with no exposed ends whatsoever

### Test record keeping and test log

The project already logs each entry to the database specifying:

* The Date-Time stamp
* The voltage for all devices
* The current for all devices

And hence the only thing left to do was to deploy a control environment. A control environment was established using special circuitry, with known loads and isolated power source, which is our very own 220 volt modulator for least amount of noise and variations in data along with a fairly predictable load.

The control environment consisted of a heating coil, a resistive load, which has uniform electricity consumption, and is predictable even with the simplest of equations.

The results were as follows.

**Logging Test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID:** | logging\_01 |  |  |  |
| **Test Priority:** | medium |  |  |  |
| **Module Name:** | The whole Project |  |  |  |
| **Test description:** | Testing for validation of values logged |  |  |  |
|  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Start module | module starts | module starts | Pass |
| 2 | Start Digital Multi-meter on the same circuit |  |  |  |
| 3 | cross reference values with Digital Multi-meter |  |  |  |
| 4 | Verify value |  |  | Pass |

Prediction Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID:** | prediction\_01 |  |  |  |
| **Test Priority:** | medium |  |  |  |
| **Module Name:** | The whole Project |  |  |  |
| **Test description:** | Testing for validation of values predicted |  |  |  |
|  |  |  |  |  |
| **Step** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Start monitoring modules | module starts | module starts | Pass |
| 2 | Start Digital Multi-meter on the same circuit |  |  |  |
| 3 | Invoke prediction modules | module inputs range | module predicts till range | Pass |
| 4 | Wait till intercept range is approached |  |  | Pass |
| 5 | Validate values using multi-meter and simple calculation | |  | Pass |
|  |  |  |  |  |

## Future Enhancements and Recommendations

There are several future enhancements and extensions to choose from that will make the system less of a mess the first and foremost being implementation of Power line communication.

Sensors can be connected using a small ATtiny micro-controller with Power line communication module and a filter. This way the sensors can send data through the very electrical grid they are monitoring. The usual electricity grid provides more than 600 Mbps of data bandwidth, which is a lot more than enough.

This extension can also allow communication of over 10 sensors, technically, up to 65536 asynchronous sensors if they are using baud rate of 9600 bits per second.

600 x 1024 x 1024 = 629145600

629145600 / 9600 = 65536

### Conclusion / Summary

This section includes the end results achieved and various observations which were monitored while deploying the system

1. Appendices
2. Project Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| ***WBS*** | ***Task Name*** | ***Duration*** | ***Start Date*** |
| *1* | Planning | 60 days | Mon 01-02-16 |
| *1.1* | software requirements | 29 days | Mon 01-02-16 |
| *1.1.1* | Features and libraries | 10 days | Mon 01-02-16 |
| *1.1.2* | Software merits | 13 days | Mon 15-02-16 |
| *1.1.3* | Report formatting | 6 days | Thu 03-03-16 |
| *1.2* | Hardware | 25 days | Fri 11-03-16 |
| *1.2.1* | Requirements | 6 days | Fri 11-03-16 |
| *1.2.2* | Specifications | 4 days | Mon 21-03-16 |
| *1.2.3* | Research | 15 days | Fri 25-03-16 |
| *1.3* | Feasibility study | 6 days | Fri 15-04-16 |
| *2* | Design | 57 days | Mon 25-04-16 |
| *2.1* | Hardware Designing | 16 days | Mon 25-04-16 |
| *2.1.1* | Circuits Design | 6 days | Mon 25-04-16 |
| *2.1.2* | Data Transmission Designing | 7 days | Tue 03-05-16 |
| *2.1.3* | Data formatting | 3 days | Thu 12-05-16 |
| *2.2* | Software Designing | 41 days | Tue 17-05-16 |
| *2.2.1* | Functional specifications | 4 days | Tue 17-05-16 |
| *2.2.2* | Background variables | 10 days | Mon 23-05-16 |
| *2.2.3* | Background Formulae | 8 days | Mon 06-06-16 |
| *2.2.4* | Report Formatting | 2 days | Thu 16-06-16 |
| *2.2.5* | Formulae implementation | 9 days | Mon 20-06-16 |
| *2.2.6* | GUI Designing | 8 days | Fri 01-07-16 |
| *3* | Development | 56 days | Wed 13-07-16 |
| *3.1* | Hardware Development | 16 days | Wed 13-07-16 |
| *3.1.1* | Circuits' construction | 3 days | Wed 13-07-16 |
| *3.1.2* | Data transmission hardware setup | 5 days | Mon 18-07-16 |
| *3.1.3* | Implementing Physical Logic | 8 days | Mon 25-07-16 |
| *3.2* | Software Development | 40 days | Thu 04-08-16 |
| *3.2.1* | Variables and Objects | 6 days | Thu 04-08-16 |
| *3.2.2* | Formulae Implementation | 10 days | Fri 12-08-16 |
| *3.2.3* | Logic and Algorithm implementation | 14 days | Fri 26-08-16 |
| *3.2.4* | GUI Development | 10 days | Thu 15-09-16 |
| *4* | Implementation | 30 days | Thu 29-09-16 |
| *4.1* | Hardware installation | 8 days | Thu 29-09-16 |
| *4.2* | Transmission installation | 7 days | Tue 11-10-16 |
| *4.3* | Software Setups | 15 days | Thu 20-10-16 |
| *5* | Testing | 35 days | Thu 10-11-16 |
| *5.1* | Hardware Testing | 10 days | Thu 10-11-16 |
| *5.1.1* | Comparison to DMM | 10 days | Thu 10-11-16 |
| *5.2* | Software Testing | 25 days | Thu 24-11-16 |
| *5.2.1* | Comparison to monthly billing | 25 days | Thu 24-11-16 |

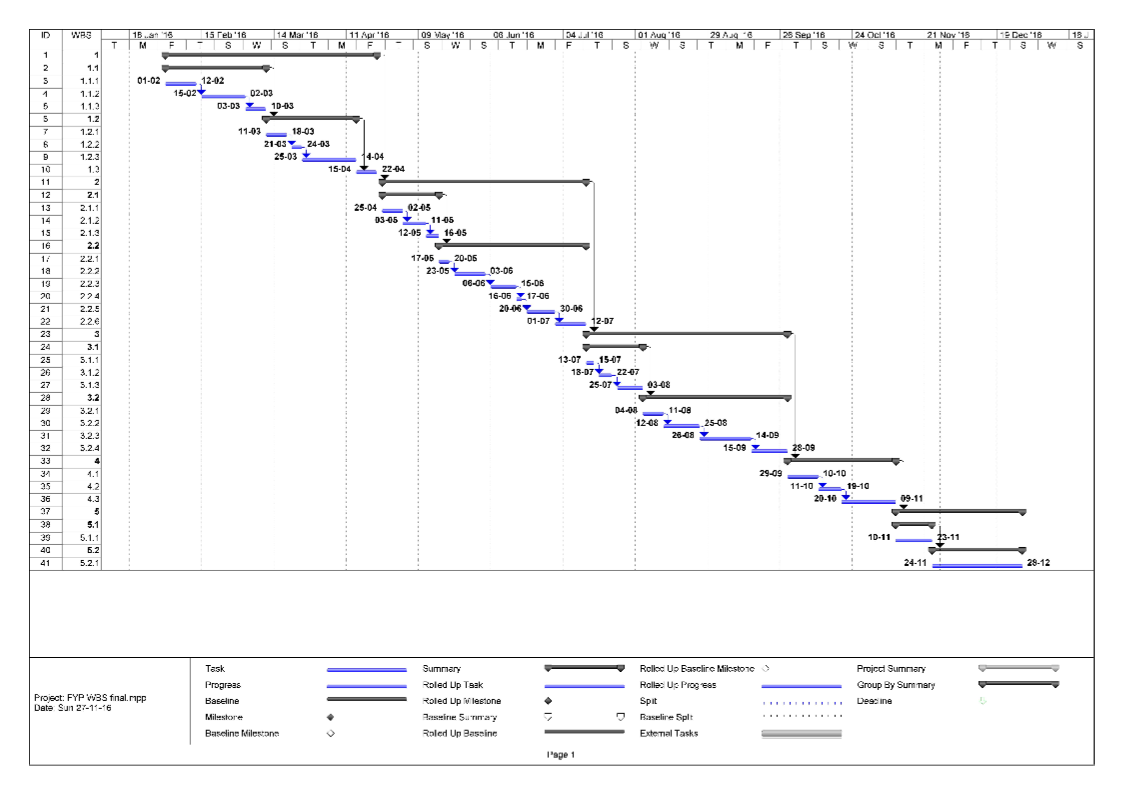


Figure Gantt Chart

Project Group Organization / Work Load Distribution

The manner in which group is organized and the mechanisms for reporting are noted.

1. Working Session / Snap shots of deployed system

Labeled Graphical User Interface (GUI) snapshots of any one work session is presented If no GUI exist than this section can be omitted or if exist, provide actual snap shots of deployed system.

1. Reference
2. https://learn.sparkfun.com/tutorials/raspberry-gpio
3. http://data.designspark.info/uploads/images/53bc258dc6c0425cb44870b50ab30621
4. https://en.wikipedia.org/wiki/Microcontroller
5. http://www.geeetech.com/wiki/images/4/4b/ArduinoMega2560\_r2\_front.jpg
6. https://en.wikipedia.org/wiki/Python\_(programming\_language)
7. <https://en.wikipedia.org/wiki/Time_series>